University of New South Wales

Software Engineering Workshop 2B

SENG2021

OG Stock Comparator

Authors:
Orion Larden
Kaan Apaydin
Damon Lee
Kevin Luo
Simon Taylor

Supervisor:
Fethi Rabhi

October 28, 2016

Contents

1 System Scope and Specification

1.1 Project Theme

For the duration of the course, the team decided to undertake a financially themed project, specifically relating to stock markets and the growth of various stocks during a given time period. The project is a web application (viewable on Chrome, Firefox, Microsoft Edge etc) that navigates and scrapes through online resources relating to stock price such as Yahoo! Finance as well as utilising the Quandl API, to collect and rank the growth of respective stock prices in a list. The purpose of the project was to show the various stocks that have experienced growth/decay over a given period of time, to help clients make an informed investment.

For example, a long term investor may want access to the top 50 stocks that have experienced the most growth over the past 10 years, however this information may not be useful to the short term investor who may want to invest in more risky stocks that are currently experiencing explosive growth, in order to make quick profit instead.

1.2 Use Cases

From the feedback given from our initial report as well as subsequent changes made to the project, the team has updated the use cases to represent the final prototype that has been developed.

1.2.1 Ranking List

Figure 1.1 displays the interactions between user and server throughout the ranking list use case. The use case begins when the user loads the website homepage. The server then responds by displaying the ranking list. Afterwards, the user can change the list's time period or industry category, or search for a particular stock. The user's actions prompts the server to update the list accordingly. Finally, the user can click on a particular stock and the server then returns information on the stock requested.

