Design and Implementation of a High Level Stock Analytics Terminal

Justin Yeap
Personal Research
Singapore, **Singapore**

December 3, 2018

“Life is like riding a bicycle. To keep your balance, you must keep moving.” - *Albert Einstein*

Contents

1 Introduction/Background [Purpose]

Quantitative Trading or Algorithmic Trading is trading stocks, options or futures that decide using buy or sell signals from programs and algorithms that have been programmed beforehand by Quants or Traders. (Chan, 2008) Such Algorithms can refer to both simple algorithms like the relative average price and complex ones like Goldman Sachs Bull Bear Indicator (this tells Quants the general condition of he market if it's in a downward or upward trend.) (Goldman Sachs, 2018)

1.1 Entry to Market

Quants refer to individuals who work in financial firms such as Banks/Hedge Funds/Proprietary Trading firms who research and craft financial models using Complex Math and transfer such knowledge into a practical and usable form that traders/executors in the company can use. Such individuals usually work in what most layman in the finance industry would commonly know as *Middle Office*.

1.1.1 Entry to Market - Education and Capital

Quants usually come from a variety of prestigious universities such as those of Ivy Leagues like Harvard, MIT or Standford University where they have had good connections as well as good knowledge of their fields having studied for about 10 years (C_Rod, 2018)as most Firms would only hire individuals with a Masters or PHDs (Two Sigma, 2018) especially since the pay is almost half a million at Entry level (Butcher, 2017) with a guaranteed base salary of about 175k including lodging and transport for most firms. (Glassdoor, 2018). On top of that, these *Quants* come from complex fields such as engineering, physics and even genetic biology as such fields have deep knowledge that would be required for a Quant role. (Butcher, 2017)

1.1.2 Entry to Market - Skills

It takes a certain amount of skills too to be come a Quant. In addition to having an education, one must both be able to program extremely well and have a thought process for it.

For example, to program a simple *Moving Average Strategy* to trade, One would be required to be able to:

1. Turn the Strategy into a Mathematical Algorithm
2. Translate the Algorithm into Coded Mathematical Expressions
3. Have a ready Program, Software Framework and Infrastructure to input those Mathematical expressions that can be used.
4. Input the Mathematical Expression into the Code

This is just merely a simple overview of the process from an annecdotal standpoint. However, to be competitive one has to give considerations to fast code and fast infrastructure (market data source and servers) as most algorithm trading is a chess game of latency (Quantinsti, 2015). Most of which demands a large amount of money; for example Blomberg's data subscription costs around 24K a year (Seward, 2013) while paying a premium to collocate servers to the exchanges to further reduce latency (Aswani, 2016).

There have been solutions which have lowered the entry bar to market like `https://www.quantopian.com/` and `https://www.quantconnect.com/`, however such platforms require knowledge of data-science python which may not be suitable for individuals good at Strategy creation but not so much, programming; this is especially crutial to older folks who do not have as much neural plasticity CITE as to learn how to program in a short period where time is money.

1.2 Speed

It is common knowledge that humans will never be as fast as machines. For example, Most Firms with Algorithmic Trading make thousands of trade orders in less than 400 milliseconds which is slower than an eye blink (Aswani, 2016) which is also the about the same amount it takes to press a button of a buy order hence to even start trading buy hand without algorithms is lost game.

It is also common knowledge that humans will never be as accurate (Wilkinson,

2003) as machines. Humans naturally will occasionally make mistakes like a simple typo or a wrong buy order. Similarly Humans make would tend to make trades based on emotions if under stress and rush yet causing more mistakes. It is simply not worth to risk tons of cash for simple mistakes, a wrong extra zero in the amount of shares bought might cost an extra ten thousand in losses while a wrong order (eg. shorting instead of longing) might cost bankruptcy as shorting has no payment obligation limit not counting the interest/margin fees. CITE

1.3 Profitability

1.3.1 Efficiency

Unlike hand trading, Machines can make thousands to trillions CITE of decisions over a wide array of variables and data in a split second due to it's speed. The time it takes a human to analyze a trade, a machine has already moved a thousand steps ahead and will never be wrong provided it follows its algorithm that will keep it profitable regardless.

Lets not forget that firms furthermore have bulks of capital and manpower to optimize their Machines/Infrastructure and Software to become much more efficient and hence more profitable.
Perhaps this is one of the reasons that Algorithmic Trading constitutes about 90% of all trading.

1.3.2 Extra Time

As there is a program doing all the work for you. Naturally more time can be spent doing research or creating/optimizing code and not stressing over to keep an eye on positions and trades during market hours. It must be noted most market hours that traders trade do not relate to the time zone they are located in. So a individual who trades US Stocks but is based in Singapore would have to work night shift and only sleep at 5AM, not a very healthy habit either.

Even if one decides to observe the trade algorithm makes in real time. One can instead dedicate the precious time during market hours to analyses and verify trades (these observations can be used for further algorithm optimizations in future) or provide suggestions to the algorithms based on personal insider knowledge or even news; there are nowadays algorithms which use NLP to analyze news and react accordingly CITE.

Finally, with the extra time provided after the initial setup while the automated algorithms run. One may choose to take the career of the occasional Quant instead; though with lesser profits, there is more time. He may choose a few stable algorithms to be utilized while he has a main job which is his main life; he merely tends to his algorithms in his spare time. These stable algorithms has an upside of requiring less initial principal too as they do not employ too much risks in the trades so as to reduce supervision which would be required in high risk algorithms in case anything goes wrong and causes catastrophic failures. In a nutshell, Algorithmic trading provides individuals which a choice on how much time they wish to spend making it a flexible career for anyone.

1.4 The Niche and Thesis Statement

All of these summed up have shown that algorithmic trading is very lucrative due to the high pay which would suggest there is large money to be made but has yet a high level of entry, a very exclusive club.

To get into a firm, cross-disciplinary skills for such as Computer Science and Math at a Masters Level are a prerequisite which is a very lengthy process to gain. Even Using layman playforms like *Quantopian*, a programming skill is still needed which might not be suitable for many.

One can go back to hand trading, but it lacks so much speed, efficiency and profitability that it's simply not worth. Considering that most *Retail traders* lose 90% of their principal in the long term while hedge funds consistently gain long term despite short term loses; a total opposite. CITE

Furthemore, in the ever increasing rise of technology in our lives in the age of Information; the 4th industrial revolution, where "a fusion of technologies that is blurring the lines between the physical, digital, and biological spheres." (Schwab, 2016). Everything involves technology as it becomes more promoted and democratized; cheaper and more accessible to many. It is only logical and sane to improve our work with what we use daily, technology and increase our competiveness;

So the question presents it self, **How may we *Democratize Quantitative trading* to the masses such that anyone especially older folks who have hope to do something more, have time and stagnant cash in the bank be involved with Quantitative Trading without too much a challenge ?**

1.5 User Scenarios (Users)

With all the above points considered, there are a few groups people this application might serve:

- Quants - Quants who despite having an already sufficient background in education as well as working in a large hedge fund might desire a simpler and straightforward interface so they may focus on the algorithm an solely the algorithm not concerning self with things programming languages. Especially with their extensive resources, knowledge and networks, they may even extend the platform and use it to its full potential. Why make your life harder, make it easier if it's possible ! Life is short ! With a good platform, more quality work is produced and hence a better pay and quality of life.

- Hedge Funds - As most Quants are working in hedge funds, as more employees use the system naturally it will become normalized standard in the company especially after he company sees the cost savings in time effort as well as entry. Quantitative Analysis will become something that everybody in a company may do and this would reduce the need for training or employing specialist which in the finance industry, are not cheap, they are one of the highest paid second only to CEO. with a platform that anyone can use, this would allow both the testers as well as main developers to work on the same platform reducing possible migration conflicts stemming from working on non-consistent platforms across the company further reducing issues that the company faced and assume was part and parcel of a hedge fund. More effort can be put to actually making money over solving issues instead and hence giving companies who use this platform and edge.

- Retirees - With the increase used of technology in our lives compounded with the Smart Nation initiative in SG, it is espected that naturally everyone knows how to use a computer and hence would know how to use our platform. Retirees are individuals who have worked for a number of years and have chosen mostly by choice to stop working and have made enough money sitting around unused, they also have tones of time. Naturally, Humans tend to want to do something and not stay still. It is also better to exercise the mind so as to prevent dementia. Hence we believe one of our users group may be retirees since they have time due to not doing anything and enough unused money sitting around that will probably stay unused if not spent. Some retirees were once workaholics and hope to sustain themselves a stable source of income not wanting to rely on their kids, this may be an outlet for them to continue work but in the comfort of their home and create financial models that they can sell or use it themselves to earn money.

- Students - just like Raspberry-pi is bare metal programming democratized and made mainstream, so may our platform become. Many students these days don't just seek entertainment or blind time fillers, they seek something new or advanced and interesting to fill their curiosity. Raspberry-pi may be the hardware which piqued many kids interest and took on widespread adoption. Perhaps kids might too now look for things that may advance their careers and finances in future and hence may look to our platform, the software which intends to pique interest of kids. Kids may desire to follow their parents in the finance industry and hence this may be their first introduction to it as it is easy to learn, they further more have tons of free time like retirees and can ask for play money from their parents to try without harm. This would exercise their thinking and thought process and hence good for their future mental health. It may also help them find a passion early, just as how some young individuals find the love for coding and hence grow up to be coders. Finally, schools may be to use this platform to teach students simple programming (QLang) as well as algorithm analysis while young to train minds and pique interest as this platform is easy and straightforward enoguh to be used by anybody.

- Working Individuals - Many working adults in their mid life in Singapore usually work office hours 9-5. This means they they usually have a small amount of time after work before sleep if the yopt to not go partying; especially since Singaporeans are very family centered and probably would go home to spend time with their children. However not 100 percent of the time is spent with the kids and parents may want a break and invest in their own interest without being away from their children, they may choose to use this platform if they have a finance background or perhaps hope to dive into something new. Furthermore, many adults have spare time not enough to find a second job(tiring too !) but desire extra cash may opt to use this platform to prototype financial models/algorithms to use or sell.

- Lastly, more straightforwardly, this platform may be used by another group, existing day/retail trades who trade by hand but are hoping to improve their portfolio with algorithms (especially since it's more efficient and trades faster)/ simply finding another way of trading, they may uses this platform to prototype it especially since being so easy to use and with a programming language a child could use, there is no problem getting up to speed. All they need is simply a web browser and a personal computer.

1.6 Use Cases Flows [Use Case + Features in text form]

All Users May access all use cases features with the exception of one user (system) which has an exclusive use case of updating the system by crawling updated news

1.6.1 Main: Search and View News

Goal: User Intends to search news relating to a new Boeing Air Plane
Conditions: User has to have a valid account and be logged in successfully

1. User Clicks Searchbar on News page - Search bar becomes foucsed with a blinking line

2. User types in "New Boeing Air Plane" - Typed text appears in search bar

3. User See desired news, clicks on news item "Boeing new 770A Dreamliner" - News item has a darker color to shows it's selected and at the

same time, the news item's full text pops up on the right panel with the title on the top

1.6.1.1 Alt: Desired Item not available at first glance

(a) User Scrolls on the Headlines Panel - More items show up as the user scrolls downwards

(b) User See desired news, clicks on news item "Boeing new 770A Dreamliner" - News item has a darker color to shows it's selected and at the same time, the news item's full text pops up on the News Text Panel with the title on the top

1.6.1.2 Alt: Desired Item not available yet or does not exist

(a) News Item panel becomes blank with "No Such Items Exist" text

1.6.2 Main: Login

Goal: User Seeks to enter application with Login Details provided by the admin
Conditions: Have a valid and existing account and know it's credentials. User is not logged in

1. User enters Main Login page on first application launch - Login Page appears
2. User clicks on Username field - text field becomes focus with a blinking line
3. User enters Login Username into Username field - Username appears in text field
4. User clicks on Password field - text field becomes focused with blinking line
5. User enters Login Password into Password field - Password appears in text field

6. User clicks Next button to login - user should be redirected to Main Page and the first thing the user sees is the news page. Their real name should appear on the User Info Panel of the page.

1.6.2.1 Alt: Username does not exist

(a) Login page does not redirect and username field is highlighted in red outline, an error text appears over it with "Username does not exist"

(b) User enters correct username

(c) User clicks Next button to login - User should be redirected to Main Page and the first thing the user sees is the news page. Their real name should appear on the User Info Panel of the page. (if both username and password are now right)

1.6.2.2 Alt: Password is wrong

(a) Login page does not redirect and password field is highlighted in red outline, an error text appears over it with "Wrong Password"

(b) User enters correct password

(c) User clicks Next button to login - User should be redirected to Main Page and the first thing the user sees is the news page. Their real name should appear on the User Info Panel of the page. (if both username and password are now right)

1.6.2.3 Alt: Username and Password does not exist or is wrong

(a) Login page does not redirect and both fields are highlighted in red outline, an error text appears over it with "Incorrect Username And Wrong Password"

(b) User enters correct username and password

(c) User clicks Next button to login - User should be redirected to Main Page and the first thing the user sees is the news page. Their real name should appear on the User Info Panel of the page. (if both username and password are now right)

1.6.3 Main: Logout

Goal: User Seeks to logout of application
Conditions: Have a valid and existing account and is currently logged in

1. User clicks on News Button on the quick links bar at the bottom - User is redirected to news page and it appears after loading

2. User clicks on the "Logout" button on the right User Info Panel - User is redirected to main login page

1.6.4 Main: Change Personalization Settings - Theme

Goal: User Seeks to change personalization settings
Conditions: Have a valid and existing account and is currently logged in. Currently on the News Main Page **Note:** Same flow for when User accesses this page from the Analyze Page

1. Click on Settings Button on the News Main Page at the User Info Panel - User Redirected to Settings page with profile settings options

2. Click on Personalization Button on the top bar - Personalization Settings Options appears on page instead of profile options

3. Default Theme is White so user clicks the "Black" button to change the theme. - The button with "Black" text is highlighted instead. Whole applications has theme colors inverted to create a dark theme on the spot (eg. White background becomes black) and because a setting was changed, save button turns from disabled to enabled state

4. User clicks save to save settings - save button turns into a disabled state again signifying a successful save

1.6.4.1 Alt: Change Personalization Settings - Main Font

(a) Default Font is Helvetica so to change user clicks select box - Select box expands below to show font options

(b) User scrolls down until they find the font they want, in this case Arial - Arial is seen as one of the select box options

(c) User clicks on Arial option in select box - Arial option highlighted, select box closes and Arial instead of Helvetica is shown as the select box selected option. The whole application's font at the same time changes to Arial and because a setting was changed, save button turns from disabled to enabled state

(d) User clicks save button to save settings - save button turns into a disabled state again signifying a successful save

1.6.4.2 Alt: Change Personalization Settings - Code Font

(a) Default Font is Consolas so to change user clicks select box - Select box expands below to show font options

(b) User scrolls down until they find the font they want, in this case Mono - Mono is seen as one of the select box options

(c) User clicks on Mono option in select box - Mono option highlighted, select box closes and Mono instead of Consolas is shown as the select box selected option. The whole application's font at the same time changes to Mono and because a setting was changed, save button turns from disabled to enabled state

(d) User clicks save button to save settings - save button turns into a disabled state again signifying a successful save

1.6.4.3 Alt: Change Personalization Settings - Cursor

(a) Default Cursor is pointer so to change user clicks select box - Select box expands below to show font options

(b) User scrolls down until they find the cursor they want, in this case Arrow - Arrow is seen as one of the select box options

(c) User clicks on Arrow option in select box - Arrow option highlighted, select box closes and Arrow instead of Pointer is shown as the select box selected option. The cursor at the same time changes to Arrow and because a setting was changed, save button turns from disabled to enabled state

(d) User clicks save button to save settings - save button turns into a disabled state again signifying a successful save

1.6.4.4 Alt: Change Personalization Settings - Alarm

(a) Default Alarm is a beep so to change user clicks select box - Select box expands below to show font options

(b) User scrolls down until they find the alarm sound they want, in this case Ring - Ring is seen as one of the select box options

(c) User clicks on ring option in select box - Ring option highlighted, select box closes and Ring instead of Beep is shown as the select box selected option. The application gives a preview by playing the sound after selecting.Because a setting was changed, save button turns from disabled to enabled state

(d) User clicks save button to save settings - save button turns into a disabled state again signifying a successful save

1.6.4.5 Alt: Change Personalization Settings - Cancel

(a) After changing the theme setting from white to black, the save button is enabled but user decides he likes the previous setting better and does not want to change settings anymore but go back, the user clicks the Cancel button - Settings are reverted by the theme going back to it's previous option, in this case the black theme turns back to white. User is redirected back to the news page.

(b) User clicks on settings button again to check settings - just as expected, the setting has reverted itself as shown by the White theme button being shown selected instead of the black theme.

1.6.4.6 Alt: Change Personalization Settings - Reset

(a) User decides he wants to change all settings back to default by pressing the reset button.- All the settings that the user previously changed are reverted and wiped. The Application goes back to it default clean state without modifications like font or theme changes. User Redirected to back News Main Page.

(b) User clicks on settings button again to check settings - just as expected, all the default settings are shown selected instead of what the user previously selected.

1.6.5 Main: Change Profile Settings - Theme

Goal: User Seeks to change Profile settings
Conditions: Have a valid and existing account and is currently logged in. Currently on the News Main Page **Note:** Same flow for when User accesses this page from the Analyze Page

1. Click on Settings Button on the News Main Page at the User Info Panel - User Redirected to Settings page with profile settings options
2. User clicks on name field. - Name field is in focus with a vertical line.
3. User Backspaces to delete existing name - text deleted from text field and because it has changed, the settings save button turns from disabled to enabled so the user might click save
4. User enters new name - new name appears in text field
5. User clicks save to save settings - save button turns into a disabled state again signifying a successful save
6. User clicks news button on bottom quicklinks bar - redirected to news main page and sees the name has changed

1.6.5.1 Alt: Change Profile Settings - Phone/Email/Address

Note: This flow all applies to phone, email and address flow.

(a) User clicks on phone field. - phone field is in focus with a vertical line.
(b) User Backspaces to delete existing phone - text deleted from text field and because it has changed, the settings save button turns from disabled to enabled so the user might click save
(c) User enters new phone number - new name appears in text field

(d) User clicks save to save settings - save button turns into a disabled state again signifying a successful save

(e) User refereshes the page - User sees the new details are still there as they were saved

1.6.5.2 Alt: Change Profile Settings - Cancel

(a) After changing name, the save button is enabled but user decides he prefers the previous name on this account better and does not want to change settings anymore but go back, the user clicks the cancel button - Settings are reverted to the old record. User is redirected back to the news page.

(b) Just as expected on the User Info Panel on the news page, the old name is shown instead.

1.6.5.3 Alt: Change Profile Settings - Invalid

(a) User clicks on phone field. - phone field is in focus with a vertical line.

(b) User Backspaces to delete existing phone - text deleted from text field and because it has changed, the settings save button turns from disabled to enabled so the user might click save

(c) Without a new value entered but an empty field, the save button turns into disabled once again.

1.6.6 Main: New Stocks List

Goal: To Create a new stocks list with 3 major tech stocks
Conditions: User must have an existing account and logged in. Currently on the watch Page

1. User clicks on the New List button on the right side bar - New List Creation Step 1 Page comes up

2. User clicks on list name text field - text field becomes focused with vertical line

3. User enters name of list he desires into text field - Name appears in text field
4. User clicks on list description text field - text field becomes focused with vertical line
5. User enters description of list he desires into text field - description appears in text field. In this case the Next button turns from a disabled to an enabled state which means the user may click it to go to the next page

1.6.6.1 Alt: New Stocks List - Cancel

(a) User decides not to create list and decides to go back, he clicks cancel button - Redirected to main watch page

(b) On the My Recent Lists Panel, the list does not appear which would mean it was not created, expected behavior

6. User Clicks next to go to next page - Page Step 2 loads up
7. User Clicks on "Type in Stock" field - it comes into focus with a vertical line

1.6.6.2 Alt: New Stocks List - Finish without Stock

(a) User Clicks finish button - redirected to list page but no data is shown as no stock was added

8. User Types in "AAPL.O" to add in Stock Ticker for Apple
9. User clicks + button to add the stock - Stock ticker is added as shown in the box below with full company name beside
10. User Types in "MSFT.O" to add in Stock Ticker for Microsoft
11. User clicks + button to add the stock - Stock ticker is added as shown in the box below with full company name beside
12. User Types in "IBM.N" to add in Stock Ticker for IBM

13. User clicks + button to add the stock - Stock ticker is added as shown in the box below with full company name beside

1.6.6.3 Alt: New Stocks List - Invalid Stock

(a) User Types in ”ABCD.N” to add in Stock Ticker for a random company

(b) User clicks + button to add the stock - red outline text field with error above saying ”stock ticker invalid”. Until a valid stock ticker is added, red outline would not disappear

14. User Clicks Finish - redirected to stocks page with all the added stock

1.6.7 Main: New Stocks to List

Goal: To add new stock to list
Conditions: User must have an existing account and logged in. Currently on the watch Page. Existing List

1. User clicks on New Stock button on right panel. - redirected to new stock page

2. User selects a list from scroll list (SP500 in this case) - select button becomes enabled

3. User Clicks on select button - redirected to step 2 page

4. User Clicks on ”Type in Stock” field - it comes into focus with a vertical line

5. User Types in ”DBSM.SI” to add in Stock Ticker for DBS

6. User clicks + button to add the stock - Stock ticker is added as shown in the box below with full company name beside

1.6.7.1 Alt: New Stocks List - Invalid Stock

(a) User Types in ”ABCD.N” to add in Stock Ticker for a random company

(b) User clicks + button to add the stock - red outline text field with error above saying ”stock ticker invalid”. Until a valid stock ticker is added, red outline would not disappear

7. User Clicks Finish button - redirected to list page in this case SP500, in that page the DBS stock is shown

1.6.8 Main: User views list from list page

Goal: User seeks to view list via the list page
Conditions: User has account and is logged in. Currently on the watch page. Has Existing List.

