Trading Platform with Prediction Algorithm

Oles Chaban - 9002

2023-2024

Contents

1 Analysis

1.1 Project Background

Over the last few years I have had a growing interest in financial markets, and methods of analysing these markets mathematically in order to make predictions on price. Whilst looking into these different methods I believed that I could implement my own machine learning methods into the field of technical analysis, and so this inspired me to build an entire portfolio management system, based off of my own prediction algorithm.

1.2 Client Discussions

My intended clients for this project are mainly day traders, and potentially small teams of people working on one fund. I therefore spoke with the company Sigma, which is creating a pattern matching software for technical analysis, but is aiming to identify only more traditional technical patterns. The key takeaway from my conversation was not to over-fit to the data I have. As I am not able to gain access to the same level of data as a proper company, I am limited by the data set of stock tickers that I have, and the dates of the historical data from Yahoo Finance (where my program downloads my data from). Although I have both a long list of tickers, and Yahoo Finance has a large range of dates to access from, there is a limit on requests that I can make to the site. This means that I have to use my data sparingly, and prevent making requests to the site as much as possible, which makes my model prone to over-fitting. They also thought that the program would need some sort of method to calculate entry and exit points, as though the general prediction of the model may be right, the entry and exit points are just as important to not lose out on gains.

1.3 Research

My first area of research was into how to prevent over-fitting in my machine learning algorithm. Over-fitting is when a model produces an output that corresponds too closely or exactly to the training data, and is therefore not necessarily accurate on new data. In my case, this would mean the polynomial that is created to match the stock price is just a polynomial that directly passes through every data point.

1.3.1 Preventing Over-Fitting

Hold-out: This technique of preventing over-fitting involves removing a subsection of your data to be used as testing data, a common ratio of training to test data is 80% to 20%. The model is then trained on the training data, and the outputs are compared to the test data. Only if the predictions made by the test data perform well on the training data are they then used as outputs of the model. Some disadvantages of this technique are that it requires a lot of data, as the training data alone needs to be enough to fully train the algorithm. There is also the chance that the test data does not contain any of the same patterns which were found in the training data, and so not all the patterns can be tested. With a sufficiently large set of testing data this should not be an issue, but increasing the amount of data requests significantly increases the run time of the code, and increases the number of issues with the API that retrieves the data.

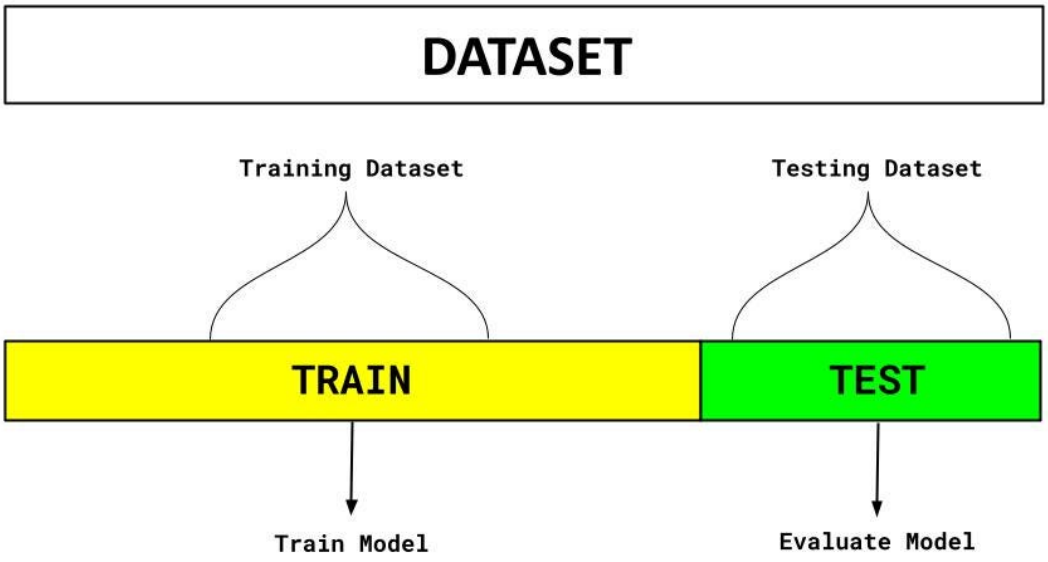

Figure 1: A visualisation of the hold-out technique

Cross-validation Similarly to hold-out, cross-validation uses a subsection of data as testing data, and the rest as training data. Unlike hold-out, cross validation splits the data into k number of evenly sized groups, usually 5 so that each is the standard 20% size. Then the algorithm is run, using one group as test data, and the rest as training data. This process repeats k times, so that each group is used as test data one time, and as training data every other time. The advantage of this over hold-out is that all of the data is used as testing data at some point, so patterns that may occur in one subsection of the data but not any others would still be found. However, as it repeats the entire algorithm x times it is much more computationally expensive than hold-out.

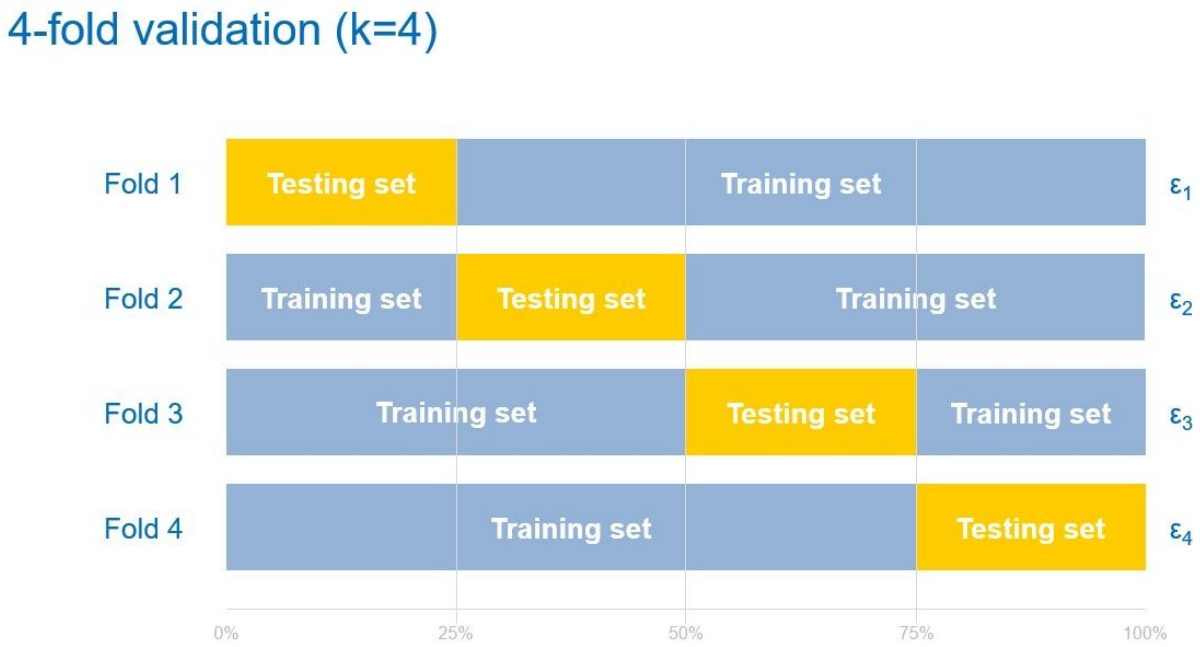

Figure 2: A visualisation of the cross-validation technique

Signal vs Noise Signal refers to the underlying, general information that the data is giving you, which can then be accurately applied if new similar data is found. Noise is the random movement of

the data within your data set, which is unimportant to the main trend of the data, and is simply just slight randomness in individual data points moving them away from the signal of the data. It is easy, especially in a stock chart, to be lost in the noise of the graph, as the price may greatly fluctuate even on a daily basis, but then stick to one general trend across a month or a year. It is therefore important for my algorithm to not fit to the noise of the data, but rather the signals, and there are a few methods of doing so. One way is to remove any obviously extreme values from the data, as this would prevent my curve from fitting towards them. Another is to use low order polynomials when fitting, this means that the polynomials would have few turning points, and so could only follow a few general trends within the data they are given, as they are not able to repeatedly change their direction to follow any tiny change in the momentum of stock price. A third way is to only look at data over longer periods of time; as mentioned before, it is common for there to be large fluctuation on a daily basis, but for the price to revert to a mean over a longer period of time.

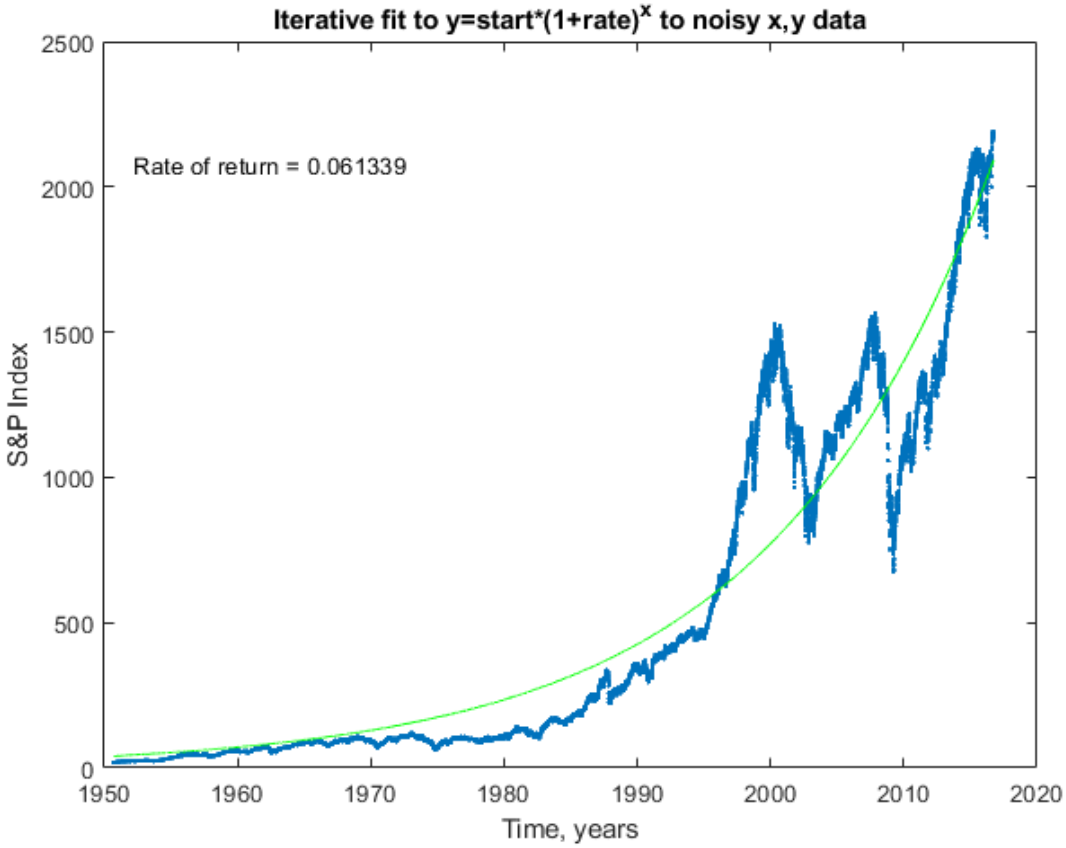

Figure 3: A visualisation of noise in a stock's chart, there may be more details in the real model

1.3.2 Entry and Exit Strategies

MACD One of the most common methods of determining entry and exit points in technical analysis is MACD, Moving Average Convergence/Divergence. This method of entry and exit involves using three different exponential moving average lines, EMA's, and the points at which they cross. In a simple moving average the time period represents the number of days from which the average was calculated, i.e. a 20 day period takes the sum of the close of the past 20 days and divides it by 20. However, in an EMA you follow this calculation:

$$EMA{=}Price_T * k + EMA(y) * (1 - k)$$

where T is today's date, y is yesterday's and k is your weighted multiplier

$$k{=}\frac{2}{\text{Time Period} + 1}$$

This equation above is a recursive process, weighting the recent days more greatly than the past days. 2 is the most commonly used number in industry for the smoothing factor, the numerator in the above equation, by increasing this number you increase the weight on the recent days more. A longer time period, for both SMA's and EMA's provide a view of long term trends, while short term trends are better represented by short term EMA's and SMA's. The advantage that EMA's have over SMA's is that they are more responsive to recent trends, so are more likely to be helpful for showing when to enter or exit on a stock, and so are used in MACD.

To calculate MACD you simply subtract the 26-period EMA of a stock from the 12-period EMA. You then need to take the 9-day EMA of the MACD line, also called the signal line, and plot it alongside the MACD, points where the MACD crosses from below to above the signal line indicate buy points, and points when the MACD line crosses from above to below the signal line indicate sell or short points.

Some disadvantages of MACD are that it often produces false positives, and gives buy or sell signals when there aren't any, and it is also is known to miss signals on occasion.

Relative Strength Index The Relative Strength Index (RSI) is a momentum indicator, measuring the speed and magnitude of recent price change to evaluate if a stock is over or undervalued. Though it signifies when a stock is overbought or oversold, they can also be used as a buy or sell indicator. The formula for the first period of RSI is:

$$RSI_1 = 100 - \frac{100}{1 + \frac{\text{Average Gain}}{\text{Average Loss}}}$$

The standard period for the RSI to be calculated is 14 days, so the averages would be taken of the first 14 data points. After this first RSI is calculated, each following RSI is calculated as follows:

$$RSI = 100 - \frac{100}{1 + \frac{(\text{Previous Average Gain} * 13) + \text{Current Gain}}{(\text{Previous Average Loss} * 13) + \text{Current Loss}}}$$

Times 13 is used, as the time period, as stated before, is 14 as a standard, and you are adding one more data point. A key note is that both the total loss and gain are divided by 14, the total number of data points in one period, not by the number of days in which there were gains or losses.

1.3.3 Assessing Risk of Investment

The most common way to calculate the risk of an investment is through the risk to reward ratio.

Simple Risk to Reward To calculate a simple risk to reward ratio, you take the maximum gain of your investment, and divide it by your maximum loss of that investment. For my program maximum gain will be taken as mean relative change when a specific trend is found, plus one standard deviation of that change, and in a simple risk to reward your maximum loss is the total input money. The issue with this, however, is that you are unlikely to ever let your position fall to zero, and it is unlikely for a position to have a value fall to zero.

Adjusted Risk to Reward To make the ratio more realistic you can incorporate a stop loss. For example if you invest 5 shares worth \$50 each, with a predicted maximum gain of 0.3 and a stop loss of \$40, your maximum gain would be \$15 and your maximum loss would be \$10, giving a risk to reward ratio of 1:1.5. If you did not include a stop loss this ratio would be 1:0.3, a much worse ratio. The standard 'good' risk to reward ratio is 1:3, therefore the trading algorithm should always aim to make trades as close to this value as possible, however any in the range of 1:2-1:3 are acceptable.

Sharpe Ratio The Sharpe ratio is commonly used to analyse and compare the portfolio of an entire fund, however its input parameters can be slightly modified to allow it to calculate risk individual investments. The formula is as follows:

$$\text{Sharpe Ratio} = \frac{R_I - R_f}{\sigma_I}$$

Where R_I is your return on investment, R_f is the risk free rate and σ_I is the standard deviation of the investment. Both R_I and σ_I will be values that I have from the trend finding algorithm, and R_f can be calculated as follows:

$$R_f = \frac{1 + \text{Government Bond Rate}}{1 + \text{Inflation Rate}} - 1$$

This is deemed to be the minimum amount of gain that you can have without any risk, as government bonds tend to be extremely stable. The Sharpe ratio is a very popular and reliable metric of risk, however in my case it will be relying on my forecasting algorithm as the 'reward' parameters, which may decrease the accuracy of the prediction.

Beta Beta is a measure of systematic risk compared to the market as a whole. Systematic risk is the general, unavoidable movement of the market as a whole, and what the value beta gives you is how an individual stock moves relatively to the whole market. This is not only useful in calculating risk, but also in making predictions on stock price, if you know its beta value and have a prediction for the market change. The formula for beta is as follows:

$$\beta = \frac{\text{Covariance}(R_e, R_m)}{\text{Variance}(R_m)}$$

Where R_e is the return on an individual stock and R_m is the return of the overall market (usually taken as the S&P 500. Covariance is the direction of the relationship between two variables, whether they are negatively or positively correlated, and is calculated by the formula:

$$\text{Covariance} = \sum \frac{(Ret_e - Avg_e) * (Ret_m - Avg_m)}{\text{Sample Size} - 1}$$

Where Ret and Avg of e and m and the days returns and average returns over the time period of the individual stock and overall market respectively, and sample size is the number of days across which you are calculating covariance.

1.4 User Interface

The final decision which needed research was the UI of the program. By looking at previous similar solutions there was a clear general standard used by trading platforms such as the one in the image below.

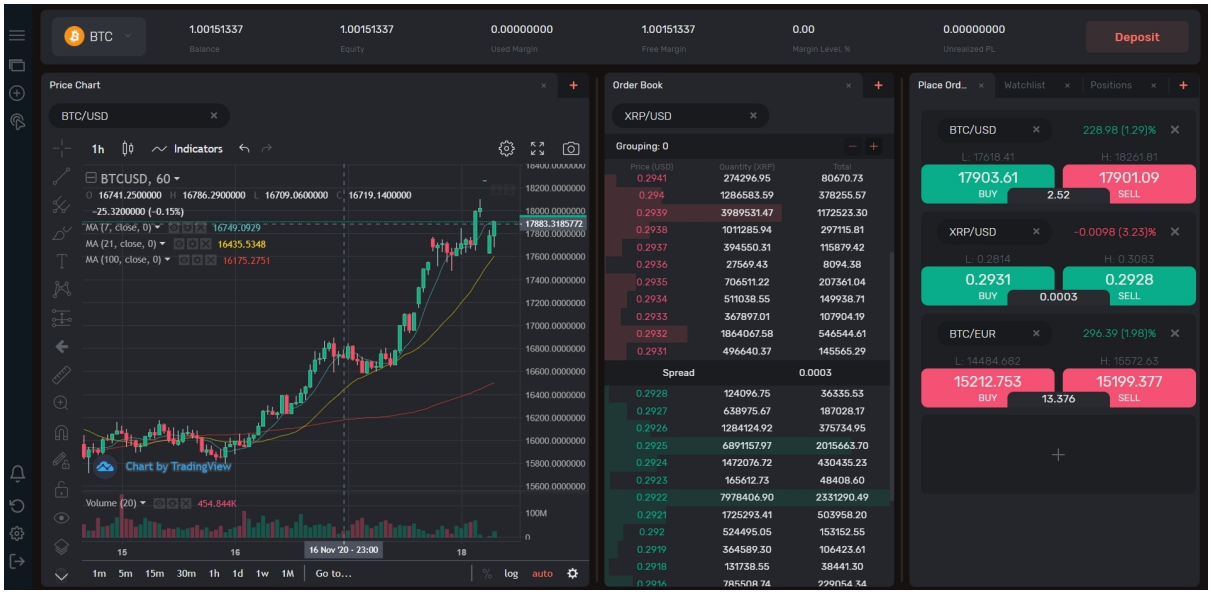

Figure 4: An example of a typical trading software screen [1]

This is just one example of a trading platform, however I believe it shows all the industry standards very well. It has a simple background, with bright and distinct buttons for commands such as buy and sell.

There is also a candlestick chart of the stock price (although the candlesticks will likely not be possible for me to implement due to lack of data). Aside from these few key highlighted features the page is very busy and full of 'hidden' information and features.

1.4.1 Client Feeback on Current UI's

After speaking with my client they highlighted positives and negatives from the UI in order to aid my decisions.

The main takeaways were that there was no need to drastically change anything, this sort of UI is extremely popular in this industry, so is the expectation of the user, especially new users. The bright contrast between the buttons was also highlighted to be important, as especially under high pressure situations such as trading it is less common to attentively read each label, and so being able to use colour to make quick decisions is important.

One thing that my client said they did not particularly like was the 'business' of the screen as 'you are unlikely to need all of this functionality all at once'. However they said they did like having some extra data also shown onscreen, so that you did not have to waste time searching for it.
This conversation influenced requirements **2.1-b** to **2.1-f**, **2.1-h** and **2.1-i**.

1.4.2 My UI:

Based off of this feedback I created some of mocks up for my own UI. The first and second images are of the welcoming screens of my user interface, the first for logging in and the second for creating an account. I chose to use a matching high contrast, dark background colour scheme to that of the previous UI, so that the user was given the impression of a trading platform from first launching the program.

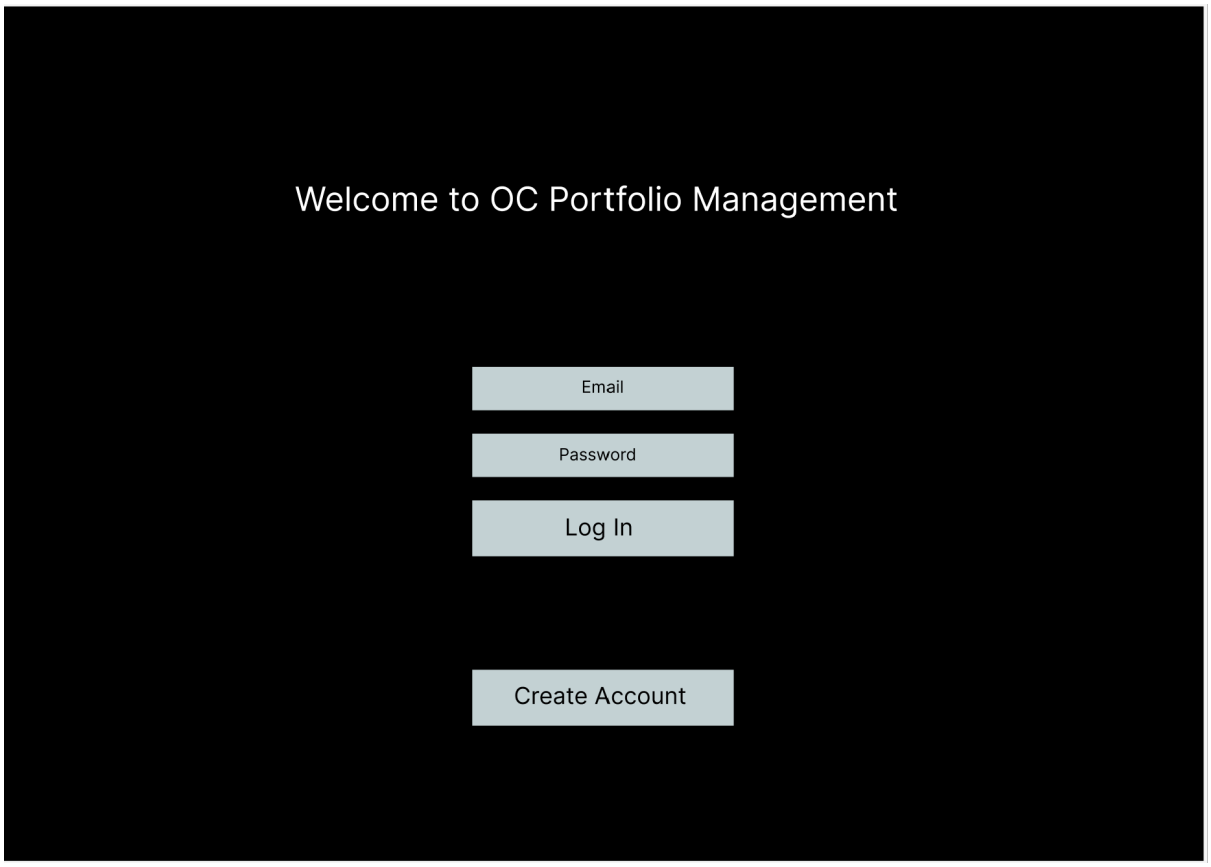

Figure 5: A mock of my login screen

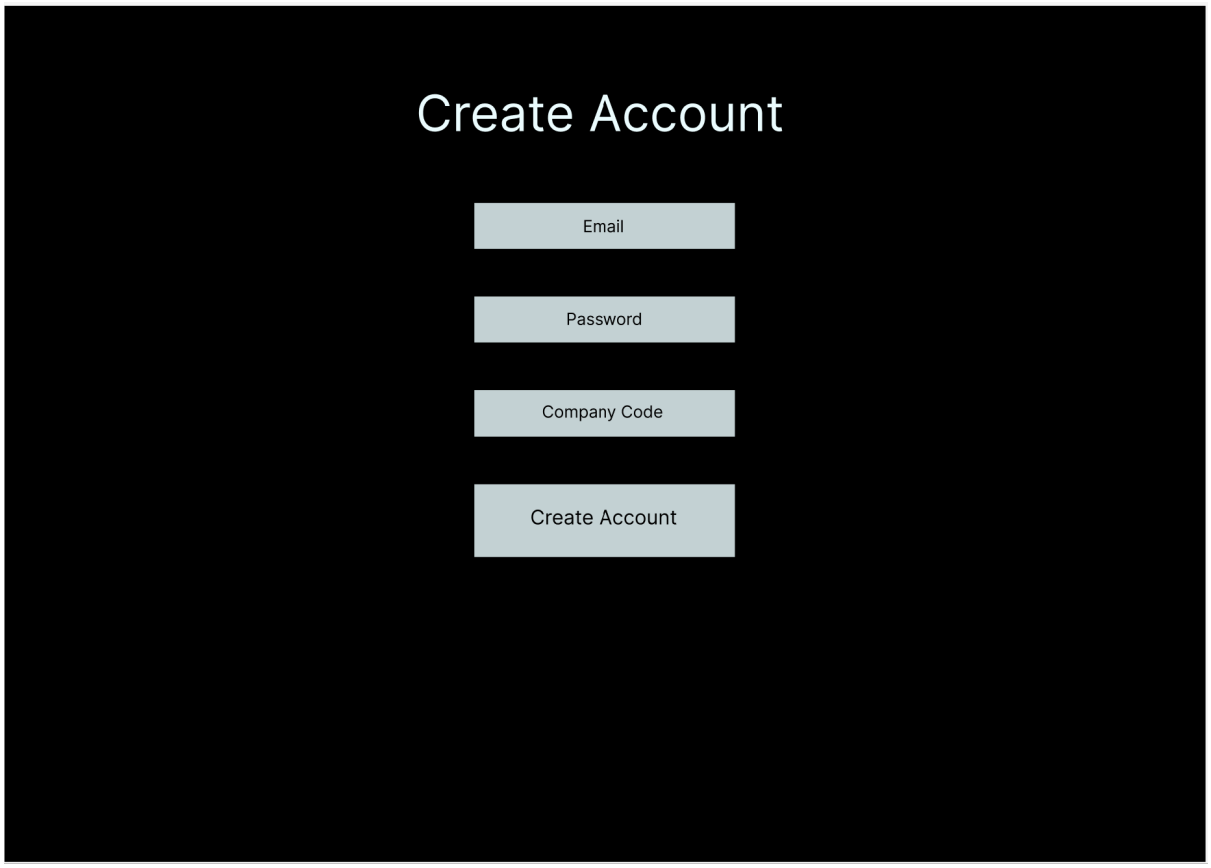

Figure 6: A mock of my account creation screen

The next figure is a mock of the home screen of the platform.

I wanted to again incorporate a similar style to the previous UI, highly contrasting the individual stocks and the background. I also wanted to incorporate some use of data into the home screen, rather than just displaying some random stocks that may be of no interest to the user. Hence I chose the two categories 'hot stocks', which show the three most bought stocks yesterday, and 'top positions' which shows the 3 stocks in which you hold the most equity. Alongside the name of these 6 stocks being displayed I also chose to include their 30 day chart and percentage change in this time. This I believe is the most key information needed for the user to immediately see, and is limited enough that it does not clutter the UI.

The other key features on the screen are the account button in the top right, the search stock function and the notification function. From these functions you can access any other information in the program, and therefore I decided it would be valuable for them to be immediately displayed. I did not create a mock for the search stock function, because it will likely be a simple pop-down of the search return. A mock of the notifications pop-down is shown below the home screen. The only notifications that the program will be able to produce are for buy and sell recommendations for stocks inside your portfolio or watch list, and so I have again limited it to a simple pop-down, so that the user can still make use of the main home screen.

After showing these mock up to my client their feedback was as follows. They liked the idea of showing popular stock and your main stocks, however said 'maybe you could add some other categories such as biggest growers of droppers and then put them on a rotation swapping with the other categories'. They also liked the contrast between stock and background and consistent colour scheme with the bright green being positive and red negative, however did not like the background colour of the notification pop-down as it looked too different to the rest of the screen. As for amount of data on screen, they said that there was a 'good amount' for it to be accessible to beginners, but that there could perhaps be some more to also attract more experienced users.