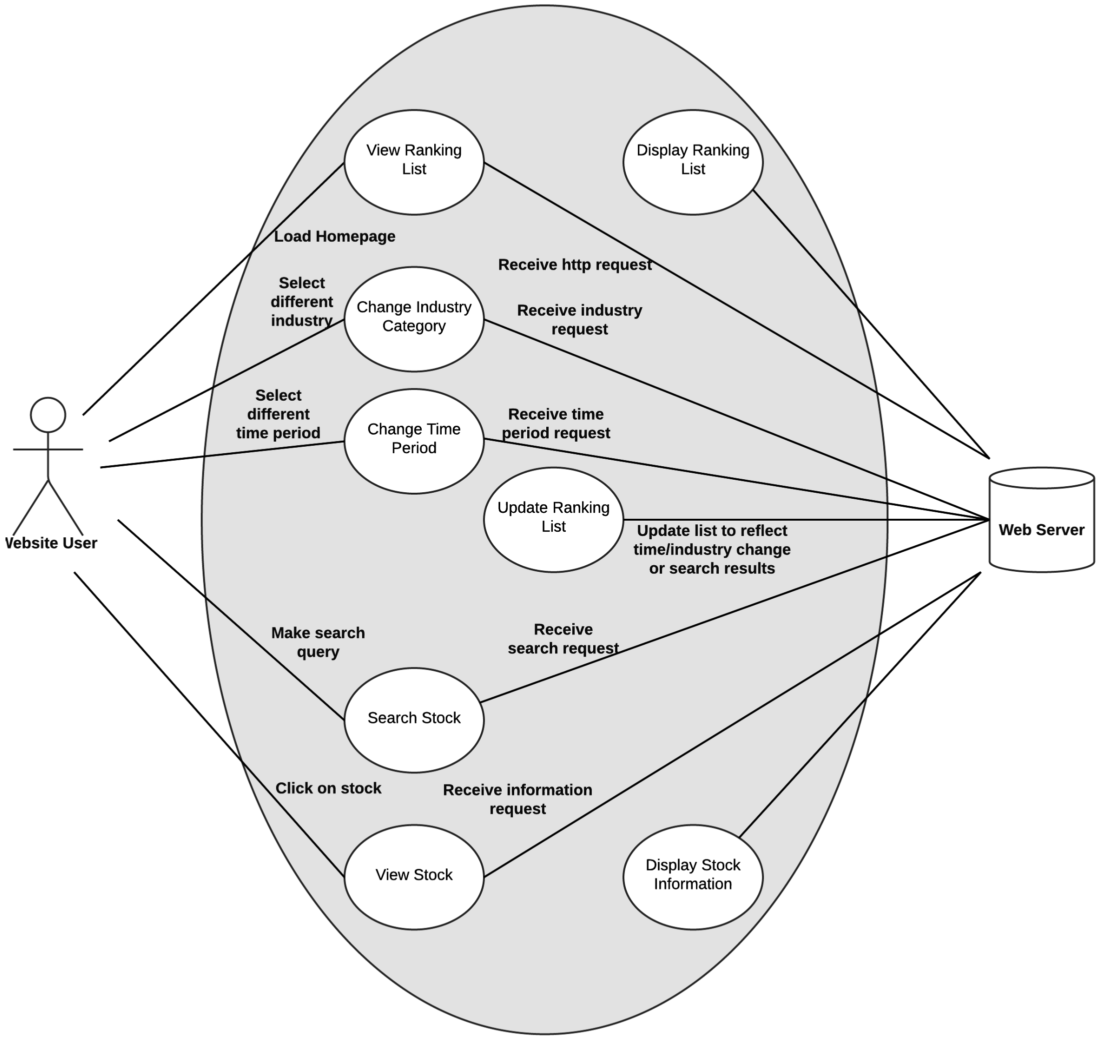


Figure 1.1: Use case for viewing the ranking list

1.2.2 Comparing Stocks

Figure 1.2 displays the user and server interactions during the stock comparison use case. The use case begins when the user navigates to the compare page or clicks on the compare button, upon which the server then loads the comparison page.

Once the user searches for a stock and the server has replied with the search results, the user can then add stocks to the comparison by clicking on them in the search result. The user can also remove stocks and change the time period of the comparison and upon doing so, the server updates the graph accordingly. Furthermore, a user can hover over individual points in the graph to obtain the percentage of stock growth/decay at that particular point in time.

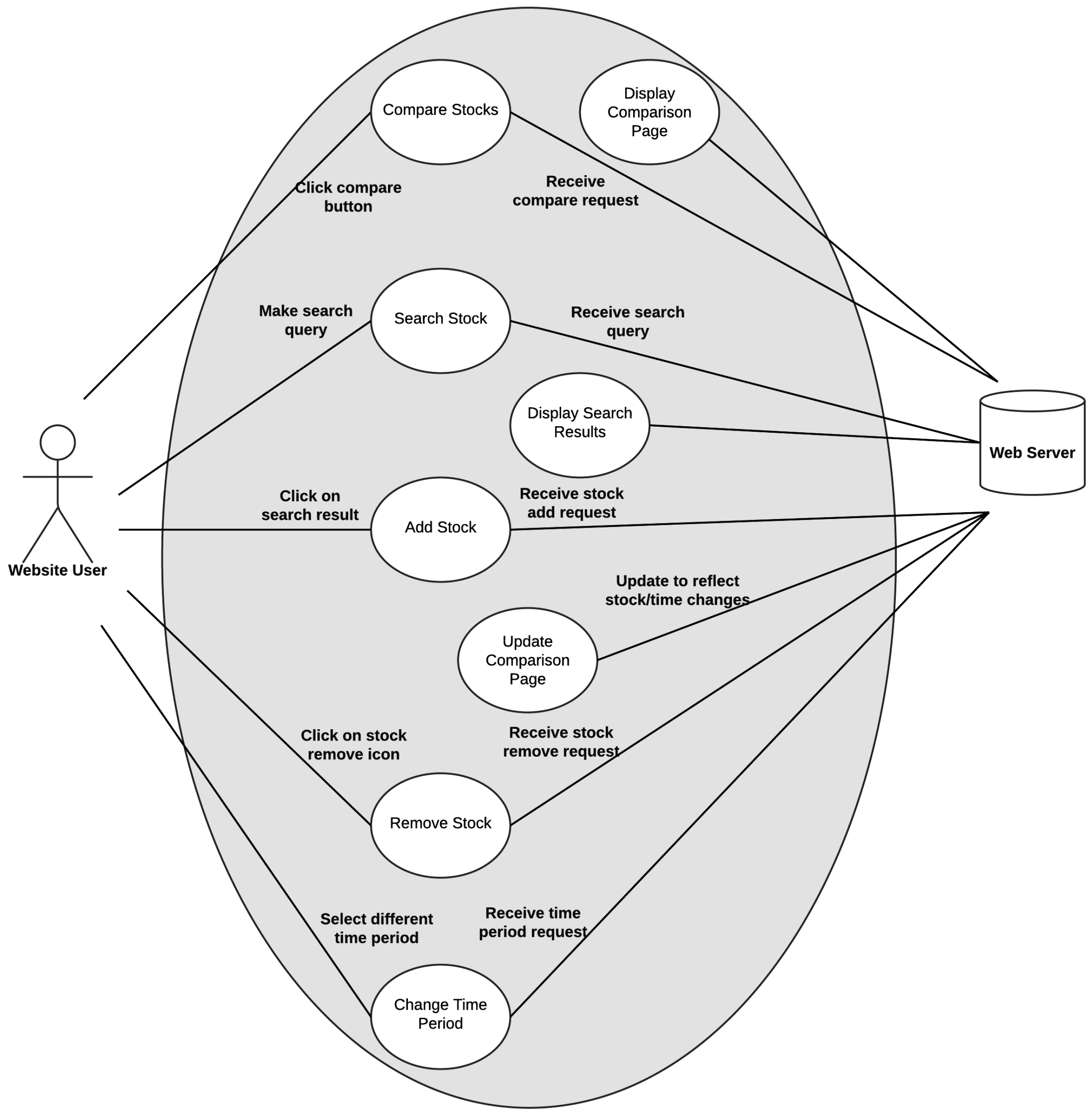


Figure 1.2: Use case for comparing stocks

1.2.3 Activity Diagrams

We have also included two activity diagrams which depict the flow our system whenever the user interacts with our website. Each diagram describes one of our use cases, displaying stocks and comparing stocks. There are no termination points, as these only happen when the user exits out of a page and moves to a different page or in other words, executes the other use case.