1. User clicks list button on the right side panel. - redirected to list page
2. User clicks on the list he wishes to see, in this case SP500 - redirected to SP500 List
3. On the Sp500 List the user is able to see the list of stocks in the list such as APPLE and Microsft, prices and quantity - these panel auto updates as the stock price changes

1.6.9 Main: User views list from My list panel

Goal: User seeks to view list via the my list panel
Conditions: User has account and is logged in. Currently on the watch page. Has Existing List.

1. User clicks on the Dow Jones button on the my recent list panel on the left side button - Dow Jones Stocks shows up in middle of page with respective price and quantity - these panel auto updates as the stock price changes

1.6.10 Main: Simulate a studies on a stock chart with mouse interaction

Goal: User seeks to view a stock on the stock chart
Conditions: User has account and is logged in. Currently on the Chart page.

1. User Clicks on Stock Text field above Chart Controls (play, pause, reset) - text field focus with vertical line
2. User types in "AAPL.O" to trigger chart for Apple Stock - Time Series Chart with Apple Stock comes up

1.6.10.1 Alt: User adds invalid ticker

(a) User types in invalid ticker "AAAPL.O" by mistake. - nothing happens while the text field is outline in red

3. User Click Adds Study Button - Studies Selection Modal Comes up
4. Use Selects Moving Average Study - section expands on the side showing text fields for parameters
5. User enters parameter values in the respective fields - text appears in the fields
6. User Clicks next button - Modal closes an Moving Average Study is shown under "Studies Information Section on the Chart Page"

1.6.10.2 Alt: Simulate a studies on a stock chart with mouse interaction - Remove Existing Study

(a) User Clicks Remove Study Button - Modal pops up with added studies

(b) User Clicks Exponential Average Study and Then Confirm Button - Modal Closes and study no longer on the study panel on the right side

1.6.10.3 Alt: Simulate a studies on a stock chart with mouse interaction - Edit Existing Study

(a) User Clicks Edit Study Button - Modal pops up with added studies

(b) User Clicks Bolinger Average Study - Section expands with text fields for parameters

(c) User enter new parameter values in the text field. - text appears in the text field

(d) User clicks confirm button - Model Closes and the corresponding study in the study panel changes to reflect data based of analysis of the new parameters

7. User Clicks Play control button - The chart starts moving from right to left. As the Charts move, the values in the study changes to reflect the respective value for that period as the chart moves. General Information Details also changes to show the respective value for that date or period as the chart moves.

8. User Decides to pause chart to examine that period by clicking the pause button. - Chart Stops moving

9. User hovers mouse over the chart and moves the mouse around - values pop up on the carton the axis to show the x and y axis value for that data point on the chart

10. User clicks on rest control button to go back to the staring point - Chart resets to the starting point and all the information on the left panel are shown as None as the chart has not be played to show any value yet.

1.6.11 Main: New Analysis

Goal: User seeks to create new analysis
Conditions: User has account and is logged in. Currently on the analyze page.

1. User clicks on Create button on the top - redirected to step 1 of analysis creation page

2. User enters name of analysis in name text field - text appears in the field and next button turns from disabled to enabled

3. User Enters description of analysis in description text field - text appears in the field

4. User clicks next button - redirected to code page

5. User types in code to be executed - code appears in mini text editor
6. User clicks build button - redirected into results page - loading screen until analysis done

1.6.11.1 Alt: New Analysis - Invalid Syntax Code

(a) Instead of expected list of results metrics being shown, a large "Invalid Syntax" text is

7. User Clicks on Sharpe Ratio Metric to expand it - section expands with time series graph that shows the Sharpe ratio overtime
8. User Clicks back button to go back to code editor - redirected back to code editor

1.6.12 Main: Search and View Analysis and respective versions

Goal: User seeks to create new analysis
Conditions: User has account and is logged in. Currently on the analyze page.

1. User clicks on search bar - search bar in foucs with vertical typing line
2. User types in "SP500 Analysis" - The list of analysis are filtered to show only SP500 Analysis
3. User Scrolls down until find the one they desire and click on it - redirected to analysis results page
4. User Clicks on edit button - redirected to code editor page for that analysis

1.6.12.1 Alt: Search and View Analysis and respective versions - Fork

(a) User Clicks on Fork this button - redirected to step 1 of analysis forking page

(b) User Enters name into Analysis Name field - Next Button turns from disabled to enabled

(c) User enters description into Analysis Description field

(d) User Clicks Next Button - user is redirected to code editor with the same code but different analysis name and record

1.6.12.2 Alt: Search and View Analysis and respective versions - Fork

(a) User Clicks on Trash this button - redirected to saved analysis page with the search bar. That respective analysis has been trash and can no longer be found

5. User clicks on versions button - redirected to versions page as they scroll through code versions with their respective timestamps - these are saved after evertime a user saves a new version of a code or it has been 5 minutes since the last save

6. There is the option to fork either of these versions but user decides not to and clicks saved button on the top to returned to main Analyze page where all saved list of analysis are shown.

1.6.13 Main: View Company Information - Ticker

Goal: User seeks to view Apple's Ticker Informations
Conditions: User has account and is logged in. Currently on the Main Page of Information

1. User clicks on search bar - search bar in focus and vertical line

2. User types in Apple in to search bar - Apple Corporation comes up as one of the possible suggestions underneath

3. User Clicks on Apple Suggestion - User redirected to Page about apple information with current page of Apple Profile

4. User sees the apple Ticker On the center of the page

1.6.13.1 Alt: View Apple Information - Company Summary, Historical, Stats, Financials, Analysis, Holders, Sustainability

Note: The flow below is the same for Historical, Stats, Financials, Analysis, Holders, Sustainability links

(a) User clicks on Summary Button on the left panel - Redirected to Summary Section with list of company summary

5. User seeks to go back to the main information page so he clicks the back button - Redirected to main information page

1.6.14 Main: View Information Profile via Recent Searches

Goal: User seeks to view IBM's Informations profile descriptions
Conditions: User has account and is logged in. Currently on the Main Page of Information. Have previous recent searches.

1. Click on IBM's Ticker that was recently searched on the right panel - Redirected to IBM Profile with the company description shown on the right side

1.6.15 Main: Add New Calender Item

Goal: User seeks to view add new calender item
Conditions: User has account and is logged in. Currently on the Main Page of Calender.

1. Click on Event Name Field on the right panel - Field Focus with vertical line
2. Enter Event Name into text field - Text appears in field
3. Click on Event Description Field on the right panel - Field Focus with vertical line
4. Enter Event Description into text field - Text appears in field
5. Click on Event Marker Field on the right panel - field expands with showing select box below to choose marker

6. Click on Blue marker option on the select box - Select box closes and Blue marker option is shown in the Event Marker field
7. Click on Event Date Field on the right panel - field to show date selector
8. Select Date from date selector by selecting a number on the year grid, then month grid which narrows down into a day grid - the date selector closes and the date is shown in the event date field
9. Click on Event Time Field on the right panel - field to show time selector
10. Select time from time selector by selecting the hour from a select box dropwdown and the minute from another select box dropdown beside - time selector closes while time appears in the even time field.
11. User Clicks confirm button - Alert box Appears on the top with "Event Added" then dissapears after a few seconds

1.6.16 Main: View Calender Item

Goal: User seeks to view delete calender item
Conditions: User has account and is logged in. Currently on the Main Page of Calender.

1. User selects a date (where the event is in) from the mini calender grid on the left panel, by first selecting the month-year pair and from the side arrows then the corresponding day on the date grid - the Agenda for that week will show up in the middle
2. User hovers upon the date item they wish to see and click on it - a modal pops up with it's description and respective color tag

1.6.17 Main: Delete Calender Item

Goal: User seeks to view delete calender item
Conditions: User has account and is logged in. Currently on the Main Page of Calender.

1. User selects a date (where the event is in) from the mini calender grid on the left panel, by first selecting the month-year pair and from the side arrows then the corresponding day on the date grid - the Agenda for that week will show up in the middle

2. User hovers upon the date item they wish to discard - a X button appears on the side

3. User clicks X button to remove it - agenda item dissapears

1.6.18 Main: User Changes Calender Marker Tag

1. User clicks on the blue market tag on the left panel - text field expands on a tool tip with "Home" inside it.

2. User clicks on the text field to remove the text and changes it to "Life" and clicks outside the text field to change focus - the tag now shows the new information - "Blue - Life"

1.6.19 Main: View Education Information

Goal: User seeks to view education item for IS() function documentation and example
Conditions: User has account and is logged in. Currently on the Main Page of Education. **Note:** This is the same flow for when education information is accessed via the qlang reference from code editor section

1. User clicks on search bar - Search bar comes into focus with vertical line

2. User types "IS()" into search bar - the page is filtered to only show records related to "IS()"

3. User scrolls down to desired record for "IS()" documentation

4. User clicks on the title for that documentation - it expands to give a detailed definition with an example of the code in usage

1.6.20 Main: Update New News

Goal: System seeks crawl the respective news pages via it's servers to update onto the database

1. System Goes to news home website - News website loads
2. System checks for new news updates on the home website

1.6.20.1 Alt: Update news - No news updates

(a) System chooses to ingore checking for that website until the next interval comes

3. System Clicks onto new news item and go to that page - news item page loads
4. System Crawls and Copy the webpage news headline and text as well as metadata and dumps it into the database - news content is now in database
5. System closes the news page

1.6.21 Main: Update Edited News

Goal: System seeks crawl the respective news pages via it's servers to check for edited news onto the database

1. System Goes to news home website - News website loads
2. System scans all the news upon the website and compare it with the database record for any changes - System detects changes

1.6.21.1 Alt: Update news - No news updates

(a) System chooses to ingore checking for that website until the next interval comes

3. For each of the news items that have changesm System Clicks onto the edited news item and goes to that page - news item page loads
4. System Crawls and Copy the webpage the edited news headline and text as well as metadata and dumps it into the database - updated eidted news content is now in database
5. System closes the news pages

2 Solution Description and Features

In this paper we would be introducing a *Full Fledge Trading Terminal with a Microservice Backend and a NoSQL Database.* that any laymen could operate and utilize to make money through Algorithmic Trading.

We will descirbe our solution using the outside-in approach meaning that we'll start from the big picture and then dive into the singular parts and components to describe how it works in depth. We will keep mind not to over-document parts that are self-explanatory and common sense.

We will be describing our solution in this order:

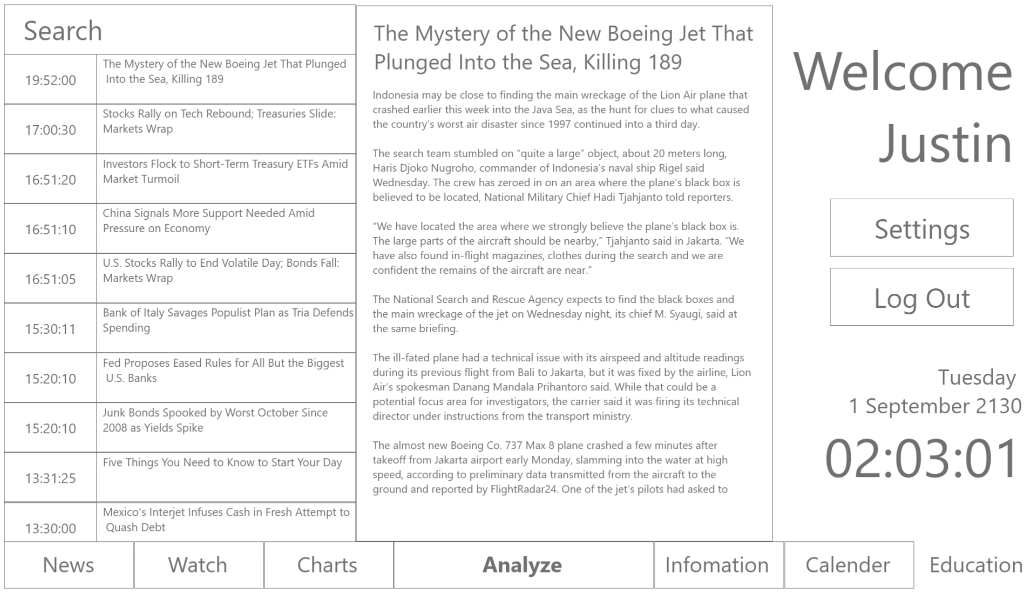
1. Main Dashboard
2. Microservices
3. Databases

2.1 Main Dashboard Overview

The Main Dashboard is just as it name says, the main dashboard. It is also known as the main trading terminal screen. The screen that you see once you login onto the application. It is the window where upon users operate and utilize the product and is the main graphical interface that utilizes and communicates with the Microservices backend. It is also the screen that encompass all the different components and functions of the application. This is the only screen the user will ever see.

2.1.1 Main Dashboard Design Mockups and Description [Screenshots]

2.1.1.1 News Page



19:52:00	The Mystery of the New Boeing Jet That Plunged Into the Sea, Killing 189
17:00:30	Stocks Rally on Tech Rebound; Treasuries Slide: Markets Wrap
16:51:20	Investors Flock to Short-Term Treasury ETFs Amid Market Turmoil
16:51:10	China Signals More Support Needed Amid Pressure on Economy
16:51:05	U.S. Stocks Rally to End Volatile Day; Bonds Fall: Markets Wrap
15:30:11	Bank of Italy Savages Populist Plan as Tria Defends Spending
15:20:10	Fed Proposes Eased Rules for All But the Biggest U.S. Banks
15:20:10	Junk Bonds Spooked by Worst October Since 2008 as Yields Spike
13:31:25	Five Things You Need to Know to Start Your Day
13:30:00	Mexico's Interjet Infuses Cash in Fresh Attempt to Quash Debt

The Mystery of the New Boeing Jet That Plunged Into the Sea, Killing 189

Indonesia may be close to finding the main wreckage of the Lion Air plane that crashed earlier this week into the Java Sea, as the hunt for clues to what caused the country's worst air disaster since 1997 continued into a third day.

The search team stumbled on "quite a large" object, about 20 meters long, Haris Djoko Nugroho, commander of Indonesia's naval ship Rigel said Wednesday. The crew has zeroed in on an area where the plane's black box is believed to be located, National Military Chief Hadi Tjahjanto told reporters.

"We have located the area where we strongly believe the plane's black box is. The large parts of the aircraft should be nearby," Tjahjanto said in Jakarta. "We have also found in-flight magazines, clothes during the search and we are confident the remains of the aircraft are near."

The National Search and Rescue Agency expects to find the black boxes and the main wreckage of the jet on Wednesday night, its chief M. Syaugi, said at the same briefing.

The ill-fated plane had a technical issue with its airspeed and altitude readings during its previous flight from Bali to Jakarta, but it was fixed by the airline, Lion Air's spokesman Danang Mandala Prihantoro said. While that could be a potential focus area for investigators, the carrier said it was firing its technical director under instructions from the transport ministry.

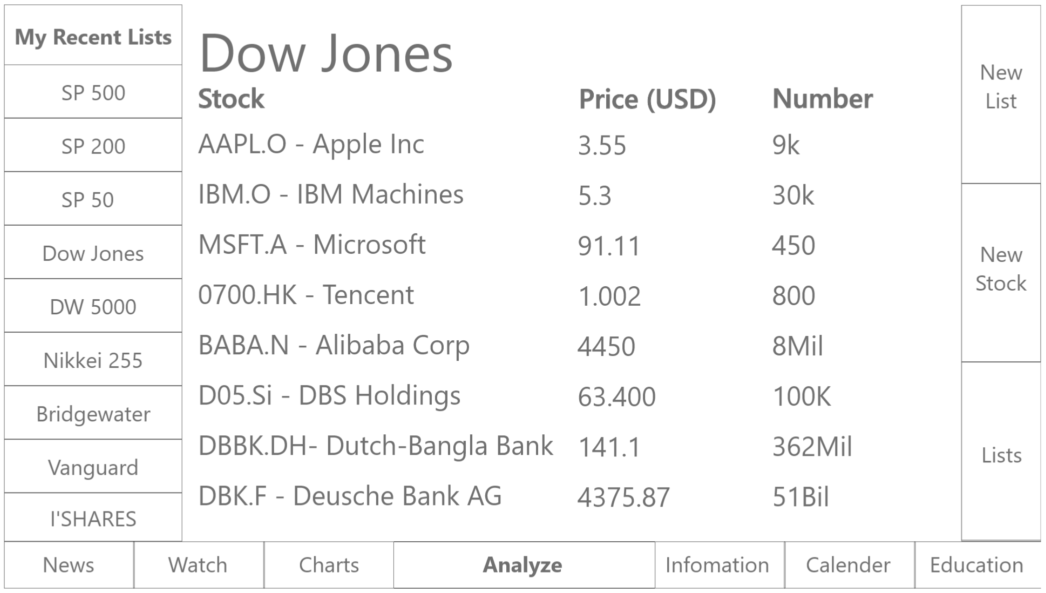
The almost new Boeing Co. 737 Max 8 plane crashed a few minutes after takeoff from Jakarta airport early Monday, slamming into the water at high speed, according to preliminary data transmitted from the aircraft to the ground and reported by FlightRadar24. One of the jet's pilots had asked to

Tuesday
1 September 2130
02:03:01

Description
The *News Page* is the first panel a user sees after loging in. It shows the overview of the news in short headlines. On clicking the *Headlines Panel* on the left, the *News Text Panel* is shown. These news information is crucial to traders who make trades on a company, for example, a company scandal would result in a stock crash and hence would be bad to buy the stock.

On the right is the *User Info Panel*. It welcomes the user with their name with settings and logout buttons. On the bottom is simply datetime information.

On the bottom, the bar is called the *Toolbar*. These are quicklinks buttons to other parts/pages of the application.



Dow Jones

Stock	Price (USD)	Number
AAPL.O - Apple Inc	3.55	9k
IBM.O - IBM Machines	5.3	30k
MSFT.A - Microsoft	91.11	450
0700.HK - Tencent	1.002	800
BABA.N - Alibaba Corp	4450	8Mil
D05.Si - DBS Holdings	63.400	100K
DBBK.DH- Dutch-Bangla Bank	141.1	362Mil
DBK.F - Deusche Bank AG	4375.87	51Bil

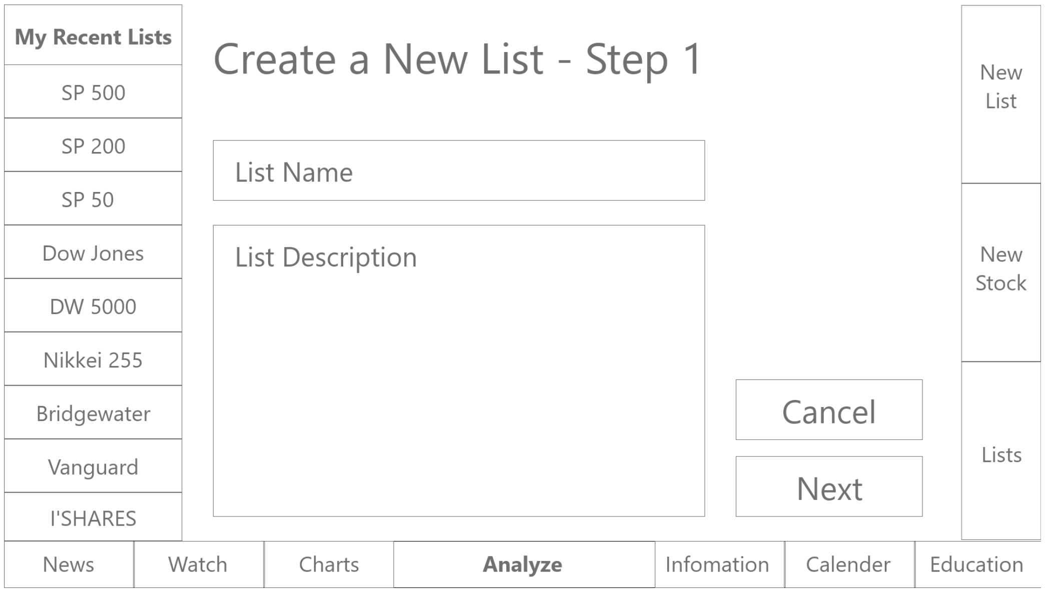
2.1.1.2 Watch Page

Description

The *Watch Page* is commonly known to many traders as the ”Scanner”. It is the page a user sees after clicking on the Watch button on the toolbar. This page is where users go to view stocks they have saved to be tracked/watched or ”scanned” as it name implies. It is useful for traders and Quants as they can observe the prices of their desired stocks in real time and react quickly as an overview lets them keep note and not forget about it as shown in the center of the page the price of stocks in the Dow Jones list created by the user. Do note that in this preview, the Dow Jones List is selected, else the center portion will be blank.

The left panel, *My Recent Lists Panel* allows individuals to have quick links of their recently created/modified lists and to that page. It is good for Quants who are in a rush reacting to news.

The right panel, *Actions Panel* is simply a panel for users to click to perform actions like creating new lists.