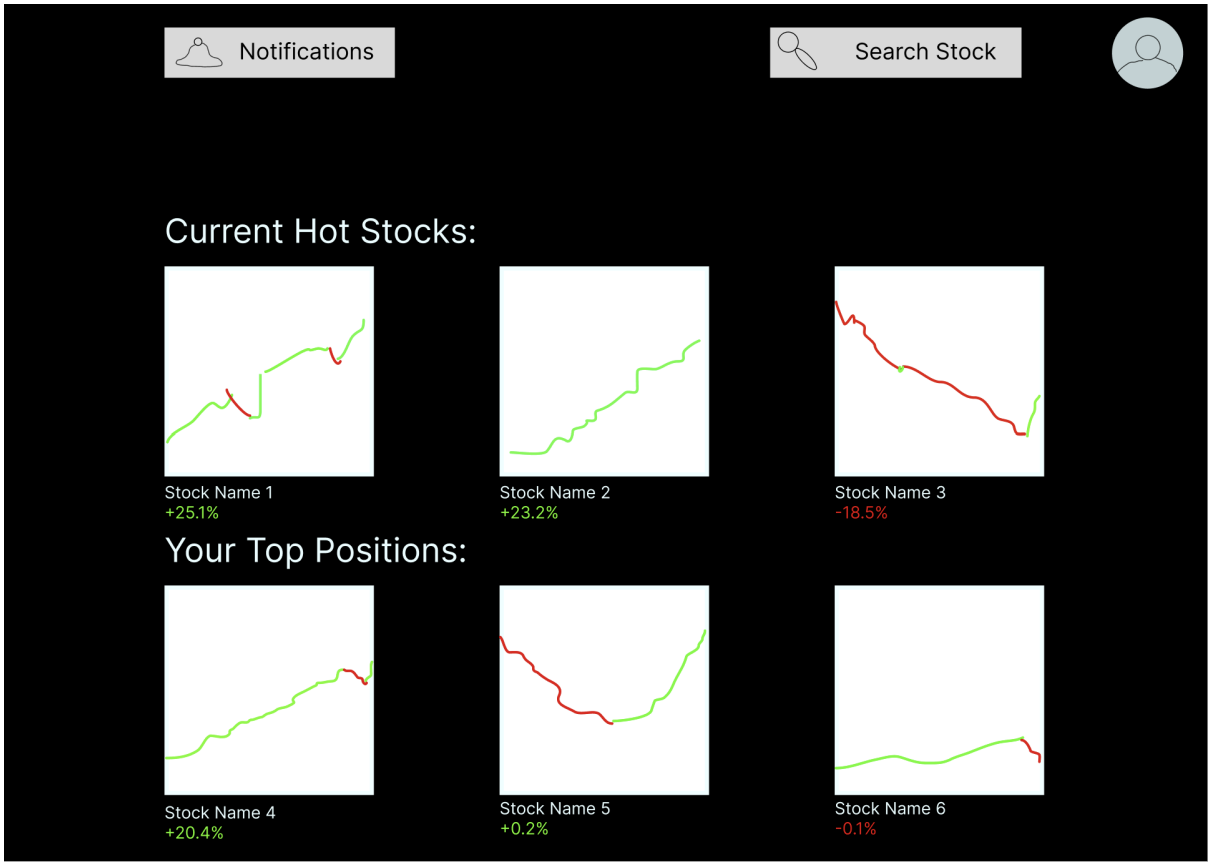

Figure 7: A mock of my home screen

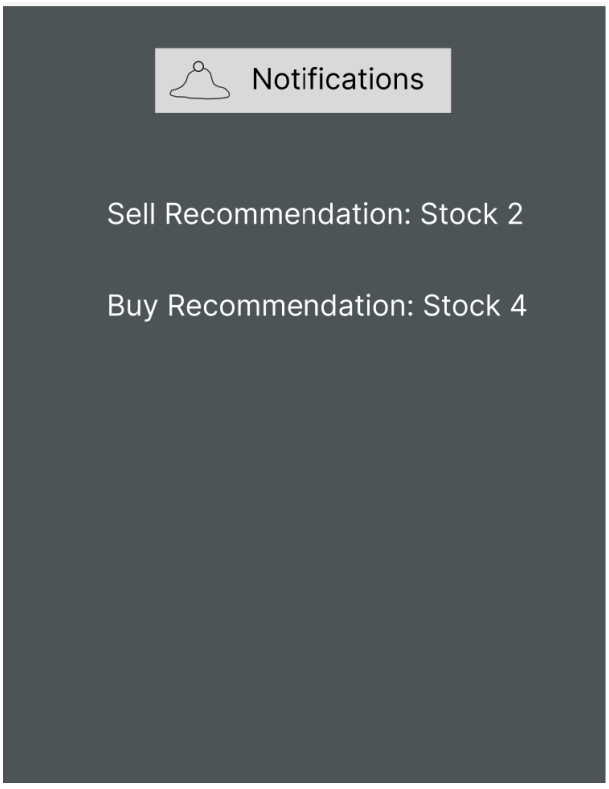

Figure 8: A mock of my notification pop-down

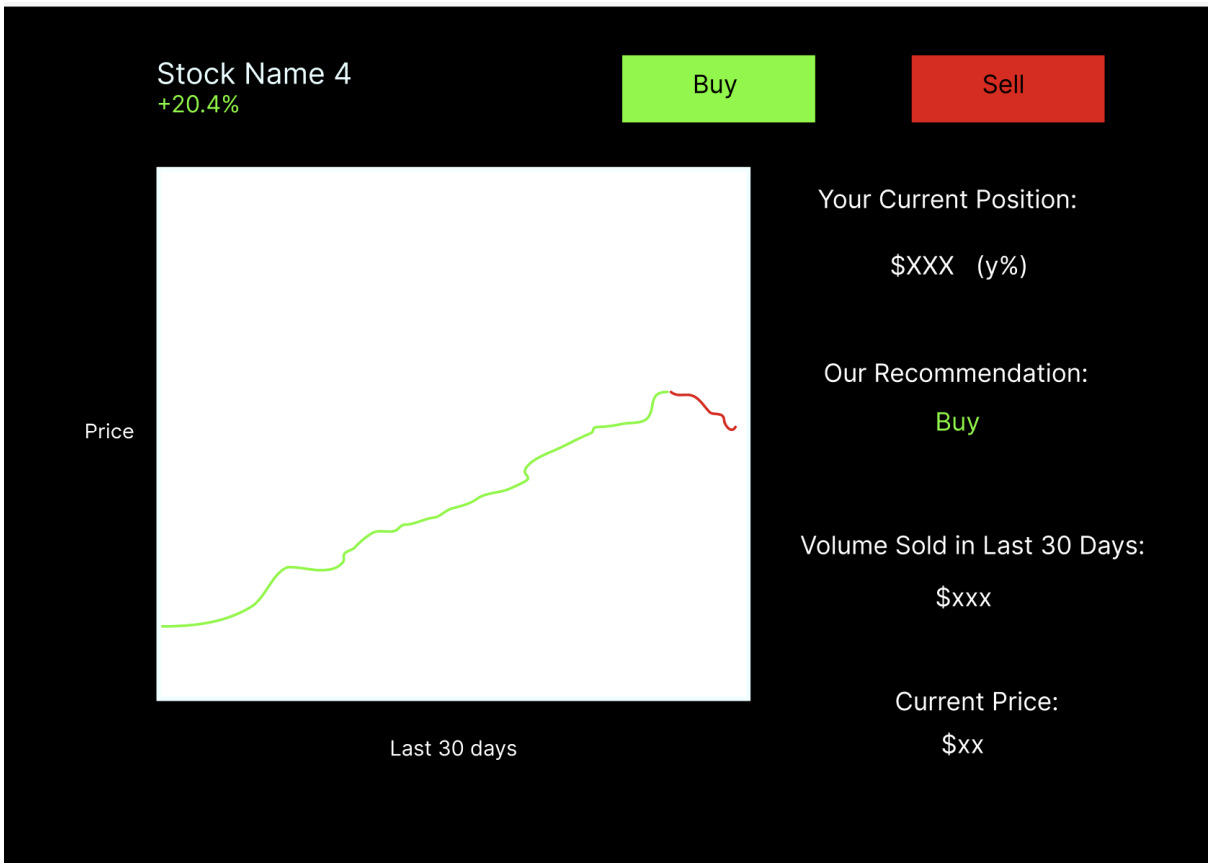

Figure 9: A mock of my stock screen

The above figure shows what I envision my 'stock screen' to look like. This screen will show when the user clicks on the name or image of a stock, or uses the search function to find a stock.

I tried to heavily incorporate elements of the previous UI from my research into this screen, as it is the one that will likely be used the most, and therefore needs to be similar to what users are used to. Therefore I used the same colour scheme, with a bright green for buy and red for sell, with a similar reasoning as earlier outlined - to make it clear in high pressure situations what you are doing.

I also wanted to add some data to the screen, as in the researched UI, unfortunately I would not have access to the same amount of data. I still decided to add as much data as I could, and this is something that my client agreed I should do, as otherwise the screen would look 'bare and leave the user unsure as to what they should do with the stock'. The information I chose to include was current position, so the user could see how much of the stock they own, 'our recommendation' which will display the result of the entry/exit calculator that the program has to inform the user of what they should do with the stock. I also decided to include volume sold in the last 30 days, as this is another common figure that I would be able to get from my available data source, and of course the current price of the stock will also be shown.

After speaking with my client they also emphasised the importance of being able to clearly see the stock chart, as 'that is what a user would immediately look at'. Therefore I added a large graph to the left of my data showing the price over the last 30 days. I chose to use the last 30 days as this is the period over which most of my calculations are done, and price before that would unlikely be a significant indicator for the user.

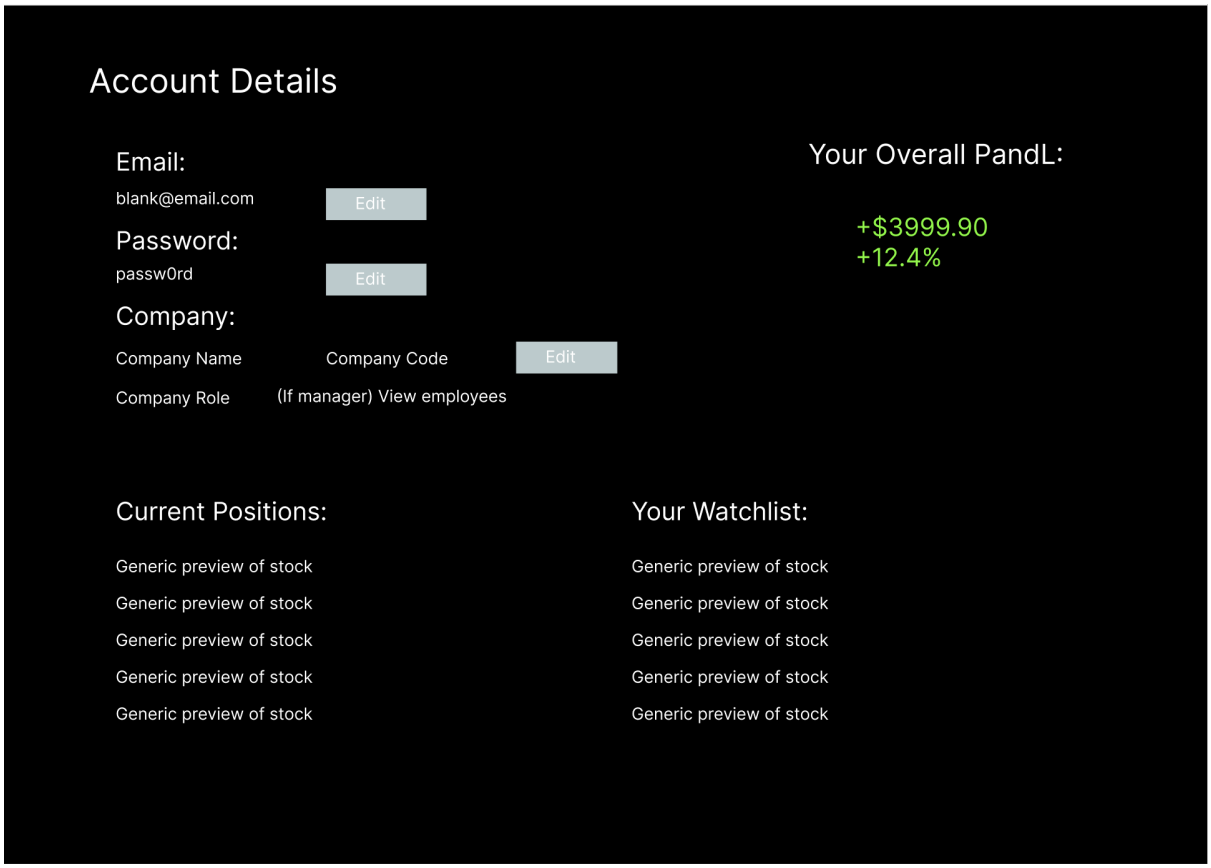

Figure 10: A mock of my account screen

Finally, above is a mock of my account screen. I again chose to use the same colour scheme, with green for positive red for negative, and a darker contrasting background.

The account system I am constructing is limited, so I was able to add all the information onto just one page. The sign up/log in system will require an email and password, and these will both be shown on the account page where the user can also alter them. When the user joins the platform they are able to join a company, or start their own, and this is done through a company code. As shown on the mock up there will be an option to change this code and join anther company. Beside this key information such as the name of the company and your access level within it is also shown beside this.

The account page will also show some information about the users stocks and their success, such as the overall profit and loss in the top right and their current positions in the bottom left. The final information that will be shown through the account page is the user's watchlist, which they can edit through this page also.

1.5 Interview with End-User

Following my research, and the conversations I had with various potential clients, I decided to isolate one specific target user, and conduct a more detailed interview. This would allow me to gain specific feedback from a potential user, which would help ensure that the requirements aptly tackled the problem at hand.

Interviewer: Oles Chaban

Interviewee: Taras Chaban - former professional stock trader

Oles: You have a lot of experience working on trading platforms. What would you expect to see on one?

Taras: Other than simply being able to view stock price, both past and present, and buy and sell stocks. It is important to see what is going on in the market as a whole, and see how your portfolio is currently diversified.

Analysis: The response clearly outlines the must haves in a trading platform, buying, selling and viewing stocks. But also provides insight into some metrics that should be looked at for helping users analyse stocks.

Oles: How familiar are you with technical analysis being used for predicting stock price?

Taras: I am very familiar with the concept, although I have never used it myself to help make trades. I have always used more classical statistical methods of analysis.

Oles: Would you be open to having a technical analysis software running on your platform in order to inform trade decisions?

Taras: It can't harm to have any more indicators running simultaneously, at the very least it would give something to compare against even if you end up using some other prediction metric. However, if you have an original indicator that is also accurate that is attractive to any user, regardless of their background with technical or any other type of analysis.

Oles: How customisable should the technical analysis algorithm

Taras: I don't think that customisation will be very important if it is to be used as an indicator. It can be thought of as just a fixed function like MACD. You also don't want to leak to competitors what your algorithm does, which customisation may give some insight to.

Analysis Having a custom indicator will help make my program stand out from the many readily available trading platforms. Customisation of this function is also not as important as I initially thought.

Oles: What would you suggest to use as the platform for the program, desktop, web or mobile

Taras: Definitely desktop. On mobile devices the screen is too small, and you don't want to miss-click on trades that could potentially lose money, and the processing power may also not be enough if the algorithm predicting price is heavy duty. With web programs you have delay as all your inputs need to be transported across the internet, this could pose some issues if you are making quick sequential trades, especially if you are using a bot.

Analysis As per my expectation a desktop application will be most suitable.

Oles: Do you value having a risk metric when making trades?

Taras: Not particularly. By the time you are ready to buy a stock you (should) have already calculated risk. Also, any risk metric that you could show when buying will likely be simple enough that an experienced trader would be able to carry out the calculation in their head.

Analysis To my surprise incorporating risk into the UI does not seem to be important.

To summarise, the interview has provided me with several insights into which features should and shouldn't be implemented. This will be helpful when confirming my research decisions, and ensuring my requirements fit the desires of the user.

1.6 Research Decisions

Based off of my research and client feedback I decided to implement some new strategies into my project.

1.6.1 Preventing Over-Fitting

The first technique that I decided to use to prevent over-fitting was hold-out. I chose hold-out over cross-validation as loading and accessing tables of data is a slow process, which takes up most of the processing time for the algorithm in MATLAB, my chosen software for the machine learning algorithm. Therefore, if I chose to use cross-validation I would need to access my tables k amount of time, where k is the number of folds, which would significantly slow the processing speed. The other option would be to load each table once, and then store is as a matrix, however this would take up a large portion of memory, and therefore again would slow processing speeds.

To prevent the effects of over-fitting to signals I decided to limit the number and orders of the polynomials I would fit, for example I would only fit polynomials of order 3-5 for each set of data. Given that I can only access data on a daily basis, I also decided that I would only allow the model to operate with time periods for the data greater than 15 days. Including the fact that there are days where the markets are closed, this would only give 11 data points as a minimum, enough for any found fits to still be valid. For these shorter periods, 15 to 25 days, I will also only consider polynomials of order 3, 4 and 5.

The requirements concerning loading data for the algorithm are detailed in **2.2-a**, **2.2b**, **2.5.1-c** and **2.5.1-g**. The requirements concerning over-fitting are detailed in **2.2-c**, **2.2-d**, **2.5.1-e** and **2.5.1-f**.

1.6.2 Deciding Entry and Exit Points

To recommend entry and exit points I decided that it would be best to incorporate a MACD calculator, as it tends to be the industry preference, and when speaking with those working at SIGMA, it was the first they recommended.

This decision is outlined in requirements under section **2.3**.

1.6.3 Incorporating Risk

Following my interview I decided against incorporating any risk calculation into my UI. From the interview it seemed that if I were to incorporate such a calculator, it would have to be heavy duty for it to be of value to the user. Given my experience in the field I was not able to find anther such metric, and so to stop the UI from becoming over saturated with functions that would end up being rarely used I decided not to include a risk ratio.

1.7 Project Outline

In traditional methods of technical analysis, there are developed theories and corresponding trends in price which can be found by humans or machines and then used for prediction. My belief was that the reasoning behind these trends was unimportant, but rather it was important that there was a consistently similar result after a specific trend occurs. This allows for me to create a larger set of trends to look for, therefore allows me to make more accurate predictions, and on a greater variety stocks. Then, by using my own algorithm alongside traditional methods I should be able to make informed recommendations for managing a portfolio.

The requirements for this algorithm are found in **2.2-e** to **2.2-j**

1.7.1 Chosen Technical Solution Method

To produce my algorithms on the back-end, I needed a programming language which could easily manipulate large data structures, as well as one that had an interface I could use to download data from the internet. MATLAB seemed suitable for both, as I had an API which could connect to Yahoo Finance, and when I had data loaded into my program I could make use of the optimised functionality that MATLAB offers with matrices. MATLAB allows for calculations to be applied simultaneously to an entire matrix, similar to a map function in functional programming, and has built in least-squared error functions that can fit polynomials to my data.

In addition to this I could also create a COM server for my local MATLAB client, and therefore call function from MATLAB directly in any other language I choose, making it flexible to implement with other languages.

A COM server is a local library or application (in my case application), which can be invoked by a running process. For my project in visual studio I will open an instance of the MATLAB command window, and then from this command window I can then call pre-programmed functions that I have made in MATLAB, which allows me to have my algorithms independent of the main code.

For the user interface WPF was most suitable, as the program is designed as a desktop application, rather than a web or mobile and WPF also allows for interaction of the back end with a UI.

The choice of UI framework then made C# and obvious choice for the running the main section of my code, as it pairs with the WPF framework, and is ideal of implementing object oriented designs. I would like to include and object oriented framework to modularise my code, which would allow me to debug sections of my code independently, making the program easier to maintain.

2 Requirements

2.1 User Interface

The User Interface should:

(a) Show a log-in or sign up page upon first opening the program

  i. The user should then be taken to the corresponding page based off their button click

(b) Show the user stocks that are currently trending in the last day

(c) Show the users total profit and loss on the platform, as well as the values of any individual positions on each stock page.

(d) Have a 'watch-list' to which users can add stock and track their progress. Updates on this watch-list such as good entry and exit points should also be shown

(e) Show an image of the stock price of each company on the screen

  i. Next to any stock code there should be an icon showing the stock price over the last 30 days

(f) The user should be able to click any stock on their screen to take them to another page with more advanced information on that stock

  i. On this page they should be able to buy or sell (if they own this stock)

  ii. They should also be able to see other information, such as recommendation to buy, sell or hold and the current price of the stock

  iii. The recommendation to buy or sell will be done using my learning algorithm - as described in requirement 2.5.1.d. Should there be no suitable prediction found by this method, due to there being no matches with the trained polynomials to the polynomial for the current data, the MACD calculator should be used as a substitute. This is detailed in requirement section 2.3.

(g) Update account information, in a separate 'account' section

  i. This includes email and password

  ii. The same verification should occur for email overlap within the account page as when logging in

(h) Display user's top 3 stock, and watchlist stocks as buttons on the account page

(i) View notifications from the software in a 'notifications' pop-down

  i. This should include and exit or entry points found in their watch-list or portfolio

(j) Have a vivid, consistent colour system which indicates to the user positive and negative/buy and sell

(k) Have navigation buttons for entire program, such as back to home page, close pop-down and close program

2.2 Trend Finding Algorithm

The trend finding algorithm should:

(a) Get real market data to use for analysis

  i. This is done by using an API to Yahoo Finance, where you are able download historical data from every previously recorded day

ii. The data is then to be stored on a separate csv file which the user can choose when to update the data on

(b) Handle issues of codes not returning data

i. Not all the codes from the data set I have return data for all the dates tested in the algorithm, exception handling is needed to ignore any null returns

(c) Split the data from the csv into individual blocks of data to be used in the algorithm

i. These blocks should contain consecutive dates, and should each only contain data from one stock, as well as all being the same size

(d) Remove a random 20% of the data blocks to be used for test data

i. This will then be later used to test the outputs of the algorithm to ensure they are accurate

(e) Fit a curves to the data and store the coefficients of these curves

i. The algorithm should fit polynomials of order 3 on each set of data that it is given

ii. The coefficients will then be stored in a file alongside other information for each fit

(f) When a new fit is found, it should be compared to the previously found fits

i. If a previously found fit is sufficiently similar then the new fit should be incorporated into the data of the old fit

ii. If a the fit is different to all previously found fits then it is added to the found fits in a new row

(g) Calculate the change in relative price for each set of fetched data after a month

i. When a new set of stock data is returned, the relative price change after a month should be calculated. If the fit found for this stock matches with a previously found fit, the mean of the two changes should be stored along with the fit

ii. The volatility of the change after a month should also be stored along with the change, if there was more than one fit found. For this model I will be using one standard deviation as the volatility

(h) Check whether there are any fits that are stored as separate fits, have converged so that they match the same new fit

i. If this scenario occurs then it indicates that the tolerance for combining two fits is too great, and so should be adjusted to be more strict

(i) Test the found fits using the test data, and then clean the found fits to only include those that are successful with the test data.

(j) Store the tested successful fits in a file

i. When saved the file stores: the coefficients of the polynomial, the mean change after 30 days for when this fit was found, and the standard deviation of this change

2.3 Entry Exit Calculator

The entry exit calculator should:

(a) Use the users stocks and watchlist stocks to determine whether or not the user should sell or buy stock

i. This should be done using the MACD calculator.

ii. The system will calculate the MACD by subtracting 26 day EMA from 12 day EMA, and then calculating the 9 day EMA of this result. Whether the MACD line goes from below to above this signal line, or from above to below, indicate buy and sell points respectively.

iii. For further details see pages 6-7 in my research section.

(b) Act as a backup to my own prediction algorithm, if it is to not produce a prediction for a certain stock then the entry exit calculator should be used as a substitute.

2.4 Database

The database should:

(a) Store all the information of users of the program

   i. Usernames, passwords etc. should be stored in a separate table

   ii. This should also include the total profit and loss of the user, and the stocks on their watchlist

(b) Store all of the stock tickers, used for the machine learning algorithm

   i. This is also to be used for comparing stock tickers/symbols and their names

   ii. The price of each stock will also be saved here

(c) Store all of the stocks owned by any user currently

   i. Alongside the stock symbols, this table should also store the last update to the price, the current amount of the stock that the user owns, and the email of the owner

2.5 Other requirements

2.5.1 In MATLAB:

(a) Return data for stock price over the last month to be used for plotting

(b) Return data for individual day prices of stocks to update the display prices when loading up the program

(c) Update stock data which is to be used in the trend finding algorithm

(d) Use the predictions made by my algorithm to predict price, and therefore give recommendations for users

   i. This is done by first fitting a third order polynomial to the price time graph of the stock for which a prediction is desired, and then comparing it with the previously found fits by the algorithm. If the polynomials match, with some tolerance, the predicted change by the algorithm is then assumed to be the same for this new stock, and this is returned as the prediction.

(e) Use a similar method as above to test each of the found fits by the algorithm, using the 20% of testing data, before they are saved as successful fits

(f) Remove any fits found by the algorithm that do not have sufficiently high changes after the fit occurs, or have too high a volatility in change after the fit occurs, from the successful fits

   i. This is because fits with too high a volatility can not be thought of as consistent, and with too low of a change there may again be slight randomness which makes the prediction wrong

(g) Have a function to return future price

   i. This is necessary in the prediction finding algorithm to see what the relative change will be after the 30 day period for which it is designed to make predictions for

2.5.2 In Visual Studio:

(a) Convert between stock codes and names through the database

   i. This to ensure that the code which is being used for data retrieval etc. matches with the name on the title of the stock page

(b) Verify that a user has entered their correct login details before loading their account

   i. When verified their watchlist stocks should be displayed on the account page, as well as their total PandL

(c) Verify that the email that a user is attempting to register with is not already in use, and stop them from being able to register if that is the case

(d) After login, the user's three most owned stocks should be displayed on the main page, alongside yesterday's most bought stocks

(e) Have a search function, using which the user can search for any stock which is saved in the database

  (i) This function should search for both stock codes and names similar to the user input

(f) Be able to buy and sell stock

  (i) When buying and selling, it should also update the database entry, or create a new entry if that stock has not yet been bought by that user, which relates to that user and stock

  (ii) Invalid inputs, such as letters or sell orders greater than the user's position should be dealt with accordingly

(g) Update profit and loss (P and L) of all users when loading this program

  (i) This is done by looking at when the price was most recently loaded price of each stock, and calculating the relative change from that to the current price. This multiplier is then applied to each position which contains that stock, and the users' overall P and L, using the sums of these changes

(h) Include logout button in account screen that clears user information and returns to login page

3 Design

In the design section I will outline my solution using the three levels of design, each with a different level of abstraction. The three levels are as follows:

1. First Level:
   The first level will consist of a flowchart outlining the general function of the program.
2. Second Level:
   The second level will consist of a class diagram, discussion of my use of object oriented programming and sequence diagrams.
3. Third Level
   The third level will consist of an identification of key data structures and algorithms, pseudo code of key algorithms and an outline of the database system.

3.1 Fist Level of Design

My program can be abstracted into one sequence of events, as there is only one 'scenario' in which the user can be. The diagram below is a simplified illustration of this event sequence.

Note that there is no 'end' state in my flowchart as there is no place at which the program will terminate by itself - it is all user driven - and adding connections at each process to an end state would have unnecessarily over complicated the flow chart.

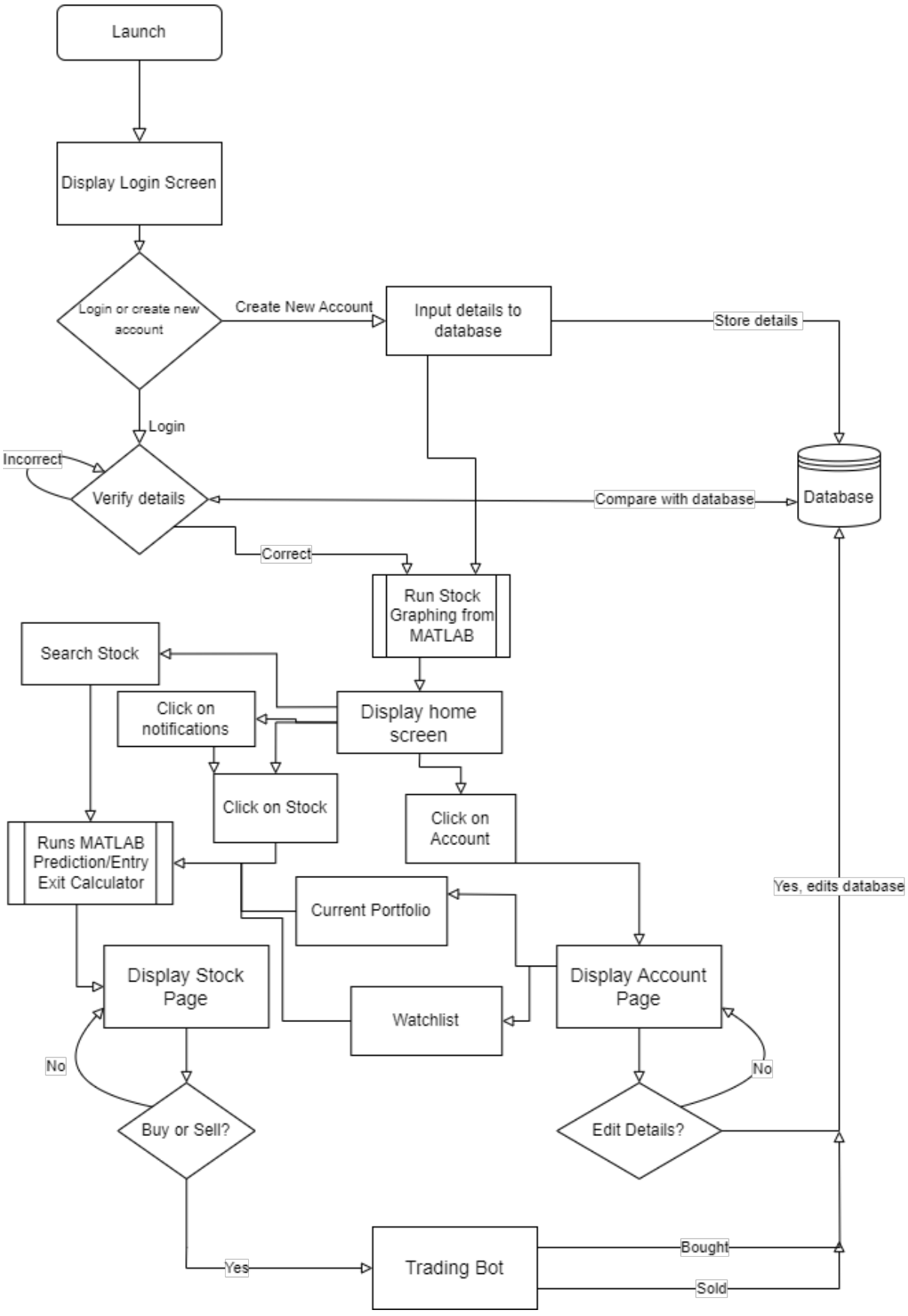

Figure 11: The general flow diagram of my program - note the lack of 'end' state

3.2 Second Level of Design

3.2.1 Class Diagram and OOP Discussion

Creating a class diagram is a key step in identifying interactions between objects in an object oriented paradigm (OOP), below is my class diagram.

I have used the standard UML syntax for representing relationships - shaded diamonds indicating composition, unshaded indicating aggregation, and arrows representing association.

Although MATLAB is not an object oriented language, I thought it would be informative to add it the MATLAB functions to the class diagram. This is because my relationships between the MATLAB functions can also be represented by OOP relationships. For example my testing and loading functions can be thought of as being in an aggregation with my main stock predicting algorithm. As they can both exist independently of the main algorithm, but are also both subsets of the algorithm. The MACD and EMA calculators can be thought of as having a similar relationship.

Note that to avoid over-cluttering the diagram some methods have been condensed into one line, or ommitted from the diagram due to their limited use.