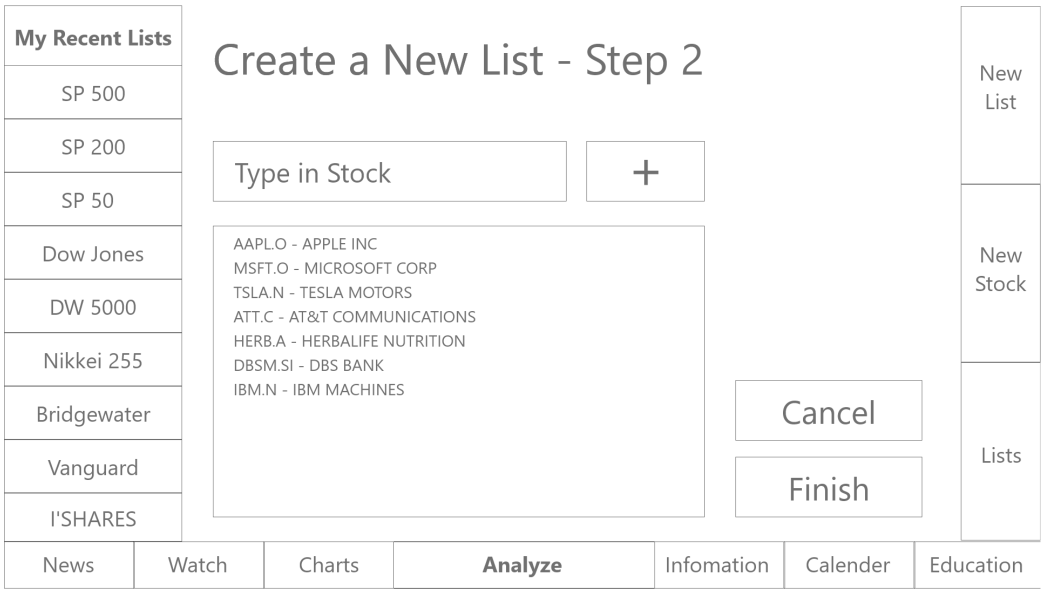
2.1.1.3 Watch Page - Creation of New List - Step 1

Description
The *Watch Page New List Creation* is simply for users to create new list where they can view their stocks. This page is step 1.

The fields are for the user to input their list name and description so they may reference/find it in the future.

The user may click the next button to proceed to step 2 or cancel if they wishes to not continue the process.

2.1.1.4 Watch Page - Creation of New List - Step 2

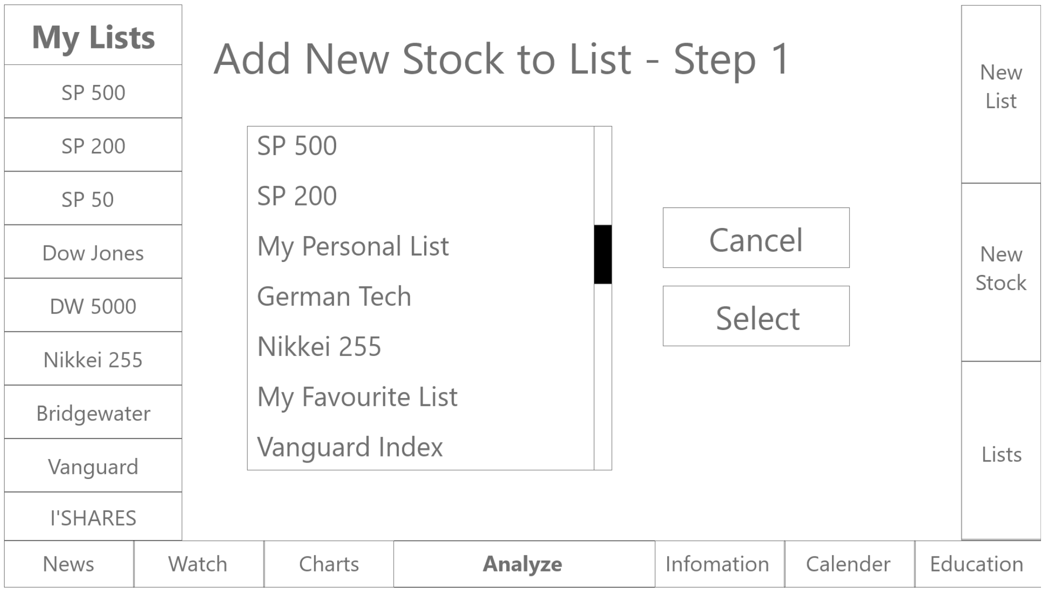


Description
Similar to the previous section. This is just step 2 of list creation. The User is required to select stocks to be added into the newly created list.

The user enters the stock symbol for the company desired in the *Type in Stock Field* and presses the ”+” button which will trigger the stock added to appear in the panel below to signify it has been added.

The user may click the *Finish Button* to confirm or *Cancel Button* if they wishes to not continue the process. If the *Finish Button* leads to the List View Page present in *Section 2.1.1.2* with the list that was created shown.

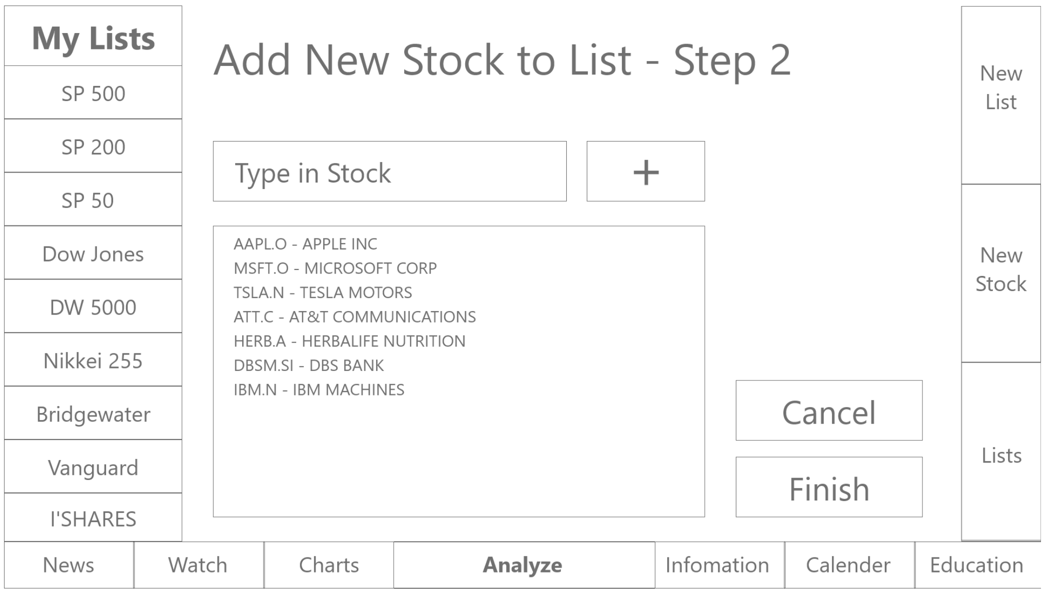
2.1.1.5 Watch Page - Creation of New Stock - Step 1



Description
The *Watch Page New Stock Creation* is simply for users to create add a stock to a existing list. This is step 1.

After the User selects an existing list from the *Center Scrollable Portion* of the page, they may click the *Cancel Button* to stop or the *Select Button* to confirm the list selection and move on to Step 2.

2.1.1.6 Watch Page - Creation of New Stock - Step 2

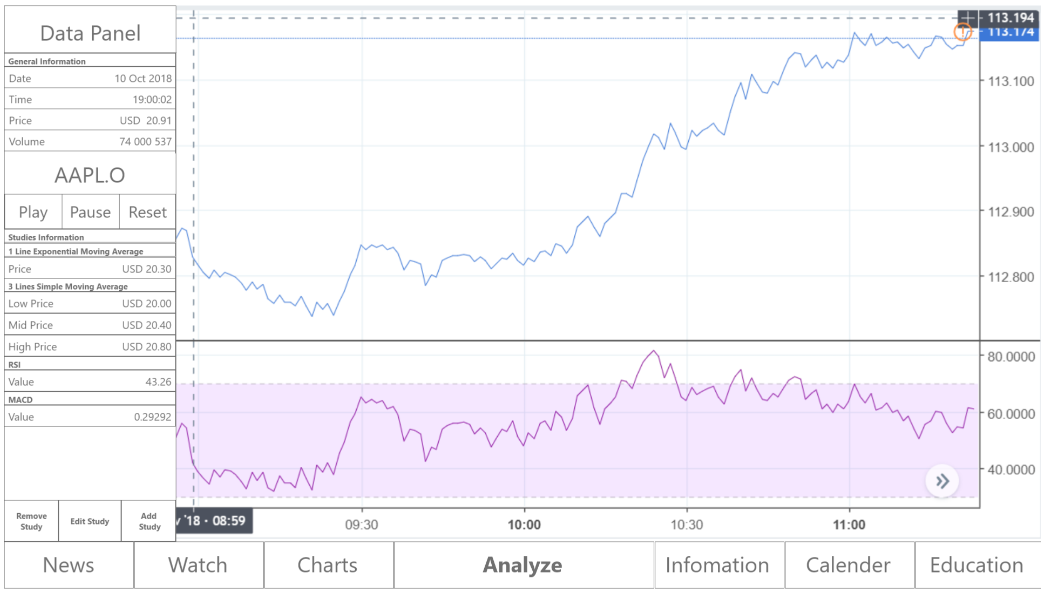


AAPL.O - APPLE INC
MSFT.O - MICROSOFT CORP
TSLA.N - TESLA MOTORS
ATT.C - AT&T COMMUNICATIONS
HERB.A - HERBALIFE NUTRITION
DBSM.SI - DBS BANK
IBM.N - IBM MACHINES

Description
Similar to the previous section however this is Part 2. This is very similar to *Section 2.1.1.4* in terms of layout and functionality just that instead of adding to a new list, it adds to an existing list.

The user may click the *Finish Button* to confirm or *Cancel Button* if they wishes to not continue the process. If the *Finish Button* leads to the List View Page present in *Section 2.1.1.2* with the list that was added to shown.

2.1.1.7 Charts Page


Data Panel

General Information	
Date	10 Oct 2018
Time	19:00:02
Price	USD 20.91
Volume	74 000 537

AAPL.O

Studies Information	
1 Line Exponential Moving Average	
Price	USD 20.30
3 Lines Simple Moving Average	
Low Price	USD 20.00
Mid Price	USD 20.40
High Price	USD 20.80
RSI	
Value	43.26
MACD	
Value	0.29292

Description
The Charts page is where a user may see a real time visualization of historical data with studies/indicators analysis visualizations

This page is suitable for individuals who wish to have a quick replay of the stocks prices visualizations.

It does not store any data on the server but is simply transient and stored as as cookie on the client side.

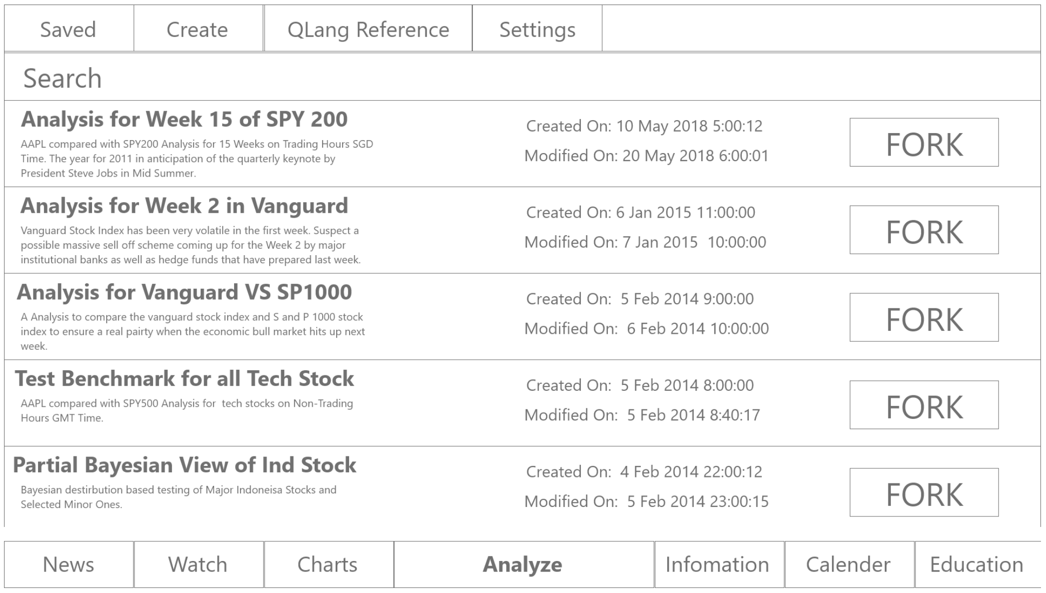
The Chart simply shows the visualization in a time series line chart fashion.

The left panel is a little complex, it displays information about the current data point when the user hovers on-top in the interactive chart. The top portion displays general information about the date,time , price and volume of the datapoint. The middle portion displays the current stock in a text box we are viewing as well as controls to start, pause or reset from the start again, we may change the stock anytime by editing the textbox to display with the stock's Symbol/RIC. Lastly, the bottom section displays indicator/study values for each indicator that we have added using the buttons at the bottom.

When one clicks either button on the extreme bottom ,different behaviors pop up. Adding a study will trigger a dialog box to select the indicator desired, removing will cause check-box to appear beside the studies on the left panel so we may select what to remove and editing will trigger a dialog to edit the parameters of the study after we have selected from the list of existing studies/indicator on the left panel.

It is to be noted that indicators and studies refer to the same thing.

2.1.1.8 Analyze Page



Analysis for Week 15 of SPY 200
AAPL compared with SPY200 Analysis for 15 Weeks on Trading Hours SGD Time. The year for 2011 in anticipation of the quarterly keynote by President Steve Jobs in Mid Summer.
Created On: 10 May 2018 5:00:12
Modified On: 20 May 2018 6:00:01

Analysis for Week 2 in Vanguard
Vanguard Stock Index has been very volatile in the first week. Suspect a possible massive sell off scheme coming up for the Week 2 by major institutional banks as well as hedge funds that have prepared last week.
Created On: 6 Jan 2015 11:00:00
Modified On: 7 Jan 2015 10:00:00

Analysis for Vanguard VS SP1000
A Analysis to compare the vanguard stock index and S and P 1000 stock index to ensure a real pairty when the economic bull market hits up next week.
Created On: 5 Feb 2014 9:00:00
Modified On: 6 Feb 2014 10:00:00

Test Benchmark for all Tech Stock
AAPL compared with SPY500 Analysis for tech stocks on Non-Trading Hours GMT Time.
Created On: 5 Feb 2014 8:00:00
Modified On: 5 Feb 2014 8:40:17

Partial Bayesian View of Ind Stock
Bayesian destirbution based testing of Major Indoneisa Stocks and Selected Minor Ones.
Created On: 4 Feb 2014 22:00:12
Modified On: 5 Feb 2014 23:00:15

Description
This *Analyze Page* is the page that appears after clicking the *Analyze Button* on the *Toolbar*. It is the main page/section a user would spend most of their time in as it is used to analyze stocks and create financial models through coding.

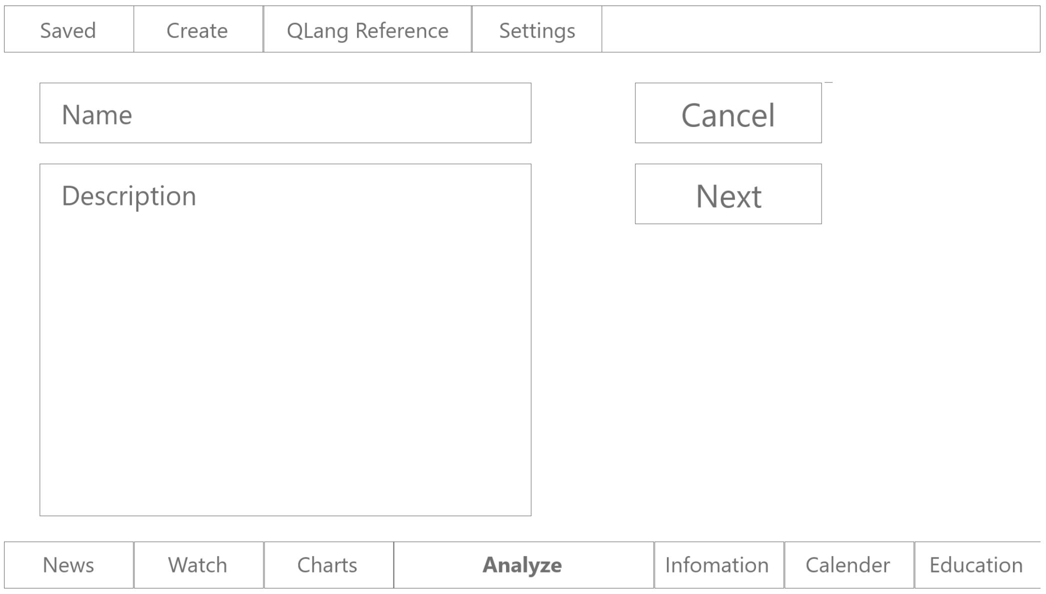
The first view a user has is on entry the *Saved Tab* of the Top Toolbar. This tab displays all the user previously saved Analysis

Moving down the User sees a *Search Bar*. This search bar is used to filter through all previous saves of the application.

Continuing Moving Down. There is a list of saves that a user may click on any to view the report for on the *Analysis View Page*. The *Analysis Title* as well as *Analysis Description* is shown. Creation and Modified datetime details are also shown for utility purposes.

On the extreme right, the *FORK Button* allows a user to "Fork" a Analysis, another word for "copying" the Analysis and save it as something else so it may be edited and re-purposed for another goal. On clicking, the User is brought to the *Fork Creation Page*.

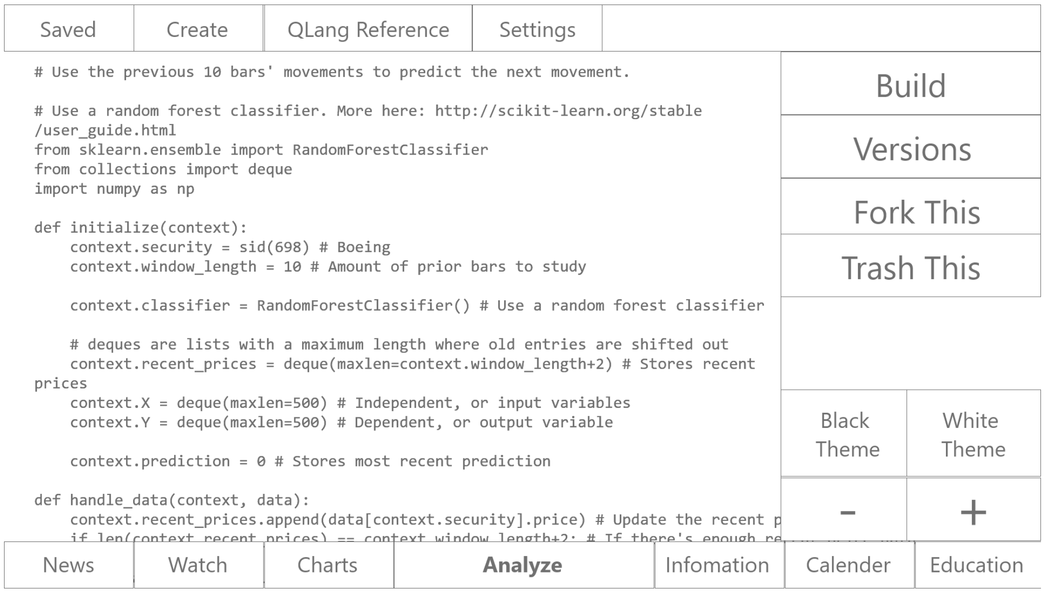
2.1.1.9 Analyze - Fork Creation Page



Description
This *Analyze Fork Creation Page* is the page that appears after clicking the *Fork Button* on the *Analyze Page*. The *Name and Description Fields* are for the user to enter details about the fork of an existing Analysis made. Clicking the *Next Button* will lead to the *Analysis Edit Page* which will be

shown in the next section.

2.1.1.10 Analyze Edit Page



```
# Use the previous 10 bars' movements to predict the next movement.

# Use a random forest classifier. More here: http://scikit-learn.org/stable
/user_guide.html
from sklearn.ensemble import RandomForestClassifier
from collections import deque
import numpy as np

def initialize(context):
    context.security = sid(698) # Boeing
    context.window_length = 10 # Amount of prior bars to study

    context.classifier = RandomForestClassifier() # Use a random forest classifier

    # deques are lists with a maximum length where old entries are shifted out
    context.recent_prices = deque(maxlen=context.window_length+2) # Stores recent
prices
    context.X = deque(maxlen=500) # Independent, or input variables
    context.Y = deque(maxlen=500) # Dependent, or output variable

    context.prediction = 0 # Stores most recent prediction

def handle_data(context, data):
    context.recent_prices.append(data[context.security].price) # Update the recent p
    if len(context.recent_prices) == context.window_length+2: # If there's enough re
```

Description
This *Analyze Edit Page* is the page that appears after clicking the *Create Tab/Fork Tab* and going through the creation /fork process or clicking on the *Fork Button* on the *Analyze Page*. This page allows the user to create, edit and build their financial algorithms and is the page a User would spend most time on the most.

The main component in on the page is the code editor in the middle. As its name states, it allows the user to create and edit their Algorithms in the QLang Language.

The right panel, the *Editor Actions Panel* allows the user to do a multitude of things.

On clicking the *Build Button*, the user may compile and run their algorithm will produce a report that will be shown on the *View Analysis Page*. It shows the metric results of the user's algorithms such as *Sharpe Ratio*, a metric of the risk from its returns of an algorithm

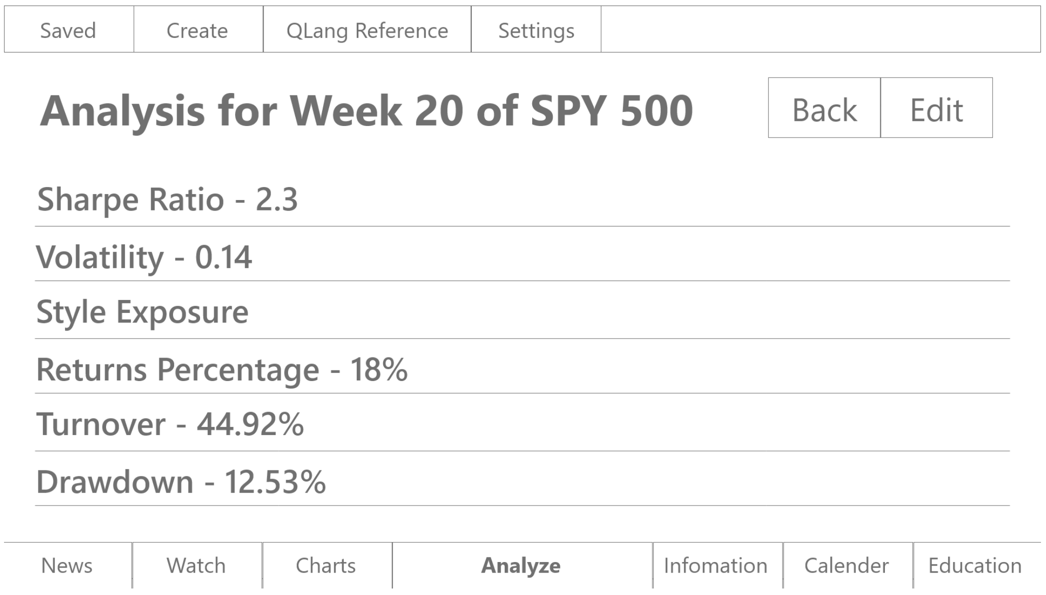
The *Versions Button* shows a past version of codes that were saved, like a repository version history. It simply shows a history list of codes all in a single page.

The *Fork This Button* is simply a quicklink for the user to fork the current project into a new one.

The *Thrash This Button* is a link for the user to delete what he is currently doing for good.

The bottom 4 buttons are very straightforward. *Black and White Theme Button changes the theme*, The 2 bottom button changes the Font Size in the editor.

2.1.1.11 Analysis View Page - Collapse/Expanded



Analysis for Week 20 of SPY 500

Sharpe Ratio - 2.3

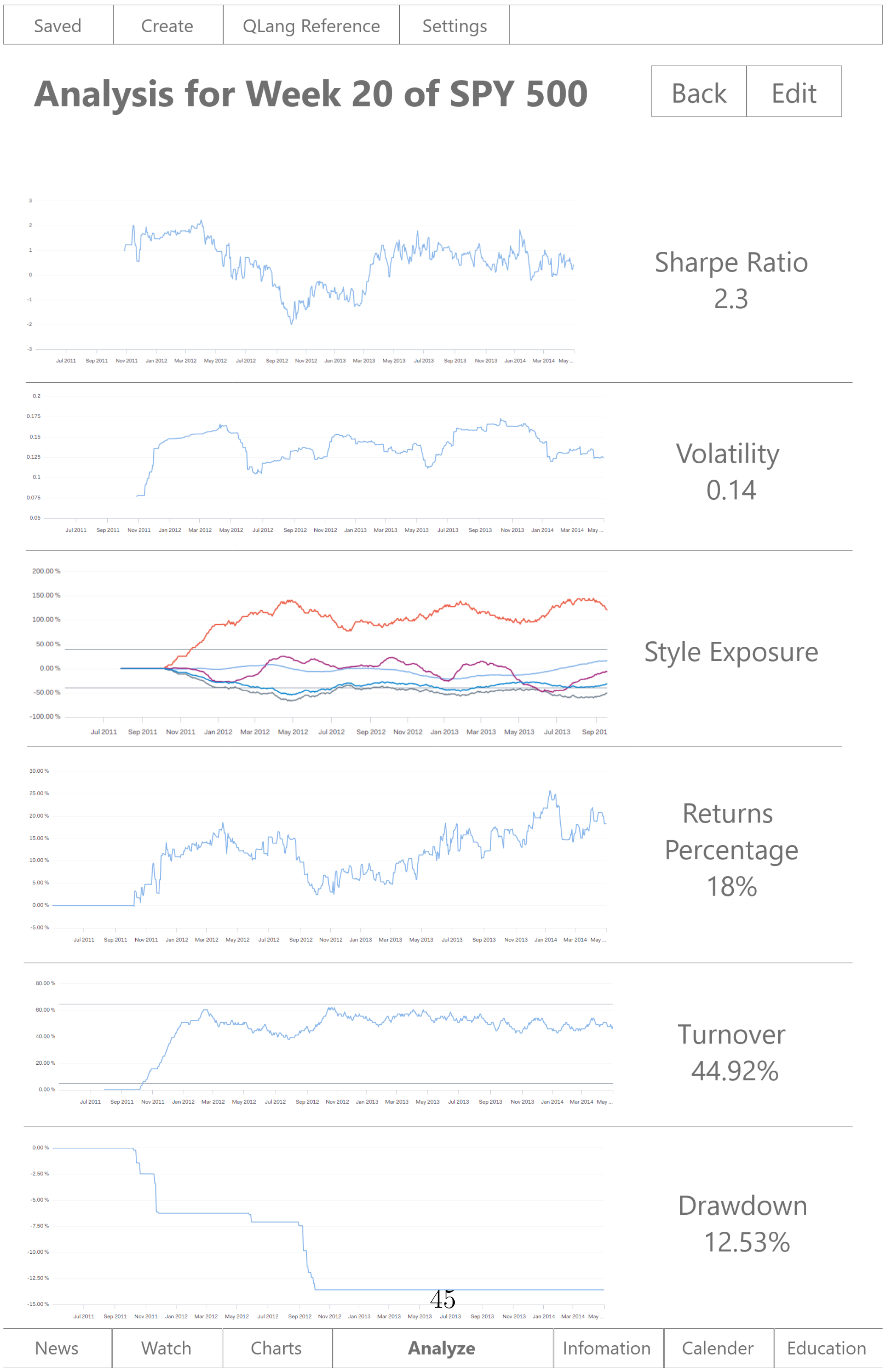
Volatility - 0.14

Style Exposure

Returns Percentage - 18%

Turnover - 44.92%

Drawdown - 12.53%



Analysis for Week 20 of SPY 500

Sharpe Ratio
2.3

Volatility
0.14

Style Exposure

Returns
Percentage
18%

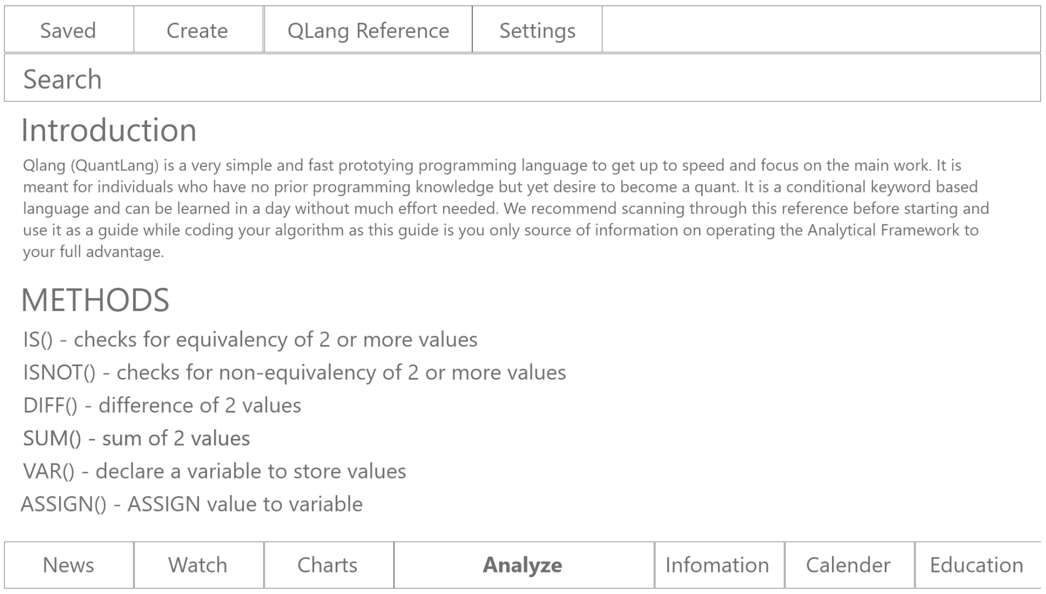
Turnover
44.92%

Drawdown
12.53%

Description
This *Analysis View Page* is the page that the user goes to view the Analysis Results and is also shown after the algorithim is built and executed in the edit page. The top 2 images show the collapse view and expanded view respectively. The expanded view appears after you click the tab resulting in the graph visualization appearing.

The 2 buttons beside the title of the analysis are quite simple. The *Edit Button* allows you to go back to the code editor shown in the previous section. The *Back Button* if directed from the editor page does not appear, but only when directed from the *Analysis Saved Tab*.

2.1.1.12 Analyze QLang Reference Tab - Collapse/Expanded

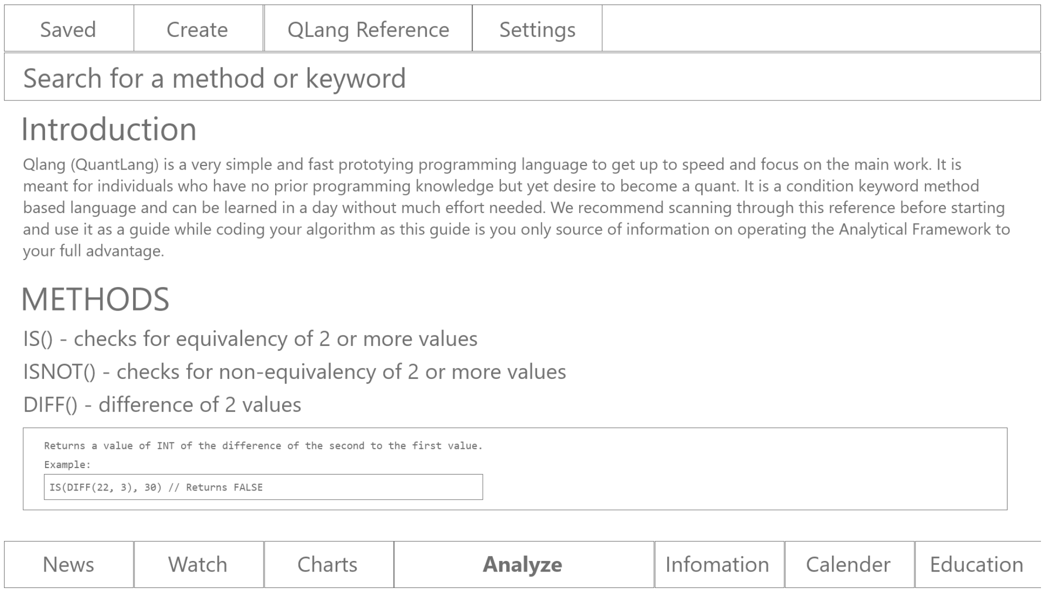


Introduction

Qlang (QuantLang) is a very simple and fast prototying programming language to get up to speed and focus on the main work. It is meant for individuals who have no prior programming knowledge but yet desire to become a quant. It is a conditional keyword based language and can be learned in a day without much effort needed. We recommend scanning through this reference before starting and use it as a guide while coding your algorithm as this guide is you only source of information on operating the Analytical Framework to your full advantage.

METHODS

IS() - checks for equivalency of 2 or more values
ISNOT() - checks for non-equivalency of 2 or more values
DIFF() - difference of 2 values
SUM() - sum of 2 values
VAR() - declare a variable to store values
ASSIGN() - ASSIGN value to variable



Introduction

Qlang (QuantLang) is a very simple and fast prototyping programming language to get up to speed and focus on the main work. It is meant for individuals who have no prior programming knowledge but yet desire to become a quant. It is a condition keyword method based language and can be learned in a day without much effort needed. We recommend scanning through this reference before starting and use it as a guide while coding your algorithm as this guide is you only source of information on operating the Analytical Framework to your full advantage.

METHODS

IS() - checks for equivalency of 2 or more values

ISNOT() - checks for non-equivalency of 2 or more values

DIFF() - difference of 2 values

Returns a value of INT of the difference of the second to the first value.

Example:

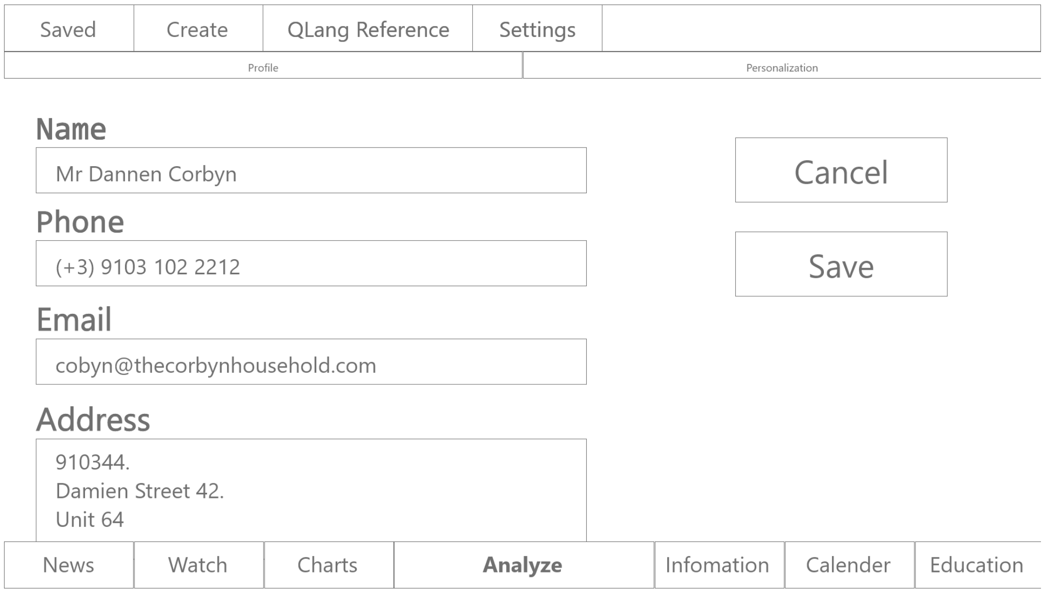
```
IS(DIFF(22, 3), 30) // Returns FALSE
```

Description

This *QLang Reference Tab* is the tab the user goes to refer to language documentation while coding. The QLang language is a conditional prototyping keyword method based programming language created for Users to use in this Trading Terminal to prototype financial algorithms.

The respective 2 images show it in different state. The first, when it is collapsed in its default state and when it's opened after a section of documentation is clicked upon to show examples or further detailed documentation.

2.1.1.13 Analyze Settings Tab/ Settings Page - Profile/Personalization

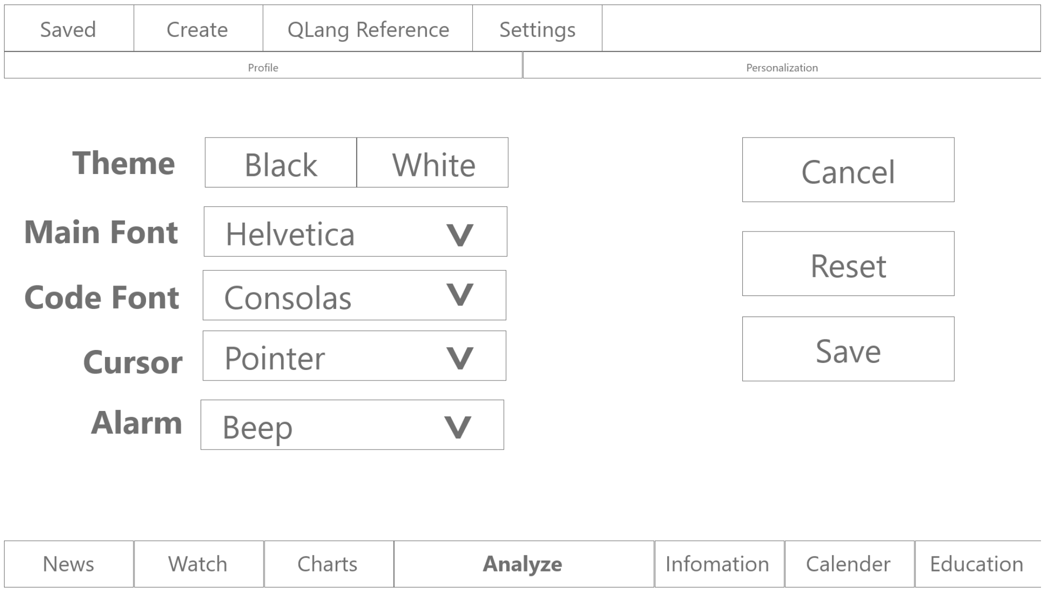


Name
Mr Dannen Corbyn

Phone
(+3) 9103 102 2212

Email
cobyn@thecorbynhousehold.com

Address
910344.
Damien Street 42.
Unit 64



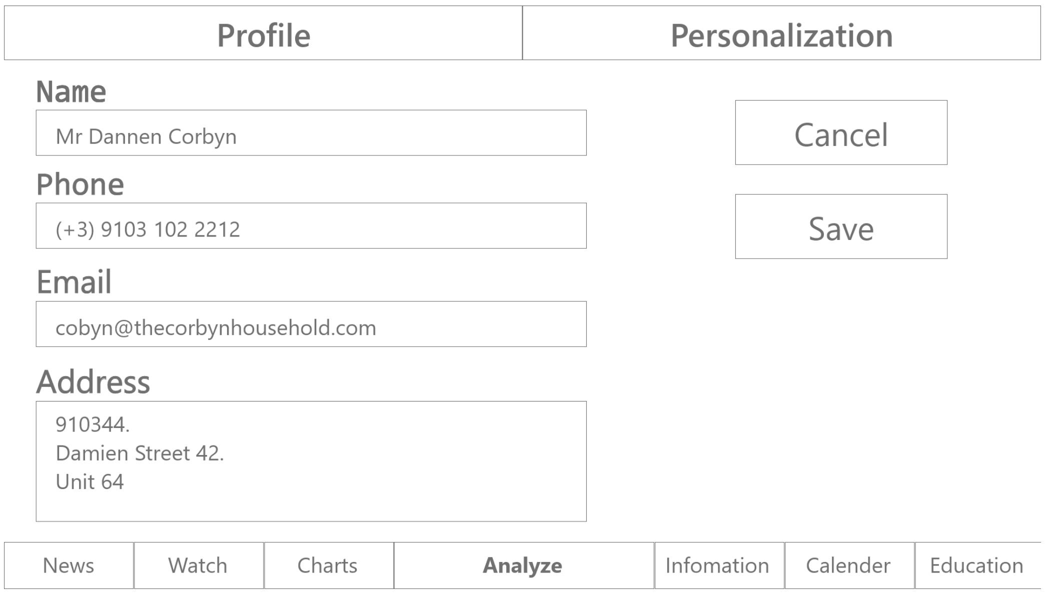
Theme	Black / White
Main Font	Helvetica
Code Font	Consolas
Cursor	Pointer
Alarm	Beep

Description

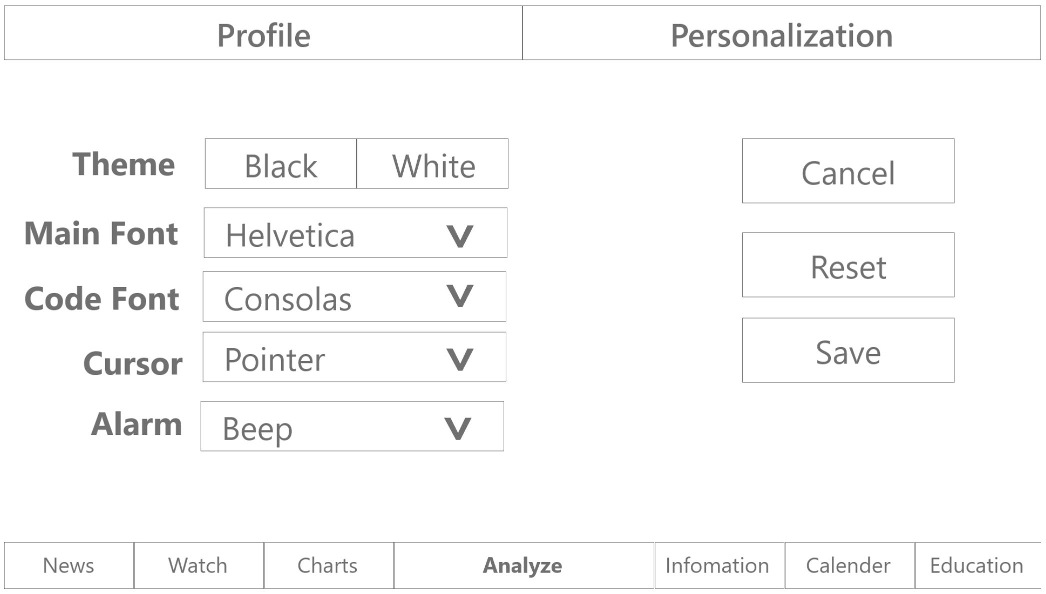
This *analyze Settings Tab* is where the user goes to change settings of the terminal, it's a quicklink to the same page the *Settings Button* on the *News Page* has but in a slightly different format as shown in the bottom 2 images (it does not have the *Analayze's Page Top Toolbar*).