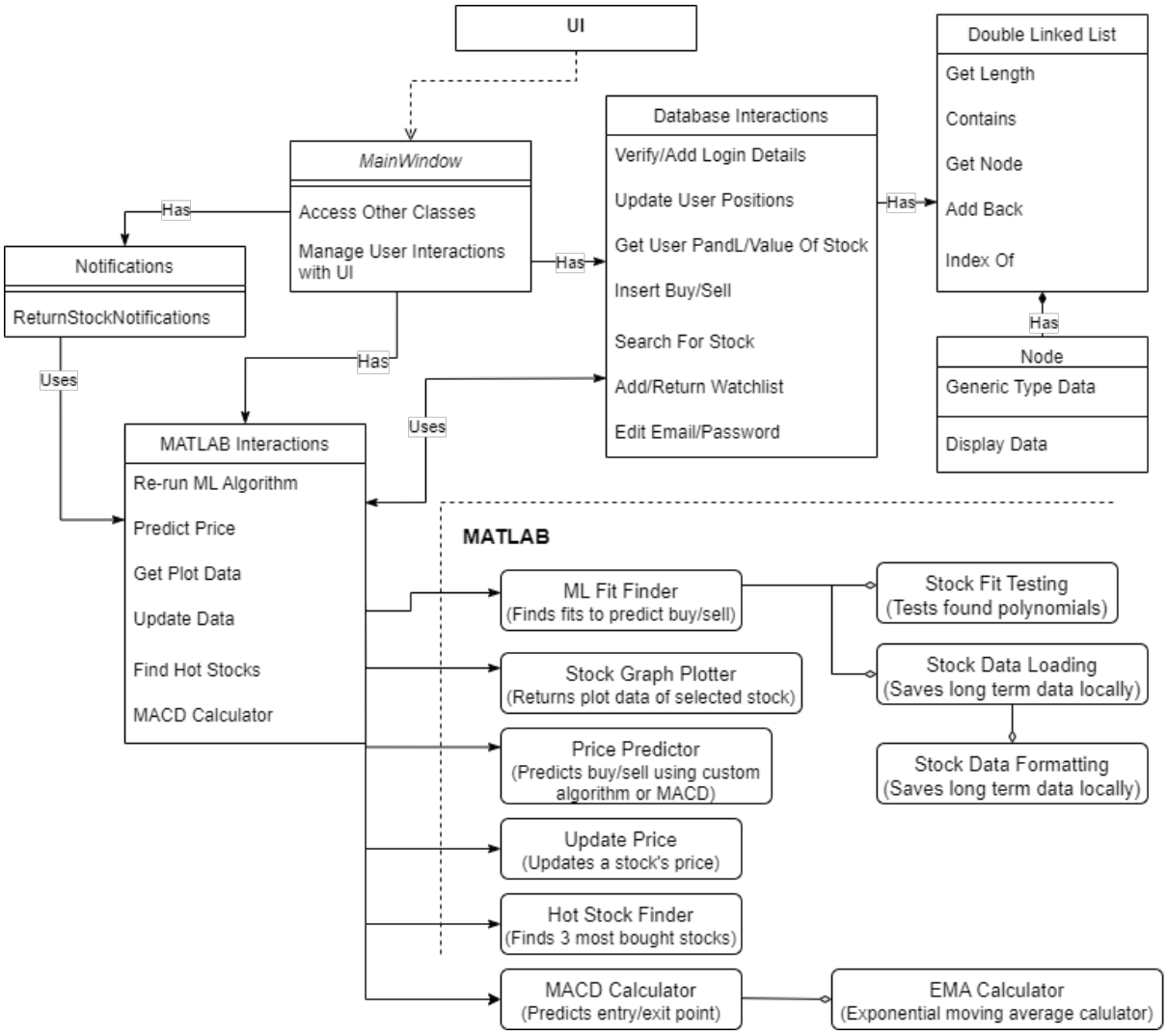

Figure 12: My Class Diagram

By choosing to use an OOP design it has allowed me to avoid significant repetitions in code, as I can call methods from each class across my entire program. It has also allowed me to decompose my code into sensible subsections, each of which has distinct functionality, therefore allowing independent testing.

By using aggregation between my custom lists and nodes, it allows me to be generic with the data being used for the lists, and also gives the potential to expand my program by using the nodes as a basis for further complex data structures.

3.2.2 Initialise Program Sequence Chart

In order to load my program, the following actions need to occur; outlined in my sequence chart below.

First the program needs to determine the most bought stocks from yesterday in order to display once the user logs in. This is done by creating a new MATLABInteractions() object, through which the function to find the most bought stocks is called in MATLAB.

The program then updates all the users' PandL's, by finding the current stock prices, and the last recorded stock price and calculating the relative difference. This relative difference is then applied to the users' positions in each stock they own, and the sum of theses changes is stored in the database as their new PandL.

Finally all of the graphing tools are formatted, and the past thirty days' worth of data for each of the hot stocks is loaded via MATLAB. This data is plotted for the home screen, and the login page is displayed.

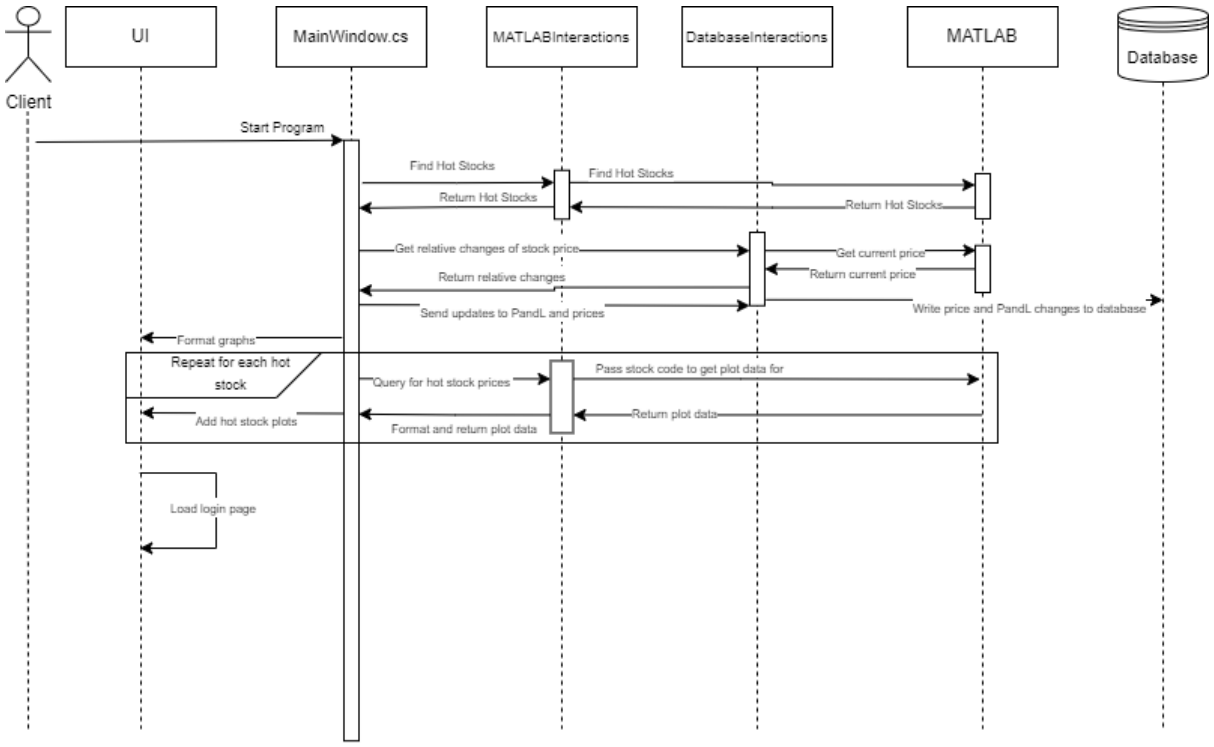

Figure 13: A sequence chart outlining what occurs when the program is run

3.2.3 Login Sequence Chart

The next sequence chart outlines the events that occur once a user attempts to login.

The first loop repeatedly verifies if the login details entered are correct, by comparing them with the entries in the database via the DatabaseInteractions() class. The user also has the option to create a new account, but this has not been shown in order to simplify the diagram.

After the details have been verified, the database is again queried via the DatabaseInteractions() class, requesting for the stocks for which the user has the 3 highest value positions. The last thirty days' worth of data for each of these stocks is then loaded via the MATLABInteractions() class and plotted onto the home page.

The next stage involves checking if the user has any notifications. First the database returns all of the user's stocks, and passes them to the MATLABInteractions() class. My entry/exit calculator, in this

case MACD calculator, is then run on each of these stocks. If any of these stocks return a sell prediction then that stock is added to the notification buttons.

Similarly the database is queried for the user's watchlist stocks and these are each added as buttons to the watchlist section in the account page.

Finally, the database is once more queried for the user's PandL, and it is also displayed on their account page.

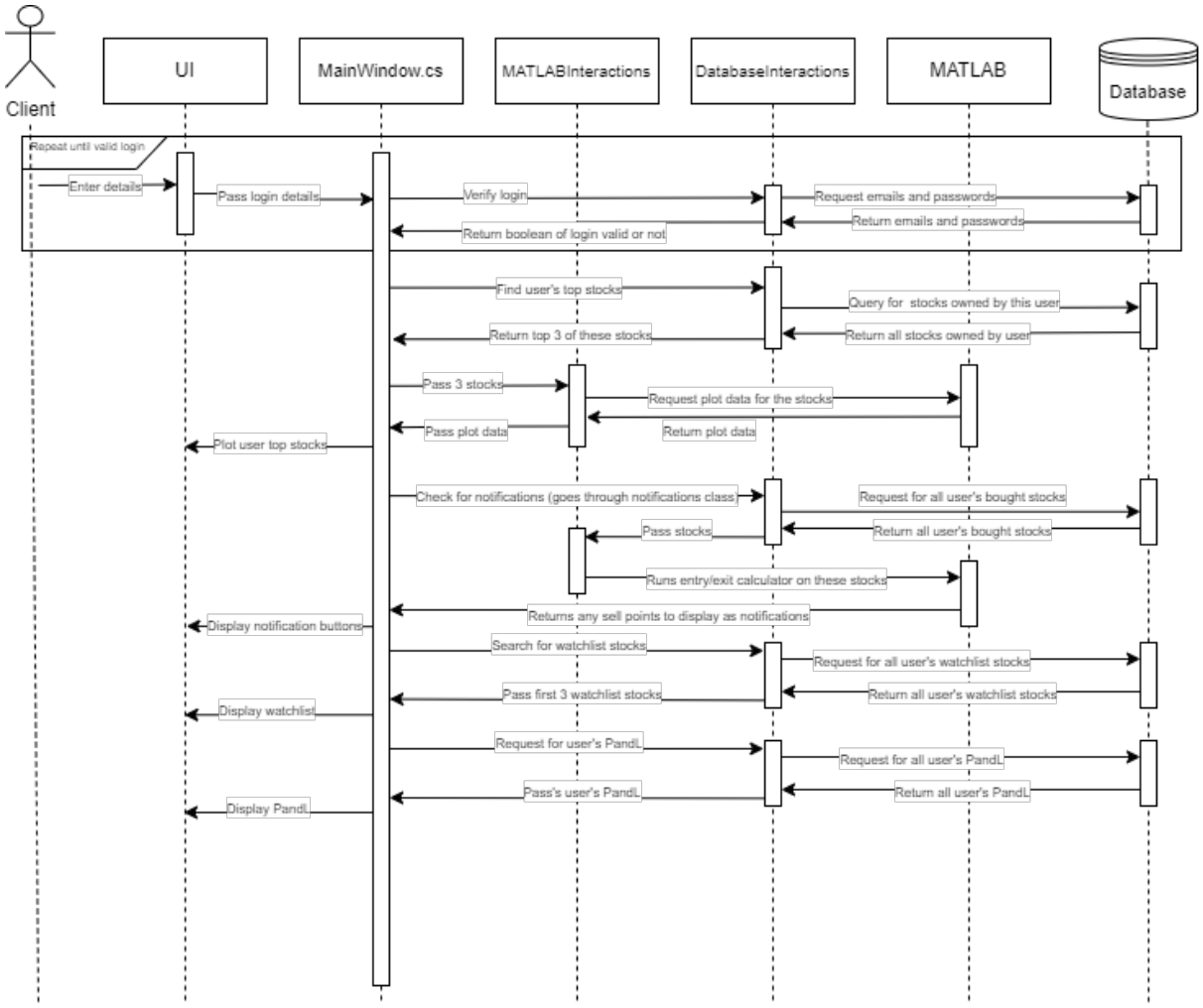

Figure 14: A sequence chart outlining what occurs when the user presses the login button

3.2.4 Load Stock Sequence Chart

This third sequence chart outlines what occurs when the user clicks on a stock page. The same process occurs regardless of where the stock is clicked, e.g. notification screen, home page or account page, the only difference would be the name of the stock which is inputted.

Initially, the database is queried for the price of the stock and this is displayed on the stock page. The email of the user is then also passed along with the stock code in order for their position to be returned and displayed.

From MATLAB the last thirty days of data is returned to be plotted on the main stock screen. And finally the my own prediction algorithm is run on the stock to give a buy/sell prediction, and if this does not return any result the MACD calculator is run as a substitute to display a prediction for the user.

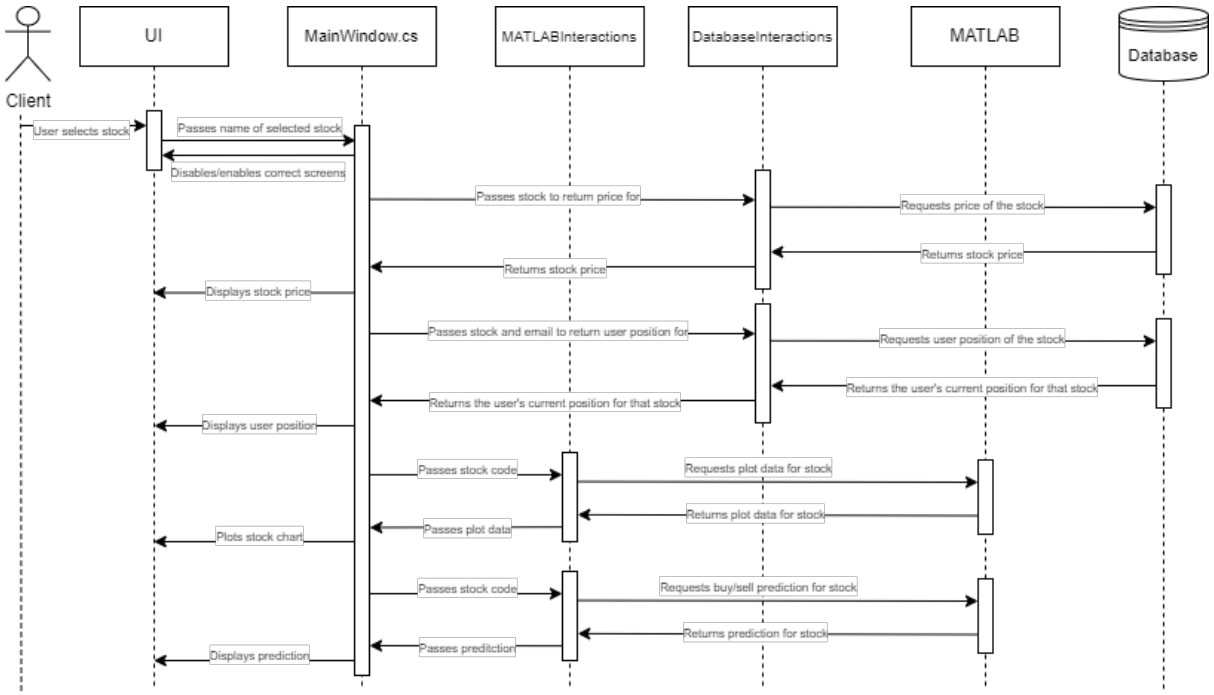

Figure 15: A sequence chart outlining what occurs when a stock is selected to be viewed

3.3 Third Level of Design

3.3.1 Database Entity Relationship Diagram

Below is a diagram outlining the relationships between my three tables, as well as the data stored in each one.

The LoginTable table contains generic information about each user, with the primary key being Email, as upon creating an account it is verified that the entered email is not already in use. It has a one to many relationship with the UserPositionsTable table, as each user, corresponding to one login record, can have multiple positions currently open, but a position cannot have more than one user own it.

The UserPositionTable table contains details of trades made by users. The primary key is a composite key made up of Email and StockSymbols. A composite key is used as a user cannot have more than one position open for the same company, but a company can still have multiple positions open by multiple users. The table also has both Email and StockSymbols as foreign keys, for the LoginTable and StockCodesTable tables respectively.

Finally the StockCodesTable table contains details for each of the companies stored in my program. Its primary key is the Symbol field, as stock tickers (or symbols) are unique to each company. The table is also in a one to many relationship with the UserPositionsTable table.

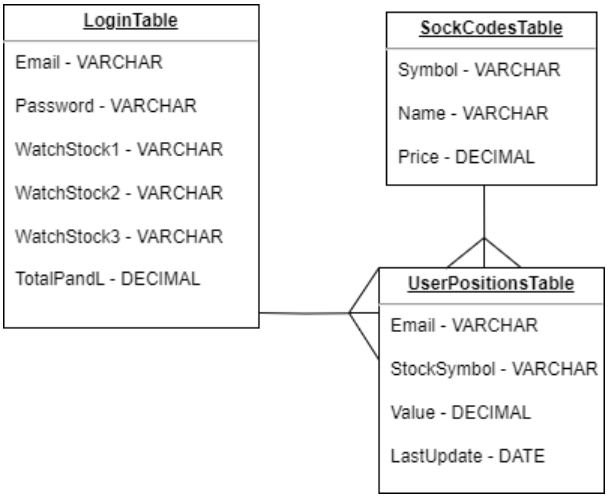

Figure 16: The entity relationship diagram for my database

3.3.2 Example Database Entries and Queries

Below is an example of entries into the UserPositionsTable table, StockSymbols and Email are stored as VARCHARs, LastUpdate as DATE, and Value as DECIMAL.

An example of a query used for this table is: "UPDATE UserPositionsTable SET Email = '0', StockSymbol = '1', Value = '2', LastUpdate = '3' WHERE Email = '0' AND StockSymbol = '1';",email, stockCode, newValue, sqlFormattedDate.

Email	StockSymbol	Value	LastUpdate
Filter	Filter	Filter	Filter
Email@gmail.com	AAPL	1231.6079192...	2024-02-28
Email@gmail.com	BABA	61.070503569...	2024-02-28
Email@gmail.com	RNST	15460838.674...	2024-02-28
Email@gmail.com	NFLX	904.50645483...	2024-02-28
Email@gmail.com	A	10235.013929...	2024-02-28

Figure 17: An example of the data stored in the UserPositionsTable table

The next figure shows examples of entries into the LoginTable table. Email, Password, and all the WatchStocks are stored as VARCHARs, and TotalPandL is a DECIMAL.

An example of queries used on this table are: "SELECT TotalPandL FROM LoginTable WHERE EMAIL = '0';", email; and "INSERT INTO LoginTable (Email, Password, WatchStock1, WatchStock2, WatchStock3, TotalPandL) VALUES('0', '1', '3','3', '3','2');", email, password, 0, " .

	Email	Password	WatchStock1	WatchStock2	WatchStock3	TotalPandL
	Filter	Filter	Filter	Filter	Filter	Filter
1	Email@gmail.c...	Passw0rd	RNST	A	NFLX	1236025.7317...

Figure 18: An example of the data stored in the LoginTable table

This final figure shows what entries into the StockSymbolsTable table look like. Symbol and name are both VARCHARs, and Price is DECIMAL.

A query used for this table is: "UPDATE StockCodesTable SET Price = '0' WHERE Symbol = '1';", price, companyCodes[x].

An example of queries used across the tables are:

This query deletes a user from the database.

"DELETE FROM LoginTable, UserPositionsTable

WHERE LoginTable.Email = UserPositionsTable.Email AND LoginTable.Email = '0';", emailToDelete

This query deletes all trades made on a certain stock, as well as the stock itself.

"DELETE FROM StockCodesTable, UserPositionsTable

WHERE StockCodesTable.Symbol = UserPositionsTable.StockSymbol AND StockCodesTable.Name = '0';", companyToDelete

This query scales all entries for value of a specific company in each table by the same amount, which can be useful for updating positions and stock price.

"UPDATE StockCodesTable, UserPositionsTable

SET StockCodesTable.Price = StockCodesTable.Price * '0' AND UserPositionsTable.Value = UserPositionsTable.Value * '0'

WHERE StockCodesTable.Symbol = UserPositionsTable.StockSymbol AND StockCodesTable.Name = '1';" relativePriceChange,companyName

Symbol	Name	Price
Filter	Filter	Filter
A	Agilent ...	132.550003
AAPL	Apple Inc.	182.630005
ABNB	Airbnb Inc.	152.059998
AMSWA	American ...	11.49
ATVI	Activision ...	-1
BA	The Boeing ...	201.399994
BABA	Alibaba Group...	77.68
BB	BlackBerry ...	2.71
BBW	Build-A-Bear ...	23.860001
BBWI	Bath & Body ...	47.41
BBY	Best Buy Co. ...	79.059998
BCS	Barclays PLC	8.56
BK	The Bank of ...	55.549999
BKCC	BlackRock ...	3.79
BLBD	Blue Bird ...	32.990002
BLK	BlackRock Inc.	800.539978
CAAS	China ...	3.25
CAKE	The Cheeseca...	35.48
CHGG	Chegg Inc.	8.84
CHNR	China Natural ...	1.32

Figure 19: An example of the data stored in the StockSymbolsTable table

3.3.3 Key Data Structures

The use of a appropriate data structures is a key part of solving problems effectively. Thus by planning where to use select data structures a more complete and efficient solution will be achieved.

Customised Double Linked List

In order to read from my database, it is very frequently sensible to use a linked list, as the number of elements being returned may not necessarily be known. Therefore, a list is sensible as you can add elements without having to pre-set a size, like in an array, and can access elements from any point in the structure, unlike a queue or stack.

To better understand the workings of the list, and to give me customisation of the functions which can be used on the list, I will write my own generic list.

An example of the list which would be returned from the database is shown below. The arrows represent references.

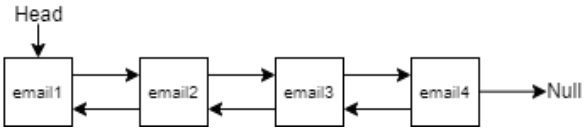

Figure 20: An example of the structure of the double linked list

2D and Single Dimension Arrays

Where the number of the data is already known, it is sensible to use an array. This is because arrays only hold a fixed size in memory, and allow for direct accessing of elements, whereas in a list you need to iterate from the head past every element in order to reach your desired element. So, for passing data on the stocks owned by a user, storing their notifications etc. it was suitable to use an array.

Matrices - MATLAB

MATLAB is extremely optimised when working with matrices, therefore, when working with large amounts of data it is sensible to store that data in matrices in order for it to be quickly processed. This is also the main reason for me choosing MATLAB. Therefore matrices are used for storing the vast majority of my data used in MATLAB, from all the coefficients and relative changes of all my found polynomials to day by day data used in my EMA calculator.

Tables - MATLAB

When reading data from CSV's into MATLAB, they are automatically stored as tables. Therefore being able to use tables and convert them to matrices where appropriate is an useful skill. The main difference between matrices and tables in MATLAB is that tables have headers for each column, and can store different data types in each column, unlike matrices. This makes them suitable for reading files as they may often contain mixed inputs, such as the files downloaded from YahooFinance which contain dates, doubles and strings all in one file.

Cell Arrays - MATLAB

Similarly to matrices in MATLAB, cell arrays must have pre-determined dimensions, however in them you are able to store whole matrices within each cell, and therefore make a collection of matrices. This will be useful in my code when splitting up stock data into even chunks to be used for training/testing, as one cell array could be used to store the entire collection of sub-datasets. An alternative could have been to use a 3D matrix, and make the third dimension the index for each dataset. However I decided against this as it would only give a slight improvement in performance for my scale of data, and the indexing is easier to confuse, therefore data along the incorrect dimensions may be accidentally used.

3.3.4 Pseudo Code

Throughout my code there are a number of complex algorithms which require some greater fore-thought before implementation. Below are some such algorithms. All of these algorithms are to be written in MATLAB.

Splitting data for validation:

Note that this function is in actuality split across two functions, but for demonstration they have been put as one.

PROCEDURE SPLIT_STOCK_DATA(StockData)

uniqueStocks - set to - number of unique elements in StockData.Symbols

for each element in uniqueStocks

  uniqueStockTables(x) = StockData(where StockData.Symbols = uniqueStocks(x))

end for

for each uniqueStockTables table

split each table containing unique stock into new tables of 30 day size - currentBlock

dataBlocks = dataBlocks + currentBlock

end for

generate matrix(number(dataBlocks), 1) of random numbers between 1 and 10 - randColumns

testingIndex = where(randColumns¿8) trainingIndex = where(randColumns¡=8)

testBlocks = dataBlocks(testingIndex, :) trainingBlocks = dataBlocks(trainingIndex, :)

Price prediction learning algorithm:

PROCEDURE STOCK_LEARNING()

data - set to - previously loaded stock data

training/testingData - set to- SPLIT_STOCK_DATA(data)

for each element in trainingData

normalise current trainingData

find relative price change from last day to 30 days in advance

fit polynomial to current trainingData

if (current polynomial within tolerance of previously found polynomial)

save mean of old and new found polynomial and save

save mean of price change after 30 days

calculate and save new standard deviation of price change after 30 days

else

create new row in savedPolynomials, adding the coefficients, relative change and 0 standard deviation

end for

clear any savedPolynomials that do not predict significant change or too volatile

TestPredictions(remaining savedPolynomials, testingData)

write passing polynomials to file

Price prediction testing algorithm:

PROCEDURE STOCK_TESTING(trainedPolynomials, testData, polynomialOrder)

for each set in testData

normalise current testingData

find relative price change from last day to 30 days in advance

fit polynomial to current testingData

if (current polynomial within tolerance of training polynomial, and relative change after 30 days is same within tolerance)

add to this fit that it is verified in testing

return all polynomials and whether or not they are verified

MACD calculator:

PROCEDURE MACD_Calculator(stockCode)

retrieve data for stockCode

EMA(stockData, currentDay)

weight = 2/(EMAperiod +1)

if (currentDay = EMAperiod)

  EMA - set to - stockData(currentDay)*weight

else

  EMA - set to - stockData(currentDay)*weight + EMA(stockData, currentDay + 1)

calculate 26 and 12 day EMA using above function

MACD - set to - 12 day EMA - 26 day EMA

for x = 1:numberEMAdays

  signalEMA(x) = EMA(MACD, x)

end for

signalDifference = MACD - signalEMA

if (signalDifference changes from positive to negative)

  return sell

else if (signalDifference changes from negative to positive)

  return buy

else

  return hold

4 Implementation

Global Coding Conventions:

Throughout my code there are some coding conventions which I have maintained throughout in order to maintain consistency and readability of the code. A number of these are outlined below:

- Camel case for naming variables e.g. stockName
- Sensible variable names in the context of the code
- Methods and functions starting with a capital letter e.g. InsertSell
- A comment at the top of each function and method describing its general functionality
- Further comments where appropriate on custom functions
- Try Catch blocks where needed, typically when reading from database
- Use of constant variables
- Minimal to no use of global variables

4.1 MATLAB - Back-end

4.1.1 Yahoo Finance API

This function was not written by me, however I implemented some small changes in order to return data more consistently. Original function can be found in reference 2, [2].

```
1  function data = getMarketDataViaYahoo(symbol, startdate, enddate, interval)
2      % Downloads market data from Yahoo Finance for a specified symbol and
3      % time range.
4      %
5      % INPUT:
6      % symbol    - is a ticker symbol i.e. 'AMD', 'BTC-USD'
7      % startdate - the date from which the market data will be requested
```

```
    % enddate   - the market data will be requested till this date
    % interval  - the market data will be returned in this intervals
    % supported intervals are '1d', '5d', '1wk', '1mo', '3mo'
    %
        % OUTPUT:
    % data - is a retrieved  dataset returned as a table
    %
    % Example:
    %   data = getMarketDataViaYahoo('AMD', '1-Jan-2018', datetime('today'), '5d');

    if(nargin() == 1)
        startdate = posixtime(datetime('1-Jan-2018'));
        enddate = posixtime(datetime()); % now
        interval = '1d';
    elseif (nargin() == 2)
        startdate = posixtime(datetime(startdate));
        enddate = posixtime(datetime()); % now
        interval = '1d';
    elseif (nargin() == 3)
        startdate = posixtime(datetime(startdate));
        enddate = posixtime(datetime(enddate));
        interval = '1d';
    elseif(nargin() == 4)
        startdate = posixtime(datetime(startdate));
        enddate = posixtime(datetime(enddate));
    else
        error('At least one parameter is required. Specify ticker symbol.');
        data = [];
        return;
    end

    %% Send a request for data
    % Construct an URL for the specific data
    uri = matlab.net.URI(['https://query1.finance.yahoo.com/v7/finance/download/',
    ↪ upper(symbol)],...
        'period1',  num2str(int64(startdate), '%.10g'),...
        'period2',  num2str(int64(enddate), '%.10g'),...
        'interval', interval,...
        'events',   'history',...
        'frequency', interval,...
        'guccounter', 1,...
        'includeAdjustedClose', 'true');

    options = weboptions('ContentType','table', 'UserAgent', 'Mozilla/5.0');
    try
        data = rmmissing(webread(uri.EncodedURI, options));

    catch ME
        %edited by me (makes the function much more consistent at returning
        %data)
        if contains(ME.message,'json')
            try
                data = parse_json(webread(uri.EncodedURI, options));
            catch
                data = [];
                warning(['Identifier: ', ME.identifier, 'Message: ', ME.message])
            end
        else
```

```
        %end of changes
            data = [];
            warning(['Identifier: ', ME.identifier, 'Message: ', ME.message])
        end
    end
end
```