The Tab/Page is split into 2 sub tabs. With the Profile tab, the User may change information relating to themselves like Name, Phone, Email and Address in the respective fields. With the Personalization tab, the user may change details relating to design of the terminal. Such as main font, code font, Theme of the overall application, mouse cursor and alarm sound.

The 3 buttons on the left are straightforward. Cancel to stop. Reset to go back to default and Save to save the settings changed.

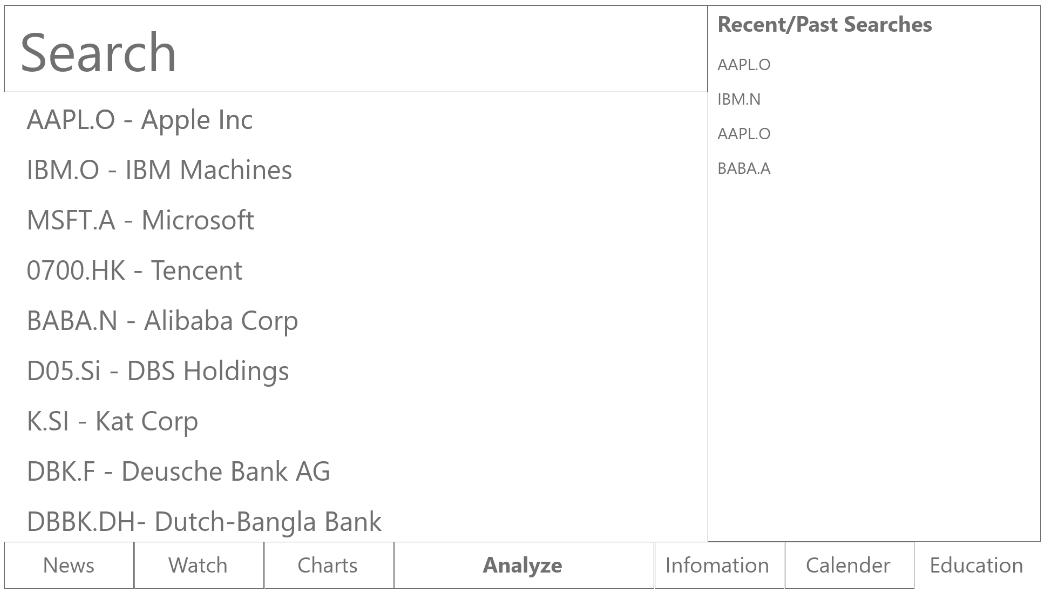


Name
Mr Dannen Corbyn
Phone
(+3) 9103 102 2212
Email
cobyn@thecorbynhousehold.com
Address
910344.
Damien Street 42.
Unit 64



Theme	Black / White
Main Font	Helvetica
Code Font	Consolas
Cursor	Pointer
Alarm	Beep

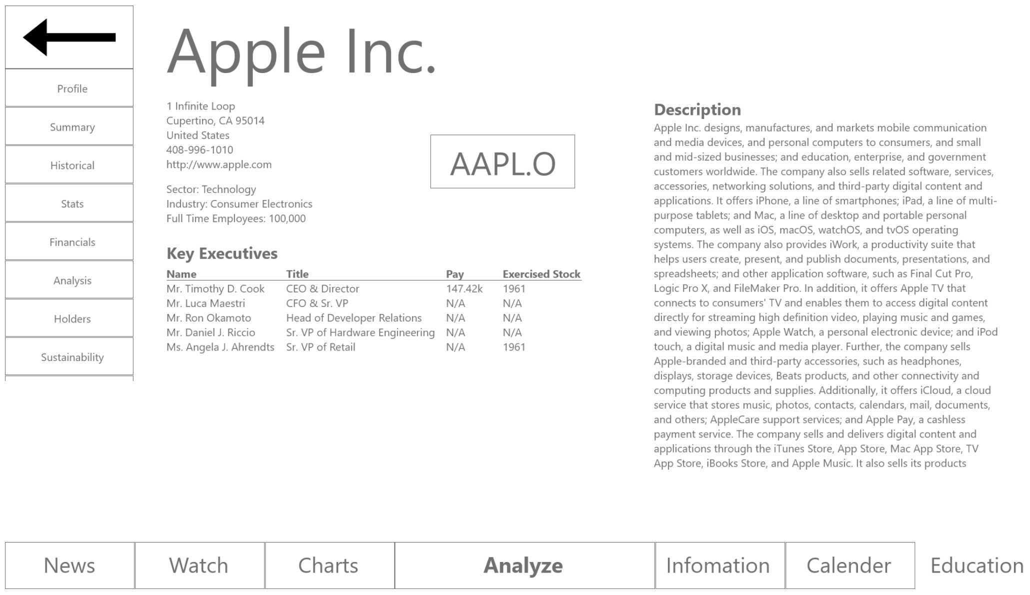
2.1.1.14 Information Page



AAPL.O - Apple Inc
IBM.O - IBM Machines
MSFT.A - Microsoft
0700.HK - Tencent
BABA.N - Alibaba Corp
D05.Si - DBS Holdings
K.SI - Kat Corp
DBK.F - Deusche Bank AG
DBBK.DH- Dutch-Bangla Bank

Recent/Past Searches

AAPL.O
IBM.N
AAPL.O
BABA.A



Apple Inc.

1 Infinite Loop
Cupertino, CA 95014
United States
408-996-1010
http://www.apple.com

Sector: Technology
Industry: Consumer Electronics
Full Time Employees: 100,000

AAPL.O

Key Executives

Name	Title	Pay	Exercised Stock
Mr. Timothy D. Cook	CEO & Director	147.42k	1961
Mr. Luca Maestri	CFO & Sr. VP	N/A	N/A
Mr. Ron Okamoto	Head of Developer Relations	N/A	N/A
Mr. Daniel J. Riccio	Sr. VP of Hardware Engineering	N/A	N/A
Ms. Angela J. Ahrendts	Sr. VP of Retail	N/A	1961

Description

Apple Inc. designs, manufactures, and markets mobile communication and media devices, and personal computers to consumers, and small and mid-sized businesses; and education, enterprise, and government customers worldwide. The company also sells related software, services, accessories, networking solutions, and third-party digital content and applications. It offers iPhone, a line of smartphones; iPad, a line of multi-purpose tablets; and Mac, a line of desktop and portable personal computers, as well as iOS, macOS, watchOS, and tvOS operating systems. The company also provides iWork, a productivity suite that helps users create, present, and publish documents, presentations, and spreadsheets; and other application software, such as Final Cut Pro, Logic Pro X, and FileMaker Pro. In addition, it offers Apple TV that connects to consumers' TV and enables them to access digital content directly for streaming high definition video, playing music and games, and viewing photos; Apple Watch, a personal electronic device; and iPod touch, a digital music and media player. Further, the company sells Apple-branded and third-party accessories, such as headphones, displays, storage devices, Beats products, and other connectivity and computing products and supplies. Additionally, it offers iCloud, a cloud service that stores music, photos, contacts, calendars, mail, documents, and others; AppleCare support services; and Apple Pay, a cashless payment service. The company sells and delivers digital content and applications through the iTunes Store, App Store, Mac App Store, TV App Store, iBooks Store, and Apple Music. It also sells its products

Description

This *Information Page* is the page that appears after clicking the `Information Button` on the *Toolbar*. It allows view *Fundamental Data* on a company. Such info would be crucial to users who wish to run analysis on a certain company stock but would like to know more about them.

The First image is the first thing a user sees on entry. They may type a company's name or stock ticker into the search bar on the top which will filter out the result below which they may click on to move to the second page. On the Right Panel, it shows quicklinks to recent searches that users may click on to access when they wish to.

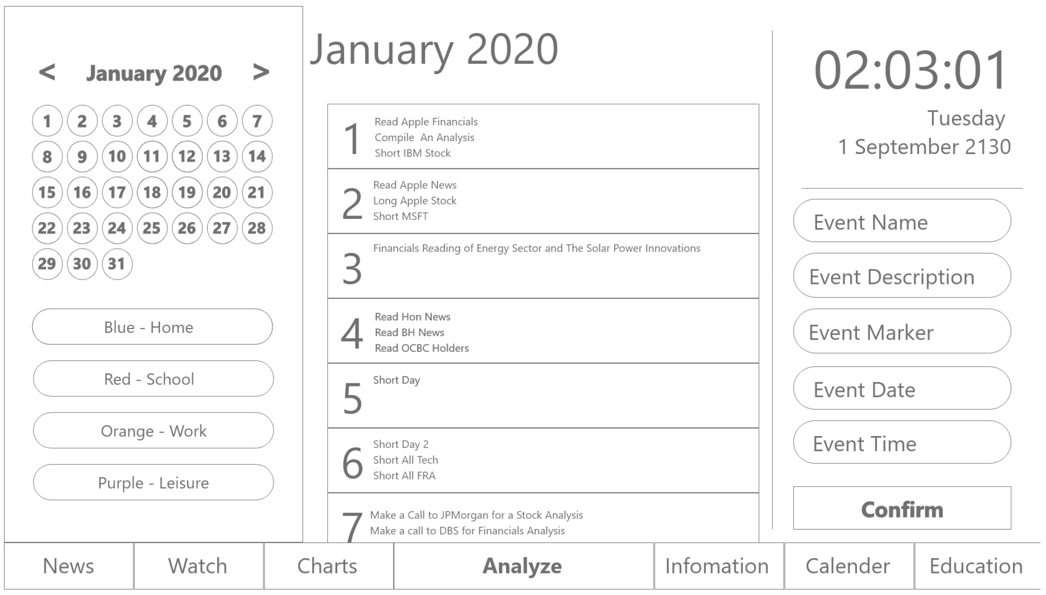
The Second Image is the image seen after clicking on the search result. Currently, the Profile Tab is selected and hence profile details about the company is shown on the right side. Details such as its name, address, contact details, board of executives, description and most importantly, its Stock Symbol (in this case, AAPL.O for Apple Inc).

The left panel shows quick links to other parts of the data. The top arrow is simply the back button to the search page.

1. Profile - Profile details about company

2. Summary - Stock Price Summary Overview such as previous close data.
3. Historical - View of all past pricing data
4. Stats - Stock Price Detailed Overview and interpretations of them like the averages and profile margin.
5. Financials - Financial data about a company like it's re venue and earnings
6. Analysis - Prediction Estimates of a company's details like its growth and earnings
7. Holders - List of Major Shareholders and Fund Holders of a company as well as a few insider information from the Holders
8. Sustainability - Eco-Friendliness Status of a company such as it's environmental effects ratings performance in comparison to its peers.

2.1.1.15 Calender Page



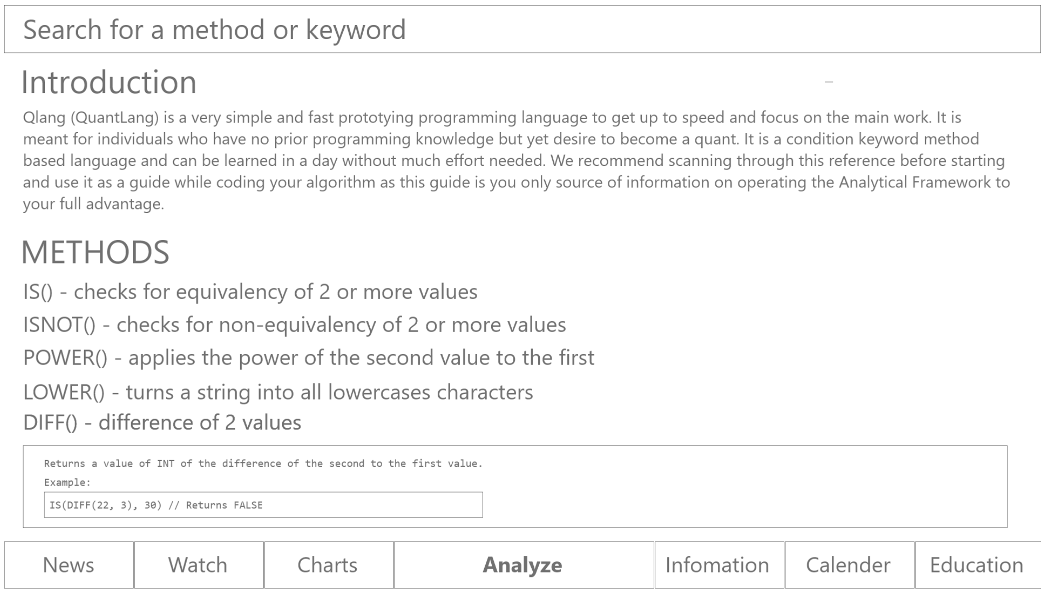
Description
This *Calender Page* is the page that appears after clicking the `Calender`

`Button` on the *Toolbar*. It allows users to use the terminal to plan their day or task lists so as to be organized. A utility page that would be very useful to many Users especially in a time sensitive industry like Finance that this application is for.

The Left Panel allows the user to view through and scroll through the months as well as gain info on which color marker represents what category. As the user scrolls through the months through the month view mini calender on the top left via the left and right arrows, the center section reacts as accordingly to show the respective month.
The bottom left section of the left panel shows the color codes for the different categories.

The Right panel has multiple things. On the top, the time and date details are shown so the user may know what is current. On the bottom, there are text fields that the user may use to input details about event he wishes to add like name, description, marker(the user choose the marker for that event) and datetime details of the event. On confirmation, the User clicks on the confirm button all the way at the bottom in which the fields will clear and the event will pop on the center portion if the date the event is added to is in view.

2.1.1.16 Education Page


Search for a method or keyword

Introduction

Qlang (QuantLang) is a very simple and fast prototying programming language to get up to speed and focus on the main work. It is meant for individuals who have no prior programming knowledge but yet desire to become a quant. It is a condition keyword method based language and can be learned in a day without much effort needed. We recommend scanning through this reference before starting and use it as a guide while coding your algorithm as this guide is you only source of information on operating the Analytical Framework to your full advantage.

METHODS

IS() - checks for equivalency of 2 or more values

ISNOT() - checks for non-equivalency of 2 or more values

POWER() - applies the power of the second value to the first

LOWER() - turns a string into all lowercases characters

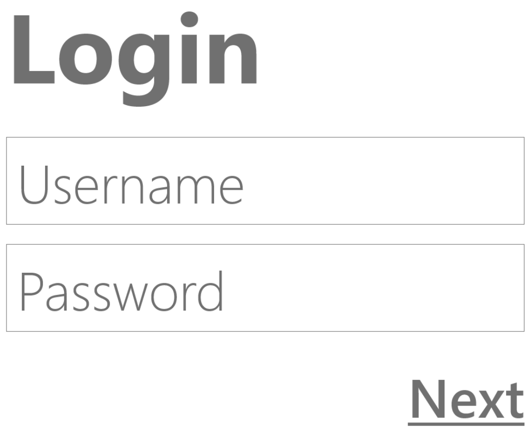
DIFF() - difference of 2 values

```
Returns a value of INT of the difference of the second to the first value.
Example:
IS(DIFF(22, 3), 30) // Returns FALSE
```

Description

The education page is simply education for how to use the application. For now as the application is not that complex, the page will simply show Help on how to use QLang (Quant Language).

2.1.1.17 Login



Description
Login page, no explanation needed. Only when logged in may the user view the main pages.

2.1.2 Main Dashboard Technology Choice and Rationale

Our Choice of technology for the Main Dashboard will be Angular 6 with SASS. Angular 6 is the 6th version of an open source client-side all-in-one framework created by Google to solve Maintainability and Scale issues that is used by Millions . It is financially and manpower supported by Google as well as Microsoft and many other companies. It uses the *Typescript Language*, a super set of *Javascript* with a type system for better debugging, auto complete and Maintainability with writing safe code; coincidentally created by Microsoft.

SASS on the other hand is a compiled superset of CSS while the same purpose, styling web pages created in HTML. However unlike CSS, it has extra features such as mixins which are functions/methods in CSS but for styling, more terse syntax such as not needed semicolon endings nor curly braces speeding up development time just like Angular6.

Hence these 2 main technologies are a good pair as they both come pre-installed with the Angular CLI and are a good pair to reduce development time needed.

2.1.2.1 Easy Setup

Angular 6 because it has a very easy installation and startup up process simply with

```
sudo apt-get install nodejs npm
npm install -g @angular/cli
ng new myappname
```

This makes it a very good candidate in this Microservices project as every server is dockerised and hence software needs to be installed everytime a new node is created.

Being simple and easy allows quicker prototyping too which this project is doing.

Moreover, the setup is not only limited to code but development environment too such as having code auto-complete in *VSCode*, a popular text editor already configured on installation.

2.1.2.2 Opinionated and Efficient

Angular is opinionated meaning that it has one way and only one way to do a certain action. This makes it good for prototyping as the developer does not have to spend too much time thinking of software paradigms and hence just focus solely on developing the product and new features. Choice is non existent and is made for the developer by the creators of angular creating a smooth workflow without hiccups as most things are simply plug and play.

2.1.2.3 Big Ecosystem

As Angular is an open source and community based project, there are many addons created by the community for it. This means there are many ready made code available freely for download that can be used in or project reducing development time.

Angular being made by Google gave it a a large name behind the project, this meant that many individuals trust in the product due to the reputation Google meaning the product will probably be good and maintained for a long while. Hence many have invested time in contributing to it on *Github* as well as developing addons for Angular and this has naturally given Angular a large ecosystem since many people including large companies contribute.

2.1.2.4 Component Based

Angular6 is component based meaning that elements are grouped into chunks of code to make a component. This gives it many benefits.

With a component based system, as code is segregated, it makes it easy to structure and break down the application into development stages during prototyping leading to better and more organized code based and work flow and hence even faster development time !

With a component based system, there is a hierarchy of parent and child components as components can be nested within each other and within each other again all the way down. This makes it clear to the developers who plan the app how to structure the app for a smooth dataflow downwards leading to a better long term maintainability.

2.1.2.5 Type-System

Many organisations' developers who create and maintain large client side applications have always complained about the lack of maintainability due to *JavaScript* allowing any variable to hold anytime of information. This leads to **More Bugs** as the wrong type of data might be passed to the wrong variable (the only way to know what type of data to pass is through self written documentation or pure memory which might not work when working across a diverse team of many individuals with different personalities, habits and coding style) and may can happen both by a "Typo" mistake or simply overlooking to convert the datatype before passing it into the system.

Naturally, many have tried to combat this by means of Continuous Testing with the downside of more work. With a Type-System, it is impossible to pass the wrong data to the wrong variable or data container as the TypeScript Language Compiler will reject compilation with errors and inform the developer of the offending section of code.

Hence a Type-System eliminates the need for Testing as well as Reduces Bugs at the same time. A Type-System furthers enhances smarter auto-completion as the IDE now knows what functions or variables to auto-complete with based of the data-type of the element being completed.

2.1.2.6 SASS DRY

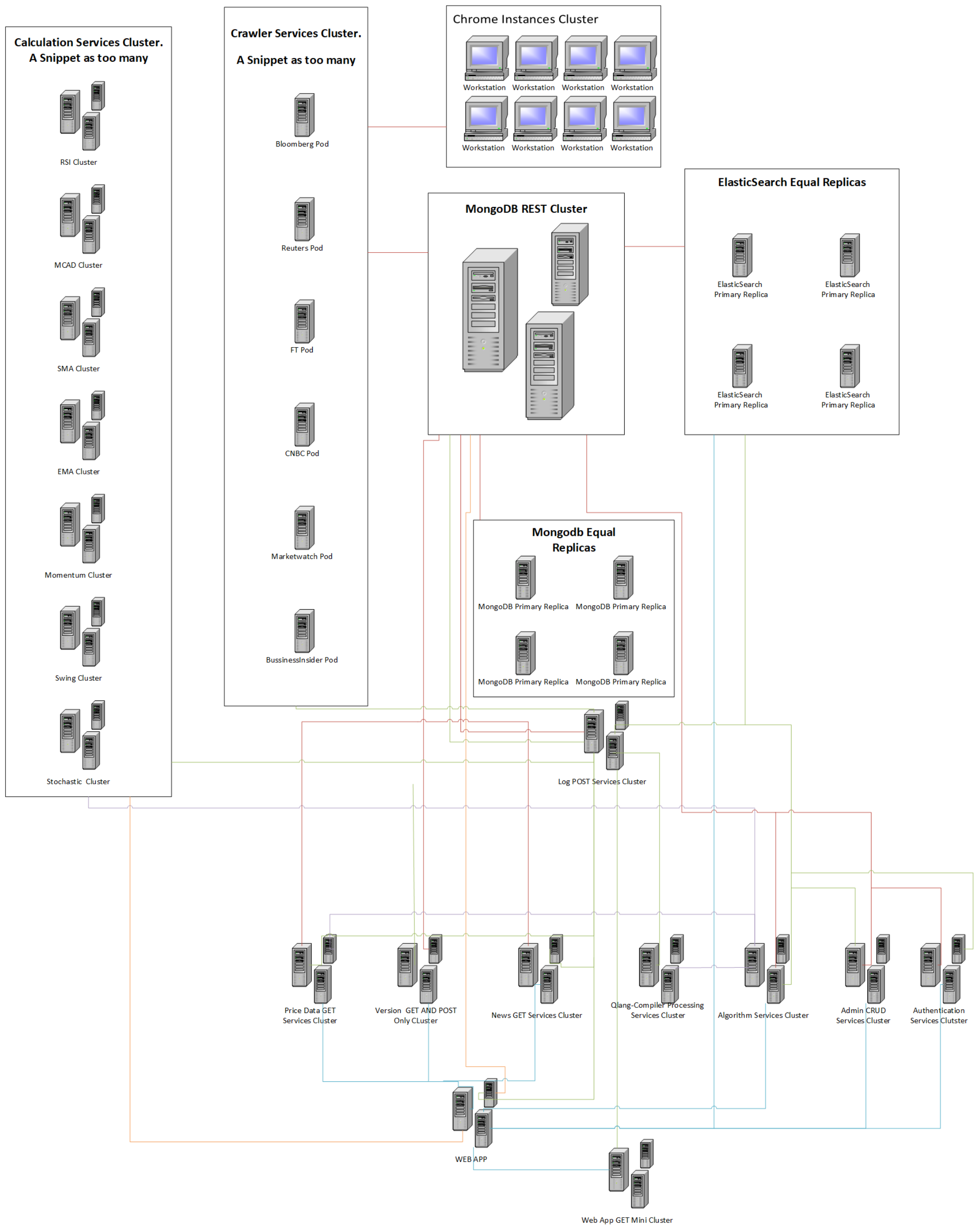
SASS is a superset of CSS meaning it is compatible with CSS but with additional features.

SASS has mixins like CSS, which are inherently functions. This allows SASS to store a block of commonly used code into a mixin which the can be reused anywhere in the code as needed eliminating the need for typing the code again. This keeps our code DRY, Dont Repeat yourself.
Similarly to mixins, CSS does not have variables. This means when a developer wants to change a value commonly used, they have to replace it at every location which makes for a redundant exercise that can be better spent developing other features. With variables, values may be referenced anywhere in the code via a variable which references one location in the code without copy pasting. When the variable's value is changed, the values in the locations where the variable is used are changed at the same time speeding up development time and keeping it DRY.
Finally, SASS unlike CSS allows interfile linking. This would mean one may separate the code into different files instead of one who file leading to a more organized process of development. This makes it a good companion for Angular as the framework is component based meaning that SASS code can be sectioned by components if desired allowing better code organization parity to the markup content code in Angular.

2.2 Microservices Overview

We have opted for a complex but low latency, ship first optimise later and multiple points of failures for fail safe architecture. These services all run on NodeJS within a Docker Container running Ubuntu Linux OS that is orchestrated by Kubernetes. They are all running locally on a single node but multiple pods to reduce latency when cross communicating between Pods while there are multiple pods to reduce processes wait time. It is to be noted there is no central server, router or dealer. All UI Elements communicate directly with the servers as these servers only serve one domain and only one to reduce fault issues that might appear from over cross communication.

2.2.1 Microservices Architecture Diagrams



2.2.2 Microservices API Documentation

2.2.2.1 Web App GET Mini Cluster

The *Web App GET Mini Cluster* is a Mini Cluster of Pods also known as "Containers". This is a special case that it's a Mini Cluster meaning not as heavy duty as a normal cluster as it merely serves to provide visitors that visit the web application the Angular Application and nothing computationally intensive.

Similarly to any other service, any operations performed are logged into the Log Service faults may be tracked if it happens.

Main Endpoint

```
GET /
Request Body: NULL
Response Body: Angular Application
```

2.2.2.2 Admin CRUD Services Cluster

The *Admin CRUD Services Cluster* is a Normal Cluster of Pods also known as "Containers". This service cluster has all the CRUD (Create, Read, Update, Delete) function as it is not used that much though it does perform database operations. This service cluster is used to perform operations on admin data such as user logins and details.

This MicroService is slightly bigger than the usual size as it is rarely used, more so than any other cluster. Usually it's used only when a user signs up, signs in or edits their information and settings.

Similarly to any other service, any operations performed are logged into the Log Service faults may be tracked if it happens.

Login Endpoint

```
POST /login
Request Body: {
   username: string
   password: string
}
Response Body: {
   authenticated: bool
   session_string: jwt
}
```

Logout Endpoint

```
POST /logout
Request Body: {
   session_string: jwt
}
Response Body: {
   logout_status: bool
}
```

Get Profile Endpoint

```
GET /profile?jwt={string}
Request Body: NULL
Response Body: {
   name: string,
   phone: int,
   email: string,
   address: string
}
```

Edit Profile Endpoint

```
PUT /profile?jwt={string}
Request Body: {
   name: string,
   phone: int,
   email: string,
   address: string
}
Response Body: {
   success: bool
}
```

Get Personalization Endpoint

```
GET /personalization?jwt={string}
Request Body: NULL
Response Body: {
   theme: string,
   mainfont: int,
   codefont: string,
   cursor: string,
   alarm: string
}
```

Edit Personalization Endpoint

```
PUT /profile?jwt={string}
Request Body: {
   theme: string,
   mainfont: int,
   codefont: string,
   cursor: string,
   alarm: string
}
Response Body: {
   success: bool
}
```

Reset Personalization Endpoint

```
GET /profilereset?jwt={string}
Request Body: NULL
Response Body: {
   success: bool
}
```

Reset Personalization Endpoint

```
GET /profilereset?jwt={string}
Request Body: NULL
Response Body: {
success: bool
}
```

2.2.2.3 Price Data GET Services Cluster

The *Price Data GET Services Cluster* is a Normal Cluster of Pods also known as "Containers". This service cluster has is simply a GET service to retrieve price data from the database, a middleman interface.

This service is used mainly by 2 other services/applications. It is used by the *Algorithm Services Cluster* as algorithms need data to be simulated against and the Main Webapp when user views historical price data in the information tab.

Similarly to any other service, any operations performed are logged into the Log Service so faults may be tracked in any event and if JWT is used, the authentication service is consumed to check user authenticity.

Price GET Endpoint with pagination
Each pagination returns next 20 results.

```
GET /prices?pagination={int}
Request Body: NULL
Response Body: {
   {
      datetime: string,
      price: double
   },
   ...
}
```

Price GET Endpoint by range

```
GET /price?datetimestart={string}&datetimeend={string}
Request Body: NULL
Response Body: {
   {
      datetime: string,
      price: double
   },
   ...
}
```

Price Websocket Endpoint with pagination
Each pagination returns next 20 results.

```
WEBSOCKET /prices
SUBSEQUENT
Request Body: {
   pagination: int
}
Response Body: {
   {
      datetime: string,
      price: double
   },
   ...
}
```

2.2.2.4 Version GET and POST Services Cluster

The *Version GET and POST Services Cluster* is a Normal Cluster of Pods also known as "Containers". This service cluster has both GET and POST services only to retrieve and commit QLang Code to and from the database, a middleman interface. It must be noted that a GET or POST endpoint is never on the same Pod. Essentially it is a self built versioning engine that saves code history that may be used client-side to find diffs.

This service is used mainly the Main Webapp when a user saves their QLang code in the editor while editing their algorithm.

Similarly to any other service, any operations performed are logged into the Log Service faults may be tracked if it happens.

Version GET Endpoint with pagination
Each pagination returns next 20 results.

```
GET /version?jwt={string}&pagination={int}
Request Body: NULL
Response Body: {
{
   versionid: string/uuid,
   code: string
},
...
}
```

Version POST Endpoint

```
POST /version
Request Body: {
   jwt: string,
   code: string
}
Response Body: {
   success: bool
}
```

2.2.2.5 News GET Services Cluster

The *News GET Services Cluster* is a Normal Cluster of Pods also known as "Containers". This service cluster has only GET services only to retrieve News data that is crawled by the Crawlers Clusters from the database, a middleman interface.

This service is used mainly the Main Webapp to retrieve and display News data and headlines on the News page. The endpoitn would.

Similarly to any other service, any operations performed are logged into the Log Service faults may be tracked if it happens.

News GET Endpoint with pagination
Each pagination returns next 20 results.

```
GET /news?jwt={string}&pagination={int}
Request Body: NULL
Response Body: {
{
   title: string,
   content: string,
   date: string,
   time: int
},
...
}
```

2.2.2.6 QLang Compile Processing Endpoint

The *QLang Compile Processing Endpoint* is a Normal Cluster of Pods also known as ”Containers”. This service cluster has only POST services only to pass QLang code. They do not communicate with any datastores and are stateless in nature, a processing cluster.

This service is used by the *Algorithm Services Cluster* as before it executes the code written by the User to test their Algorithm it needs to compile the Qlang code first to be usable. This is a processing service.

Similarly to any other service, any operations performed are logged into the Log Service so faults may be tracked in any event and if JWT is used, the authentication service is consumed to check user authenticity.

QLang Compile Processing Endpoint

```
POST /compile?jwt={string}
Request Body: {
   code: string
}
Response Body: {
   machinecode: string
}
```

2.2.2.7 Algorithm Services Endpoints

The *Algorithm Services Endpoints* is a Normal Cluster of Pods also known as ”Containers”. This service is one of the larges services and is the main service that serves the analyze page to allow it to perform algorithm executions that the user has created. It is the controller of the whole Algorithm Analysis pipeline and has the follow responsibilities:

1. Commit Logs
2. Commit Code Versions
3. Calculate algorithms with supplemented with high performance calculation service.
4. Send code to QLang Compile Service to compile the user's code
5. Execute code returned from QLang COmpile service
6. Return results to the real time chart client side
7. **Perform overall calculation like volume percentage change**

It is only used by the client side.
We have opted for a Websocket connection over HTTP as the client side will connect to the service till the algorithm has been done doing its work and have finished sending all the data points of results and hence will be a sustained connection which With HTTP we would have to keep polling for details with multiple requests as there is no push request and that multiple HTTP request will be needed hence increasing latency and work since a single requests is less heavy than multiple request. This is good too as in future the service may be extended with duplex communication as needed.

The reason for a service instead of building it built the client is so that we may offload the work from the client so that it may focus on displaying he chart and not take up resources for the user pc and hence a good user experience. The algorithms perform loads of waiting too and hence we would not want it to have programs in a wait state on the client side as we want the user to freely refresh. Furthermore, centralizing into a service means that if anything goes wrong i would not affect the client and that there is a neater code beign centralized into a service, it's own domain.

Similarly to any other service, any operations performed are logged into the Log Service so faults may be tracked in any event and if JWT is used, the authentication service is consumed to check user authenticity.

```
EXAMPLE
WEBSOCKET /algoservice
START
Request Body:{
code: string
id: string
}
THEN
Response Body: {
   status: string
   lengend: {
      SMA: blue
      EMA: red,
      BB: orange,
      AAPL: silver,
      IBM: gold
   }
}
THEN for every datapoint
Response Body: {
datetime-for-point: string
point-value: string
}
```

QLang Compile Processing Endpoint

```
POST /compile?jwt={string}
Request Body: {
code: string
}
Response Body: {
machinecode: string
}
```

2.2.2.8 Authentication Services Endpoint

The *Authentication Services Endpoint* is a Normal Cluster of Pods also known as "Containers". This service cluster has only GET services to check for user's token authenticity that is provided to the user for the session on login.

This service is used by every service that may demand a JWT token so as to keep out unauthorized operators.

We have opted to use jwt tokens as it is industry standard, the obvious choice

---

Authentication Endpoint

```
GET /auth?jwt={string}
Request Body: NULL
Response Body: {
   authenticated: bool
}
```

2.2.2.9 Log POST Services Cluster

The *Log POST Services Endpoint* is a Normal Cluster of Pods also known as "Containers". This service cluster has only POST services to commit Logs to the database that may be used to find faults in a failure event. It serves a simple interface to the complex operations behind.

This service is used by every service to commit logs and is non accessible to any user.

Log POST Endpoint

```
POST /log
Request Body: {
   username: string
   service-node-name: string
   log: string
   datetime: string
}
Response Body: {
   success: bool
}
```

2.2.2.10 Calculation Services Clusters

The *Calculation Services Clusters* is a Cluster of Normal Clusters of Pods also known as "Containers". This service cluster's purpose serves as a companion for the *Algorithm Services Cluster* to perform complex calculations. One may think of these clusters of clusters of Calculation Pods to partition it away from the main server, offsetting the load as these complex calculations hog the main server time preventing other request from coming in addition to being heave computationally intensive.

Calculations performed might have as much to a hundred thousand data points in a 3 dimensional matrix or even more for popular stocks.

These endpoint are only communicated via the *Algorithm Services Endpoint* and not user accessible directly.

It must be noted this endpoint unlike the other endpoints is **Websocket based** and hence it will be a sustained connection till it ends due to the nature of algorithm calculation. Calculation scheduling will be managed by the Algorithm Service while this is merely a dumb blackbox Websocket server for computationally intensive operations.

Amount of Values passed and returned depending on what kind of algorithm the Pod/Container Serves however the type of values passed are all the same, doubles.

It is to be noted that a Pod Cluster is allocated per calculation type AKA "indicator" such as *Simple Moving Average*(SMA) or *Relative Strength Index*(RSI). How many indicators there are is dependent of how many different calculation types QLang currently support. (Dependent on the how much time or effort I may have to implement such QLang features)

Calculation Services Endpoint

```
WEBSOCKET /calc_{indicator}
Request Body: {
   values: double_array
}
Response Body: {
    results: double_array
}
```

2.2.2.11 Crawler Services Clusters

The *Crawler Services Clusters* is a Normal Cluster of Pods also known as ”Containers”. These Crawler Services serve to actually web crawl news data from different financial news websites that users may use to decide how they might analyze their algorithm and target it for that particular date and time of the even (eg. Plane crash of Boeing = Stock price drops). The news data is then committed into the database.

As there is not much activity in crawling, this simply is a cluster of pods instead of a nested cluster. Each pod being dedicated to one website source like financial times or Bloomberg.

This endpoint is not actually controlled by any other service nor user nor webapp. It simply is executed on a time basis to check the respective website for the respective pod for updates and if there happens to be such, it commits the new news into the database that may be retrieved by the webapp or other services later from the database.

The endpoint communicates with 2 other clusters. It firstly communicates with the database where there happens to be an update in the news and hence data is transfered into the database. It secondly communicates with the *Chrome Instance Clusters* to crawl data via a Websocket client interface. A Chrome Instance is used instead of the classic Request via HTTP as most websites now demand Javascript to be used to render the page which the HTTP Request methods does not support.

The amount of pods for soley depends on the number of sites crawled, the more sites to crawl is implemented, the more pods there will be. One pod per site only and nothing more.

There are no important endpoints as it's not user triggered, however it has a utility endpoint that allows a crawl to be triggered ahead of time, the respective pod for the respective site url will be called once a GET Request is sent to the endpoint.

Crawl Services Endpoint

```
GET /crawl?url={string}
Request Body: Null
Response Body:
   crawltrigged: bool
}
```

2.2.2.12 Chrome Instances Services Clusters

The *Chrome Instances Services Clusters* is a Normal Cluster of Pods also known as "Containers". This cluster is simply just a cluster of pods containing Chrome Instances that may be controlled via websocket commands. These services contain no application/business logic and simply are just modified Chrome. They do not communicate with any datastores and are stateless in nature, a processing cluster.

The cluster is routinely used by the *Crawler Services Cluster* when it performs a crawl operation on schedule. These crawl operations work by the Crawl Cluster which contains the application logic sending Chrome commands via it's Websocket client to the Chrome Cluster which has a Websocket server per pod/instance to receive such commands and then sends back the result in which the *Crawl Services Cluster* may filter and apply application business logic.

Another way of seeing these chrome instances cluster are rendering engines for websites just that in this case, there is no visual feedback but a textual one.

Crawl Services Endpoint There are simply too many endpoints for the chrome instances to be documented meaningfully (480 A4 Pages), however if one wishes to read the documentation for it, look at we recommend visiting `https://raw.githubusercontent.com/ChromeDevTools/debugger-protocol-viewer/master/_data/tot/protocol.json`

2.2.2.13 Mongodb Equal Replicas

The *MongoDB Services Clusters* is a Normal Cluster of Pods also known as "Containers". This cluster is simply just a cluster of pods containing MongoDB Database Instances in a primary replica set kubernetes structure. Replica to reduce chance of failure and redundant data while accepting data additions from any pod in the cluster.

MongoDB is a free opensource document nosql database that is suitable for big data due to it's features like horizontal replication and multi sequential writes. It has no fixed schema and hence is good for prototyping.

Due to the heavy usage and input as well as output, we have opted for a minimum for 4 instances running at any point of time meaning there are 4 physical points of entry and output for data flow though virtually, at least a thousand as MongoDB is nosql and is suitable for large input streams for use cases like **Big Data** and for **Scale**.

Heavier usage might cause a higher chance of failure, hence why we have opted for replicas and not purely a single primary pod which the monolithic architecture tends to favour.

There is no HTTP endpoint as this is operated via a language API and it's a database hosted similar to a http service.

2.2.2.14 Mongodb Service Clusters

Unlike ElasticSearch, the MongoDB team do not offer a REST server as this it's their proprietary product and hence we will not use as we intend to keep costs low. However, there are other open source community created options like RestHeart(`https://restheart.org`) which we will be using.

RestHeart is basically a HTTP Server built upon the MongoDB SDK. It exposes REST Endpoints that one may used to create/update/delete or read documents into MongoDB without any coding but a simple plug and play. We will be opting to run a cluster of these that will be load balanced and provisioned based on the usage.
It is also fast, built with a Java Backend and the Famous Undertow Java Webserver, one of the fastest webservers in the world. It has a small memory footprint of 15mb only and low ram usage.
Most importantly, it has a ready made docker container making it easy to be deployed into our Microservices architecture !. Commercial support is provided with a fee too.

2.2.2.15 ElasticSearch Equal Replicas

The *ElasticSearch Equal Replicas* is a Normal Cluster of Pods also known as "Containers". This cluster is simply just a cluster of pods containing ElasticSearch Search Server Instances in a primary replica set kubernetes structure. Replica to reduce chance of failure and redundant data while accepting data additions from any pod in the cluster.

ElasticSearch is a database search server that is suitable for big data. We opted to use ElasticSearch for a simple reason as MongoDB full text search function is not that optimized unlike ElasticSearch which is search server that specializes in not storing data but reducing latency in text lookup and hence it's the right tool for the job.
Furthermore, ElasticSearch has added features such as search filters by text grammar complexity which though is not implemented in this project, may be used for future project enhancements; future proofing the project.

Due to the heavy usage and input as well as output, we have opted for a minimum for 4 instances running at any point of time meaning there are 4 physical points of entry and output for data flow though virtually, at least a thousand as ElasticSearch is a server suitable for large input/output data streams for use cases like **Big Data** and for **Scale**.

Heavier usage might cause a higher chance of failure, hence why we have opted for replicas and not purely a single primary pod which the monolithic architecture tends to favour.

ElasticSearch has further features, in addition to the ElasticSearch search server, there is an REST API layer implemented on top the server that was created by the ElasticSearch Team that we will be using directly to reduce development time and not have "NIH (not invented here) syndrome. For Documentation on the endpoints, look at `https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html`.

2.2.3 Microservices Features/Extra Features and Technology

2.2.3.1 Websockets

The Websocket technology was used in a few of our Services both client and server side.

1. Web App - Used to communicate with the Algorithm Services Cluster from the Analyze page to perform analysis over a extended period of time and get streaming back results real time.

2. Calculation Services Clusters - Websocket server utilized by the Websocket client in Algorithm Services Cluster to perform sustained extended connections to calculate and reduce latency

3. Crawler Services Cluster - Websocket Client Side internally to control Chrome Instance Services to crawl. The main service exposed to the webapp is still REST HTTP though internally it's not.

4. Chrome Instances Services Clusters - Websocket Server to control the Web Browser to crawl over an extended period of time.

3 Reasons Why

1. Pricing data migh include tens to millions of data points and hence it would be impractical, impossible to send a million HTTP Request and Receive a million HTTP responses as both the client and server may not have enough memory to store so much textual data for every request times the amount of user not enough amount of cache.

2. In building upon the previous point, it would be more sensible to have one long connection than multiple connections due to the startup time cost of a request which increases latency, one also may not desire to receive the same header information that the client already knows on every request.

3. As Websockets has server push, a duplex communication feature, it is better for services like the Calculation Service as one may not know how long a calculation takes and having server push meaning the client might not needed to keep polling for results. A duplex communication also reduces latency as it's half the time required as no "request" is needed.

2.2.3.2 NOSQL Database - MongoDB

We were taught in class to use MSSQL, a SQL table relational database, however in our project we are opting to use the *MongoDB* database via a HTTP interface (RestHeart),a open source REST Interface for MongoDB.

It is used in/used by the following services

1. Price Data GET Services Cluster - To store and obtain pricing data of stocks

2. Version GET and POST - To store, obtain and commit Qlang Codes crated by the user in the webapp

3. News GET Services Cluster - To Obtain and Store news data that is crawled and committed by the Crawler Services Clusters that internally uses Chrome Instance Clusters to retrieve the data.

4. Log POST Services cluster - To Store and Commit log data generated by all other services that may be used by the developer to examine the application in the event of a fault.

5. Algorithm Services Cluster - to execute algorithms created by the user in the analyze page and return the results such as *Sharpe Ratio* of the algorithm

6. Admin CRUD Services Cluster - to tore, obtain and edit user details like personal details and personalization settings.