4.1.2 Initial stock data loader

```
%function to load intial stock prices before running of full program

function FirstLoad(stockCodes)


StartDate = datetime('today') - 730;

RawData = [];
for x = 1:length(stockCodes)
    Newraw = getMarketDataViaYahoo(stockCodes{x}, string(StartDate),
↪   string(datetime('today')-30), '1d');
    if (isempty(Newraw))
        %pause(1);
        continue;
    end
    NewrawClean = removevars(Newraw,{'Open','Close', 'High','Low','Volume'});
    CurrentSymbol = stockCodes{x};
    SymbolsForMerge = repmat({CurrentSymbol}, size(NewrawClean, 1),1);
    SymbolsForMerge = table(SymbolsForMerge);
    ReadyData = horzcat(SymbolsForMerge,NewrawClean);
    %ReadyData = vertcat(subTables{x},ReadyData);
    RawData = vertcat(RawData,ReadyData);
end


folderPath = 'C:\Users\olesc\OneDrive\Documents\MATLAB\MATLABData';
homedir = pwd;
cd(folderPath);
flnm = 'RawStockData.csv';
fid = fopen(flnm,'wt+');
writetable(RawData, flnm)
fclose(fid);
cd(homedir);
```

4.1.3 Stock data updater

```
%function to update the data stored on the stocks

function StockDataLoader()

dataDir = 'C:\Users\olesc\OneDrive\Documents\MATLAB\MATLABData\';
dataFile = 'RawStockData';
PrevData = readtable([dataDir, dataFile]);

subTables = StockDataSplitter(PrevData);
uniqueStocks = unique(PrevData.SymbolsForMerge);
startDate = datetime(subTables{1}.Date(end));

rawData = [];

```

```
for x = 1:length(uniqueStocks)
    newRaw = getMarketDataViaYahoo(uniqueStocks{x}, string(startDate),
    ↪ string(datetime('today')), '1d');
    if (isempty(newRaw))
        %pause(1);
        continue;
    end
    newRawClean = removevars(newRaw,{'Open','Close', 'High','Low','Volume'});
    currentSymbol = uniqueStocks{x};
    symbolsForMerge = repmat({currentSymbol}, size(newRawClean, 1),1);
    symbolsForMerge = table(symbolsForMerge);
    readyData = horzcat(symbolsForMerge,newRawClean);
    readyData = vertcat(subTables{x},readyData);
    rawData = vertcat(rawData,readyData);
end

%writes the stock data to a csv
folderPath = 'C:\Users\olesc\OneDrive\Documents\MATLAB\MATLABData';
homedir = pwd;
cd(folderPath);
flnm = 'RawStockData.csv';
fid = fopen(flnm,'wt+');
writetable(rawData, flnm)
fclose(fid);
cd(homedir);
```

4.1.4 Formatting stock data

```
%function to split the raw data into data for each individual stock

function splitStocks = StockDataSplitter(StockData)

uniqueStocks = unique(StockData.SymbolsForMerge);
subTables = cell(length(uniqueStocks), 1);

for i = 1:length(uniqueStocks)
    stock = uniqueStocks{i};

    % Logical indexing to filter rows
    filteredRows = strcmp(StockData.SymbolsForMerge, stock);

    % Create a sub-table based on filtered rows
    subTables{i} = StockData(filteredRows, :);
end

splitStocks = subTables;
```

4.1.5 Finding relative price change of a stock

```
%finds the relative change in price after given time

function CHNG = FuturePriceChange(startDate, TimeL,stockCode)
endDate = startDate + caldays(TimeL);
endDate = string(endDate);
startDate = string(startDate);
stockCode = convertStringsToChars(stockCode);
pRaw = getMarketDataViaYahoo(stockCode, startDate, endDate, '1d');
%need to normalise
if isempty(pRaw) ~= true
```

```
    relativeChange = pRaw.AdjClose(end)/pRaw.AdjClose(1);
    relativeGrowth = relativeChange-1;
    CHNG = relativeGrowth;
else
    CHNG = 'no data';
end
```

4.1.6 Formatting dates for API

```
function date  = DateFormatter(InpDate)

InpDate = string(InpDate);
Year = extractAfter(InpDate,6);
Month = extractBefore(extractAfter(InpDate,3),3);
Day = extractBefore(InpDate,3);

if Month == "01"
    Month = "Jan";
elseif Month == "02"
    Month = "Feb";
elseif Month == "03"
    Month = "Mar";
elseif Month == "04"
    Month = "Apr";
elseif Month == "05"
    Month = "May";
elseif Month == "06"
    Month = "Jun";
elseif Month == "07"
    Month = "Jul";
elseif Month == "08"
    Month = "Aug";
elseif Month == "09"
    Month = "Sep";
elseif Month == "10"
    Month = "Oct";
elseif Month == "11"
    Month = "Nov";
elseif Month == "12"
    Month = "Dec";
end

date = datetime(Day + '-' + Month + '-' + Year);
```

4.1.7 Price prediction learning algorithm

```
%custom machine learning function which is used to predict
%buying or selling of stock
%(passed from c#) model ReRun
function StockfitML()

dataDir = 'C:\Users\olesc\OneDrive\Documents\MATLAB\MATLABData\';
dataFile = 'RawStockData';
StockData = readtable([dataDir, dataFile]);

%Splitting data into individual stocks
%Split stocks is an 1xn matrix with each cell having
% one table of data corresponding to a stock's price
splitStocks = StockDataSplitter(StockData);
```

```
14
15
16  %Time period for which prediction is made
17  timeExtrap = 30;
18  %misc variables
19  polyOrder = 3;
20  polyTol = 0.1; %changed
21
22  %number of extra columns required to contain extra data about fit
23  %1st is number of matching fits, 2nd is predicted relative change
24  %after x time
25  %3rd is volatility, 4th is sum of increase squared (needed for sd)
26  XtraCol = 4;
27
28  %number of comanies included in simulation
29  numIter = size(splitStocks,1);
30
31  %periodicity of fits in days (must be even as data taken bi-periodically)
32  numDays = 30;
33
34  %%Change here - random 20% of data taken for testing - but this
35  % 20% must be of 30 day blocks
36  %split all data into 30 day blocks (for each stock) and then
37  %take a random 20% of this for test
38  %Variable containing a table of a 30 day period of data in each cell
39  testBlocks = [];
40
41  %Loop which splits each different stock ticker into 30 day periods
42  % of data
43  for c = 1:numIter
44      %Determines current stock and how many blocks that ticker can
45      % be split into
46      currentStock = splitStocks{c,1};
47      numTestBlocks = fix(size(currentStock,1)/numDays)*2;
48      lastRow = 1;
49
50      %Iterated around the current ticker placing each block of 30
51      % days into a separate cell in TestBlocks
52      %The blocks of data overlap 15 days each, to avoid potential loss
53      % of interesting fits in periods across 30 day blocks
54      for i = 1:(numTestBlocks-1)
55          currentStockCell = {currentStock(lastRow:(lastRow+(numDays-1)),:)};
56          testBlocks = [testBlocks; currentStockCell];
57          lastRow = lastRow+(numDays/2);
58      end
59  end
60
61  randCols = randi(10,size(testBlocks,1),1);
62  testingIndex = find(randCols>8);
63  trainingIndex = find(randCols<=8);
64  testData = {testBlocks{testingIndex, :}};
65
66  %minimum relative change in price for a stock to be valid when predicting
67  minChange = 0.1;
68
69  %maximum volatility relative to growth
70  maxrVol = 0.5;
71
```

```
%final found fits with their fit, change, volatility and order of
%polynomial
successFits = [];

poly3Found = ones(1,XtraCol+polyOrder+1);
convergedFits = [];

%loop to adjust data block
for d = 1:size(trainingIndex,1)

    %defining new data which is going to be used for training
    newData = testBlocks{trainingIndex(d),:};
    currentStock = string(newData.symbolsForMerge(1));

    %defining boundary
    lastRowIndex = size(newData,1);
    endDate = newData.Date(lastRowIndex,1);
    endDate = DateFormatter(endDate);

    %normsalises all stock price
    normalisedNewPrice = newData.AdjClose ./ newData.AdjClose(1);
    newdays = 1:size(normalisedNewPrice,1);
    newdays = newdays(:);
    %polynomial fit of the new stock
    pNew = polyfit(newdays, normalisedNewPrice, polyOrder);
    %fNew = polyval(pNew, Newdays); %only neccessary if want to see fit

    %tolerance of fit for comparing to previously identified trends
    %mean is formatted to dimensions of previously found fits matrix
    %and has absolute values taken
    %plot(Newdays, fNew);
    %hold on
    formattedMean = abs(pNew(ones(size(poly3Found, 1),1),:));
    tol = polyTol .* formattedMean;

    %new prediction for price change, using new fit
    nChangeP = FuturePriceChange(endDate, timeExtrap, currentStock);

    %error check if there is extrapolated data found
    if nChangeP == 'no data'
        continue;
    end

    %checks if there is a fit that has been found that matches
    %the identified trend
    difference = poly3Found(:,1:(size(pNew, 2))) - pNew(ones(1,
    ↪ size(poly3Found,1)),:);
    withinTol = abs(difference) <= tol;
    isPresent = sum(withinTol, 2);

    %sees if a similar fit has already been logged
    replaceIndex = find(isPresent == length(pNew));

    if size(replaceIndex,1) >=1
        for c = 1:size(replaceIndex, 1)

            %identifies correct index to replace
            index = replaceIndex(c,1);
```

```
129
130             %edits logged mean fit to incorporate newly found regression
131             matchedFits = poly3Found(index, 1:(size(pNew, 2)));
132             replacement = rdivide(plus(matchedFits, pNew), 2);
133             poly3Found(index, 1:(size(pNew, 2))) = replacement;
134
135             %new number of this fit found
136             newFound = plus(poly3Found(index, 5),1);
137             nMean = rdivide(plus(nChangeP, poly3Found(index, 6).*(newFound-1)),
    ↪  newFound);
138
139             %new prediction for volatility (one standard deviation)
140             nSD = NewSD(poly3Found(index,8), nMean, nChangeP, newFound);
141
142             %increases number of this fit found by 1
143             poly3Found(index, 5) = newFound;
144
145             %adds new mean, s.d., and sum of changes squared
146             poly3Found(index, 6) = nMean;
147             poly3Found(index, 7) = nSD;
148             poly3Found(index,8) = plus(poly3Found(index,8), (nChangeP.^2));
149         end
150         if size(replaceIndex(:,1)) > 1
151             convergedFits = [convergedFits,-1,replaceIndex];
152         end
153      else
154         %%nChangeP = FuturePriceChange(startDate, timeExtrap,
    ↪  StockSymbols.Symbol{x});
155         nSD = 0;
156         nSumsq = nChangeP.^2;
157         poly3Found = [poly3Found; pNew, 1 , nChangeP, nSD, nSumsq];
158     end
159 end
160
161
162 %Tests any found fits and writes the successful fits to a file
163 minChange = 0.05;
164 maxrVol = 0.5;
165 polyOrder = 3;
166 %tested successful fits should be used to write to a database of prediction
167 successFits = CleanMatches(poly3Found, minChange, maxrVol, polyOrder);
168
169 %tests the successful fits
170 postTestSuc = StockFitTesting(successFits, testData, polyOrder);
171 postTestSuc = sortrows(postTestSuc,size(postTestSuc,2));
172 %only keeps actual polynomial data, and change + s.d.
173 postTestSuc = postTestSuc(2:end,1:6);
174
175 homedir = cd;
176 flnm = 'Poly3Found.csv';
177 fid = fopen(flnm,'wt+');
178 writematrix(postTestSuc, flnm)
179 fclose(fid);
180 cd(homedir);
```

4.1.8 Cleaning found predictor polynomials

```
%function which clears and found polynomial fits from my algorithm that are
%not consistent enough, e.g. do not change enough from original price to be
%worth buying/selling or are too volatile in their predicted price

function SucFits = CleanMatches(potentialFits, minChange, maxrVol, polyOrder)
fitChangeVol = [];
if (isempty(potentialFits) == false)
    %stores all the found repeated fits
    requiredCols = [potentialFits(:,1:(polyOrder+1))];
    requiredCols = [requiredCols, potentialFits(:,6:7)];
    fitChangeVol = requiredCols;
    sucIndex1 = abs(fitChangeVol(:,5))>minChange;
    sucIndex2 = abs(fitChangeVol(:,5)*maxrVol)>fitChangeVol(:,6);
    sucIndex = and(sucIndex1, sucIndex2);
    fitChangeVol(sucIndex,:) = [];
end

SucFits = fitChangeVol;
```

4.1.9 Testing found predictor polynomials

```
%% Testing fits and predictions
%returns martix with the found fits and with one extra column indicating
%accuracy of the fit when predicting, if 0 is in accuracy column then
%the fit was not found in the test data

function fits = StockFitTesting(sucFits, testData, polyOrder)

newCol = zeros(size(sucFits,1),2);
sucFits = [sucFits, newCol];
timeExtrap = 30;
tol = 0.1;
%counter to see if there are issues with returning stock codes
counter = 0;


for x = 1:size(testData,1)

    currentDataCell = testData(1,x);
    currentData = currentDataCell{1,1};
    currentStock = currentData.symbolsForMerge(1);
    currentStock = currentStock{1,1};

    %normalises stock price
    normalisedPrice = currentData.AdjClose ./ currentData.AdjClose(1);
    newdays = 1:size(normalisedPrice,1);
    newdays = newdays(:);

    %polynomial fit of the new stock
    pNew = polyfit(newdays, normalisedPrice, polyOrder);

    %gets the price change to use for comparison
    testEnd = datetime(currentData.Date(end));
    nChangeP = FuturePriceChange(testEnd, timeExtrap, currentStock);

    %error check if there is extrapolated data found
    if nChangeP == 'no data'
```

```
      continue;
    end

    %checks if there is a fit that has been found that matches
    %the identified trend
    difference = sucFits(:,1:(size(pNew, 2))) - pNew(ones(1, size(sucFits,1)),:);
    withinTol = abs(difference) <= tol;
    isPresent = sum(withinTol, 2);

    %sees if a similar fit has been logged
    replaceIndex = find(isPresent == size(pNew,2));
    if size(replaceIndex,1) >=1
        for m = 1:size(replaceIndex, 1)
           %identifies correct index to compare
           index = replaceIndex(m,1);
           %finds the difference between newly found change and old
           %mean change for this particular fit
           changeSimi = sucFits(index,6)-nChangeP;
           %finds Z-Value for this fit
           ZVal = abs(changeSimi/sucFits(index,6));
           if(sucFits(index,end) == 0)
               %logs new Z value and adds that a test fit was used
               sucFits(index, end) = ZVal;
               sucFits(index,size(sucFits,2)-1) = sucFits(index,size(sucFits,2)-1)+1;
           else
               %updates mean Z val and adds that another test fit
               %was used
               sucFits(index, end) = plus(sucFits(index,end),ZVal)/2;
               sucFits(index,size(sucFits,2)-1) = sucFits(index,size(sucFits,2)-1)+1;
           end
        end
    end
    %drops out of program if data is consistently returning erros
    if counter>100
        fits = sucFits;
        return;
    end
end

fits = sucFits;
```

4.1.10 Using found predictor polynomials to predict buy/sell of stock

```
%function which uses data from my algorithm in order to give
%a buy or sell prediction

function price = PricePredictor(stockCode)

dataDir = 'C:\Users\olesc\OneDrive\Documents\MATLAB\';
dataFile = 'Poly3Found';
poly3Data = readtable([dataDir, dataFile]);

poly3Coefs = poly3Data{:, 1:4};
poly3ChangeSD = poly3Data{:, 5:6};
chngToSD = poly3ChangeSD(:,1)./poly3ChangeSD(:,2);


startDate = datetime('today') - days(30);
```

```
endDate = datetime('today');

polyOrder = 3;
polyTol = 0.1;
maxRangeOfChange = 0.2;

%Contains fit of each order of polynomial, 3,4,5 respectively in each cell
fits = cell(1,3);

%Getting data from Yahoo Finance and normalising the price
pRaw = getMarketDataViaYahoo(stockCode, string(startDate), string(endDate), '1d');

if isempty(pRaw) ~= true
    normalisedPrice = pRaw.AdjClose./pRaw.AdjClose(1);
else
    price = 'no data available for this stock';
end

Newdays = 1:size(normalisedPrice,1);
Newdays = Newdays(:);
for x = 1:3
    pNew = {polyfit(Newdays, normalisedPrice, polyOrder)};
    fits{1,x} = pNew;
    polyOrder = polyOrder + 1;
end

%tolerance of fit for comparing to previously identified trends
%mean is formatted to dimensions of previously found fits matrix
new3rdCoefs = fits{1,1};
%This weird line needs to be added as each cell in Fits is itself stored as
%a cell
new3rdCoefs = new3rdCoefs{1};
formattedMean = abs(new3rdCoefs(ones(size(poly3Coefs, 1),1),:));
tol = polyTol .* formattedMean;

%checks if there is a fit that has been found that match
%the identified trend
difference = poly3Coefs(:,1:(size(pNew, 2))) - new3rdCoefs(ones(1,
↪  size(poly3Coefs,1)),:);
withinTol = abs(difference) <= tol;
isPresent = sum(withinTol, 2);

%sees if a similar fit has already been logged
replaceIndex = find(isPresent == length(pNew));

potentialPredicts = poly3ChangeSD(replaceIndex,:);
potentialRatios = chngToSD(replaceIndex,:);

removeRatioIndex = find(abs(potentialRatios) == Inf);
removeRatioIndex = [removeRatioIndex; find(abs(potentialRatios) < 0.5)];

keepIndex = true(size(potentialRatios, 1), 1);
keepIndex(removeRatioIndex) = false;


potentialPredicts = potentialPredicts(keepIndex,:);
potentialPredicts = sortrows(potentialPredicts,1);
if size(potentialPredicts, 1) ~= 0
```

```
    if size(potentialPredicts, 1) == 1
        price = sprintf("%.2f", round(pRaw{1,"AdjClose"}*potentialPredicts(1,1),2));
        if price>0
            price = "Buy";
        elseif price == 0
            price = "Hold";
        else
            price = "Sell";
        end
    elseif (potentialPredicts(end,1)-potentialPredicts(1,1))>maxRangeOfChange
        price = "No consistent predictions from algorithm";
    else
        price = mean(potentialPredicts,1);
    end
else
    price = "No consistent predictions from algorithm";
end
```

4.1.11 MACD calculator

```
%function which uses moving average convergence/divergence (MACD) in order
%to give entry or exit recommendation for stock

function enterExit = MACDCalculator(stockCode)

period26 = 26;
period12 = 12;
signalPeriod = 9;

numEMAdays = 5;

EMA26 = [];
EMA12 = [];
signalEMA = [];

%get data up to period26 + 22 back, to give 14 data points of EMA,
%enough for 5 days of signal EMA to be calculated, and then the most
%recent 5 days of MACD and signal compared to check for entry/exit
%+30 as 5 days of signal, 9 days needed for each signal data point
%and excess 16 to account for some days not having data i.e. weekend

%getting current data for the stock
startDate = datetime('today') - days(period26+(numEMAdays+signalPeriod+20));
endDate = datetime('today');
stockDataAll= getMarketDataViaYahoo(stockCode, string(startDate), string(endDate),
↪  '1d');
stockDataAll = stockDataAll.AdjClose;
stockDataAll = flip(stockDataAll,1);

for x = 1:(numEMAdays + signalPeriod)
    StockData26 = stockDataAll(x:x+period26-2,:);
    EMA26(x) = EMACalculator(StockData26,1);

    StockData12 = stockDataAll(x:x+period12-1,:);
    EMA12(x) = EMACalculator(StockData12,1);
end

MACD = EMA12(:) - EMA26(:);
```

```

for x = 1:numEMAdays
   signalEMA(x) = EMACalculator(MACD, x);
   signalEMA = signalEMA(:);
end

MACDSignalDif = MACD(1:numEMAdays,:) - signalEMA;
%Is there change in market, and if so buy or sell
%0 in first column is no change, 1 is change
%0 in second is no buy or sell, -1 is sell, 1 is buy
changeInMarket = [0,0];

for x = 1:numEMAdays-1
    sumofDif = abs(MACDSignalDif(x,1)+MACDSignalDif(x+1,1));
    MagnitudeDif = abs(MACDSignalDif(x,1)) + abs(MACDSignalDif(x+1,1));
    if sumofDif<MagnitudeDif
        changeInMarket(1,1) = 1;
        if MACDSignalDif(x,1) < 0
           changeInMarket(1,2) = 1;
        else
            changeInMarket(1,2) = -1;
        end
    end
end

if changeInMarket(1,1) == 1
    if changeInMarket(1,2) == -1
        enterExit = "Sell";
    elseif changeInMarket(1,2) == 1
        enterExit = "Buy";
    else
        enterExit = "Hold";
    end
else
    enterExit = "Hold";
end
```

4.1.12 EMA calculator

```
%function which calculates the exponential moving average (EMA) of a
%stock's price

function EMA = EMACalculator(stockData, currentDay)

%the weighted multiplier used in each period
WMultiplier = 2/(size(stockData,1) + 1);

if currentDay == size(stockData,1)
   EMA = stockData(currentDay) * WMultiplier;
else
   EMA = stockData(currentDay)* WMultiplier + (1-WMultiplier)*EMACalculator(stockData,
↪  currentDay+1);
end
```

4.1.13 Hot stock finder

```
%function which returns the three most bought stocks in last day,
%the if statements are necessary as there is no data on weekends

```

```
function hotStocks = HotStocksFinder(stockCodes)

startDate = datetime('today') - 1;
pRaw = [];

for x = 1:length(stockCodes)
    stockData = getMarketDataViaYahoo(stockCodes{x}, startDate, datetime('today'), '1d');
    pRaw = [pRaw; stockData];
    if isempty(pRaw) == true
        startDate = startDate - 1;
        stockData = getMarketDataViaYahoo(stockCodes{x}, startDate, datetime('today'), '1d');
        pRaw = [pRaw; stockData];
        if isempty(pRaw) == true
            startDate = startDate - 1;
            stockData = getMarketDataViaYahoo(stockCodes{x}, startDate, datetime('today'), '1d');
            pRaw = [pRaw; stockData];
        end
    end
end

[~, sortIndex] = sort(pRaw.Volume, 'descend');
sortedStockCodes = stockCodes(sortIndex);
% return 3 highest volume stocks
hotStocks = sortedStockCodes(1:3);
```

4.1.14 Stock price returner

```
%function which returns most recent value of a share of a specified stock

function price = PriceReturner(stockCode)

startDate = datetime('today')-4;
stockData = getMarketDataViaYahoo(stockCode, startDate, datetime('today'), '1d');
if isempty(stockData)
    price = -1;
else
    price = stockData.AdjClose(end);
end
```

4.1.15 Stock graph data returner

```
%function which returns the last 30 days of data to be used for graph
%plotting

function stockData = ReturnPlotData(stockCode)

startDate = datetime('today') - 31;

rawStockData = getMarketDataViaYahoo(stockCode, startDate, datetime('today'), '1d');
try
   newRawClean = removevars(rawStockData,{'Open','Close', 'High','Low','Volume', 'Date'});
   newRawClean = table2array(newRawClean);
catch
    newRawClean = {};
end
```

```
15
16
17 stockData = newRawClean;
```

4.2 C#

UI definitions - all in one class

4.2.1 Welcome page XAML

```
1  <Window x:Class="MATLABintegrationTest.MainWindow"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6          xmlns:ScottPlot="clr-namespace:ScottPlot;assembly=ScottPlot.WPF"
7          xmlns:local="clr-namespace:MATLABintegrationTest"
8          mc:Ignorable="d"
9          Title="Window" Height="732" Width="1297" Background="Black"
10         WindowState ="Maximized"
11         WindowStyle = "None">
12     <Window.Resources>
13         <Style x:Key="CenteredPasswordBoxStyle" TargetType="PasswordBox">
14             <Setter Property="HorizontalContentAlignment" Value="Center" />
15             <Setter Property="VerticalContentAlignment" Value="Center" />
16             <Setter Property="Template">
17                 <Setter.Value>
18                     <ControlTemplate TargetType="PasswordBox">
19                         <Border Background="{TemplateBinding Background}"
20                             BorderBrush="{TemplateBinding BorderBrush}"
21                             BorderThickness="{TemplateBinding BorderThickness}">
22                             <ScrollViewer x:Name="PART_ContentHost" />
23                         </Border>
24                     </ControlTemplate>
25                 </Setter.Value>
26             </Setter>
27         </Style>
28     </Window.Resources>
29     <Grid Margin="0,0,0,0">
30
31
32         <Grid Name="WelcomeScreen" Visibility="Visible">
33             <Grid.RowDefinitions>
34                 <RowDefinition Height="*"/>
35                 <RowDefinition Height="*"/>
36                 <RowDefinition Height="*"/>
37                 <RowDefinition Height="*"/>
38                 <RowDefinition Height="*"/>
39             </Grid.RowDefinitions>
40             <Grid.ColumnDefinitions>
41                 <ColumnDefinition Width="*"/>
42                 <ColumnDefinition Width="*"/>
43                 <ColumnDefinition Width="*"/>
44                 <ColumnDefinition Width="*"/>
45                 <ColumnDefinition Width="*"/>
46             </Grid.ColumnDefinitions>
47
48             <Border Grid.Column ="1" Grid.Row="0"  Grid.ColumnSpan="3"
↪  Grid.RowSpan="1">
```

```
                <TextBlock
                    Background="Black"  Foreground="WhiteSmoke"
                    FontSize="40" HorizontalAlignment="Center"
↪ VerticalAlignment="Center">
                    Welcome to OC Stock Management
                </TextBlock>
            </Border>

            <Border Grid.RowSpan="1" Grid.Row="3" Grid.Column="2" Grid.ColumnSpan="1">
                <Button Content="LOG-IN"  Click="Button_Click_Login"
                     Background="WhiteSmoke" FontSize ="20" Width="170" Height="60"/>
            </Border>

            <Button   Click="Button_Click_CreateAccountScreen"
            Width="170" Height ="60" Background="WhiteSmoke" FontSize ="20"
↪ Grid.Row="5" Grid.Column="2"
               HorizontalAlignment="Center">Create Account</Button>

            <PasswordBox Name="PasswordLogin" Width="170" Height="50"
↪ Background="WhiteSmoke"
                         PasswordChanged="passwordBoxLogin_PasswordChanged"
↪ IsEnabled="True" Style="{StaticResource CenteredPasswordBoxStyle}"
                         Grid.Column="2" Grid.Row="2"
↪ VerticalContentAlignment="Center"
             HorizontalContentAlignment="Center"/>

            <Border Grid.RowSpan="1" Grid.Row="2" Grid.Column="2" Grid.ColumnSpan="1">
                <TextBlock Name="PassWordTextLogin" Text="Enter Password"
↪ Foreground="Gray"
                       IsEnabled="False" Background="Transparent" Panel.ZIndex="1"
↪ HorizontalAlignment="Center"
                           VerticalAlignment="Center"/>
            </Border>

            <TextBox Grid.Row ="1" Grid.Column="2" HorizontalAlignment="Center"
↪ Height="50" TextWrapping="Wrap"
                     Text="Enter Email" Width="170" VerticalAlignment="Bottom"
↪ VerticalContentAlignment="Center"
             HorizontalContentAlignment="Center" Name="WelcomeEmailBox"/>

            <Button
                Name="CloseProgramButtonWelcomeScreen" Content="X" Grid.Column="4"
↪ Grid.Row="0" Height="25"
                Width="40" HorizontalAlignment="Right" VerticalAlignment="Top"
↪ Click="CloseProgramButtonWelcomeScreen_Click"/>

            <TextBox Name="InvalidLogin" Text="Invalid Login, Try Again"
↪ Grid.Column="3" Grid.Row="3"
                     Foreground="WhiteSmoke" Background="Transparent"
↪ BorderThickness="0"
                     VerticalAlignment="Center" Visibility="Collapsed"/>

        </Grid>
```

4.2.2 Account page XAML

```
        <Grid Name="AccountCreation" Visibility="Collapsed">
            <Grid.RowDefinitions>
```

```
3                 <RowDefinition Height="*"/>
4                 <RowDefinition Height="*"/>
5                 <RowDefinition Height="*"/>
6                 <RowDefinition Height="*"/>
7                 <RowDefinition Height="*"/>
8             </Grid.RowDefinitions>
9             <Grid.ColumnDefinitions>
10                <ColumnDefinition Width="*"/>
11                <ColumnDefinition Width="*"/>
12                <ColumnDefinition Width="*"/>
13                <ColumnDefinition Width="*"/>
14                <ColumnDefinition Width="*"/>
15            </Grid.ColumnDefinitions>
16
17            <TextBlock Name="CreatAccountTitle" Text="Create Account"
    ↪ HorizontalAlignment="Center" VerticalAlignment="Bottom"
18             Grid.Column ="2" Grid.Row = "0" Foreground="WhiteSmoke" FontSize="30"/>
19
20            <Button Width ="150" Height="55" Content="Create New Account"
21                Click="Button_Click_CreateAccount" Grid.Column="2" Grid.Row="3"/>
22
23            <Button Width ="150" Height="55" Content="Back to login"
24                Click="Button_Click_BackToLogin" Grid.Column="2" Grid.Row="4"
    ↪ VerticalAlignment="Top"/>
25
26
27            <TextBox Name="EmailCreateAccountBox" Text="Enter Email"
    ↪ VerticalAlignment="Top"
28             Grid.Column="2" Grid.Row="2" Height="40" Width="150"
29            Background="WhiteSmoke" Foreground="Black" FontSize="10"
    ↪ VerticalContentAlignment="Center"
30             HorizontalContentAlignment="Center" />
31
32            <PasswordBox Name ="PasswordCreateAccountBox" VerticalAlignment="Center"
33                PasswordChanged="passwordBox_PasswordChanged" Panel.ZIndex="0"
34             Grid.Column="2" Grid.Row="2" Height="40" Width="150"
35                IsEnabled="True" FontSize="20" Style="{StaticResource
    ↪ CenteredPasswordBoxStyle}"/>
36
37            <TextBlock Name="PassWordText" Text="Enter Password" Foreground="Gray"
38                 HorizontalAlignment="Center"  Grid.Column="2" Grid.Row="2"
    ↪ VerticalAlignment="Center"
39                 IsEnabled="False" Background="Transparent" Panel.ZIndex="1"/>
40
41            <Button
42                Name="CloseProgramButtonCreateAccountScreen" Content="X"
    ↪ Grid.Column="4" Grid.Row="0" Height="25"
43                Width="40" HorizontalAlignment="Right" VerticalAlignment="Top"
    ↪ Click="CloseProgramButtonWelcomeScreen_Click"/>
44
45            <TextBox Name="EmailUsedMessage" Text="This email is already in use"
    ↪ Grid.Column="3" Grid.Row="3"
46                     Foreground="WhiteSmoke" Background="Transparent"
    ↪ BorderThickness="0"
47                     VerticalAlignment="Center" Visibility="Collapsed"/>
48
49        </Grid>
```

4.2.3 Home page XAML

```
        <Grid Name="HomeScreen" Visibility="Collapsed">

            <Grid.RowDefinitions>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
            </Grid.ColumnDefinitions>

            <ScottPlot:WpfPlot Name="TLPlot" Grid.Row="1" Grid.Column="1" Grid.RowSpan="2"
            Height="200" Width="210" VerticalAlignment="Center" HorizontalAlignment="Left"/>

            <ScottPlot:WpfPlot Name="TMPlot" Grid.Row="1" Grid.Column="2" Grid.RowSpan="2"
            Height="200" Width="210" VerticalAlignment="Center" HorizontalAlignment="Left"/>

            <ScottPlot:WpfPlot Name="TRPlot" Grid.Row="1" Grid.Column="3" Grid.RowSpan="2"
            Height="200" Width="210" VerticalAlignment="Center" HorizontalAlignment="Left"/>

            <ScottPlot:WpfPlot Name="BLPlot" Grid.Row="3" Grid.Column="1" Grid.RowSpan="2"
            Height="200" Width="210" VerticalAlignment="Center" HorizontalAlignment="Left"/>

            <ScottPlot:WpfPlot Name="BMPlot" Grid.Row="3" Grid.Column="2" Grid.RowSpan="2"
            Height="200" Width="210" VerticalAlignment="Center" HorizontalAlignment="Left"/>

            <ScottPlot:WpfPlot Name="BRPlot" Grid.Row="3" Grid.Column="3" Grid.RowSpan="2"
            Height="200" Width="210" VerticalAlignment="Center" HorizontalAlignment="Left"/>

            <Button Content="Notifications"
                Click="Button_Click_Notifications" Grid.Column="1" Grid.Row="0" Height="50" Width="120"
                VerticalAlignment="Center" HorizontalAlignment="Left"/>

            <Grid Name="NotificationScreen" Visibility="Collapsed" Background="SlateGray" Grid.Row="0"
                  Grid.RowSpan="2" Grid.Column="1" Panel.ZIndex="10">
                <Grid.RowDefinitions>
                    <RowDefinition Height="*"/>
```

```
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>

                <Button Name="CloseNotifications" Content="X" Grid.Row="0" Grid.Column="4" Width="25" Height="20"
                        HorizontalAlignment="Right" VerticalAlignment="Top" Click="CloseNotifications_Click"/>

                <TextBlock Name ="NotificationTitle" Grid.Row="0" Grid.Column="1" Grid.ColumnSpan="3"
                           Foreground="WhiteSmoke" Text = "Notifications" TextAlignment="Center" FontSize="25"
                           IsEnabled="False"/>

                <Button Name ="FirstNotification" Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="3"
                           Foreground="Black" Content = "NOTI 1" FontSize="13"
                           Click="Noti1Click" Background="Transparent" IsEnabled="True"/>

                <Button Name ="SecondNotification" Grid.Row="2" Grid.Column="1" Grid.ColumnSpan="3"
                           Foreground="Black" Content = "NOTI 2" FontSize="13"
                           IsEnabled="True" Click="Noti2Click" Background="Transparent"/>

                <Button Name ="ThirdNotification" Grid.Row="3" Grid.Column="1" Grid.ColumnSpan="3"
                           Foreground="Black" Content = "NOTI 3" FontSize="13"
                           IsEnabled="True" Click="Noti3Click" Background="Transparent"/>

                <Button Name ="FourthNotification" Grid.Row="4" Grid.Column="1" Grid.ColumnSpan="3"
                           Foreground="Black" Content = "NOTI 4" FontSize="13"
                           IsEnabled="True" Click="Noti4Click" Background="Transparent"/>
            </Grid>

            <Button Name = "TopLStock" Content="STOCK NAME TL" Background="Transparent"
                    Foreground="WhiteSmoke" HorizontalAlignment="Left" VerticalAlignment="Bottom"
                    BorderThickness="0" FontSize="16" Grid.Column="1" Grid.Row="2" Click="TopLStock_Click"
                    HorizontalContentAlignment="Left"/>

```

```
86              <Button Name = "TopMStock" Content="STOCK NAME TM"
    ↪ Background="Transparent"
87                      Foreground="WhiteSmoke" HorizontalAlignment="Left"
    ↪ VerticalAlignment="Bottom"
88                      BorderThickness="0" FontSize="16" Grid.Column="2" Grid.Row="2"
    ↪ Click="TopMStock_Click"
89                      HorizontalContentAlignment="Left"/>
90
91              <Button Name = "TopRStock" Content="STOCK NAME TR"
    ↪ Background="Transparent"
92                      Foreground="WhiteSmoke" HorizontalAlignment="Left"
    ↪ VerticalAlignment="Bottom"
93                      BorderThickness="0" FontSize="16" Grid.Column="3" Grid.Row="2"
    ↪ Click="TopRStock_Click"
94                      HorizontalContentAlignment="Left"/>
95
96              <Button Name = "BottomLStock" Content="STOCK NAME BL"
    ↪ Background="Transparent"
97                      Foreground="WhiteSmoke" HorizontalAlignment="Left"
    ↪ VerticalAlignment="Bottom"
98                      BorderThickness="0" FontSize="16" Grid.Column="1" Grid.Row="4"
    ↪ Click="BottomLStock_Click"
99                      HorizontalContentAlignment="Left"/>
100
101             <Button Name = "BottomMStock" Content="STOCK NAME BM"
    ↪ Background="Transparent"
102                     Foreground="WhiteSmoke" HorizontalAlignment="Left"
    ↪ VerticalAlignment="Bottom"
103                     BorderThickness="0" FontSize="16" Grid.Column="2" Grid.Row="4"
    ↪ Click="BottomMStock_Click"
104                     HorizontalContentAlignment="Left"/>
105
106             <Button Name = "BottomRStock" Content="STOCK NAME BR"
    ↪ Background="Transparent"
107                     Foreground="WhiteSmoke" HorizontalAlignment="Left"
    ↪ VerticalAlignment="Bottom"
108                     BorderThickness="0" FontSize="16" Grid.Column="3" Grid.Row="4"
    ↪ Click="BottomRStock_Click"
109                     HorizontalContentAlignment="Left"/>
110
111             <TextBlock Name = "TopStocksTitle" Text ="Yesterday's Hot Stocks:"
    ↪ Grid.Column="1" Grid.Row="1"
112                        FontSize="30" Foreground="WhiteSmoke" IsEnabled="False"
    ↪ Grid.ColumnSpan="2"
113                        Panel.ZIndex="2"/>
114
115             <TextBlock Name = "BottomStocksTitle" Text ="Your Top Positions:"
    ↪ Grid.Column="1" Grid.Row="3"
116                        FontSize="30" Foreground="WhiteSmoke" IsEnabled="False"
    ↪ Grid.ColumnSpan="2"
117                        Panel.ZIndex="2"/>
118
119             <TextBox Name="SearchBar" TextAlignment="Center" Background="WhiteSmoke"
    ↪ Foreground="Black"
120                      Grid.Column="3" Grid.Row="0" Height="50" Width="120"
    ↪ HorizontalAlignment="Right" VerticalAlignment="Center"
121                      TextChanged="SearchBarTextChangedEventHandler"/>
122
```

```
123             <Button Name="SearchedStock" Grid.Row="0" Grid.Column="3"
    ↪ VerticalAlignment="Bottom" HorizontalAlignment="Right"
124                     Height="50" Width="120" Background="WhiteSmoke" Foreground="Black"
    ↪ Visibility="Collapsed"
125                     Click="SearchedStock_Click"/>
126 
127             <TextBlock Name="SearchBarText" Text="Search 0~" Foreground="Black"
128                  HorizontalAlignment="Right"  Grid.Column="3" Grid.Row="0"
    ↪ VerticalAlignment="Center"
129                  IsEnabled="False" Background="Transparent" Panel.ZIndex="1"
130                  TextAlignment="Center" FontSize="18" Visibility="Visible"/>
131 
132             <Button Name="SearchButton" Content="SEARCH" FontSize="15" Grid.Column="4"
133                     Grid.Row="0" HorizontalAlignment="Left" Width="60" Height="30"
    ↪ Background="WhiteSmoke"
134                     VerticalAlignment="Center" Visibility="Collapsed"
    ↪ Click="SearchButton_Click"/>
135 
136             <Button
137                 Name="CloseProgramButtonHomeScreen" Content="X" Grid.Column="4"
    ↪ Grid.Row="0" Height="25"
138                 Width="40" HorizontalAlignment="Right" VerticalAlignment="Top"
    ↪ Click="CloseProgramButtonHomeScreen_Click"/>
139 
140             <Button Name="AccountButton" Content="Account" Foreground="Black"
141                  HorizontalAlignment="Right"  Grid.Column="4" Grid.Row="0"
    ↪ VerticalAlignment="Center"
142                  Background="WhiteSmoke" Panel.ZIndex="1" Width="120" Height="50"
143                  FontSize="18" Visibility="Visible"
    ↪ Click="ButtonToAccountScreen_Click"/>
144 
145         </Grid>
```

4.2.4 Stock page XAML

```
1         <Grid Name="StockScreen" Visibility="Collapsed">
2             <Grid.RowDefinitions>
3                 <RowDefinition/>
4                 <RowDefinition/>
5                 <RowDefinition Height="Auto"/>
6                 <RowDefinition Height="2*"/>
7                 <RowDefinition Height="2*"/>
8                 <RowDefinition Height="2*"/>
9             </Grid.RowDefinitions>
10            <Grid.ColumnDefinitions>
11                <ColumnDefinition Width="0.6667*"/>
12                <ColumnDefinition Width="0.6667*"/>
13                <ColumnDefinition Width="0.6667*"/>
14                <ColumnDefinition Width="*"/>
15                <ColumnDefinition Width="*"/>
16                <ColumnDefinition Width="*"/>
17            </Grid.ColumnDefinitions>
18 
19            <ScottPlot:WpfPlot Name="StockScreenPlot" Grid.Row="2" Grid.Column="1"
    ↪ Grid.RowSpan="3"
20            Grid.ColumnSpan ="3" VerticalAlignment="Center" HorizontalAlignment="Left"
21            Height="370" Width="500"/>
22 
```

```
23              <Button
24                  Name="CloseProgramButtonStockScreen" Content="X" Grid.Column="5"
    ↪ Grid.Row="0"
25                  Width="40" Height="25" HorizontalAlignment="Right"
    ↪ VerticalAlignment="Top"
26                  Click="CloseProgramButtonStockScreen_Click"/>
27
28              <Button
29                  Name="BackButtonToHomeScreen" Content="Back" Grid.Column="0"
    ↪ Grid.Row="0"
30                  Width="40" Height ="30" HorizontalAlignment="Left"
    ↪ VerticalAlignment="Top"
31                  Click="BackButtonToHomeScreen_Click" />
32
33              <TextBox Name = "StockScreenTitle" Background="Transparent"
    ↪ Foreground="WhiteSmoke"
34                       Grid.Column="1" Grid.Row="1" Grid.ColumnSpan="2"
35                       HorizontalAlignment="Left" VerticalAlignment="Top" FontSize="30"
    ↪ Text="TEST LONGGGGG"
36                       BorderThickness="0" IsReadOnly ="True"/>
37
38              <Button Name ="BuyButton" Content="Buy" Background="Lime"
    ↪ FontWeight="Bold"
39                      FontSize ="25" Grid.Row="1" Grid.RowSpan="2" Grid.Column = "4"
    ↪ HorizontalAlignment="Center"
40                      Width ="150" Click="BuyButton_Click" BorderThickness="0"/>
41
42              <Button Name ="SellButton" Content="Sell" Background="Red"
    ↪ FontWeight="Bold"
43                      FontSize ="25" Grid.Row="1" Grid.RowSpan="2" Grid.Column = "5"
    ↪ HorizontalAlignment="Center"
44                      Width ="150" Click="SellButton_Click" BorderThickness="0"/>
45
46              <TextBox Text="Your Current Position:" Grid.Row="3" Grid.Column="4"
    ↪ Background="Transparent"
47                       Foreground="WhiteSmoke" FontSize="23" BorderThickness="0"
    ↪ VerticalAlignment="Center"
48                       HorizontalAlignment="Right" IsReadOnly ="True"/>
49
50              <TextBox Text="Our Recommendation:" Grid.Row="4" Grid.Column="4"
    ↪ Background="Transparent"
51                       Foreground="WhiteSmoke" FontSize="23" BorderThickness="0"
    ↪ VerticalAlignment="Top"
52                       HorizontalAlignment="Right" IsReadOnly ="True"/>
53
54              <TextBox Text="Current Price:" Grid.Row="4" Grid.Column="4"
    ↪ Background="Transparent"
55                       Foreground="WhiteSmoke" FontSize="23" BorderThickness="0"
    ↪ VerticalAlignment="Bottom"
56                       HorizontalAlignment="Right" IsReadOnly ="True"/>
57
58              <TextBox Name="CurrentPositionBox" Text="1Test" Grid.Row="3"
    ↪ Grid.Column="5" Background="Transparent"
59                       Foreground="WhiteSmoke" FontSize="23" BorderThickness="0"
    ↪ VerticalAlignment="Center"
60                       HorizontalAlignment="Center" IsReadOnly ="True"/>
61
```

```
62              <TextBox Name="OurRecommendationBox" Text="2Test" Grid.Row="4"
    ↪ Grid.Column="5" Background="Transparent"
63                       Foreground="WhiteSmoke" FontSize="23" BorderThickness="0"
    ↪ VerticalAlignment="Top"
64                       HorizontalAlignment="Center" IsReadOnly ="True"/>
65
66              <TextBox Name="CurrentPriceBox" Text="3Test" Grid.Row="4" Grid.Column="5"
    ↪ Background="Transparent"
67                       Foreground="WhiteSmoke" FontSize="23" BorderThickness="0"
    ↪ VerticalAlignment="Bottom"
68                       HorizontalAlignment="Center" IsReadOnly ="True"/>
69
70              <Button Name="AddToWatchlist" Content="Add to watchlist +" Grid.Column="5"
    ↪ Grid.Row="5"
71                      Width="200" Height="60" Background="Transparent"
    ↪ Foreground="WhiteSmoke"
72                      FontSize="22" Click="AddToWatchlist_Click"/>
73
74              <TextBox Name="WatchlistError" Text="Watchlist full" Grid.Column="4"
    ↪ Grid.Row="5"
75                       Background="Transparent" Foreground="WhiteSmoke" FontSize="18"
76                       TextAlignment="Center" HorizontalAlignment="Right" Height="40"
    ↪ BorderThickness="0"
77                       Visibility="Collapsed"/>
78
79              <Grid Name="BuyGrid" Visibility="Collapsed" Grid.Row="1" Grid.Column="4"
    ↪ Grid.RowSpan="3" Background="MediumAquamarine">
80                  <Grid.RowDefinitions>
81                      <RowDefinition Height="*"/>
82                      <RowDefinition Height="*"/>
83                      <RowDefinition Height="*"/>
84                      <RowDefinition Height="*"/>
85                      <RowDefinition Height="*"/>
86                  </Grid.RowDefinitions>
87                  <Grid.ColumnDefinitions>
88                      <ColumnDefinition Width="*"/>
89                      <ColumnDefinition Width="*"/>
90                      <ColumnDefinition Width="*"/>
91                      <ColumnDefinition Width="*"/>
92                      <ColumnDefinition Width="*"/>
93                  </Grid.ColumnDefinitions>
94
95                  <TextBox Text="Enter Amount:" Grid.Column="1" Grid.Row="0"
    ↪ Grid.ColumnSpan="3"
96                           TextAlignment="Center" Background="Transparent"
    ↪ BorderThickness="0"
97                           VerticalAlignment="Center" FontSize="20" IsReadOnly="True"/>
98
99                  <TextBox Name = "BuyAmount" TextAlignment="Center" Grid.Column="1"
    ↪ Grid.ColumnSpan="3"
100                          Grid.Row="1" Background="WhiteSmoke" Panel.ZIndex="1"
    ↪ IsReadOnly="False"
101                          FontSize="17"/>
102
103                 <TextBox Text="$" Grid.Column="1" Grid.Row="1" Panel.ZIndex="2"
    ↪ Background="WhiteSmoke"
104                          TextAlignment="Center" VerticalAlignment="Center"
    ↪ BorderThickness="0"
```

```
                        FontSize="18" Width="20"/>

                <Button Name="ConfirmBuy" Grid.Column="1" Grid.Row="3"
    ↪ Grid.ColumnSpan="3" Content="Confirm Buy"
                        Background="Lime" Height="55" Width="156"
    ↪ VerticalAlignment="Center" HorizontalAlignment="Right"
                        Click="ConfirmBuy_Click" BorderThickness="0"/>

                <Button Name="CloseBuyGrid" Click="CloseBuyGrid_Click" Content="X"
                        Grid.Row="0" Grid.Column="4"  Width="25" Height="20"/>

            </Grid>

            <Grid Name="SellGrid" Visibility="Collapsed" Grid.Row="1" Grid.Column="5"
    ↪ Grid.RowSpan="3" Background="Coral">
                <Grid.RowDefinitions>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>

                <TextBox Text="Enter Amount:" Grid.Column="1" Grid.Row="0"
    ↪ Grid.ColumnSpan="3"
                        TextAlignment="Center" Background="Transparent"
    ↪ BorderThickness="0"
                        VerticalAlignment="Center" FontSize="20" IsReadOnly="True"/>

                <TextBox Name = "SellAmount" TextAlignment="Center" Grid.Column="1"
    ↪ Grid.ColumnSpan="3"
                        Grid.Row="1" Background="WhiteSmoke" Panel.ZIndex="1"
    ↪ IsReadOnly="False"
                        FontSize="17"/>

                <TextBox Text="$" Grid.Column="1" Grid.Row="1" Panel.ZIndex="2"
    ↪ Background="WhiteSmoke"
                        TextAlignment="Center" VerticalAlignment="Center"
    ↪ BorderThickness="0"
                        FontSize="18" Width="20"/>

                <Button Name="ConfirmSell" Grid.Column="1" Grid.Row="3"
    ↪ Grid.ColumnSpan="3" Content="Confirm Sell"
                        Background="Red" Height="55" Width="156" Grid.RowSpan="2"
    ↪ VerticalAlignment="Top"
                        Click="ConfirmSell_Click" BorderThickness="0"/>

                <Button Name="CloseSellGrid" Click="CloseSellGrid_Click" Content="X"
                        Grid.Row="0" Grid.Column="4" Width="25" Height="20"/>

            </Grid>
```

```
152
153         </Grid>
```

4.2.5 Account page XAML

```
1           <Grid Name="AccountScreen" Visibility="Collapsed">
2               <Grid.RowDefinitions>
3                   <RowDefinition Height="*"/>
4                   <RowDefinition Height="*"/>
5                   <RowDefinition Height="*"/>
6                   <RowDefinition Height="*"/>
7                   <RowDefinition Height="*"/>
8               </Grid.RowDefinitions>
9               <Grid.ColumnDefinitions>
10                  <ColumnDefinition Width="*"/>
11                  <ColumnDefinition Width="*"/>
12                  <ColumnDefinition Width="*"/>
13                  <ColumnDefinition Width="*"/>
14                  <ColumnDefinition Width="*"/>
15              </Grid.ColumnDefinitions>
16
17              <TextBox Text="Account Details:" FontSize="38" Grid.Column="0"
    ↪ Grid.Row="0" Background="Transparent"
18                      Foreground="WhiteSmoke" Grid.ColumnSpan="2"
    ↪ HorizontalAlignment="Center"
19                      VerticalAlignment="Center" BorderThickness="0"
    ↪ IsReadOnly="True"/>
20
21              <TextBox Text="Email:" Background="Transparent" Foreground="WhiteSmoke"
    ↪ FontSize="27"
22                      Grid.Column="0" Grid.Row="1" HorizontalAlignment="Right"
    ↪ VerticalAlignment="Top"
23                      IsReadOnly = "True" BorderThickness="0"/>
24
25              <TextBox Text="Password:" Background="Transparent" Foreground="WhiteSmoke"
    ↪ FontSize="27"
26                      Grid.Column="0" Grid.Row="1" HorizontalAlignment="Right"
    ↪ VerticalAlignment="Bottom"
27                      IsReadOnly = "True" BorderThickness="0"/>
28
29              <TextBox Text="Email" Background="Transparent" Foreground="WhiteSmoke"
    ↪ FontSize="17"
30                      Grid.Column="0" Grid.Row="1" HorizontalAlignment="Right"
    ↪ VerticalAlignment="Center"
31                      IsReadOnly = "True" BorderThickness="0"
    ↪ Name="AccountScreenEmail"/>
32
33              <TextBox Name= "AccountScreenEmailError" Text="Invalid Email"
    ↪ FontSize="10"
34                      BorderThickness="0" IsReadOnly="True" Grid.Row="1"
    ↪ Grid.Column="0" Background="Transparent"
35                      Foreground="WhiteSmoke" VerticalAlignment="Center"
    ↪ HorizontalAlignment="Center" Visibility="Collapsed"/>
36
37              <TextBox Text="Passw0rd" Background="Transparent" Foreground="WhiteSmoke"
    ↪ FontSize="17"
38                      Grid.Column="0" Grid.Row="2" HorizontalAlignment="Right"
    ↪ VerticalAlignment="Top"
```

```
39                      IsReadOnly = "True" BorderThickness="0"
    ↪ Name="AccountScreenPassword"/>
40
41              <Button Name="EditEmailBox" Background="WhiteSmoke" Content="Edit Email"
42                      Grid.Column="1" Grid.Row="1" HorizontalAlignment="Center"
    ↪ VerticalAlignment="Center"
43                      Width="100" Height="30" Click="EditEmailBox_Click"/>
44
45              <Button Name="EditPasswordBox" Background="WhiteSmoke" Content="Edit
    ↪ Password"
46                      Grid.Column="1" Grid.Row="2" HorizontalAlignment="Center"
    ↪ VerticalAlignment="Top"
47                      Width="100" Height="30" Click="EditPasswordBox_Click"/>
48
49              <Button Name = "SaveEmail" Content="Save" Visibility="Collapsed"
    ↪ Click="SaveEmail_Click"
50                      Width = "50" Height="30" Grid.Column="1" Grid.Row="1"
    ↪ HorizontalAlignment="Right"
51                      VerticalAlignment="Center"/>
52
53              <Button Name = "SavePassword" Content="Save" Visibility="Collapsed"
    ↪ Click="SavePassword_Click"
54                      Width = "50" Height="30" Grid.Column="1" Grid.Row="2"
    ↪ HorizontalAlignment="Right"
55                      VerticalAlignment="Top"/>
56
57              <Button Name="UpdateData" Content="Update stock data for algorithm"
    ↪ Grid.Column="1" Grid.Row="3"
58                      Width="180" Height="40" Background="Transparent"
    ↪ Foreground="WhiteSmoke" Click="UpdateData_Click"/>
59
60              <TextBox Text="Watchlist:" FontSize="27" Foreground="WhiteSmoke"
    ↪ Background="Transparent"
61                       BorderThickness="0" Grid.Column="4" Grid.Row="2"
    ↪ VerticalAlignment="Center"
62                       HorizontalAlignment="Left" IsReadOnly="True"/>
63
64              <TextBox Text="Current Positions:" FontSize="27" Foreground="WhiteSmoke"
    ↪ Background="Transparent"
65                       BorderThickness="0" Grid.Column="3" Grid.Row="2"
    ↪ VerticalAlignment="Center"
66                       HorizontalAlignment="Left" IsReadOnly="True"/>
67
68              <TextBox Text="Overall PandL:" FontSize="27" Foreground="WhiteSmoke"
    ↪ Background="Transparent"
69                       BorderThickness="0" Grid.Column="3" Grid.Row="1"
    ↪ VerticalAlignment="Top"
70                       HorizontalAlignment="Right" IsReadOnly="True"/>
71
72              <TextBox Text="$$$$" FontSize="24" Foreground="WhiteSmoke"
    ↪ Background="Transparent"
73                       BorderThickness="0" Grid.Column="3" Grid.Row="1"
    ↪ VerticalAlignment="Center"
74                       HorizontalAlignment="Center" IsReadOnly="True"
    ↪ Name="AccountPandL"/>
75
76              <Button  FontSize="17" Foreground="WhiteSmoke" Background="Transparent"
```

```
77                      BorderThickness="0" Grid.Column="4" Grid.Row="2"
    ↪ VerticalAlignment="Bottom"
78                      HorizontalAlignment="Left" Click="WatchList1_Click"
    ↪ Name="WatchListStock1"/>
79
80              <Button  FontSize="17" Foreground="WhiteSmoke" Background="Transparent"
81                      BorderThickness="0" Grid.Column="4" Grid.Row="3"
    ↪ VerticalAlignment="Center"
82                      HorizontalAlignment="Left" Click="WatchList2_Click"
    ↪ Name="WatchListStock2"/>
83
84              <Button  FontSize="17" Foreground="WhiteSmoke" Background="Transparent"
85                      BorderThickness="0" Grid.Column="4" Grid.Row="4"
    ↪ VerticalAlignment="Top"
86                      HorizontalAlignment="Left" Click="WatchList3_Click"
    ↪ Name="WatchListStock3"/>
87
88              <Button  Content="YourStock1" FontSize="17" Foreground="WhiteSmoke"
    ↪ Background="Transparent"
89                      BorderThickness="0" Grid.Column="3" Grid.Row="2"
    ↪ VerticalAlignment="Bottom"
90                      HorizontalAlignment="Left" Click="YourStock1_Click"
    ↪ Name="YourStock1"/>
91
92              <Button  Content="YourStock2" FontSize="17" Foreground="WhiteSmoke"
    ↪ Background="Transparent"
93                      BorderThickness="0" Grid.Column="3" Grid.Row="3"
    ↪ VerticalAlignment="Center"
94                      HorizontalAlignment="Left" Click="YourStock2_Click"
    ↪ Name="YourStock2"/>
95
96              <Button  Content="YourStock3" FontSize="17" Foreground="WhiteSmoke"
    ↪ Background="Transparent"
97                      BorderThickness="0" Grid.Column="3" Grid.Row="4"
    ↪ VerticalAlignment="Top"
98                      HorizontalAlignment="Left" Click="YourStock3_Click"
    ↪ Name="YourStock3"/>
99
100             <Button Name="LogOutButton" Content="Log Out" Width="60" Height="30"
101                     Grid.Column="3" HorizontalAlignment="Right"
    ↪ Click="LogOutButton_Click"/>
102
103             <Button Name="CloseProgramButtonAccountScreen" Content="X" Grid.Column="5"
    ↪ Grid.Row="0"
104                 Width="40" Height="25" HorizontalAlignment="Right"
    ↪ VerticalAlignment="Top"
105                 Click="CloseProgramButtonStockScreen_Click"/>
106
107             <Button Name="BackButtonToHomeScreenAccount" Content="Back"
    ↪ Grid.Column="0" Grid.Row="0"
108                 Width="40" Height ="30" HorizontalAlignment="Left"
    ↪ VerticalAlignment="Top"
109                 Click="BackButtonToHomeScreenAccount_Click" />
110
111             <Button Name="RunML" Content="Re-Run Prediction Algorithm" Grid.Column="0"
    ↪ Grid.Row="3"
112                     Background="Transparent" Foreground="WhiteSmoke" Width="190"
    ↪ Height="40"
```

```
                    Click="RunML_Click"/>

        </Grid>

    </Grid>

</Window>

```

4.2.6 Main window UI interactions

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Forms;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Drawing;
using System.Drawing.Imaging;
using System.Windows.Controls;
using System.Printing;
using System.Data.Entity;
using System.Threading;
using System.Windows.Forms.DataVisualization.Charting;
using ScottPlot;
using ScottPlot.WPF;
using Color = System.Drawing.Color;
using ScottPlot.Plottable;
using System.Net.NetworkInformation;

namespace MATLABintegrationTest
{
    /// <summary>
    /// Interaction logic for all of the pages
    /// </summary>
    public partial class MainWindow
    {

        //defining variables global to this class

        string email = "";
        string password = "";
        string stockCodeG = "";
        string stockNameG = "";
        string[] hotStocks = new string[3];
        string[,] notifications = new string[4,2];

        public MainWindow()
        {
            InitializeComponent();
```

```
48
49              DatabaseInteractions database = new DatabaseInteractions();
50
51              //database.CreateTable(database.CreateConnection());
52              //database.InsertStocks(database.CreateConnection());
53              string[,] stockChanges =
    ↪  database.RelChangeInStockPrice(database.CreateConnection());
54              database.UpdateUserPositions(database.CreateConnection(), stockChanges);
55
56
57              MATLABinteractions MATLABloader = new MATLABinteractions();
58
59              hotStocks = MATLABloader.FindHotStocks();
60              TopLStock.Content = database.GetNameFromCode(database.CreateConnection(),
    ↪  hotStocks[0]);
61              TopMStock.Content = database.GetNameFromCode(database.CreateConnection(),
    ↪  hotStocks[1]);
62              TopRStock.Content = database.GetNameFromCode(database.CreateConnection(),
    ↪  hotStocks[2]);
63
64              database.UpdateStockPrices(database.CreateConnection());
65
66              TLPlot.Plot.Style(figureBackground: Color.Black);
67              TLPlot.Plot.Style(dataBackground: Color.Black);
68              TLPlot.Plot.Style(grid: Color.Black);
69              TLPlot.Plot.Style(axisLabel: Color.WhiteSmoke);
70              TLPlot.Plot.XAxis.Color(Color.WhiteSmoke);
71              TLPlot.Plot.YAxis.Color(Color.WhiteSmoke);
72              TLPlot.Plot.XLabel("Days");
73              TLPlot.Plot.YLabel("Price/$");
74              BMPlot.Refresh();
75
76              TMPlot.Plot.Style(figureBackground: Color.Black);
77              TMPlot.Plot.Style(dataBackground: Color.Black);
78              TMPlot.Plot.Style(grid: Color.Black);
79              TMPlot.Plot.Style(axisLabel: Color.WhiteSmoke);
80              TMPlot.Plot.XAxis.Color(Color.WhiteSmoke);
81              TMPlot.Plot.YAxis.Color(Color.WhiteSmoke);
82              TMPlot.Plot.XLabel("Days");
83              TMPlot.Plot.YLabel("Price/$");
84              BMPlot.Refresh();
85
86              TRPlot.Plot.Style(figureBackground: Color.Black);
87              TRPlot.Plot.Style(dataBackground: Color.Black);
88              TRPlot.Plot.Style(grid: Color.Black);
89              TRPlot.Plot.Style(axisLabel: Color.WhiteSmoke);
90              TRPlot.Plot.XAxis.Color(Color.WhiteSmoke);
91              TRPlot.Plot.YAxis.Color(Color.WhiteSmoke);
92              TRPlot.Plot.XLabel("Days");
93              TRPlot.Plot.YLabel("Price/$");
94              BMPlot.Refresh();
95
96              BLPlot.Plot.Style(figureBackground: Color.Black);
97              BLPlot.Plot.Style(dataBackground: Color.Black);
98              BLPlot.Plot.Style(grid: Color.Black);
99              BLPlot.Plot.Style(axisLabel: Color.WhiteSmoke);
100             BLPlot.Plot.XAxis.Color(Color.WhiteSmoke);
101             BLPlot.Plot.YAxis.Color(Color.WhiteSmoke);
```