7. Crawler Service Cluster - To store news data crawled via Chrome Instances Cluster.

4 Reasons Why

1. JSON parity, as mongo Db is a documented based data store meaning it stores data in JSON format while our api communication (Microservices) uses the same communication protocol format, it's logical to stick to it to reduce development time and unnecessary latency or performance reduction by writing converter/parser code from JSON into SQL Tables. The developer may not have to plan nor think of too many things too leading to better focus on other more important features

2. Free and Open Source. Being free and open source unlike MSSQL means the developer is free to do whatever he wants with it at no cost even modify and redistribute it lessening worries or cost and never be inhabited by the vendor. **Multiple Servers which Microservices so dearly rely on would have to pay for additional licensee if it uses MSSQL which is a hurdle for many startups**

3. Scalability - MSSQL doesn't have a sharding ability like MongoDB which would allow it to be scaled and replicated for redundancy over multiple servers which is a key Principe of Microservices. This in turn makes MongoDB more scalable and performance hence better to scale and future proof an application. - this was the key reasons nosql took on large adoption.

4. Because it data is stored in JSON, this would mean that nested data would be accepted. Unlike MSSQL where to nest such data one would

have to relationally link with a new table This wastes space especially in many-to-many relationship where there needs to be an additional many-many table linking the relations. This makes development smooth and easy to think about data. One simply need not think or spend much time thinking abt the data organization.

2.2.3.3 Docker

Docker is used for the containerization. It was not taught it class but it's a cornerstone in Microservices industry and almost every tech company one can think of uses it. It is further supported and continuously by the big 5 (Apple, Amazon, Google, Microsoft and Facebook).

It is used by **every** service and database in our application.

4 Reasons Why

1. Containerization creates consistent environments. Docker may be seen as a Linux Server that has been coded to be operated on any kinds of server. This makes it easy to build Microservices as any kinds of server can be used from any point on the glob and be connected to the Internet. The services or servers or applications inside the container will not face any difficulty operating as the system it was built for is the same practically anywhere. Another Upside is that development teams who work globally on a service or project may see the same system reducing development times.

2. It is easy to install and ship anywhere. A docker container is installed and operated via a Dockerfile, a file of commands that may be stored and reused to start a new container anytime. This means startup simply loading up the dockerfile and executing it and good to go. Being so easily installed, development time is reduced greatly since it only needs to be configured once and to update would be to configure a single file that will be deployed to all the microservices. "Dependency Hell and Installation troubles" is not in a developer's dictionary any longer.

3. Better operations. With containerization, one may operate multiple applications on a single machine. This leads to reduction costs as one

may easily run multiple application without conflict nor resource performance stealing which leads to the need to develop a complex resource management system between applications which would be redundant if docker is used and resource is sensibly allocated by the container. Another upside to management by docker means that application metrics like performance , latency and resource time hogs may be retrieved so the developer could optimize the application.

4. Docker may be installed on our desktop development environment, this allows us to truly develop the services on the environment that the services will be operation on unlike how developers traditionally developed applications by coding on their personal operation system before uploading the application to the container on the main servers. With this capability, Microservices developers are able to have confidence the MicroService will work on the Server just like it had in their development environment.

2.2.3.4 Kubernetes - Auto Scaling

Kubernetes is a containerization orchestration framework, it works hand in hand with docker to deliver the full micro-service experience streamlining manging microservices across servers through automation. It is opensource and was a project from Google.

Like Docker, it is used by every service even databases and our webapp.

3 Reasons Why

1. With auto-scaling, time is saved, one does not have to administrate the application on a time sensitive basis like increasing during peak to hodl the load and reducing during unpeak hours to save cost.

2. One may say, i can create my own scripts, why create your own if you could use a system that has barely any coding needed that has been tested by ten and thousands of developers and fixed and tweaked for the highest performance but users and developers all over the world as ell as large companies like the BIG 5 as this is an open source project.

3. Millimeter accurate tuning. As most of our application is structured into pods(container) inside a Kubernetes system, one may autoscale to

the needs at that time. Such as ready made scaling to scale to usage percentage, server temperature, ram usage, number of request etc, this means that it can react fast and scale for only what is needed for that long or short moment. One may even scale over multiple servers instead. One may fine tune the scaling to only scale certain service and not all blindly.

2.2.3.5 Kubernetes - Auto Heal

Kubernetes is a containerization orchestration framework, it works hand in had with docker to deliver the full micro-service experience streamlining manging microservices across servers through automation. It is opensource and was a project from Google.

Like Docker, it is used by every service even databases and our webapp.

3 Reasons Why

1. Auto healing is very imoprotant as incase in the instance of a container failure or application failure, the specific container would be restarted and hence if replication/redundancy is enabled, the user virtually seens no downtime ever.

2. Auto Healing is further better than creating a script to simply restart an application if it fails as that does not reclaim any lost memory that ay have leaked or not deleted which when pilled up over time would bring down th whole server and catastrophic failure. As docker inside Kubernetes is a self contained OS, kuebernetes would be able to monitor and fully control the application such that is deletes any lost/leak/undeleted/stray memory which is something that manually created via a script would cause large development work; kubernetes is virtually supported by almost all large tech MNC.

3. Auto Healing is not just restarting an application if it fail but detecting when there are memory leaks or issues like lackluster sluggish performance, restart or heal the application in which simple self created scripts cannot do without much work.

2.2.3.6 Kubernetes - Load Balancing

Kubernetes a containerization orchestration framework, it works hand in hand with docker to deliver the full micro-service experience streamlining manging microservices across servers through automation. It is opensource and was a project from Google.

Like Docker, it is used by every service even databases and our webapp.

Reason Why
Load Balancing is needed as one does not one to unfairly allocate most of the work to a single pod based of default behavior but rather balance it out between the pods so that they may perform equally as fast and utilizes performance efficiently an not cause a bottleneck through a overloaded service at a single point.

2.2.3.7 Kubernetes - DNS for all Pods/ Service Discovery

Kubernetes is a containerization orchestration framework, it works hand in hand with docker to deliver the full micro-service experience streamlining manging microservices across servers through automation. It is opensource and was a project from Google.

Like Docker, it is used by every service even databases and our webapp.

Reason Why
To manually manage the list of services endpoints and manually schedule to utilize them on a balanced basis especially if there are replicas would be developer hell to do so.
For example, for every new service u build , one has to add the url into every other microservice's code or opt to create a url list server which itself may be called a psuedo DNS. It is better to use Kubernetes' DNS server for all it pods. One May simply go to `localhost:8888/admin` to simply target the admin cluster not need to go into a url like `localhost:8888/admin/node/1` to target a specific node or pod as kubernetes may be configured by the developer to auto assign and direct based of loads or which node/pod is free to serve. This feature would also apply to services and versioning, we may configure Kubernetes to target a specific version without changing the url

and hence breaking changes client side.

It is to be noted that DNS Management does not simply limit to a single server but across multiple servers making horizontal scaling easy.

2.2.3.8 Kubernetes - Replication

Kubernetes is a containerization orchestration framework, it works hand in hand with docker to deliver the full micro-service experience streamlining manging microservices across servers through automation. It is opensource and was a project from Google.

Like Docker, it is used by every service even databases and our webapp.

3 Reasons Why

1. Replication is very good for Microservices, a key point of Microservices concept. It creates redundancy so that in the event of a failure, there is a backup/failover system to take care of it ensuring a never failing service. This is fully managed by Kubernetes after configuration

2. Concurrency, it is something that companies do to allow a better latency. With replication there are multiple servers and multiple entry and exit points that programs may be processed and operated at the same time. This greatly increases performance and reduces cost.

3. Globally Replicate. Replication allows geo based replication meaning that a user may access a server that is nearer to their location to reduce time for communication between client an sever. Replication also allows allocation of serve resources based of demand (eg more servers in more populated dense areas)

2.2.3.9 Kubernetes - Metrics Collection + UI

Kubernetes is a containerization orchestration framework, it works hand in hand with docker to deliver the full micro-service experience streamlining manging Microservices across servers through automation. It is open-source and was a project from Google.

Like Docker, it is used by every service even databases and our webapp.

3 Reasons Why

1. Kubernetes may be configured to collect metrics about out application and visualize it, this is good as it allows developers to observe their applications and know how much to configure. Visual representations of data allows better mental understanding.

2. With Monitoring, developers may know how much to configure (eg see high loads visualization in central Singapore so increase amount of servers in that area). It would be easier 5o optimize services configuration like replication , auto healing, load balancing etc.

3. Most Faults are not seen unless the analytics is performed over it. In this case with metrics, one may see in great detail the bigger picture and know for example that on average this certain service fails more often than others and might have a certain memory leak that was unknown previously.

2.2.3.10 Realism - Polygot/Cross Languages

In an effort to be as realistic as possible in our MicroService implementation and for multiple other benefits we have chosen to code in multiple languages for our various Microsevices for various reasons. A Polygot Development Model
3 Reasons Why

1. With a polyglot approach, there is no tight coupling mindset, this means that developers may develop or improve existing languages with any languages and no be limited to a languages' toolkit. With this mindset developers are hence forced to use non tight coupling concepts like Microservices and Communication thorough HTTP so that their tool-sets may be used. It further allows the employment on any type of developer as all the developers need to understand is to built it in compliance with the HTTP endpoints but may use any languages, coding style or version as the system is not tightly coupled. This additionally also allows graceful upgrading such that certain applications which need to be improved may be written in a faster language without affecting

the other services and hence saving cost on improving and working on features that matters.

2. In the trading industry, time is literally money due to high frequency of trading and picking the right time to enter or exit the market and hence high performance and low memory code that can execute thousands of orders or analyze them is very crucial and the industry standard. Unlike C++, NodeJS or python does not have real kernel level threads and consumes more memory as it is an interpreted languages hence we have opted to use c++ which solves these 2 issues. Performance gain not as a desire or idea but an actual need due to the fact we may calculate up to a million data points for multiple users.

3. We have lastly chosen to do it cross language to use the right tool for the job to save time and effort. c++ is good for high performance code but development time is longer and more complex demanding a higher skill and more time planning, it is also not as suitable for prototyping as NodeJS is in a few lines. Furthermore taking time to set up the environment while NodeJS is simply plug and play. However the thousand times performance gains in C++ simply can't be forgone. NodeJS is not all bad as it excels in quick syntax such as async/await primitives allowing fast prototyping and clean code. It has more libraries that C++ and they are written for async code. The downside to NodeJS are its memory, single threadedness and speed compared to nodejs, but it is good for prototyping and hence will be used for less mission critical services.

2.2.3.11 Realism - C++ build pipeline

C++ is one of the programming languages we will be using for our services in our polygot approach in addition to NodeJS. However unlike NodeJS, C++ is a compiled language and hence needs a compiler. Libraries will be used in the c++ programs such as HTTP`https://github.com/meltwater/served`/Websocket`https://github.com/uNetworking/uWebSockets`(this library is used by popular cryptocurrency exchanges worldwide, the same industry as we are in, finance) servers and BOOST`https://www.boost.org/` C++ libraries hence a build pipeline is further needed. This is standard practice in the industry and hence realistic. We have chosen CMAKE `https://cmake.org/` as it is

1. CMake is a simple build too which ties all the whole build line together. It is really simple as it basically is a text file with compilation commands designed as simple functions that trigger the functionality. Anyone can learn in an hour.

2. Like Kubernetes, Open Source and Big 5 supported meaning that one may change it however they wish to suit their needs for customisability and being supported by the Big 5 Tech companies has made it an industry standard and used everywhere. It is hence reliable and open.

3. CMake is compatible with everything. It is describe as a cross-platform cross compiler build system meaning it can run anywhere making it sensible as a choice for Microservices as it may run on multiple services.

It is used in only in development before the binary is shipped into the services.

2.2.3.12 Realism - RestHeart, external tools

In most realistic user applications, there will definitely be externally created tools that are operated as services such a s RestHeart which is an API Rest layer on top MongoDB. We will use this API layer MicroService for anything related to database operation like retrieval and committing.
3 Reason Why

1. This prevents developer time wastage of reinventing the wheel and Not-Invented-Here syndrome. This allows that time be better spent creating better products. Sometimes it's better to use something already created, company time is money spent.

2. Since RestHeart is community maintained and used by thousands of individuals including companies, it is hence guaranteed to be supported for a long time unlike closed enterprise applications as if the main company stops support, the community that uses it so extensively especially since it's free, would support it. It is even officially supported and promoted by the MongoDB Company `https://docs.mongodb.com/ecosystem/tools/http-interfaces/`. One may also see open source as delegating work to the community.

3. It may be seen as contribute to a social good, open source as it is free software. As we use it we might pass changes and help develop addition

fixes/features (we contribute the feature and fixes as we need to use it for our use case and we want these features to move upstream so that it is compatible with future versions) that others might use and since it is such a good software as the community incrementally helps, others will use it more and in turn contribute.

2.2.3.13 Chrome Browser Instances

The Chrome Instance cluster is a cluster of chrome browser instances that is used and operated/controlled by the Crawler Clusters to crawl/mine web pages. It is a extra features as it is running a headless browser as a service, a application with a Websocket interface to wrap as a service, this was not taught in class.
3 Reasons Why

1. With Chrome Instances being placed into it's own Service and sandboxed into its own OS, if it crashes this means it wont affect other services running on the same machine but in a different container, reducing downtime (Kubernetes may auto restart it). With it's own container we may update easily just by hot switching the container without much fuss as opposed to running natively on a server where th whole server has to be paused so that one application (Chrome) may lock the installation drive and update nd hence hogging the server not letting it do other task increasing our cost such tha we may need to obtain more servers to compensate for updates hog(Chrome updates regularly).

2. We use Chrome Browser to crawl instead of HTTP Request as most website nowadays rely on Javascript rendering. We would be immensely limiting ourselves if we simply only crawl sites that can be accessed with a Simple GET Request and crawl. Chrome would render the Webpage in an actual browser before crawling and pulling the data through our crawl enigne. However one must be wary as Chrome tends to have many memory leaks and uses loads of RAM which is another reason we opted to sandbox it into a service instead so we may restart at will and if there are any faults in operation, no other service.

3. There are further benefits on the Employment side too. For example, with Chrome put in it's own service, a team that is in charge of manag-

ing the chrome instances would not affect the other teams (especially if they acid-dentally triggered something wrongly while testing) working on the other Microservices, as they do not share the same server and hence do not compete to use the server so they may perform their updates for example. Making decoupling into its own service very suitable for the MicroService Architecture.

2.2.3.14 Crawl Engines

The Crawl Engine is simply a service that controls the Chrome Instances to crawl financial WebPages for financial news to aid the application user. It contains the business logic and is a Websocket client that controls the Websocket sever included in Chrome Instances. Websocket and Crawling was not taught in class and hence is a extra feature.

3 Reasons Why

1. A Crawl engine is immensely useful as it may be used for virtually any websie and hence not every website that we desire information from has an API, a Crawl Engine would give us that info regardless of the existence of an API.
2. Unlike an API, a crawl engine allows access to any data that may be viewed with a web browser. With an API, you are only allowed a limited functionality which is bad when one desires to display crucial news that cannot be limited. Moreover one is limited to the API's Vendor format
3. This Crawl Engine may be extended for future use such as Other kinds of news other than financial news without much issue such as the need to Sign up for APIs or study the API format for every specific site.

2.2.3.15 Scheduled Instances on Crawler Services

This is an extra feature (not taught in class)we will be implementing. In this case we are scheduling the crawl service on an minute basis to crawl for data on the financial website, it is totally independent to itself without user intervention. It crawls on a minute basis so as to obtain the latest updated information and sends it directly to the database without any intervention.

2.2.3.16 Sass

An Extra Feature we will be implemented is a technical internal one. We will be using SASS CSS preprocessor in our development pipeline which will affect the Main App.

3 Reasons Why

1. With the ability to use variables and functions, we may keep our code DRY (DONT REPEAT YOURSELF) not having to repeat or change every value at every location leading to better development times. It also allows global variables and hence how it's used in the main application client side is global changes to theme may be done with the simple code of triggering a function which changes all he color variables.

2. SASS Allows reactive code, meaning that the developer is allowed to write code that eg if a there are 10 elements on screen, the font decreases in size and color changes. This makes it very easy and convenient to write complex code that react to UI changes or User changes with simple SASS codes allowing easy prototyping and development.

3. SASS allowing separate files means that that files may be organized better into it's components in the Angular Framework Client side that we are using and into its respective domains. It will also allow teams to work across borders as Microservices teams usually do as they only need to focus on their own file in their domain and they will not affect each other.

2.2.3.17 Linux

Linux Server administration and operating applications on top a Linux stack was not taught in class and hence it's an extra feature. All our services operates on top a Linux Operating system.
3 Reasons Why

1. Linux is free and open source(meaning on may tweak it for their own needs). This means that one may use it at any point of time for any purpose and even resell it without worries. Where-else for windows, this is not the case, Widnows charges for servers are extremely high and is limited by the number of processors or nodes that u use which is not

financially wise for a startup or for a company using the Microservices architecture as MicroServices rely on multiple servers with each server doing one domain or small functionality. A Microservices approach using Windows is not financially sensible.

2. Most high performance libraries for servers and math are written for soley Linux due to manpower limitation or simply that everyone uses Linux and hence it's only logical we use Linux that we may have access to such libraries.

3. Freedom. Unlike windows which locks you in to Microsoft API for example, servers, Linux enables and allows an individual to use whatever protocol or API he or she wishes meaning that there is a greater access to a other bigger world of software and hence better products can be made without compromises. Windows API is usuable on Linux but not vice-versa too.

2.2.3.18 Elastic Search

The Elasticsearch software and usage was no taught in class and hence it's a extra feature. ElasticSearch is an ultra fast search server based off lucene search server and is document based in that in stores files in JSON.

4 Reasons Why

1. We chose elastic search as it is document based meaning that it has the same data format as our REST Services and Our database MongoDB and hence it would be good to have a compatible data format. This reduces development time and increases performance as no additional code is needed. Furthermore, it causes less development confusion.

2. ElasticSearch has many extra features. For example it has filtering by keywords grammar and Ngrams which would allow very accurate results as well as performance increase as with Ngrams (a dictionary of words broken down into their parts and linked to the correct location) one is able to target the exact word fast without much computation. This all being built in means that configuration is only needed without much coding.

3. ElasticSearch provides a REST server on top unlike other kinds of search servers aas well as natively support data redundancy and replication making it very good and suitable for the Microservices architecture without much extra work.

4. Due to being built with special algorithms and years of research it is super fast in looking up text which gives the user and developer both a good experiences using it. It achieves this by indexing via Ngrams and a inverted index (instead of indexing by document like traditional SQL it indexes by word)

2.2.3.19 Low Latency Practices

In finance, Time is literally money, a minute late may cost a few thousand lost in Trade Positions for the average retail trader, that number is in the millions for Hedge Funds/Banks. This would mean that we would need to write non blocking high performance low latency code for a good experience for the user. For example the user might want to view the most updated information news and hence would need a system that crawls data but not hogged by other processors which would delay the data transfer from crawl service to database and finally to screen.

How have we tackled this

1. Removing headers for certain services - in our internal system where no outsiders consume our api, headers are not needed since our developers and applications already understand what kind of data it is and what to expect. There is no need to transfer useless metadata that would cause an overhead due to more data being sent hence the amount of time it has to spend in transit would be longer as the data is longer. This s industry standard in HPC communities.

2. Everything is Locally Hosted - it is an industry standard to run all application in a Localized Server as it's universally recognized in the world of high performance computing, the Internet is a thousand to a million times slower than direct SSD to Processor connection and that difference of a thousand may lie within the range of 1MS !

3. Nano Services, Many moving Parts - due to our deployment of kubernetes, if a service fails, it is immediately restarted or replaced. All

services are bound to fail one way or another, nothing is bug or error free. So instead of bringing down the whole system, we have split everything into as many small tiny services which serves one domain and one method eg GET Server and POST server are different and no in the same one. This reduce developer complication and confusion while allowing for failure that will happen regardless, it also lightens developer complexity as each service is truly single purpose and hence single domain in development.

4. We have chosen to use Async primitives in our C++ and NodeJS services, this would mean that a server may be able to serve multiple requests without being blocked waiting for a operation as it may come back to it later. Asynchronous means that it a request may enter now but the response may be after another request that entered later has been solved as this request takes a longer time due to perhaps more data. This immensely saves server resources as now a server has no wasted time and any time it's in operation ,its being used while in another side of the same service it is waiting for a response for the database. This is industry standard.

2.2.3.20 Logging

We have opted to have a logging server, every operation is logged and the corresponding log is committed into the database. It is a high performance C++ server as it it used by every service.

3 Short Reasons Why

1. With a logging server, one is able to properly debug a problem to find the cause of a bug as the log is basically "history recorded" and hence this would enable developers to fix issues without worry of not knowing the cause and hence reducing development time and decreasing cost.

2. With logging server, we are able to track all the Microservices health and history in one central place as opposed to logging into the possibly hundreds of Microservices and examine each log file in every service's file system.

3. A log server enables additional features. For example if there is a widespread down time for all services, perhaps instead of the services

having bugs the server room is on fire and it's detected though super hot temperatures of the temperature sensor in our metrics, a log server might receive such logs within the millisecond and send us a phone alert via voice or text so that we will know to quickly run to take care of the situation. Further more, such logging servers may simply just notify us on a hourly basis that the machines are still running and healthy. A log server provides a central location too fo us to collect metrics or download metrics from which simple log files that are not able to interpret the logs would no be able to do so.