```
            BLPlot.Plot.XLabel("Days");
            BLPlot.Plot.YLabel("Price/$");
            BMPlot.Refresh();

            BMPlot.Plot.Style(figureBackground: Color.Black);
            BMPlot.Plot.Style(dataBackground: Color.Black);
            BMPlot.Plot.Style(grid: Color.Black);
            BMPlot.Plot.Style(axisLabel: Color.WhiteSmoke);
            BMPlot.Plot.XAxis.Color(Color.WhiteSmoke);
            BMPlot.Plot.YAxis.Color(Color.WhiteSmoke);
            BMPlot.Plot.XLabel("Days");
            BMPlot.Plot.YLabel("Price/$");
            BMPlot.Refresh();

            BRPlot.Plot.Style(figureBackground: Color.Black);
            BRPlot.Plot.Style(dataBackground: Color.Black);
            BRPlot.Plot.Style(grid: Color.Black);
            BRPlot.Plot.Style(axisLabel: Color.WhiteSmoke);
            BRPlot.Plot.XAxis.Color(Color.WhiteSmoke);
            BRPlot.Plot.YAxis.Color(Color.WhiteSmoke);
            BRPlot.Plot.XLabel("Days");
            BRPlot.Plot.YLabel("Price/$");
            BMPlot.Refresh();

            StockScreenPlot.Plot.Style(figureBackground: Color.Black);
            StockScreenPlot.Plot.Style(dataBackground: Color.Black);
            StockScreenPlot.Plot.Style(grid: Color.Black);
            StockScreenPlot.Plot.Style(axisLabel: Color.WhiteSmoke);
            StockScreenPlot.Plot.XAxis.Color(Color.WhiteSmoke);
            StockScreenPlot.Plot.YAxis.Color(Color.WhiteSmoke);
            StockScreenPlot.Plot.XLabel("Days");
            StockScreenPlot.Plot.YLabel("Price/$");
            StockScreenPlot.Refresh();

            double[] priceForPlot = MATLABloader.PlotData(hotStocks[0]);
            priceForPlot = priceForPlot.Where(T => T != 0).ToArray();
            int[] numberOfDays = Enumerable.Range(1, priceForPlot.Length).ToArray();
            double[] days = numberOfDays.Select(x => (double)x).ToArray();

            TLPlot.Plot.AddScatter(days, priceForPlot);
            TLPlot.Refresh();

            priceForPlot = MATLABloader.PlotData(hotStocks[1]);
            priceForPlot = priceForPlot.Where(T => T != 0).ToArray();
            numberOfDays = Enumerable.Range(1, priceForPlot.Length).ToArray();
            days = numberOfDays.Select(x => (double)x).ToArray();

            TMPlot.Plot.AddScatter(days, priceForPlot);
            TMPlot.Refresh();

            priceForPlot = MATLABloader.PlotData(hotStocks[2]);
            priceForPlot = priceForPlot.Where(T => T != 0).ToArray();
            numberOfDays = Enumerable.Range(1, priceForPlot.Length).ToArray();
            days = numberOfDays.Select(x => (double)x).ToArray();

            TRPlot.Plot.AddScatter(days, priceForPlot);
            TRPlot.Refresh();
        }
```

```
160
161         //Welcome Screen
162
163         //performs verification of login details and loads all user specific metrics
164         private void Button_Click_Login(object sender, RoutedEventArgs e)
165         {
166             DatabaseInteractions database = new DatabaseInteractions();
167             MATLABinteractions MATLABloader = new MATLABinteractions();
168             Notifications getNotifications = new Notifications();
169
170             email = WelcomeEmailBox.Text;
171             password = PasswordLogin.Password;
172
173
174             if (database.VerifyLoginData(database.CreateConnection(), email, password)
    ↪  == true)
175             {
176                 WelcomeScreen.Visibility = Visibility.Collapsed;
177                 HomeScreen.Visibility = Visibility.Visible;
178
179                 AccountScreenEmail.Text = email;
180                 AccountScreenPassword.Text = password;
181
182                 SearchedStock.Visibility = Visibility.Collapsed;
183
184                 string[] userTopSymbols =
    ↪  database.ReturnUserTopStocks(database.CreateConnection(), email);
185                 int counter = 0;
186                 string[] userTopNames = new string[3];
187
188                 while (counter < 3 && (userTopSymbols[counter] != null ||
    ↪  userTopSymbols[counter] != ""))
189                 {
190                     userTopNames[counter] =
    ↪  database.GetNameFromCode(database.CreateConnection(), userTopSymbols[counter]);
191                     counter++;
192                 }
193                 BottomLStock.Content = userTopNames[0];
194                 BottomMStock.Content = userTopNames[1];
195                 BottomRStock.Content = userTopNames[2];
196                 YourStock1.Content = userTopNames[0];
197                 YourStock2.Content = userTopNames[1];
198                 YourStock3.Content = userTopNames[2];
199
200                 double[] priceForPlot;
201                 int[] numberOfDays;
202                 double[] days;
203
204                 if (userTopSymbols[0] != null)
205                 {
206                     priceForPlot = MATLABloader.PlotData(userTopSymbols[0]);
207                     priceForPlot = priceForPlot.Where(T => T != 0).ToArray();
208                     numberOfDays = Enumerable.Range(1, priceForPlot.Length).ToArray();
209                     days = numberOfDays.Select(x => (double)x).ToArray();
210
211                     BLPlot.Plot.AddScatter(days, priceForPlot);
212                     BLPlot.Refresh();
213                 }
```

```
214
215
216                 if (userTopSymbols[1] != null)
217                 {
218                     priceForPlot = MATLABloader.PlotData(userTopSymbols[1]);
219                     priceForPlot = priceForPlot.Where(T => T != 0).ToArray();
220                     numberOfDays = Enumerable.Range(1, priceForPlot.Length).ToArray();
221                     days = numberOfDays.Select(x => (double)x).ToArray();
222
223                     BMPlot.Plot.AddScatter(days, priceForPlot);
224                     BMPlot.Refresh();
225                 }
226
227
228                 if (userTopSymbols[2] != null)
229                 {
230                     priceForPlot = MATLABloader.PlotData(userTopSymbols[2]);
231                     priceForPlot = priceForPlot.Where(T => T != 0).ToArray();
232                     numberOfDays = Enumerable.Range(1, priceForPlot.Length).ToArray();
233                     days = numberOfDays.Select(x => (double)x).ToArray();
234
235                     BRPlot.Plot.AddScatter(days, priceForPlot);
236                     BRPlot.Refresh();
237                 }
238
239                 notifications = getNotifications.ReturnStockNotifications(email);
240
241                 FirstNotification.Content = String.Format("{0} - {1}",
    ↪  notifications[0,0], notifications[0,1]);
242                 SecondNotification.Content = String.Format("{0} - {1}",
    ↪  notifications[1, 0], notifications[1, 1]);
243                 ThirdNotification.Content = String.Format("{0} - {1}",
    ↪  notifications[2, 0], notifications[2, 1]);
244                 FourthNotification.Content = String.Format("{0} - {1}",
    ↪  notifications[3, 0], notifications[3, 1]);
245
246                 if (notifications[0,0] == null || notifications[0,0] == "" ||
    ↪  notifications[0,0] == " ")
247                 {
248                     FirstNotification.IsEnabled = false;
249                 }
250                 else
251                 {
252                     FirstNotification.IsEnabled = true;
253                 }
254                 if (notifications[1, 0] == null || notifications[1, 0] == "" ||
    ↪  notifications[1, 0] == " ")
255                 {
256                     SecondNotification.IsEnabled = false;
257                 }
258                 else
259                 {
260                     SecondNotification.IsEnabled = true;
261                 }
262                 if (notifications[2, 0] == null || notifications[2, 0] == "" ||
    ↪  notifications[2, 0] == " ")
263                 {
264                     ThirdNotification.IsEnabled = false;
```

```
265                 }
266                 else
267                 {
268                     ThirdNotification.IsEnabled = true;
269                 }
270                 if (notifications[3, 0] == null || notifications[3, 0] == "" ||
    ↪ notifications[3, 0] == " ")
271                 {
272                     FourthNotification.IsEnabled = false;
273                 }
274                 else
275                 {
276                     FourthNotification.IsEnabled = true;
277                 }
278 
279                 string[] watchlist =
    ↪ database.ReturnUserWatchlist(database.CreateConnection(), email);
280                 WatchListStock1.Content =
    ↪ database.GetNameFromCode(database.CreateConnection(), watchlist[0]);
281                 WatchListStock2.Content =
    ↪ database.GetNameFromCode(database.CreateConnection(), watchlist[1]);
282                 WatchListStock3.Content =
    ↪ database.GetNameFromCode(database.CreateConnection(), watchlist[2]);
283                 if (watchlist[0] == null || watchlist[0] == "" || watchlist[0] == " ")
284                 {
285                     WatchListStock1.IsEnabled = false;
286                 }
287                 else
288                 {
289                     WatchListStock1.IsEnabled = true;
290                 }
291                 if (watchlist[1] == null || watchlist[1] == "" || watchlist[1] == " ")
292                 {
293                     WatchListStock2.IsEnabled = false;
294                 }
295                 else
296                 {
297                     WatchListStock2.IsEnabled = true;
298                 }
299                 if (watchlist[2] == null || watchlist[2] == "" || watchlist[2] == " ")
300                 {
301                     WatchListStock3.IsEnabled = false;
302                 }
303                 else
304                 {
305                     WatchListStock3.IsEnabled = true;
306                 }
307 
308                 double pandL = database.ReturnUserPandL(database.CreateConnection(),
    ↪ email);
309                 AccountPandL.Text = String.Format("${0:.00}", pandL);
310             }
311             else
312             {
313                 InvalidLogin.Visibility = Visibility.Visible;
314             }
315 
316         }
```

```
317
318         //loads the create account screen
319         private void Button_Click_CreateAccountScreen(object sender, RoutedEventArgs
    ↪   e)
320         {
321             WelcomeScreen.Visibility = Visibility.Collapsed;
322             AccountCreation.Visibility = Visibility.Visible;
323         }
324
325         //hides password message
326         private void passwordBoxLogin_PasswordChanged(object sender, RoutedEventArgs
    ↪   e)
327         {
328             if (string.IsNullOrEmpty(PasswordLogin.Password))
329                 PassWordTextLogin.Visibility = Visibility.Visible;
330             else
331                 PassWordTextLogin.Visibility = Visibility.Collapsed;
332         }
333
334         //closes program
335         private void CloseProgramButtonWelcomeScreen_Click(object sender,
    ↪   RoutedEventArgs e)
336         {
337             this.Close();
338         }
339
340
341         //Create Account
342
343         //returns to login page from account creation page
344         private void Button_Click_BackToLogin(object sender, RoutedEventArgs e)
345         {
346             WelcomeScreen.Visibility = Visibility.Visible;
347             AccountCreation.Visibility = Visibility.Collapsed;
348         }
349
350         //adds account details to database
351         private void Button_Click_CreateAccount(object sender, RoutedEventArgs e)
352         {
353             DatabaseInteractions Database = new DatabaseInteractions();
354             email = EmailCreateAccountBox.Text;
355             password = PasswordCreateAccountBox.Password;
356
357             if (Database.CheckValidEmail(Database.CreateConnection(), email) == true)
358             {
359                 AccountCreation.Visibility = Visibility.Collapsed;
360                 HomeScreen.Visibility = Visibility.Visible;
361
362                 AccountScreenEmail.Text = email;
363                 AccountScreenPassword.Text = password;
364
365                 FirstNotification.Content = "";
366                 FirstNotification.IsEnabled = false;
367                 SecondNotification.Content = "";
368                 SecondNotification.IsEnabled = false;
369                 ThirdNotification.Content = "";
370                 ThirdNotification.IsEnabled = false;
371                 FourthNotification.Content = "";
```

```
372                 FourthNotification.IsEnabled = false;
373 
374                 SearchedStock.Visibility = Visibility.Collapsed;
375 
376                 Database.InsertLoginData(Database.CreateConnection(), email,
    ↪ password);
377             }
378             else
379             {
380                 EmailUsedMessage.Visibility = Visibility.Visible;
381             }
382 
383             WatchListStock1.IsEnabled = false;
384             WatchListStock2.IsEnabled = false;
385             WatchListStock3.IsEnabled = false;
386 
387             FirstNotification.IsEnabled = false;
388             SecondNotification.IsEnabled = false;
389             ThirdNotification.IsEnabled = false;
390             FourthNotification.IsEnabled = false;
391         }
392 
393         //hides password message
394         private void passwordBox_PasswordChanged(object sender, RoutedEventArgs e)
395         {
396             if (string.IsNullOrEmpty(PasswordCreateAccountBox.Password))
397                 PassWordText.Visibility = Visibility.Visible;
398             else
399                 PassWordText.Visibility = Visibility.Collapsed;
400         }
401 
402         //closes program
403         private void CloseProgramButtonAccountScreen_Click(object sender,
    ↪ RoutedEventArgs e)
404         {
405             this.Close();
406         }
407 
408 
409         //Home Screen
410 
411         //opens notifications page
412         private void Button_Click_Notifications(object sender, RoutedEventArgs e)
413         {
414             NotificationScreen.Visibility = Visibility.Visible;
415         }
416 
417         //closes program
418         private void CloseProgramButtonHomeScreen_Click(object sender, RoutedEventArgs
    ↪ e)
419         {
420             this.Close();
421         }
422 
423         //opens top left stock (hot stock 1)
424         private void TopLStock_Click(object sender, RoutedEventArgs e)
425         {
426             OpenStockScreen(Convert.ToString(TopLStock.Content));
```

```
427         }
428
429         //opens top middle stock (hot stock 2)
430         private void TopMStock_Click(object sender, RoutedEventArgs e)
431         {
432             OpenStockScreen(Convert.ToString(TopMStock.Content));
433         }
434
435         //opens top right stock (hot stock 3)
436         private void TopRStock_Click(object sender, RoutedEventArgs e)
437         {
438             OpenStockScreen(Convert.ToString(TopRStock.Content));
439         }
440
441         //opens bottom left stock (user's most valuable stock)
442         private void BottomLStock_Click(object sender, RoutedEventArgs e)
443         {
444             OpenStockScreen(Convert.ToString(BottomLStock.Content));
445         }
446
447         //opens bottom middle stock (user's 2nd most valuable stock)
448         private void BottomMStock_Click(object sender, RoutedEventArgs e)
449         {
450             OpenStockScreen(Convert.ToString(BottomMStock.Content));
451         }
452
453         //opens bottom right stock (user's 3rd most valuable stock)
454         private void BottomRStock_Click(object sender, RoutedEventArgs e)
455         {
456             OpenStockScreen(Convert.ToString(BottomRStock.Content));
457         }
458
459         //hides search bar message
460         private void SearchBarTextChangedEventHandler(object sender,
    ↪ TextChangedEventArgs args)
461         {
462             if (string.IsNullOrEmpty(SearchBar.Text))
463             {
464                 SearchBarText.Visibility = Visibility.Visible;
465                 SearchButton.Visibility = Visibility.Collapsed;
466             }
467             else
468             {
469                 SearchBarText.Visibility = Visibility.Collapsed;
470                 SearchButton.Visibility = Visibility.Visible;
471             }
472         }
473
474         //triggers stock search sequence
475         private void SearchButton_Click(object sender, RoutedEventArgs e)
476         {
477             string stockToFind = SearchBar.Text;
478
479             DatabaseInteractions Database = new DatabaseInteractions();
480             string foundCode = Database.StockSearch(Database.CreateConnection(),
    ↪ stockToFind);
481             stockCodeG = foundCode;
482
```

```
483             stockNameG = Database.GetNameFromCode(Database.CreateConnection(),
    ↪  stockCodeG);
484 
485             if (foundCode == "" || foundCode == null)
486             {
487                 SearchedStock.Content = "No found stock";
488                 SearchedStock.Visibility = Visibility.Visible;
489                 SearchedStock.IsEnabled = false;
490             }
491             else
492             {
493                 //convert code to name
494                 SearchedStock.IsEnabled = true;
495                 SearchedStock.Content = foundCode;
496                 SearchedStock.Visibility = Visibility.Visible;
497             }
498         }
499 
500         //opens stock page of the searched stock
501         public void SearchedStock_Click(object sender, RoutedEventArgs e)
502         {
503             OpenStockScreen(stockNameG);
504             SearchedStock.Visibility = Visibility.Collapsed;
505         }
506 
507         //opens account page
508         private void ButtonToAccountScreen_Click(object sender, RoutedEventArgs e)
509         {
510             HomeScreen.Visibility = Visibility.Collapsed;
511             AccountScreen.Visibility = Visibility.Visible;
512         }
513 
514         //opens first stock notification
515         private void Noti1Click(object sender, RoutedEventArgs e)
516         {
517             //add if statement to check if company notification or stock notification
518             OpenStockScreen(notifications[0, 0]);
519         }
520 
521         //opens second stock notification
522         private void Noti2Click(object sender, RoutedEventArgs e)
523         {
524             OpenStockScreen(notifications[1,0]);
525         }
526 
527         //opens third stock notification
528         private void Noti3Click(object sender, RoutedEventArgs e)
529         {
530             OpenStockScreen(notifications[2, 0]);
531         }
532 
533         //opens fourth stock notification
534         private void Noti4Click(object sender, RoutedEventArgs e)
535         {
536             OpenStockScreen(notifications[3, 0]);
537         }
538 
539         //Notification screen
```

```
540
541         //closes the notification pop-down
542         private void CloseNotifications_Click(object sender, RoutedEventArgs e)
543         {
544             NotificationScreen.Visibility = Visibility.Collapsed;
545         }
546
547         //Stock Screen
548
549         //returns to home screen from stock screen
550         private void BackButtonToHomeScreen_Click(object sender, RoutedEventArgs e)
551         {
552             HomeScreen.Visibility = Visibility.Visible;
553             StockScreen.Visibility = Visibility.Collapsed;
554             BuyGrid.Visibility = Visibility.Collapsed;
555             SellGrid.Visibility = Visibility.Collapsed;
556             SearchedStock.Visibility = Visibility.Collapsed;
557
558             DatabaseInteractions Database = new DatabaseInteractions();
559
560             double currentValue = Database.ReturnValue(Database.CreateConnection(),
    ↪  email, stockCodeG);
561             CurrentPositionBox.Text = String.Format("${0:0.00}", currentValue);
562         }
563
564         //adds the current stock to the user's watchlist, if watchlist is not full
565         private void AddToWatchlist_Click(object sender, RoutedEventArgs e)
566         {
567             DatabaseInteractions database = new DatabaseInteractions();
568             string watchlistPos = database.AddWatchlist(database.CreateConnection(),
    ↪  stockCodeG, email);
569
570             if (watchlistPos == "WatchStock1")
571             {
572                 WatchListStock1.Content = stockNameG;
573                 WatchListStock1.IsEnabled = true;
574             }
575             else if(watchlistPos == "WatchStock2")
576             {
577                 WatchListStock2.Content = stockNameG;
578                 WatchListStock1.IsEnabled = true;
579             }
580             else if (watchlistPos == "WatchStock3")
581             {
582                 WatchListStock3.Content = stockNameG;
583                 WatchListStock1.IsEnabled = true;
584             }
585             else
586             {
587                 WatchlistError.Visibility = Visibility.Visible;
588             }
589         }
590
591         //opens sell stock pop-down
592         private void SellButton_Click(object sender, RoutedEventArgs e)
593         {
594             SellGrid.Visibility = Visibility.Visible;
595         }
```

```
596
597         //opens buy stock pop-down
598         private void BuyButton_Click(object sender, RoutedEventArgs e)
599         {
600             BuyGrid.Visibility = Visibility.Visible;
601         }
602
603         //writes the user's sell order to database
604         private void ConfirmSell_Click(object sender, RoutedEventArgs e)
605         {
606             try
607             {
608                 double SellVol = Convert.ToDouble(SellAmount.Text);
609
610                 DatabaseInteractions Database = new DatabaseInteractions();
611
612                 double OldValue = Database.ReturnValue(Database.CreateConnection(),
    ↪  email, stockCodeG);
613
614                 Database.InsertSell(Database.CreateConnection(), email, stockCodeG,
    ↪  SellVol, OldValue);
615
616                 double currentValue =
    ↪  Database.ReturnValue(Database.CreateConnection(), email, stockCodeG);
617                 CurrentPositionBox.Text = String.Format("${0:0.00}", currentValue);
618             }
619             catch
620             {
621
622             }
623
624         }
625
626         //writes the user's buy order to database
627         private void ConfirmBuy_Click(object sender, RoutedEventArgs e)
628         {
629             try
630             {
631                 double BuyVol = Convert.ToDouble(BuyAmount.Text);
632
633                 DatabaseInteractions Database = new DatabaseInteractions();
634
635                 double OldValue = Database.ReturnValue(Database.CreateConnection(),
    ↪  email, stockCodeG);
636
637                 Database.InsertBuy(Database.CreateConnection(), email, stockCodeG,
    ↪  BuyVol, OldValue);
638
639                 double currentValue =
    ↪  Database.ReturnValue(Database.CreateConnection(), email, stockCodeG);
640                 CurrentPositionBox.Text = String.Format("${0:0.00}", currentValue);
641             }
642             catch
643             {
644
645             }
646         }
647
```

```
        //closes user's buy pop-down
        private void CloseBuyGrid_Click(object sender, RoutedEventArgs e)
        {
            BuyGrid.Visibility= Visibility.Collapsed;
            BuyAmount.Text = "";
        }

        //closes user's sell pop-down
        private void CloseSellGrid_Click(object sender, RoutedEventArgs e)
        {
            SellGrid.Visibility = Visibility.Collapsed;
            SellAmount.Text = "";
        }

        //closes program
        private void CloseProgramButtonStockScreen_Click(object sender,
↪   RoutedEventArgs e)
        {
            this.Close();
        }

        //Account screen

        //enables email editing
        private void EditEmailBox_Click(object sender, RoutedEventArgs e)
        {
            AccountScreenEmail.IsReadOnly = false;
            SaveEmail.Visibility = Visibility.Visible;
            AccountScreenEmailError.Visibility = Visibility.Collapsed;
        }

        //enables password editing
        private void EditPasswordBox_Click(object sender, RoutedEventArgs e)
        {
            AccountScreenPassword.IsReadOnly = false;
            SavePassword.Visibility = Visibility.Visible;
        }

        //writes new email to database
        private void SaveEmail_Click(object sender, RoutedEventArgs e)
        {
            AccountScreenEmail.IsReadOnly = true;
            SaveEmail.Visibility = Visibility.Collapsed;

            DatabaseInteractions Database = new DatabaseInteractions();

            bool validEmail = Database.CheckValidEmail(Database.CreateConnection(),
↪   AccountScreenEmail.Text);

            if (validEmail == true)
            {
                Database.EditEmail(Database.CreateConnection(),
↪   AccountScreenEmail.Text, email);
                email = AccountScreenEmail.Text;
            }
            else
            {
                AccountScreenEmailError.Visibility = Visibility.Visible;
```

```
703                 AccountScreenEmail.Text = email;
704             }
705         }
706 
707         //writes new password to database
708         private void SavePassword_Click(object sender, RoutedEventArgs e)
709         {
710             AccountScreenPassword.IsReadOnly = true;
711             SavePassword.Visibility= Visibility.Collapsed;
712 
713             DatabaseInteractions Database = new DatabaseInteractions();
714 
715             password = AccountScreenPassword.Text;
716             Database.EditPassword(Database.CreateConnection(), password, email);
717         }
718 
719         //opens employee management pop-down
720         private void ManageEmployees_Click(object sender, EventArgs e)
721         {
722 
723         }
724 
725         //opens stock screens for watchlist stocks
726         private void WatchList1_Click(object sender, RoutedEventArgs e)
727         {
728             OpenStockScreen(Convert.ToString(WatchListStock1.Content));
729         }
730 
731         private void WatchList2_Click(object sender, RoutedEventArgs e)
732         {
733             OpenStockScreen(Convert.ToString(WatchListStock2.Content));
734         }
735 
736         private void WatchList3_Click(object sender, RoutedEventArgs e)
737         {
738             OpenStockScreen(Convert.ToString(WatchListStock3.Content));
739         }
740 
741         //opens stock screens for user's most bought stocks
742         private void YourStock1_Click(object sender, RoutedEventArgs e)
743         {
744             OpenStockScreen(Convert.ToString(YourStock1.Content));
745         }
746 
747         private void YourStock2_Click(object sender, RoutedEventArgs e)
748         {
749             OpenStockScreen(Convert.ToString(YourStock2.Content));
750         }
751 
752         private void YourStock3_Click(object sender, RoutedEventArgs e)
753         {
754             OpenStockScreen(Convert.ToString(YourStock3.Content));
755         }
756 
757         //returns to home screen from account screen
758         private void BackButtonToHomeScreenAccount_Click(object sender,
    ↪  RoutedEventArgs e)
759         {
```

```
760             HomeScreen.Visibility = Visibility.Visible;
761             AccountScreen.Visibility = Visibility.Collapsed;
762             SearchedStock.Visibility = Visibility.Collapsed;
763         }
764 
765         //logs user out and returns to login page
766         private void LogOutButton_Click(object sender, RoutedEventArgs e)
767         {
768             AccountScreen.Visibility= Visibility.Collapsed;
769             WelcomeScreen.Visibility = Visibility.Visible;
770 
771             PasswordLogin.Password = "";
772             WelcomeEmailBox.Text = "Enter Email";
773             InvalidLogin.Visibility = Visibility.Collapsed;
774             EmailCreateAccountBox.Text = "Enter Email";
775             PasswordCreateAccountBox.Password = "";
776             EmailUsedMessage.Visibility = Visibility.Collapsed;
777 
778             BottomLStock.Content = "";
779             BottomMStock.Content = "";
780             BottomRStock.Content = "";
781             YourStock1.Content = "";
782             YourStock2.Content = "";
783             YourStock3.Content = "";
784 
785             FirstNotification.Content = "";
786             SecondNotification.Content = "";
787             ThirdNotification.Content = "";
788             FourthNotification.Content = "";
789 
790             WatchListStock1.IsEnabled = false;
791             WatchListStock2.IsEnabled = false;
792             WatchListStock3.IsEnabled = false;
793 
794             AccountPandL.Text = "";
795         }
796 
797         //re-runs machine learning algorithm in MATLAB
798         private void RunML_Click(object sender, RoutedEventArgs e)
799         {
800             MATLABinteractions MATLABloader = new MATLABinteractions();
801             MATLABloader.RunML();
802         }
803 
804         //updates the data used in the prediction algorithms
805         private void UpdateData_Click(object sender, RoutedEventArgs e)
806         {
807             MATLABinteractions matlab = new MATLABinteractions();
808             matlab.UpdateData();
809         }
810 
811         //General
812 
813         //general sequence for opening stock page
814         private void OpenStockScreen(string stockButton)
815         {
816             HomeScreen.Visibility = Visibility.Collapsed;
817             AccountScreen.Visibility = Visibility.Collapsed;
```

```
818             NotificationScreen.Visibility = Visibility.Collapsed;
819             StockScreen.Visibility = Visibility.Visible;
820             WatchlistError.Visibility = Visibility.Collapsed;
821             StockScreenTitle.Text = Convert.ToString(TopLStock.Content);
822             stockNameG = Convert.ToString(stockButton);
823             StockScreenTitle.Text = stockNameG;
824 
825             DatabaseInteractions database = new DatabaseInteractions();
826             MATLABinteractions MATLABloader = new MATLABinteractions();
827 
828             stockCodeG = database.GetCodeFromName(database.CreateConnection(),
    ↪  stockNameG);
829 
830             double stockPrice = database.ReturnPrice(database.CreateConnection(),
    ↪  stockCodeG);
831             CurrentPriceBox.Text = String.Format("${0:0.00}", stockPrice);
832 
833             double currentValue = database.ReturnValue(database.CreateConnection(),
    ↪  email, stockCodeG);
834             CurrentPositionBox.Text = String.Format("${0:0.00}", currentValue);
835 
836             double[] priceForPlot = MATLABloader.PlotData(stockCodeG);
837             priceForPlot = priceForPlot.Where(T => T != 0).ToArray();
838             int[] numberOfDays = Enumerable.Range(1, priceForPlot.Length).ToArray();
839             double[] days = numberOfDays.Select(x => (double)x).ToArray();
840 
841             StockScreenPlot.Plot.Clear();
842             StockScreenPlot.Refresh();
843             StockScreenPlot.Plot.AddScatter(days, priceForPlot);
844             StockScreenPlot.Refresh();
845 
846             //run program to determine buy or sell
847             string prediction = MATLABloader.Predict(stockCodeG);
848             OurRecommendationBox.Text = prediction;
849         }
850 
851     }
852 }
```

4.2.7 MATLAB interactions class

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data.SQLite;
5  using System.Data.SqlTypes;
6  using System.Linq;
7  using System.Reflection;
8  using System.Text;
9  using System.Threading.Tasks;
10 
11 namespace MATLABintegrationTest
12 {
13 
14     //This class deals with all usage of MATLAB functions in the C# code
15     internal class MATLABinteractions
16     {
17 
```

```
18          //re-runs my machine learning algorithm to produce updated predictor fits
19          public void RunML()
20          {
21              MLApp.MLApp matlab = new MLApp.MLApp();
22              DatabaseInteractions Database = new DatabaseInteractions();
23
24              List<string> StockCodes =
    ↪  Database.ReturnStockCodes(Database.CreateConnection());
25              object objStockCodes = StockCodes.Select(x => x as object).ToArray();
26
27              matlab.Execute(@"cd C:\Users\olesc\OneDrive\Documents\MATLAB\");
28              matlab.Feval("StockfitML", 0, out object nullObj, objStockCodes);
29
30
31              matlab.Quit();
32          }
33
34          //returns the three most bought stocks from the last open market day
35          public string[] FindHotStocks()
36          {
37              MLApp.MLApp matlab = new MLApp.MLApp();
38              DatabaseInteractions Database = new DatabaseInteractions();
39              string[] threeStocks = new string[3];
40
41              List<string> StockCodes =
    ↪  Database.ReturnStockCodes(Database.CreateConnection());
42              object objStockCodes = StockCodes.Select(x => x as object).ToArray();
43
44              matlab.Execute(@"cd C:\Users\olesc\OneDrive\Documents\MATLAB\");
45              matlab.Feval("HotStocksFinder", 1, out object stocksOut, objStockCodes);
46              try
47              {
48                  object[]? hotStocks = stocksOut as object[];
49                  object[,] hotStockTest = hotStocks[0] as object[,];
50
51                  threeStocks[0] = hotStockTest[0,0].ToString();
52                  threeStocks[1] = hotStockTest[1,0].ToString();
53                  threeStocks[2] = hotStockTest[2,0].ToString();
54              }
55              catch
56              {
57                 threeStocks = null;
58              }
59              matlab.Quit();
60
61
62              return threeStocks;
63          }
64
65          //runs prediction on inputted stock, if custom alogrithm is unable to produce
    ↪  a prediction then
66          //MACD calculator used as a substitute
67          public string Predict(string stockCode)
68          {
69              string prediction = "";
70
71              MLApp.MLApp matlab = new MLApp.MLApp();
72
```

```
73              matlab.Execute(@"cd C:\Users\olesc\OneDrive\Documents\MATLAB\");
74              matlab.Feval("PricePredictor", 1, out object predictionObj, stockCode);
75              try
76              {
77                  object[]? predictionObjArray = predictionObj as object[];
78                  prediction = (string)predictionObjArray[0];
79              }
80              catch
81              {
82              }
83
84              if (prediction == "No consistent predictions from algorithm" || prediction
    ↪  == "")
85              {
86                  matlab.Execute(@"cd C:\Users\olesc\OneDrive\Documents\MATLAB\");
87                  matlab.Feval("MACDCalculator", 1, out object predictionMACDObj,
    ↪  stockCode);
88                  try
89                  {
90                      object[]? predictionMACDObjArray = predictionMACDObj as object[];
91                      prediction = (string)predictionMACDObjArray[0];
92                  }
93                  catch
94                  {
95                  }
96              }
97              if (prediction == "" || prediction == null)
98              {
99                  prediction = "Not available";
100             }
101
102             return prediction;
103         }
104
105         //returns the last available price of the desired stock
106         public double Price(string stockCode)
107         {
108             MLApp.MLApp matlab = new MLApp.MLApp();
109             double price = 0;
110
111             matlab.Execute(@"cd C:\Users\olesc\OneDrive\Documents\MATLAB\");
112             matlab.Feval("PriceReturner", 1, out object priceObj, stockCode);
113             try
114             {
115                 object[]? priceObjArray = priceObj as object[];
116                 price = Convert.ToDouble(priceObjArray[0]);
117             }
118             catch
119             {
120             }
121             matlab.Quit();
122
123             return price;
124         }
125
126
127         //loads new live data into local spreadsheet to be used for prediction
    ↪  algorithm
```

```
128         public void UpdateData()
129         {
130             MLApp.MLApp matlab = new MLApp.MLApp();
131 
132             matlab.Execute(@"cd C:\Users\olesc\OneDrive\Documents\MATLAB\");
133             matlab.Feval("StockDataLoader", 0, out object nullReturn);
134 
135             matlab.Quit();
136         }
137 
138         //loads last 30 days of data to be used for plotting stock graphs
139         public double[] PlotData(string stockCode)
140         {
141             double[] data = new double[31];
142             MLApp.MLApp matlab = new MLApp.MLApp();
143 
144             matlab.Execute(@"cd C:\Users\olesc\OneDrive\Documents\MATLAB\");
145             matlab.Feval("ReturnPlotData", 1, out object stockData, stockCode);
146 
147             object[]? priceObjArray = stockData as object[];
148             double[,] priceArray = (double[,])priceObjArray[0];
149 
150             for (int i = 0; i < priceArray.Length; i++)
151             {
152                 data[i] = Convert.ToDouble(priceArray[i,0]);
153             }
154 
155             matlab.Quit();
156             return data;
157         }
158 
159 
160         //runs the MACD calculator on the desired stock
161         public string MACDCalculator(string stockCode)
162         {
163             string entryExit = "";
164 
165             MLApp.MLApp matlab = new MLApp.MLApp();
166 
167             matlab.Execute(@"cd C:\Users\olesc\OneDrive\Documents\MATLAB\");
168             matlab.Feval("MACDCalculator", 1, out object priceObj, stockCode);
169             try
170             {
171                 object[]? priceObjArray = priceObj as object[];
172                 entryExit = (string)priceObjArray[0];
173             }
174             catch
175             {
176             }
177             matlab.Quit();
178 
179             return entryExit;
180         }
181 
182     }
183 }
```

4.2.8 Database interactions class

```
using System;
using System.Collections.Generic;
using System.DirectoryServices;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
//using Microsoft.Data.Sqlite;
using System.Data.SQLite;
using System.Windows.Documents;
using System.IO;
using System.Reflection;
using Microsoft.Data.Sqlite;
using System.Windows.Media.Animation;
using System.Net;
using System.Linq.Expressions;
using System.Xml.Schema;
using System.Windows.Input;

namespace MATLABintegrationTest
{

    //This class contains all the interactions with the database
    internal class DatabaseInteractions
    {

        //Creates the database (only used on first running of program)
        public void CreateDatabaseTest()
        {
            SQLiteConnection sqlite_conn;
            sqlite_conn = CreateConnection();
            CreateTable(sqlite_conn);
        }

        //creates connection to the database
        public SQLiteConnection CreateConnection()
        {

            SQLiteConnection sqlite_conn;
            // Create a new database connection:
            // sqlite_conn = new SqliteConnection(@"Data
↪ Source=file:C:\\Users\\olesc\\source\\repos\\AlevelNEA
↪ 2023-24\\Database\\Database.db;");
            sqlite_conn = new SQLiteConnection("Data Source=database.db");
            // Open the connection:
            try
            {
                sqlite_conn.Open();
            }
            catch (Exception ex)
            {
                Console.WriteLine("NOO");
            }
            return sqlite_conn;
        }

        //creates table in the database (only used in first running of program)
        public void CreateTable(SQLiteConnection conn)
```

```
56          {
57              SQLiteCommand sqlite_cmd;
58              //string Createsql = "CREATE TABLE LoginTable(Email VARCHAR, Password
    ↪  VARCHAR, TotalPandL DECIMAL, WatchStock1 VARCHAR, WatchStock2 VARCHAR, WatchStock3
    ↪  VARCHAR)";
59              //string Createsql = "CREATE TABLE StockCodesTable(Symbol VARCHAR, Name
    ↪  VARCHAR, Price DECIMAL)";
60              string Createsql = "CREATE TABLE UserPositionsTable(Email VARCHAR,
    ↪  StockSymbol VARCHAR, Value DECIMAL, LastUpdate DATE)";
61
62              //string Createsql = "DROP TABLE StockCodesTable";
63
64              sqlite_cmd = conn.CreateCommand();
65              sqlite_cmd.CommandText = Createsql;
66              sqlite_cmd.ExecuteNonQuery();
67              conn.Close();
68          }
69
70          //finds all the stock symbols owned by a particular user
71          public string[] ReturnUserStocks(SQLiteConnection conn, string email)
72          {
73              List<string> stockCodes = new List<string>();
74
75              SQLiteDataReader sqlite_datareader;
76              SQLiteCommand sqlite_cmd;
77              sqlite_cmd = conn.CreateCommand();
78              sqlite_cmd.CommandText = String.Format("SELECT StockSymbol FROM
    ↪  UserPositionsTable WHERE Email = '{0}';", email);
79              sqlite_datareader = sqlite_cmd.ExecuteReader();
80
81              while (sqlite_datareader.Read())
82              {
83                  stockCodes.Add(sqlite_datareader.GetString(0));
84              }
85              sqlite_datareader.Close();
86              conn.Close();
87
88              string[] userCodes = stockCodes.Select(i => i.ToString()).ToArray();
89
90              return userCodes;
91          }
92
93          //inserts all stock codes/names into table (only used in first running of
    ↪  program)
94          public void InsertStocks(SQLiteConnection conn)
95          {
96              SQLiteCommand sqlite_cmd;
97              sqlite_cmd = conn.CreateCommand();
98
99              string[,] companyCodes = LoadStockCode();
100             int length = companyCodes.GetLength(1);
101
102             for (int x = 0; x<length; x++)
103             {
104                 string Symbol = companyCodes[0,x];
105                 string Name = companyCodes[1,x];
106
```

```
107                 string Command = String.Format("INSERT INTO StockCodesTable (Symbol,
    ↪  Name, Price) VALUES('{0}', '{1}', '0');", Symbol, Name);
108                 sqlite_cmd.CommandText = Command;
109                 sqlite_cmd.ExecuteNonQuery();
110 
111             }
112 
113             conn.Close();
114         }
115 
116         //compares last recorded to current stock price to return their relative
    ↪  change in order to calculate
117         //user PandL
118         public string[,] RelChangeInStockPrice(SQLiteConnection conn)
119         {
120             SQLiteCommand sqlite_cmd;
121             sqlite_cmd = conn.CreateCommand();
122             DatabaseInteractions databaseAccess = new DatabaseInteractions();
123             MATLABinteractions MATLABprices = new MATLABinteractions();
124             double newPrice = -1;
125             double oldPrice = -1;
126 
127             List<string> companyCodes =
    ↪  databaseAccess.ReturnStockCodes(databaseAccess.CreateConnection());
128             int length = companyCodes.Count();
129             double[] relativeChanges = new double[length];
130             string[,] codesAndChanges = new string[2, length];
131 
132             for (int x = 0; x<length; x++)
133             {
134                 SQLiteDataReader sqlite_datareader;
135                 try
136                 {
137                     newPrice = MATLABprices.Price(companyCodes[x]);
138                 }
139                 catch
140                 {
141                     newPrice = -1;
142                 }
143                 string Command = String.Format("SELECT Price FROM StockCodesTable
    ↪  WHERE Symbol = '{0}';", companyCodes[x]);
144                 sqlite_cmd.CommandText = Command;
145                 sqlite_datareader = sqlite_cmd.ExecuteReader();
146                 try
147                 {
148                     while (sqlite_datareader.Read())
149                     {
150                         oldPrice = sqlite_datareader.GetDouble(0);
151                     }
152                 }
153                 catch (Exception ex)
154                 {
155                     oldPrice = -1;
156                 }
157                 sqlite_datareader.Close();
158 
159                 relativeChanges[x] = newPrice / oldPrice;
160                 codesAndChanges[0,x] = companyCodes[x];
```

```
161                 codesAndChanges[1,x] = Convert.ToString(relativeChanges[x]);
162             }
163             conn.Close();
164             return codesAndChanges;
165         }
166 
167         //updates the value of each position owned by users based of the relative
    ↪  change in stock price
168         public void UpdateUserPositions(SQLiteConnection conn, string[,] stockChanges)
169         {
170             SQLiteCommand sqlite_cmd;
171             sqlite_cmd = conn.CreateCommand();
172             SQLiteDataReader sqlite_datareader;
173             DoubleLinkList<string> emails = new DoubleLinkList<string>();
174             DoubleLinkList<string> boughtSymbols = new DoubleLinkList<string>();
175             DoubleLinkList<double> values = new DoubleLinkList<double>();
176             DoubleLinkList<string> codes = new DoubleLinkList<string>();
177             DoubleLinkList<double> changes = new DoubleLinkList<double>();
178             DateTime MyDateTime = DateTime.Now;
179             string sqlFormattedDate = MyDateTime.ToString("yyyy-MM-dd");
180 
181             for (int x = 0; x < stockChanges.GetLength(1); x++)
182             {
183                 codes.AddBack(stockChanges[0,x]);
184                 changes.AddBack(Convert.ToDouble(stockChanges[1,x]));
185             }
186 
187             string Command = String.Format("SELECT Email, StockSymbol, Value FROM
    ↪  UserPositionsTable;");
188             sqlite_cmd.CommandText = Command;
189             sqlite_datareader = sqlite_cmd.ExecuteReader();
190             try
191             {
192                 while (sqlite_datareader.Read())
193                 {
194                     emails.AddBack(sqlite_datareader.GetString(0));
195                     boughtSymbols.AddBack(sqlite_datareader.GetString(1));
196                     values.AddBack(sqlite_datareader.GetDouble(2));
197                 }
198             }
199             catch (Exception ex)
200             {
201             }
202             sqlite_datareader.Close();
203             SQLiteDataReader sqlite_datareader1;
204 
205             for (int x = 0; x < emails.GetLength(); x++)
206             {
207                 int codeIndex = codes.IndexOf(boughtSymbols.GetNode(x).DisplayNode());
208                 double stockChange = changes.GetNode(x).DisplayNode();
209                 double newValue = values.GetNode(x).DisplayNode() * stockChange;
210                 double currentPandL = 0;
211 
212                 Command = String.Format("SELECT TotalPandL FROM LoginTable WHERE Email
    ↪  = '{0}';", emails.GetNode(x).DisplayNode());
213                 sqlite_cmd.CommandText = Command;
214                 sqlite_datareader1 = sqlite_cmd.ExecuteReader();
215                 try
```

```
216                 {
217                     while (sqlite_datareader1.Read())
218                     {
219                         currentPandL = sqlite_datareader1.GetDouble(0);
220                     }
221                 }
222                 catch (Exception ex)
223                 {
224                 }
225                 sqlite_datareader1.Close();
226 
227                 double newPandL = currentPandL + newValue -
    ↪ values.GetNode(x).DisplayNode();
228 
229                 Command = String.Format("UPDATE LoginTable SET TotalPandL = '{0}'
    ↪ WHERE Email = '{1}';", newPandL, emails.GetNode(x).DisplayNode());
230                 sqlite_cmd.CommandText = Command;
231                 sqlite_cmd.ExecuteNonQuery();
232 
233                 Command = String.Format("UPDATE UserPositionsTable SET Value = '{0}',
    ↪ LastUpdate = '{1}' WHERE Email = '{2}' AND StockSymbol = '{3}';", newValue,
    ↪ sqlFormattedDate, emails.GetNode(x).DisplayNode(),
    ↪ boughtSymbols.GetNode(x).DisplayNode());
234                 sqlite_cmd.CommandText = Command;
235                 sqlite_cmd.ExecuteNonQuery();
236             }
237 
238             conn.Close();
239         }
240 
241         //returns a specified user's PandL
242         public double ReturnUserPandL(SQLiteConnection conn, string email)
243         {
244             SQLiteCommand sqlite_cmd;
245             SQLiteDataReader sqlite_datareader;
246             sqlite_cmd = conn.CreateCommand();
247             sqlite_cmd.CommandText = String.Format("SELECT TotalPandL FROM LoginTable
    ↪ WHERE EMAIL = '{0}';", email);
248 
249             double pandL = 0;
250             sqlite_datareader = sqlite_cmd.ExecuteReader();
251             try
252             {
253                 while (sqlite_datareader.Read())
254                 {
255                     pandL = sqlite_datareader.GetDouble(0);
256                 }
257             }
258             catch (Exception ex)
259             {
260                 pandL = 0;
261             }
262             sqlite_datareader.Close();
263             conn.Close();
264             return pandL;
265         }
266 
267         //updates stock table to include most recent prices
```

```
268         public void UpdateStockPrices(SQLiteConnection conn)
269         {
270             SQLiteCommand sqlite_cmd;
271             sqlite_cmd = conn.CreateCommand();
272             MATLABinteractions MATLABprices = new MATLABinteractions();
273             DatabaseInteractions databaseAccess = new DatabaseInteractions();
274             double price;
275 
276             List<string> companyCodes =
    ↪ databaseAccess.ReturnStockCodes(databaseAccess.CreateConnection());
277             int length = companyCodes.Count();
278 
279             for (int x = 0; x < length; x++)
280             {
281                 try
282                 {
283                     price = MATLABprices.Price(companyCodes[x]);
284                 }
285                 catch
286                 {
287                     price = -1;
288                 }
289                 string Command = String.Format("UPDATE StockCodesTable SET Price =
    ↪ '{0}' WHERE Symbol = '{1}';", price, companyCodes[x]);
290                 sqlite_cmd.CommandText = Command;
291                 sqlite_cmd.ExecuteNonQuery();
292             }
293 
294             conn.Close();
295         }
296 
297         //adds the details of a new user to the LoginTable table
298         public void InsertLoginData(SQLiteConnection conn, string email, string
    ↪ password)
299         {
300             SQLiteCommand sqlite_cmd;
301             sqlite_cmd = conn.CreateCommand();
302 
303             string Command = String.Format("INSERT INTO LoginTable (Email, Password,
    ↪ WatchStock1, WatchStock2, WatchStock3, TotalPandL) VALUES('{0}', '{1}',
    ↪ '{3}','{3}', '{3}','{2}');", email, password, 0, " ");
304 
305             sqlite_cmd.CommandText = Command;
306             sqlite_cmd.ExecuteNonQuery();
307 
308             conn.Close();
309         }
310 
311         //updates UserPositionsTable table after a user buys a stock
312         public void InsertBuy(SQLiteConnection conn, string email, string stockCode,
    ↪ double volume, double currentValue)
313         {
314             SQLiteCommand sqlite_cmd;
315             sqlite_cmd = conn.CreateCommand();
316             string Command = "";
317 
318             DateTime myDateTime = DateTime.Now;
319             string sqlFormattedDate = myDateTime.ToString("yyyy-MM-dd");
```

```
320
321             double newValue = currentValue + volume;
322             if (currentValue == 0)
323             {
324                 Command = String.Format("INSERT INTO UserPositionsTable (Email,
    ↪ StockSymbol, Value, LastUpdate) VALUES('{0}', '{1}', '{2}', '{3}');", email,
    ↪ stockCode, newValue, sqlFormattedDate);
325             }
326             else
327             {
328                 Command = String.Format("UPDATE UserPositionsTable SET Email = '{0}',
    ↪ StockSymbol = '{1}', Value = '{2}', LastUpdate = '{3}' WHERE Email = '{0}' AND
    ↪ StockSymbol = '{1}';",email, stockCode, newValue, sqlFormattedDate);
329             }
330
331             sqlite_cmd.CommandText = Command;
332             sqlite_cmd.ExecuteNonQuery();
333
334             conn.Close();
335         }
336
337         //updates UserPositionsTable table after a user sells a stock
338         public void InsertSell(SQLiteConnection conn, string email, string stockCode,
    ↪ double volume, double currentValue)
339         {
340             SQLiteCommand sqlite_cmd;
341             sqlite_cmd = conn.CreateCommand();
342
343             DateTime MyDateTime = DateTime.Now;
344             string sqlFormattedDate = MyDateTime.ToString("yyyy-MM-dd");
345
346             double newValue = currentValue - volume;
347
348             if (newValue < 0)
349             {
350                 newValue = 0;
351             }
352
353             string Command = String.Format("UPDATE UserPositionsTable SET Email =
    ↪ '{0}', StockSymbol = '{1}', Value = '{2}', LastUpdate = '{3}' WHERE Email = '{0}'
    ↪ AND StockSymbol = '{1}';", email, stockCode, newValue, sqlFormattedDate);
354
355             sqlite_cmd.CommandText = Command;
356             sqlite_cmd.ExecuteNonQuery();
357
358             conn.Close();
359         }
360
361         //finds the value of a user's position for a specified stock
362         public double ReturnValue(SQLiteConnection conn, string email, string
    ↪ StockCode)
363         {
364             SQLiteCommand sqlite_cmd;
365             SQLiteDataReader sqlite_datareader;
366             sqlite_cmd = conn.CreateCommand();
367             sqlite_cmd.CommandText = String.Format("SELECT Value FROM
    ↪ UserPositionsTable WHERE EMAIL = '{0}' AND StockSymbol = '{1}'", email,
    ↪ StockCode);
```

```
368
369             double value = 0;
370             sqlite_datareader = sqlite_cmd.ExecuteReader();
371             try
372             {
373                 while (sqlite_datareader.Read())
374                 {
375                     value = sqlite_datareader.GetDouble(0);
376                 }
377             }
378             catch (Exception ex)
379             {
380                 value = 0;
381             }
382             sqlite_datareader.Close();
383             conn.Close();
384             return value;
385         }
386
387         //returns most recent recorded stock price to be used on stock screen
388         public double ReturnPrice(SQLiteConnection conn, string stockCode)
389         {
390             SQLiteCommand sqlite_cmd;
391             SQLiteDataReader sqlite_datareader;
392             sqlite_cmd = conn.CreateCommand();
393             sqlite_cmd.CommandText = String.Format("SELECT Price FROM StockCodesTable
    ↪  WHERE Symbol = '{0}'", stockCode);
394
395             double price = 0;
396             sqlite_datareader = sqlite_cmd.ExecuteReader();
397             try
398             {
399                 while (sqlite_datareader.Read())
400                 {
401                     price = sqlite_datareader.GetDouble(0);
402                 }
403             }
404             catch (Exception ex)
405             {
406                 price = -1;
407             }
408             sqlite_datareader.Close();
409             conn.Close();
410             return price;
411         }
412
413         //compares entered details with those in the LoginTable table in order to
    ↪  allow or deny user access
414         public bool VerifyLoginData(SQLiteConnection conn, string email, string
    ↪  password)
415         {
416             bool validLogin = false;
417
418             SQLiteDataReader sqlite_datareader;
419             SQLiteCommand sqlite_cmd;
420             sqlite_cmd = conn.CreateCommand();
421             sqlite_cmd.CommandText = "SELECT Email, Password FROM LoginTable";
422
```

```
423             DoubleLinkList<string> emails = new DoubleLinkList<string>();
424             DoubleLinkList<string> passwords = new DoubleLinkList<string>();
425             int positionEmail = -1;
426             int positionPassword = -1;
427 
428             sqlite_datareader = sqlite_cmd.ExecuteReader();
429             while (sqlite_datareader.Read())
430             {
431                 emails.AddBack(sqlite_datareader.GetString(0));
432                 passwords.AddBack(sqlite_datareader.GetString(1));
433             }
434 
435             try
436             {
437                 positionEmail = emails.IndexOf(email);
438                 positionPassword = passwords.IndexOf(password);
439                 if (positionEmail == positionPassword && (positionEmail !=-1 ||
    ↪ positionPassword!=-1))
440                 {
441                     validLogin = true;
442                 }
443             }
444             catch (Exception ex)
445             {
446                 validLogin=false;
447             }
448             sqlite_datareader.Close();
449             conn.Close();
450 
451             return validLogin;
452         }
453 
454         //verifies that an email is not already in use
455         public bool CheckValidEmail(SQLiteConnection conn, string email)
456         {
457             bool validEmail = false;
458 
459             SQLiteDataReader sqlite_datareader;
460             SQLiteCommand sqlite_cmd;
461             sqlite_cmd = conn.CreateCommand();
462             sqlite_cmd.CommandText = "SELECT Email FROM LoginTable";
463 
464             DoubleLinkList<string> emails = new DoubleLinkList<string>();
465 
466             sqlite_datareader = sqlite_cmd.ExecuteReader();
467             while (sqlite_datareader.Read())
468             {
469                 emails.AddBack(sqlite_datareader.GetString(0));
470             }
471 
472             if (emails.Contains(email) == true)
473             {
474                 validEmail = false;
475             }
476             else
477             {
478                 validEmail=true;
479             }
```

```
480
481             sqlite_datareader.Close();
482             conn.Close();
483             return validEmail;
484         }
485
486         //edits the LoginTable table after a user updates their email
487         public void EditEmail(SQLiteConnection conn, string emailNew, string emailOld)
488         {
489             SQLiteCommand sqlite_cmd;
490             sqlite_cmd = conn.CreateCommand();
491
492             string command = String.Format("UPDATE LoginTable SET Email = '{0}' WHERE
    ↪  Email = '{1}';", emailNew, emailOld);
493
494             sqlite_cmd.CommandText = command;
495             sqlite_cmd.ExecuteNonQuery();
496
497             conn.Close();
498         }
499
500         //edits the LoginTable table after a user updates their password
501         public void EditPassword(SQLiteConnection conn, string password, string email)
502         {
503             SQLiteCommand sqlite_cmd;
504             sqlite_cmd = conn.CreateCommand();
505
506             string Command = String.Format("UPDATE LoginTable SET Password = '{0}'
    ↪  WHERE Email = '{1}';", password, email);
507
508             sqlite_cmd.CommandText = Command;
509             sqlite_cmd.ExecuteNonQuery();
510
511             conn.Close();
512         }
513
514         //searches for any matches of stock to those entered by user
515         public string StockSearch(SQLiteConnection conn, string stockToFind)
516         {
517             string stockCode = "";
518             DoubleLinkList<string> stockCodes = new DoubleLinkList<string>();
519
520             SQLiteDataReader sqlite_datareader;
521             SQLiteCommand sqlite_cmd;
522             sqlite_cmd = conn.CreateCommand();
523             sqlite_cmd.CommandText = String.Format("SELECT Symbol FROM StockCodesTable
    ↪  WHERE Name LIKE '{0}%' OR Symbol LIKE '{0}%'", stockToFind);
524             sqlite_datareader = sqlite_cmd.ExecuteReader();
525
526             while (sqlite_datareader.Read())
527             {
528                 stockCodes.AddBack(sqlite_datareader.GetString(0));
529             }
530
531             try
532             {
533                 stockCode = stockCodes.GetNode(0).DisplayNode();
534             }
```

```
535             catch (Exception e)
536             {
537                 stockCode = "";
538             }
539             sqlite_datareader.Close();
540             conn.Close();
541 
542             return stockCode;
543         }
544 
545         //converts from the code of a stock to the name of the company
546         public string GetNameFromCode(SQLiteConnection conn, string code)
547         {
548             SQLiteDataReader sqlite_datareader;
549             SQLiteCommand sqlite_cmd;
550             sqlite_cmd = conn.CreateCommand();
551 
552             sqlite_cmd.CommandText = String.Format("SELECT Name FROM StockCodesTable
    ↪  WHERE Symbol = '{0}';", code);
553             sqlite_datareader = sqlite_cmd.ExecuteReader();
554 
555             DoubleLinkList <string> names = new DoubleLinkList<string>();
556             string stockName = "";
557 
558 
559             while (sqlite_datareader.Read())
560             {
561                 names.AddBack(sqlite_datareader.GetString(0));
562             }
563 
564             try
565             {
566                 stockName = names.GetNode(0).DisplayNode();
567             }
568             catch (Exception e)
569             {
570                 stockName = "";
571             }
572             sqlite_datareader.Close();
573             conn.Close();
574 
575             return stockName;
576         }
577 
578         //converts from the name of a company to its stock code
579         public string GetCodeFromName(SQLiteConnection conn, string name)
580         {
581             SQLiteDataReader sqlite_datareader;
582             SQLiteCommand sqlite_cmd;
583             sqlite_cmd = conn.CreateCommand();
584 
585             sqlite_cmd.CommandText = String.Format("SELECT Symbol FROM StockCodesTable
    ↪  WHERE Name = '{0}';", name);
586             sqlite_datareader = sqlite_cmd.ExecuteReader();
587 
588             DoubleLinkList<string> codes = new DoubleLinkList<string>();
589             string stockCode = "";
590 
```

```
591
592             while (sqlite_datareader.Read())
593             {
594                 codes.AddBack(sqlite_datareader.GetString(0));
595             }
596
597             try
598             {
599                 stockCode = codes.GetNode(0).DisplayNode();
600             }
601             catch (Exception e)
602             {
603                 stockCode = "";
604             }
605             sqlite_datareader.Close();
606             conn.Close();
607
608             return stockCode;
609         }
610
611         //returns the 3 most valuable positions currently held by a user
612         public string[] ReturnUserTopStocks(SQLiteConnection conn, string email)
613         {
614             string[] stockSymbols = new string[3];
615             DoubleLinkList<string> stockCodes = new DoubleLinkList<string>();
616
617             SQLiteDataReader sqlite_datareader;
618             SQLiteCommand sqlite_cmd;
619             sqlite_cmd = conn.CreateCommand();
620             sqlite_cmd.CommandText = String.Format("SELECT StockSymbol FROM
    ↪  UserPositionsTable WHERE Email = '{0}' ORDER BY VALUE DESC", email);
621             sqlite_datareader = sqlite_cmd.ExecuteReader();
622
623             while (sqlite_datareader.Read())
624             {
625                 stockCodes.AddBack(sqlite_datareader.GetString(0));
626             }
627
628             try
629             {
630                 stockSymbols[0] = stockCodes.GetNode(0).DisplayNode();
631                 try
632                 {
633                     stockSymbols[1] = stockCodes.GetNode(1).DisplayNode();
634                     try
635                     {
636                         stockSymbols[2] = stockCodes.GetNode(2).DisplayNode();
637                     }
638                     catch { }
639                 }
640                 catch { }
641             }
642             catch (Exception e)
643             {
644             }
645             sqlite_datareader.Close();
646             conn.Close();
647
```

```
            return stockSymbols;
        }

        //adds a stock to the users watchlist
        public string AddWatchlist(SQLiteConnection conn, string stockCode, string email)
        {
            SQLiteDataReader sqlite_datareader;
            SQLiteCommand sqlite_cmd;

            DoubleLinkList<string> stockCodes = new DoubleLinkList<string>();

            sqlite_cmd = conn.CreateCommand();
            sqlite_cmd.CommandText = String.Format("SELECT WatchStock1 FROM LoginTable WHERE Email = '{0}';", email);
            sqlite_datareader = sqlite_cmd.ExecuteReader();
            try
            {
                while (sqlite_datareader.Read())
                {
                    stockCodes.AddBack(sqlite_datareader.GetString(0));
                }
                if (stockCodes.GetNode(0).DisplayNode() == " " || stockCodes.GetNode(0).DisplayNode() == "" || stockCodes.GetNode(0).DisplayNode() == null)
                {
                    sqlite_cmd = conn.CreateCommand();
                    sqlite_cmd.CommandText = String.Format("UPDATE LoginTable SET WatchStock1 = '{0}' WHERE Email = '{1}';", stockCode, email);
                    sqlite_datareader = sqlite_cmd.ExecuteReader();
                    sqlite_datareader.Close();
                    conn.Close();
                    return "WatchStock1";
                }
                else
                {
                    sqlite_cmd = conn.CreateCommand();
                    sqlite_cmd.CommandText = String.Format("SELECT WatchStock2 FROM LoginTable WHERE Email = '{0}';", email);
                    sqlite_datareader = sqlite_cmd.ExecuteReader();

                    try
                    {
                        while (sqlite_datareader.Read())
                        {
                            stockCodes.AddBack(sqlite_datareader.GetString(0));
                        }

                        if (stockCodes.GetNode(1).DisplayNode() == " " || stockCodes.GetNode(1).DisplayNode() == "" || stockCodes.GetNode(1).DisplayNode() == null)
                        {
                            sqlite_cmd = conn.CreateCommand();
                            sqlite_cmd.CommandText = String.Format("UPDATE LoginTable SET WatchStock2 = '{0}' WHERE Email = '{1}';", stockCode, email);
                            sqlite_datareader = sqlite_cmd.ExecuteReader();
                            sqlite_datareader.Close();
                            conn.Close();
```

```
697                             return "WatchStock2";
698                         }
699                         else
700                         {
701                             sqlite_cmd = conn.CreateCommand();
702                             sqlite_cmd.CommandText = String.Format("SELECT WatchStock3
    ↪ FROM LoginTable WHERE Email = '{0}';", email);
703                             sqlite_datareader = sqlite_cmd.ExecuteReader();
704 
705                             try
706                             {
707                                 while (sqlite_datareader.Read())
708                                 {
709 
    ↪ stockCodes.AddBack(sqlite_datareader.GetString(0));
710                                 }
711                                 if (stockCodes.GetNode(2).DisplayNode() == " " ||
    ↪ stockCodes.GetNode(2).DisplayNode() == "" || stockCodes.GetNode(2).DisplayNode()
    ↪ == null)
712                                 {
713                                     sqlite_cmd = conn.CreateCommand();
714                                     sqlite_cmd.CommandText = String.Format("UPDATE
    ↪ LoginTable SET WatchStock3 = '{0}' WHERE Email = '{1}';", stockCode, email);
715                                     sqlite_datareader = sqlite_cmd.ExecuteReader();
716                                     sqlite_datareader.Close();
717                                     conn.Close();
718                                     return "WatchStock3";
719                                 }
720                                 else
721                                 {
722                                     return "Watchlist Full";
723                                 }
724                             }
725                             catch
726                             {
727                                 return "Watchlist Full";
728                             }
729                         }
730                     }
731                     catch
732                     {
733                         return "Watchlist Full";
734                     }
735                 }
736             }
737             catch (Exception e)
738             {
739                 return "Watchlist Full";
740             }
741         }
742 
743         //returns a users watchlist
744         public string[] ReturnUserWatchlist(SQLiteConnection conn, string email)
745         {
746             string[] codes = new string[3];
747 
748             SQLiteDataReader sqlite_datareader;
749             SQLiteCommand sqlite_cmd;
```

```
750             sqlite_cmd = conn.CreateCommand();
751             sqlite_cmd.CommandText = String.Format("SELECT WatchStock1 FROM LoginTable
    ↪  WHERE Email = '{0}';", email);
752             sqlite_datareader = sqlite_cmd.ExecuteReader();
753 
754             while (sqlite_datareader.Read())
755             {
756                 codes[0] = sqlite_datareader.GetString(0);
757             }
758 
759             sqlite_cmd = conn.CreateCommand();
760             sqlite_cmd.CommandText = String.Format("SELECT WatchStock2 FROM LoginTable
    ↪  WHERE Email = '{0}';", email);
761             sqlite_datareader = sqlite_cmd.ExecuteReader();
762 
763             while (sqlite_datareader.Read())
764             {
765                 codes[1] = sqlite_datareader.GetString(0);
766             }
767 
768             sqlite_cmd = conn.CreateCommand();
769             sqlite_cmd.CommandText = String.Format("SELECT WatchStock3 FROM LoginTable
    ↪  WHERE Email = '{0}';", email);
770             sqlite_datareader = sqlite_cmd.ExecuteReader();
771 
772             while (sqlite_datareader.Read())
773             {
774                 codes[2] = sqlite_datareader.GetString(0);
775             }
776 
777             sqlite_datareader.Close();
778             conn.Close();
779             return codes;
780         }
781 
782         //returns all the stock codes
783         public List<string> ReturnStockCodes(SQLiteConnection conn)
784         {
785             List<string> stockCodes = new List<string>();
786 
787             SQLiteDataReader sqlite_datareader;
788             SQLiteCommand sqlite_cmd;
789             sqlite_cmd = conn.CreateCommand();
790             sqlite_cmd.CommandText = String.Format("SELECT Symbol FROM
    ↪  StockCodesTable;");
791             sqlite_datareader = sqlite_cmd.ExecuteReader();
792 
793             while (sqlite_datareader.Read())
794             {
795                stockCodes.Add(sqlite_datareader.GetString(0));
796             }
797             sqlite_datareader.Close();
798             conn.Close();
799 
800             return stockCodes;
801         }
802 
803         //returns all the names of companies and their corresponding stock codes
```

```
804         static string[,] LoadStockCode()
805         {
806             List<string> StockCodes = new List<string>();
807             List<string> StockNames = new List<string>();
808             string filePath =
809 
    ↪  @"C:\Users\olesc\OneDrive\Documents\MATLAB\MATLABData\CondensedStockCodes.csv";
810             StreamReader reader = null;
811             int counter = 0;
812 
813             if (File.Exists(filePath))
814             {
815                 reader = new StreamReader(File.OpenRead(filePath));
816 
817                 while (!reader.EndOfStream)
818                 {
819                     var line = reader.ReadLine();
820                     var values = line.Split(',');
821                     foreach (var item in values)
822                     {
823                         if (counter%2 == 0)
824                         {
825                             StockCodes.Add(item);
826                         }
827                         else
828                         {
829                             StockNames.Add(item);
830                         }
831                         counter++;
832                     }
833                 }
834             }
835             else
836             {
837                 Console.WriteLine("File doesn't exist");
838             }
839 
840             counter = 0;
841             string[,] codesAndNames = new string[2,StockCodes.Count];
842 
843             foreach (var item in StockCodes)
844             {
845                 codesAndNames[0, counter] = StockCodes[counter];
846                 codesAndNames[1,counter] = StockNames[counter];
847                 counter++;
848             }
849 
850             return codesAndNames;
851         }
852 
853 
854     }
855 }
```