2.2.3.21 NodeJS for Every service except those stated

We chose NodeJS for use in our Microservices as they tend to not be computationally intensive services and hence do not need the capability of C++ or the speed of it. The pros of c++ comes with a cost of development time however time is extremely valuable in the finance industry.
3 Reasons Why

1. NodeJS is meant for prototyping. Unlike C++, NodeJS ecosystem allwos for quick codes that are of comparable speeds even faster than python. With such ability to write code in fast fashion due to its short syntax and auto import without the need for headers nor semi colons or compilation or compilation pipeline building, it makes it easy to get up a server in 2 mins, an i mean it. Take a look below for the full server commands needed in this case with an industry standard webserver, express, a simple GET endpoint to serve HTML to any users complete with PORT configuration and SSL which in C++ from an anecdotal standpoint takes at least 30 lines of code which is more than the whole nodejs server nothing the fact JS does not have types and hence no need for type declaration. Simply compare the examples and remember that NodeJS does not even need a setup of a build pipeline like Cmake or manual installation of libraries into the OS ONLY AFTER building them.

   NodeJS Example

   ```
   var fs = require('fs'),
      http = require('http'),
      https = require('https'),
      express = require('express');
   ```

```
var port = 8000

var options = {
   key: fs.readFileSync('./ssl/privatekey.pem'),
   cert: fs.readFileSync('./ssl/certificate.pem'),
}

var app = express()

var server = https.createServer(options, app).listen(port, function(){
   console.log("Express server listening on port " + port)
});

app.get('/', function (req, res) {
   res.writeHead(200)
   res.end("hello world\n")
})
```

C++ Example

```
#include <cstdlib>
#include <iostream>
#include <boost/bind.hpp>
#include <boost/asio.hpp>
#include <boost/asio/ssl.hpp>

typedef boost::asio::ssl::stream<boost::asio::ip::tcp

class session
{
public:
   session(boost::asio::io_service& io_service, boost
: socket_(io_service, context)
{
}
```

```
ssl_socket::lowest_layer_type& socket()
{
   return socket_.lowest_layer();
}

void start()
{
    socket_.async_handshake(boost::asio::ssl::stream_
   boost::bind(&session::handle_handshake, this,
   boost::asio::placeholders::error));
}

void handle_handshake(const boost::system::error_code
   if (!error){
      socket_.async_read_some(boost::asio::buffer(dat
      boost::bind(&session::handle_read, this,
      boost::asio::placeholders::error,
      boost::asio::placeholders::bytes_transferred));
   }else{
      delete this;
   }
}

void handle_read(const boost::system::error_code& err
size_t bytes_transferred){
        if (!error){
           boost::asio::async_write(socket_,
           boost::asio::buffer(data_, bytes_transferr
           boost::bind(&session::handle_write, this,
           boost::asio::placeholders::error));
        }else{
           delete this;
        }
}

void handle_write(const boost::system::error_code& er
        if (!error){
                socket_.async_read_some(boost::asio::
```

```
                boost::bind(&session::handle_read, th
                boost::asio::placeholders::error,
                boost::asio::placeholders::bytes_tran
        }else{
                delete this;
        }
}

private:
        ssl_socket socket_;
        enum { max_length = 1024 };
        char data_[max_length];
};

class server{
public:
        server(boost::asio::io_service& io_service, u
        : io_service_(io_service),
        acceptor_(io_service,
        boost::asio::ip::tcp::endpoint(boost::asio::i
        context_(io_service, boost::asio::ssl::contex
{
        context_.set_options(
        boost::asio::ssl::context::default_workaround
        | boost::asio::ssl::context::no_sslv2
        | boost::asio::ssl::context::single_dh_use);
        context_.set_password_callback(boost::bind(&s
        context_.use_certificate_chain_file("server.p
        context_.use_private_key_file("server.pem", b
        context_.use_tmp_dh_file("dh512.pem");

        session* new_session = new session(io_service
        acceptor_.async_accept(new_session->socket(),
        boost::bind(&server::handle_accept, this, new
        boost::asio::placeholders::error));
}

std::string get_password() const{
        return "test";
}

void handle_accept(session* new_session,
const boost::system::error_code& error){
        if (!error){
                new_session->start();
                new_session = new session(io_service_
                acceptor_.async_accept(new_session->s
                boost::bind(&server::handle_accept, t
                boost::asio::placeholders::error));
        }else{
                delete new_session;
        }
}

private:
        boost::asio::io_service& io_service_;
        boost::asio::ip::tcp::acceptor acceptor_;
        boost::asio::ssl::context context_;
};

int main(int argc, char* argv[]){
try{
        if (argc != 2){
                std::cerr << "Usage: server <port>\n"
                return 1;
        }

        boost::asio::io_service io_service;

        using namespace std; // For atoi.
        server s(io_service, atoi(argv[1]));

        io_service.run();
}catch (std::exception& e){
        std::cerr << "Exception: " << e.what() << "\n
}
```

```
return 0;
}
```

A side note too that NodeJS was made for the modern world with the recently release ES2018 with shorthand syntax while C++ was in 2011, when Netflix or Amazon barely existed in the public sphere of influence and that Apple had just become known big and hadn't haven't had a Singapore presence nor could you buy and Iphone Yet in Sg.

2. Ready Made - Unlike C++ where high performance asynchronous severs may only be used with a library, NodeJS has this by default in its standard library that is written underlying in C++ (but because JS is used it loses much of it's performance and memory usage is high) and moreover the server is already made and simply a line of code to import and start it. Less development time is needed as a result and more money and effort is saved.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
   res.statusCode = 200;
   res.setHeader('Content-Type', 'text/plain');
   res.end('Hello, World!\n');
});

server.listen(port, hostname, () => {
   console.log(`Server running at http://${hostname}:${port}/`);
});
```

Lastly NodeJS since it's Server Side Javascript, by default it already has JSON (JavaScript Object Notation) built in and may be used without libraries making it good for development as our database, rest communication data format and search severs are all JSON-Based.

3. Easy Install - NodeJS is known to be easy to install and not needing to be built from source like c++ as it's interpreted. I further more is cross platform and has its own Package Manager (NPM - Node Package manager). For example

   ```
   # To install NodeJS
   curl -sL https://deb.nodesource.com/setup_11.x | bash -
   apt-get install -y nodejs

   # To Install Angular
   npm install angular
   ```

   This makes it very easy for developers to perform dev ops and that it would be easy for docker to install packages without much fuss on any os installation making it cross platform fit for Microservices Architecture. A package manager also allows one stop access for all packages in the node system unlike C++ where u have to manually download from vendor sites . With easy install we are directly connected to the vendor's server source and hence with a simple `update` command, we can simply plug and play updates with much fuss or needed to manually replace as the package manager takes care of installation location and referencing from the app to the packages for us and includes it in Nodejs's ready made compilation for us, all of these C++ is not able to do.

2.2.3.22 Versioning

This is an extra feature as it was not taught in class. One of our bigger extra features, this is basically a versioning service that was self made. It is a companion to the QLang Language used in the Quant Editor on the main application to save versions of our past code, this is reasons why.
3 Reasons Why

1. With Versioning/Version Control, the user may refer back to his/her own code in future for reference to compare with their current code perhaps to measure performance or some metric. This allows the user to work on an algorithm while not forgetting the past and hence are able to optimize with greater confidence after knowing that perhaps in

the upgrading cycle they have not forgotten to review the past with the present.It may also be used to revert back to an old code if the user recognizes an issue.

2. It helps the user keep track of their changes and know how much they progressed which may lead them to feel good about their progress so that to know how far they have gone and how near they are to their goal with the Birds Eye view version control gives them.

3. With versioning, the User may fork his old algorithm, eg if he wishes to work on his old code but does not want to modify it but make a copy so that he may create another prototype or perform tests he may fork that old code and create a new tool. This allows developers an ease of mind of not losing things when using the platform that they ma go back to it anytime. With forking too a user may maintain multiple versions of a code if they wish to.

2.2.3.23 QLang Language

One of our big features, is our own Language !. Qlang(Quant Lang) is a opinionated conditional method based programming language that is made for quick prototyping of Algorithms in our QLang Editor. It's meant to be as easy for anyone to use even old folks and needs as little typing as possible. Internally it's a C Language Transpiler written by me.
3 Reaons Why

1. With easy to remember shorthand syntax, the coder is able to understand all features and use it to their most of their ability. This leads to better and faster algorithms created as all features of the language are used. It is more simpler than Javascript has it has only one purpose, to code algorithm ! A visual GUI was considered but found to be limiting as the programmer was limited to what he may change on the screen and not the full functionality. With a simple language, algorithm building can be the main focus instead of language semantics

2. QLang is an opinionated language. Being an opinionated language means there is only one way and one way o do something. This is generally good as one does not have to spend time thinking about what datastructure ot use or type of function or variable type for all of this

is decided for the user. These has other benefits. ith same syntax all around, developers working together may share the same code as both codes will definitely be compatible even to the data structure type so on one dosent have to write wrappers; the similar group think is there too as both codes are same.

With a same opinionated code, this means that it is very easy to create future language updates as not much code will be change and hardly will there be breaking changes, everyone can agree on updates. With opinionated code, libraries may especially integrate wit h existing code. This also leads to easier language implementation.

QLang consist of 3 main semantics, keywords, methods and conditions. A simple condition in Qlang might look like this with keywords/variables and methods. It is really really simple and short.

```
EQUALS(200, AB) THEN
LOAD_CSV(aapl.csv)
```

2.2.3.24 Real Time Chart Engine

Our last extra feature, is our custom made real time engine. We chose to make our own engine as custom current solutions do not fit our needs nor serve our high performance needs with a million data points. A real time chart engine is a chart which displays pricing data as well as algorithms results and visualizations of our indicators on the client side with data from the sever side algorithm service a Websocket data pipe.
IN this case we are building a library layer over the high performance CanvasJS library.

3 Reasons Why

1. Current solutions are too slow. We opted to find a solution that would support a million data points or so at minimum but could never find any that were free, existing solutions were at least 5 digits in costs per month. Other existing solutions demanded that we share our intellectual property of the application to opt to use their company's charting libraries. This is understandable as most trading softwares are written in either C# or Java and structured in a desktop application which

means that non of the language nor chart libraries that were of substantial quality supports the Web as traditionally web applications were too slow for such applications. However the web has modernized and hence trading on it is suitable right now to create real time charts.

2. Most existing charting libraries on Javascript did not support the idea of streaming charts from Websocket in real time especially one that supports fast streaming which we desire as we would like immediate feedback on the screen and not a chart engine where the chart is compiled and then a complete chart is outputted as there is a million data points this would mean that it would take long for it to compute and complete. This would be bad for the user experience as they would be left with a loading screen for 10 mins waiting without feedback or notice.

3. The finance industry uses alot of custom diagrams and indicators that no other industry uses such as candlesticks (a visual form of documenting prices over a minute interval on the chart) as well as bollinger bands (a trading indicator) which are combined moving averages. Hence because of the lack of sch a solution which general libraries do not solve, we have opted to create our own instead.

2.3 Databases Overview [Database Model]

For our database we have chosen MongoDB for the reasons stated in the previous chapters. Document Based means it has no tables. We keep our data as compact and separated as possible meaning we dont try to dump all the data into one collection but small and separated just like Microservices so its easy to modify and lookup times would not be long as it does not need to transverse as much data.
It references other data in other collections/tables by storing its uuid of that data.

2.3.1 Databases Model Diagrams

2.3.1.1 Logs Collection

```
{
  id: uuid,
  date: date
  time: time,
  log: string,
  service: string,
  user: id
}
```

2.3.1.2 Users Collection

```
{
   id: uuid,
   name: sring
   phone: string,
   eamil: string,
   address: string
}
```

2.3.1.3 Personalization Collection

```
{
   id: uuid,
   user: uuid
   theme: string,
   main_font: string,
   code_font: string,
   pointer: string,
   alarm: string
}
```

2.3.1.4 Price Collection

1 collection per stock eg `AAPL_Collection`

```
{
   id: numerical_uuid,
   price: double,
   time: time,
   date: date
}
```

2.3.1.5 Session Collection

```
{
   id: uuid,
   user: uuid,
   jwt: jwt,
   date: date,
   time: time,
   status_expired: bool //expires when set amount of time
}
```

2.3.1.6 Lists Collection

Uses RestHeart to retrieve and commit

```
{
   id: uuid,
   user: uuid,
   name: string,
   created_on_date: date,
   created_on_time: time,
   modified_on_date: date,
   modified_on_time: time,
   stocks: {
      uuid,
          ...
   }
}
```

2.3.1.7 News Collection

Uses RestHeart to retrieve and commit

```
{
   id: uuid,
   site: uuid,
   title: string,
   content: HTML,
   date: date,
   time: time
}
```

2.3.1.8 Sites Collection

Manually added by Admin

```
{
   id: uuid,
   url: string,
   name: string,
}
```

2.3.1.9 Versions Collection

```
{
   id: numerical_uuid,
   analysis: uuid,
   user: uuid,
   qlangcode: code,
   date: date,
   time : time
}
```

2.3.1.10 Analysis Collection

Uses RestHeart to retrieve and commit

```
{
   id: uuid,
   name: string,
   description: uuid,
   results: uuid,
   lastest-code-version: uuid
   created-on-date: date,
   created-on-time: time,
   modified-on-date: date,
   modified-on-time: time
}
```

2.3.1.11 Results Collection

Uses RestHeart to retrieve and commit

```
{
   id: uuid,
   time: time,
   date: date,
   sharperatio: double,
   sharpechartimage: url,
   turnover: double,
   turnoverchartimage: url,
   volume: double,
   volumechartimage: url,
   volatility: double,
   volatilitychartimage: url,
   variety: {
      energy: double,
      ...
   },
   drawdown: double,
   drawdownchartimage: double
}
```

2.3.1.12 Calender Legend Collection

Uses RestHeart to retrieve and commit

```
{
   id: uuid,
   user: uuid,
   work: color_string,
   school: color_string,
   home: color_string,
   leisure: color_string
}
```

2.3.1.13 Calender Events Collection

Uses RestHeart to retrieve and commit

```
{
   id: uuid,
   user: uuid,
   title: string
   description: string,
   marker: uuid,
   date: date,
   time: time,
}
```

2.3.1.14 Information/Education Endpoint

Uses RestHeart to retrieve and commit

```
{
id: uuid,
type: method | keyword,
tite: string,
caption: string
description: markdown_string
}
```

2.3.1.15 Fundamental Collection

Uses RestHeart to retrieve and commit

```
{
   uuid: uuid
   name: string,
   profile: uuid,
   summary: uuid,
   historical: collection_uuid
   statistics: uuid,
   financials: uuid,
   analysis: uuid,
   holders: uuid,
   sustainability: uuid
}
```

2.3.1.16 Fundamental Profile Collection

```
{
id: uuid,
RIC: string,
description: string,
address: string,
sector: string,
industry: string,
employees: int,
executives: uuid
}
```

2.3.1.17 Fundamental Profile Executives Collection

```
{
   id: uuid,
   details: {
      {
      name: string,
      title: string,
      pay: int,
      exercised: int
      },
      ...
}
```

2.3.1.18 Fundamental Summary Collection

```
{
    id: uuid,
open: double,
close: dobule,
bid: double,
ask: double,
avgvolume: int,
volume: int,
cap: int,
dayrange: string,
52weekrange: string,
pe_ratio: string,
target_estimate: string
}
```

2.3.1.19 Fundamental Historical Collection

One collection per company

```
{
   id: uuid,
   price: double,
   date: date,
   time: time
}
```

2.3.1.20 Fundamental Statistics Collection

```
{
   id: uuid,
   valuation_measures: {
      cap_intraday: double,
      enteprise_value: string,
      price_sales: double
   },
   financial_highlights: {
      profitablity: {
         profit_margin: double,
         operationg_margin: double
      },
      income_statement: {
         revenue: int,
         EDIBTA: int
      }
   },
   trading_information: {
      stock_price_history: {
         beta: double,
         52-week-change: double,
         52-week-high: double,
         52-week-low: double,
         sp-52-week-change: double,
         200-moving-average: double
      },
      share_statistics: {
         avg_vol: string,
         float: string,
         short_ratio: string,
         shares_short: string
      },
      dividends_and_splits: {
         payout_ratio: double,
         dividend_ratio: double,
         last_split_date: date,
         last_split_factor: double
      }
   }

}
```

2.3.1.21 Fundamental Financials Collection

```
{
   id: uuid,
   income_statement: {
      revenue: {
         total_revenue: int
         cost_of_revenue: int
      },
      operation_expense: {
         selling_general: {
          quarter_1: int,
          quarter_2: int,
          quarter_3: int,
          quarter_4: int
         },
         research: {
            quarter_1: int,
            quarter_2: int,
            quarter_3: int,
            quarter_4: int
         }
      },
      income_continuing_operations: {
         total_income_net: int,
         interest_expense: int,
        minority_interst: int
      }

   },
   balance_sheet: {
      long_term_investments: {
         quarter_1: int,
         quarter_2: int,
         quarter_3: int,
         quarter_4: int
      },
      good_will: {
         quarter_1: int,
                quarter_2: int,
                quarter_3: int,
                quarter_4: int
      },
      intangible_assets: int
   },
   cash_flow: {
      period_ending: {
        net_income: int
      },
      operating_activities: {
         depreciation: int,
         adjustments: int,
         changes_inventory: {
            quarter_1: int,
            quarter_2: int,
                quarter_3: int,
                quarter_4: int
            }
        },
        captial_expenditures: {
            changes_inventory: {
                quarter_1: int,
                quarter_2: int,
                quarter_3: int,
                quarter_4: int
            }
        }

    }


}
```

2.3.1.22 Fundamental Analysis Collection

```
{
    id: uuid,
    earnings_estimate: {
        no_analysts: {
            current_q: int,
            next_q: int,
            current_year: int,
            next_year: int
        },
        avg_est: {
            current_q: int,
            next_q: int,
            current_year: int,
            next_year: int
        },
        high_est: {
            current_q: int,
            next_q: int,
            current_year: int,
            next_year: int
        },
        low_est: {
            current_q: int,
            next_q: int,
            current_year: int,
            next_year: int
        },
        year_ago_esp: {
            current_q: int,
            next_q: int,
            current_year: int,
            next_year: int
        }
    },
    eps_history:{
        current_estimate: {
            current_q: int,
            next_q: int,
            current_year: int,
            next_year: int
        },
        7_days_ago: {
            current_q: int,
            next_q: int,
            current_year: int,
            next_year: int
        },
        30_days_ago: {
            current_q: int,
            next_q: int,
            current_year: int,
            next_year: int
        },
        60_days_ago: {
            current_q: int,
            next_q: int,
            current_year: int,
            next_year: int
        },
        90_days_ago: {
            current_q: int,
            next_q: int,
            current_year: int,
            next_year: int
        }
    },
    ...
 }
```

2.3.1.23 Fundamental Holders Collection

```
{
id: uuid,
   major:{
      breakdown:{
      insider_shares: double,
      institutions_shares: double,
      institutions_float: double,
      no_institutions: int
      },
         top_institutional:{},
      top_mutal_funds: {}
   },
   insider_roster: {
          {
             name: string,
             most_recent_transaction: string,
             date: date,
             shares_owned: int
          },
          ...
   },
   insider_transactions: {
      purchases: {
        amount: int,
        volume: int
       net_volume: int
      },
      insider_transactions_reported: {
        {
          name: string,
          transaction: string,
          type: string
          value: int,
          date: date,
          shares: int
        },
        ...
      }
   }
}
```

References

Aswani, S. (2016, May). *Leveraging Co-location for Competitive Advantage in HFT.* HFTWire. Retrieved 2018-10-30, from https://www.hpcwire.com/solution_content/hpe/financial-services/leveraging-co-location-competitive-advantage-hft/

Butcher, D. (2017, November). *Top quant PhDs are getting $400k pay packages to become e-traders.* efinancialcareers. Retrieved 2018-10-30, from https://news.efinancialcareers.com/sg-en/301445/quant-trader-pay-phd-e-traders-automated-trading-salaries

Chan, E. P. (2008). *Quantitative Trading: How to Build Your Own Algorithmic Trading Business* (1st ed., Vol. 1). Wiley.

C_Rod. (2018, October). *PhD duration in the USA.* StackOverflow. Retrieved from https://academia.stackexchange.com/questions/18478/phd-duration-in-the-usa

Glassdoor. (2018, October). *Citadel Salaries on Glassdoor.* Glassdoor. Retrieved 2018-10-30, from https://www.glassdoor.com/Salary/Citadel-Salaries-E14937.htm

Goldman Sachs. (2018, September). *From Our Briefings Newsletter.* Goldman Sachs. Retrieved 2018-10-01, from https://www.goldmansachs.com/insights/pages/from_briefings_10-sep-2018.html

Quantinsti. (2015, August). *Latency War: Why is Low Latency Important?* Quantinsti. Retrieved 2018-10-30, from https://www.quantinsti.com/blog/latency-war-why-is-low-latency-important

Schwab, K. (2016, January). *The Fourth Industrial Revolution: what it means, how to respond.* weforum.com. Retrieved 2018-10-30, from https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/

Seward, Z. M. (2013, May). *This is how much a Bloomberg terminal costs.* Quartz. Retrieved from https://qz.com/84961/this-is-how-much-a-bloomberg-terminal-costs/

Two Sigma. (2018, October). *Two Sigma Careers.* Two Sigma. Retrieved 2018-10-30, from https://careers.twosigma.com/careers/SearchJobs/ANALYST

Wilkinson, T. (2003, January). *Computers Are Never Wrong.* ComputerWorld. Retrieved 2018-10-30, from https://www.computerworld.com/article/2580628/computers-are-never-wrong.html

Fin