4.2.9 Notifications class

```
1 using System;
2 using System.Collections.Generic;
```

```
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Controls;

namespace MATLABintegrationTest
{
    internal class Notifications
    {
        //returns any sell calls for owned stock, and any sell or buy calls for watchlist stocks
        public string[,] ReturnStockNotifications(string email)
        {
            string[,] notifications = new string[4, 2];

            DatabaseInteractions database = new DatabaseInteractions();
            MATLABinteractions MATLABloader = new MATLABinteractions();

            string[] ownedStockCodes = database.ReturnUserStocks(database.CreateConnection(), email);
            DoubleLinkList<string> ownedStockActions = new DoubleLinkList<string>();

            string[] watchlistStockCodes = database.ReturnUserWatchlist(database.CreateConnection(), email);
            DoubleLinkList<string> watchlistStockActions = new DoubleLinkList<string>();

            for (int x = 0; x < ownedStockCodes.Length; x++)
            {
                string entryExitOwned = MATLABloader.MACDCalculator(ownedStockCodes[x]);
                if (entryExitOwned == "Sell")
                {

                    ownedStockActions.AddBack(database.GetNameFromCode(database.CreateConnection(), ownedStockCodes[x]));
                    ownedStockActions.AddBack(entryExitOwned);
                }
            }


            for (int x = 0; x < watchlistStockCodes.Length; x++)
            {
                if (watchlistStockCodes[x] != " ")
                {
                    string entryExitWatchlist = MATLABloader.MACDCalculator(watchlistStockCodes[x]);
                    if ((entryExitWatchlist == "Sell" || entryExitWatchlist == "Buy") && ownedStockCodes.Contains(watchlistStockCodes[x]) == false)
                    {

                        watchlistStockActions.AddBack(database.GetNameFromCode(database.CreateConnection(), watchlistStockCodes[x]));
                        watchlistStockActions.AddBack(entryExitWatchlist);
                    }
                }

```

```
            }


            int numOwnedNotifications = ((ownedStockActions.GetLength())/2)-1;
            int numWatchlistNotifications = ((ownedStockActions.GetLength()) / 2) - 1;
            if (numOwnedNotifications > 3)
            {
                numOwnedNotifications= 3;
            }

            while (numOwnedNotifications >= 0)
            {
                notifications[numOwnedNotifications, 0] =
↪   ownedStockActions.GetNode((numOwnedNotifications)*2).DisplayNode();
                notifications[numOwnedNotifications, 1] =
↪   ownedStockActions.GetNode((numOwnedNotifications)*2+1).DisplayNode();
                numOwnedNotifications--;
            }
            int counter = 0;
            while (notifications.Length < 4 && numWatchlistNotifications>0)
            {
                notifications[counter, 0] = watchlistStockActions.GetNode((counter) *
↪   2).DisplayNode();
                notifications[counter, 1] = watchlistStockActions.GetNode((counter) *
↪   2 + 1).DisplayNode();
                counter++;
                numWatchlistNotifications--;
            }

            return notifications;
        }
    }
}
```

4.2.10 Custom linked list class

```
using System;
using System.CodeDom;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MATLABintegrationTest
{
    //custom generic linked list defined with data type T
    internal class DoubleLinkList<T> where T : IComparable
    {
        //defining head and temporary nodes for use in list
        private Node<T> head;
        private Node<T> temp;
        //variable that keeps track of length of list
        private int length = 0;

        //constructor making empty list
        public DoubleLinkList()
        {
            temp = head;
```

```
23          }
24
25          //methods
26
27          //adds element to front of list
28          public void AddFront(T inp)
29          {
30              Node<T> a = new Node<T>(inp);
31              a.next = head;
32              a.previous = null;
33              head = a;
34              temp = head;
35
36              length++;
37          }
38
39          //adds element to specified position in list
40          public void AddMid(T inp, int pos)
41          {
42              int count = 1;
43              Node<T> node = new Node<T>(inp);
44              if (pos == 1)
45              {
46                  AddFront(inp);
47              }
48              else if (pos == 2)
49              {
50                  node.next = head.next;
51                  node.previous = head;
52                  head.next = node;
53                  node.next.previous = node;
54              }
55              else
56              {
57                  while (count < pos)
58                  {
59                      temp = temp.next;
60                      count++;
61                  }
62                  node.next = temp;
63                  node.previous = temp.previous;
64                  temp.previous.next = node;
65                  temp.previous = node;
66                  temp = head;
67              }
68
69              length++;
70
71          }
72
73          //adds element to the back of the list
74          public void AddBack(T inp)
75          {
76              Node<T> newnode = new Node<T>(inp);
77              if (temp != null)
78              {
79                  while (temp.next != null)
80                  {
```

```
81                      temp = temp.next;
82                  }
83                  temp.next = newnode;
84                  newnode.previous = temp;
85                  newnode.next = null;
86                  temp = head;
87              }
88              else
89              {
90                  head = newnode;
91                  temp = head;
92              }
93
94              length++;
95
96          }
97
98          //removes element at specified position in list
99          public Node<T> RemoveAt(int pos)
100         {
101             int x = 1;
102             Node<T> temp1 = head;
103             while (x < pos)
104             {
105                 temp1 = temp1.next;
106                 x++;
107             }
108             temp1.previous.next = temp1.next;
109             temp1.next.previous = temp1.previous;
110
111             length--;
112
113             return temp1;
114
115         }
116
117         //returns node at desired position
118         public Node<T> GetNode(int pos)
119         {
120             int x = 0;
121             Node<T> temp1 = head;
122             while (x < pos)
123             {
124                 temp1 = temp1.next;
125                 x++;
126             }
127             return temp1;
128         }
129
130         //returns length of list
131         public int GetLength()
132         {
133             return length;
134         }
135
136         //returns boolean indicating whether or not the passed element is in the list
137         public bool Contains(T element)
138         {
```

```
            bool contains = false;

            while (temp.next != null && contains == false)
            {
                if (element.CompareTo(temp.DisplayNode()) == 0)
                {
                    contains = true;
                }
                temp = temp.next;
            }

            temp = head;
            return contains;
        }

        //returns the index of the passed element in the list
        public int IndexOf(T element)
        {
            int index = -1;
            int counter = 0;

            while (temp.next != null && index == -1)
            {
                if (element.CompareTo(temp.DisplayNode()) == 0)
                {
                    index = counter;
                }
                temp = temp.next;
                counter++;
            }

            temp = head;
            return index;
        }
    }
}
```

4.2.11 Node class for lists

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MATLABintegrationTest
{
    internal class Node<T> where T : IComparable
    {
        public T data;
        public Node<T> next;
        public Node<T> previous;

        public Node(T inp)
        {
            data = inp;
        }

```

```
20          public T DisplayNode()
21          {
22              return data;
23          }
24      }
25  }
```

5 Testing

5.1 Testing Strategy

The testing of the solution needs to be systematic and organised in order to ensure it is complete and matches the outlined requirements. This will be done by creating a custom set of test cases, which cover all of the requirements, in order to ensure that all requirements are indeed met. The types of tests are broken down into 3 categories:

- Normal tests - that test the expected functions of the program
- Boundary tests - that test allowed but borderline conditions of the program. E.g. buying a stock for the first time
- Erroneous tests - that test invalid inputs into the program

Following this I created the below test plan. The rows are colour coded according to which screen in the code they are tied to, in this order:

1. Login/welcome screen
2. Home screen
3. Stock screen
4. Account screen

5.2 Test Plan

Test ID	Purpose	Requirement(s)	Testing Method	Type	Expected Outcome
1	Check user can switch between login and create account pages	2.1.a	Click menu buttons	Normal	When click on 'create account' see create account screen, when click on 'back to login' see login screen
2	Stop user from logging in with incorrect details	2.5.2.b, 2.4a	Attempt incorrect and correct passwords	Erroneous	With incorrect password/email error message shown, with correct password/email takes user to main page
3	Stop user from creating an account with an email that is already in use	2.5.2c	Enter an email which is already in use into the email box on the create account screen	Erroneous	Doesn't save the entered details into database and displays error message
4	Display icons on the home page showing the stock price over the last month	2.5.1.a	Visual on home screen	Normal	Different charts shown above each stock name, representing their price trends
5	Display user's top 3 stocks, and overall 3 most bought stocks on home page	2.5.1.a, 2.4.c, 2.1.e, 2.5.2.d	Visually displayed on home page, click on all user's top stocks to show their decreasing value	Normal/ Boundary	3 stocks shown on top row of home page, and a maximum of three shown on the bottom, depending on if user does or doesn't own 3 stocks

6	View any buy/sell notifications in notifications pop-down	2.1.i, 2.3.a, 2.4.c	Click on notifications pop-down	Normal	4 different buttons appear on screen, if less than 4 notifications found the spare buttons should be disabled
7	Open stock page from any stock button in main window/notifications pop-down	2.1.f	Click on stock from each page and ensure correct corresponding pages open	Normal	Opens the generic stock page, but with the information of the stock that was clicked on
8	Search for any stock saved in database	2.5.2.e, 2.4.b	Type into search bar and view response	Normal	Either a button which leads to the stock page of the searched stock is shown, or an error message that no such stock was found is shown
9	Open account page from home screen	2.1.g	Click account button on home screen	Normal	New screen should open, displaying all relevant account details
10	Open correct stock after clicking	2.1.f, 2.4.b, 2.5.2.a	Open stock page	Normal	Title of stock page should match the stock which was clicked
11	Display large stock chart on stock screen	2.1.e, 2.5.1.a	Visual on stock screen	Normal	Large chart shown on left hand side of stock screen, which matches the small chart shown before clicking on the stock
12	Display current stock price	2.1.f, 2.5.1.b	Visual on stock screen	Normal	Price of the stock in dollars to two decimal places shown on stock screen, can compare the price with the prices on chart to verify it is correct
13	Display the current value of a user's position of that stock	2.1.c, 2.4.c	Compare the value displayed on the stock page to the value saved in the database	Normal	The value shown on the stock screen and saved in the database should match, to 2 decimal places
14	Display current prediction for the displayed stock	2.1.f.ii, 2.3.b, 2.5.1.d	Visual on stock screen	Normal	Either hold, sell or buy displayed on the stock screen. This prediction should be made by my algorithm, or if not possible by the MACD calculator
15	Check clear buy and sell pop-downs	2.1.f.i, 2.1.j, 2.5.2.f	Click on buy and sell buttons on stock screen	Normal	Bright buy and sell buttons should have a pop-down that requests for buy/sell amount and confirmation of order after being clicked
16	Buy or sell a stock	2.5.2.f, 2.1.c, 2.4.c	Type in value into buy and sell pop-down and confirm order	Normal/Bo	Writes the order to the database, and displays the user's new position on the stock page
17	Test invalid buy/sell inputs	2.5.2.f.ii, 2.4.c	Enter letters, and sell orders greater than the user's position into the buy/sell pop-downs	Erroneous	If a letter is entered it should be ignored, and if a sell order greater than the user's position is entered it should set their position to zero
18	Add stock to watchlist	2.1.d, 2.4.a.ii	Click on watchlist button on stock page	Boundary	If the watchlist is full, an error message should be displayed, if it is not full then it should be written to the database

19	Display user's overall PandL on account page	2.1.c, 2.4.a.ii, 2.5.2.g	Visual on account screen	Normal	The total PandL of the user in dollars to 2 decimal places should be shown. It should match with the database, or differ slightly due to the update which occurs when loading the program
20	Display the user's 3 top positions and watchlist stocks on the account page	2.1.h	Visual on account screen, click on stock names to see if they link correctly	Normal	Stock screen for watchlist stocks and top position stocks should be displayed when clicked. The three top stock should match those on the home screen
21	Edit user's email from account page	2.1.g, 2.4.a	Click on edit email button and type new email	Normal	The email should update on the account screen, and also be changed in the database
22	Block invalid edit of user's email from account page	2.1.g, 2.4.a, 2.5.2.c	Click on edit email button and type email which is already used in another account	Erroneous	The email error message should be displayed on the account page, and the email should not be updated both in the account page and database
23	Edit user's password from account page	2.1.g, 2.4.a	Click on edit password button and type new email	Normal	The password should update on the account screen, and also be changed in the database
24	Update the stock data used for the learning algorithm	2.5.1.c	Click on update data button on account page	Normal	The data in the csv containing all available data for the algorithm should update with the prices since the last update
25	Update the predictions made by the learning algorithm	All of 2.2, 2.5.1.e-g	Click on update re-run prediction algorithm button on account page	Normal	The data in the csv containing all predictor variables from the algorithm should update with the new predictor variables
26	Test logout button	2.5.2.h	Click on logout button	Normal	The login page should be displayed, and upon next login none of the old user's information should be displayed
27	Test navigation buttons	2.1.k	Click on back button on each page, each pop-down close and exit program button	Normal	Each button should carry out their respective purpose, going back to the home page, closing its pop-down or closing the entire program

6 Evaluation

6.1 Testing results

The video containing my test results can be found using the following link:
https://youtu.be/4ufk2vywvaA

Note: There are some cuts in the video, particularly after reloading data and running the learning algorithm. This is due to the fact that these took ¿30 seconds to run because of the large volume of data they were processing. No other processes occurred during this time and so the video was cut.

Below is the table containing the results for each test case, as well as time stamps where each is displayed in the video:

Test ID	Outcome	Timestamp	Requirement(s) Met	Further Comments
1	Passed	1:10	Yes	
2	Passed	1:23	Yes	The email field is case sensitive in my program therefore it suffices to just change the one letter. Identical verification is performed for password field
3	Passed	1:14	Yes	The email used in the demonstration can be seen when looking at the database initially
4	Passed	1:47	Yes	
5	Passed	2:00 onwards	Yes	The three values are shown to be in descending order
6	Passed	1:55	Yes	
7	Passed	2:00 onwards	Yes	This is repeatedly evidenced after this time point
8	Passed	3:42	Yes	The returned button only shows the code of the stock, but when searching both code and name give return data
9	Passed	3:09	Yes	
10	Passed	2:00 onwards	Yes	This is repeatedly evidenced after this time point
11	Passed	2:00 onwards	Yes	This is repeatedly evidenced after this time point. The graphs are of slightly stretched on the larger scale so may appear slightly different, but by comparing points you can see they are indeed correct
12	Passed	2:00 onwards	Yes	This is repeatedly evidenced after this time point
13	Passed	2:00 onwards	Yes	This is repeatedly evidenced after this time point
14	Passed	2:00 onwards	Yes	This is repeatedly evidenced after this time point
15	Passed	2:20	Yes	Need to confirm with external client that colour scheme requirement is met
16	Passed	2:20	Yes	All three cases, sell, buy and new buy can be seen as written to database at the end of the video
17	Passed	2:20	Yes	Correct responses to both types of invalid input, non-numerical and too great a numerical
18	Passed	3:03 and 4:33	Yes	At the timestamps first an invalid and then valid add is shown
19	Passed	3:13	Yes	
20	Passed	3:18	Yes	
21	Passed	4:47	Yes	Updates can be seen in the database at the end of the video
22	Passed	4:47	Yes	
23	Passed	4:47	Yes	Updates can be seen in the database at the end of the video

24	Passed	3:55	Yes	The updated data is evidenced by the difference between the data in the file at the start and end of the video
25	Passed	4:00	Yes	Again, there is a clear difference in data in the polynomial file from the start to the end of the video, due to the button click and running of the algorithm
26	Passed	4:03	Yes	Evidence of the data being cleared after logging out is shown after the account is created and the user is back on the main screen
27	Passed	1:58 - 5:14	Yes	Evidenced numerous times in the time interval, when swapping between various screens

6.2 Analysis of results

From the above table we can see that each requirement was successfully met. The only one that is still in need of clarification is 2.1.j, which will be assessed with my end-user as it is not directly measurable.

The majority of the requirements were clear and measurable, as evidenced by the video. However, requirements in sub-section 2.2 are to do with the workings of the learning algorithm, and so cannot be visualised as easily in the video. Although they were shown to give a measurable output, in the form of the csv 'Poly3Found' used for making predictions. One way I could think to individually test each sub-requirement and ensure the algorithm was working as desired would be to step though the code and view the changing of the various parameters as I stepped through. This is not a very sensible method as it is laborious and not easy to follow for someone needing to verify the requirements. The other way in which I thought to more explicitly demonstrate meeting these requirements was to plot graphs of the fitted polynomials, and their respective prediction distributions. However, with so many predicted polynomials the graph became quickly too messy, and this still would miss out all of the training and testing split requirements and more.

6.3 Feedback Interview with End-User

During my research I interviewed Taras Chaban in order to gain insight into what a set of suitable requirements for my project would look like. It therefore seems suitable to interview Taras again in order to determine how well the solution met his demands.

Interviewer: Oles Chaban

Interviewee: Taras Chaban - former professional stock trader

Oles: Overall, does the program live up to what you would expect from a trading platform?

Taras: Vastly yes. There were some minor inconveniences with placements of text/text boxes, but these were only minor, mainly aesthetic inconveniences. The actual backbone of the program performed what was expected, as expected.

Analysis: The overall problem has been effectively tackled, and the user is pleased with the general implementation of the solution. This suggests that the set of requirements was complete and well defined.

Oles: What are your thoughts on the account system as a whole?

Taras: It was definitely a sensible implementation of an account system. It was defensive in not allowing any repetitions of user and secure enough for any sensible user with the password system. Splitting this sort of software with a clear account system is a must.

I was however slightly confused with the email system, most emails are not case sensitive, and yet your program was. Similarly there were no restrictions on the characters a user could enter to make it pass as an email. Perhaps usernames would have been more appropriate?

Analysis: The accounts were a necessary and functional addition. However there is room for improvement due to lose definitions around the primary key of the user.

Oles: What did you think of the predictions made by my algorithm?

Taras: Surprisingly I found the predictions to be accurate. When clicking through some stocks and looking at the predictions given for each they largely lined up with what my thinking was for that stock. Looking at the returns on your testing account further gives me the impression that it was not just luck that the stocks I looked at were sensibly predicted.

Analysis: This is very positive feedback on the algorithm, which helps to justify that all the requirements for it were aptly met and it exceeded expectations for its returns. Looking at the testing video it can be seen that on my training account there was a return of around 1.2 million dollars in the time spent testing, which I estimate to be roughly a 12% return per month given how much was invested.

Oles: Finally, what are your thoughts on the UI, in particular the colour scheme? Would you make any changes?

Taras: The UI does not bombard you with information like a lot of other trading platforms, which is most definitely a positive. It doesn't give the impression that it wants to rush you into trades to make money from commission like most platforms. The colour scheme was also simple and effective, every different action was clearly distinguishable.

Analysis: The UI is designed well and meets the overall requirements of the user. The subjective requirement regarding the colour scheme can now be confidently thought of as met.

6.4 Potential Improvements

6.4.1 Improve Account system:

From my user interview after my testing they expressed the fact that they would have liked some more work on the account system.

This could be done either by:

1. Adjusting the way in which emails are accepted/rejected
    - This could be done by making them no longer case sensitive, but also adding more strict rules as to what symbols have to be present and in what sequence. E.g. an @ followed by a .com or .co.uk
    - This can be added by adding clauses onto the SQl queries dealing with the account emails
2. Changing the email system to a username system
    - This requires no changes to the SQl, however may be unhelpful if when improving the software you want to start contacting users

6.4.2 Shorting Stocks

Another feature common in many trading platforms is the ability to short stocks. This is when you bet on the value of a stock decreasing. A user first borrows the shares from someone, and pays them some premium. They then sell the shares for the higher value and wait for the price to decrease. At the lower price they buy back the same number of shares and return them to the original owner, making a profit along the way.

This is difficult as it requires some collaboration between users, which is difficult to implement on one device. A possible implementation would to be to display a notification to each user that owns the stock that someone would like to short it. However, with this there will always be a delay between the request for a short and the time at which it actually occurs, which is not acceptable for a stock platform. Automatic shorting would lie considerably outside the scope of difficulty of this project.

6.4.3 Graph Functionality

One common feature of trading platforms that utilise technical analysis is the functionality which they provide around the stock graphs. This includes things such as adding pitchforks, support lines and other technical metrics to the graphs of each stock.

The graphing tool which I used in WPF was static, and required a programmatic refresh every time new data was added onto the plot.

Possible solutions to this are:

1. Add binding to the graphs in WPF and have them refresh on a timer
    - This however still does not display the items added by the user instantaneously, and may cause lag on other parts of the program due to the frequent refreshing
2. Use an entirely new language to implement the UI
    - When choosing a language for the UI this time it can specifically be chosen to have good customisation of graphs

6.4.4 Trading Bot

Another tool used by many algorithmic trading firms is a trading bot, which automatically makes trades based off the predictions made in its algorithm. The advantage of this is that trades are made instantaneously when there is a change in signal, which means that users can gain extra profit between the times they are online.

The difficulty with implementing this into my current program is that I would need to use the quantitative outputs from my prediction algorithm. The buy or sell calls produces may be accurate, however predicting more precise numeric values for a bot to follow may have large uncertainties, and therefore produce loses without any user interference.

Another difficulty would be that the bot would need to be running around the clock. Therefore this implementation may be more suitable to add later down the line, when the program has a server.

6.4.5 Broaden Algorithm Training

Currently the algorithm is training on only one type of polynomial, with a fairly restricted dataset. The predictions of the algorithm are likely to improve with more data, and a larger scope of polynomials or other functions to find trends for.

Initially I was operating using a larger dataset, however the API stops working after a certain amount of calls are made in a given time. A solution to this would be buying a connection to some stock exchange, as it would provide a much larger variety of data and without any limit on calls. However, these are extremely expensive so not sensible for such a small scale project.

Increasing the scope of the training is fairly simple to implement, just adding another fit function to each loop and saving it to another file. However, there is always a risk of over-fitting, and especially with my more limited dataset I decided against greatly increasing the scope.

6.4.6 Project Summary

Overall I believe that my project is able to successfully carry out its desired function, and has met, or in some cases exceeded, the requirements outlined.

Creating a complete solution all rests upon a comprehensive analysis, in the form of user interviews, user feedback and current system analysis. I think that a major positive of my analysis was my continued back and forth between my potential user and myself. This gave me an agile approach which ensured that all of my requirements were well targeted. To improve my UI I could have asked my user to feedback on requirements after I made them before implementing. In order to further cement the connection between user desire and project output.

The next stage of the project was my documented design. By separating this section into the three levels of design, it allowed me to view my project from all levels of abstraction, and therefore gain a more

complete understanding of what my solution should look like. Utilising an object oriented design, COM server and complex data structures such as lists allowed for the larger problem to be distributed across a number of smaller, more manageable problems, ultimately making or a more manageable solution. In order to improve my design I could have explored more into my first level of design, by creating multiple flow charts, or going into more details with the one I did create. That being said my flowchart was still able to provide an abstracted view of the problem suitable for the first level of design.

My technical solution was written in a consistent style and managed to satisfy every project requirement. Using the objects outlined in my design, and decomposing my functions in MATLAB allowed me to remove significant complexity from my solution. This use of consistent style, good practices and loosely coupled modules also means that the code is maintainable and easy to refactor for future developers. In order to improve my technical solution I could implement any of the afore mentioned things. In particular the improvements to the account system could extend as far as making companies within my program with their own hierarchies of accounts. Implementing this could then allow me to leverage further object oriented techniques such as polymorphism and inheritance.

The testing section was similarly well structured and successful. The improvement in this section would be to do with the more rigorous testing of my algorithm. Some specific programmatic tests, such as sensitivity analysis, could be designed in order to probe it. A more visual response could be to create a sort of adaptive graph, that shows how a specific polynomial adapts as more training data is added to edit it.

To summarise, the problem which I desired to solve was that of producing a trading platform with a prediction algorithm unbiased to any human factors or ideals, targeted towards any skill of trader. Upon completion of the project it is clear that an effective solution to the problem has been given

7 References

[1] B2Broker. *Launch of New Trading Platform*. URL: https://www.businesswire.com/news/home/20201118005593/en/B2Broker-Launches-B2Margin-White-Label-Margin-Exchange-Trading-Platform.

[2] Artem Lensky. *Yahoo Finance API*. URL: https://uk.mathworks.com/matlabcentral/fileexchange/68361-yahoo-finance-and-quandl-data-downloader.