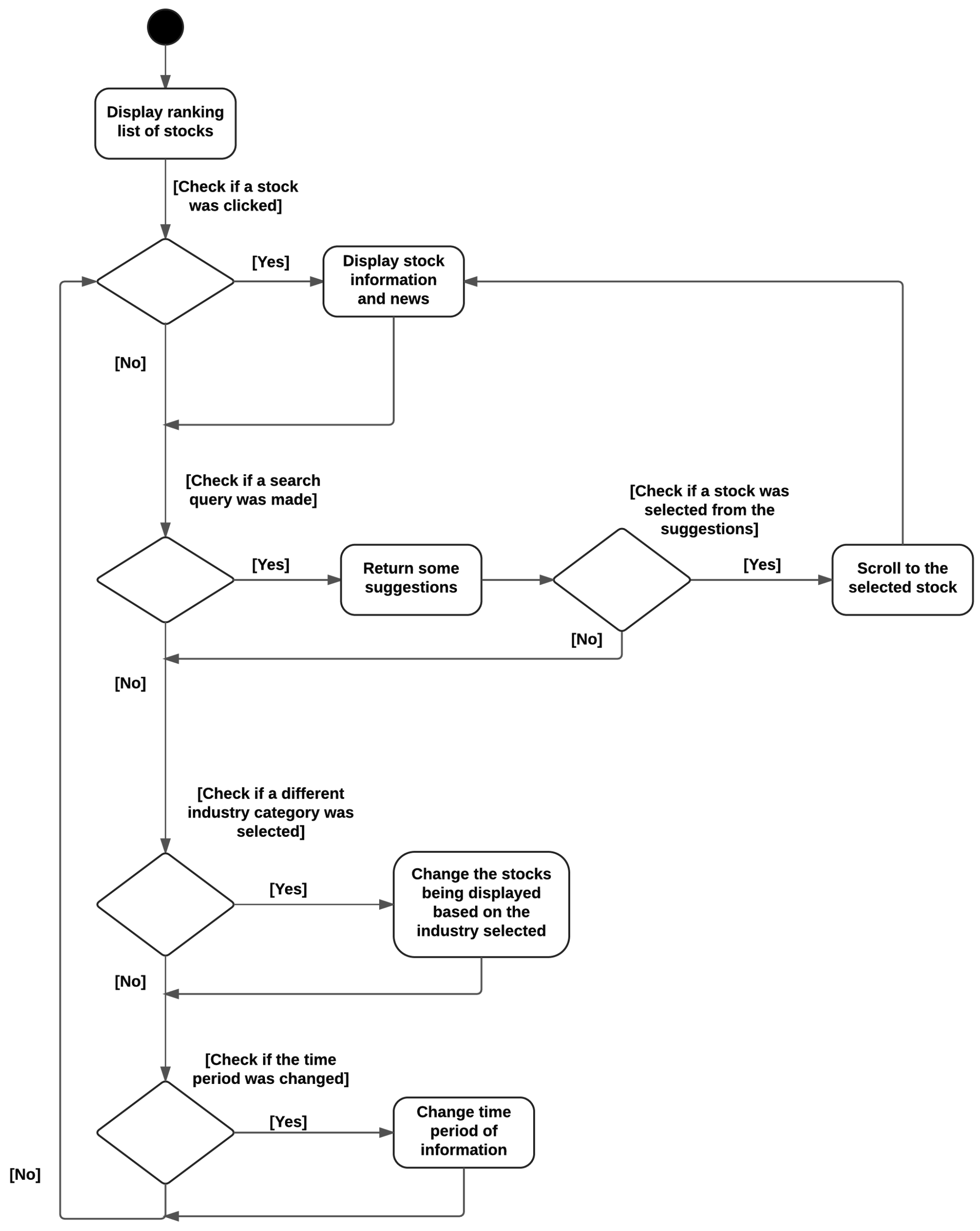

Display ranking list of stocks
[Check if a stock was clicked]
[Yes]
Display stock information and news
[No]
[Check if a search query was made]
[Yes]
Return some suggestions
[Check if a stock was selected from the suggestions]
[Yes]
Scroll to the selected stock
[No]
[No]
[Check if a different industry category was selected]
[Yes]
Change the stocks being displayed based on the industry selected
[No]
[Check if the time period was changed]
[Yes]
Change time period of information
[No]

Figure 1.3: Activity diagram for displaying stocks

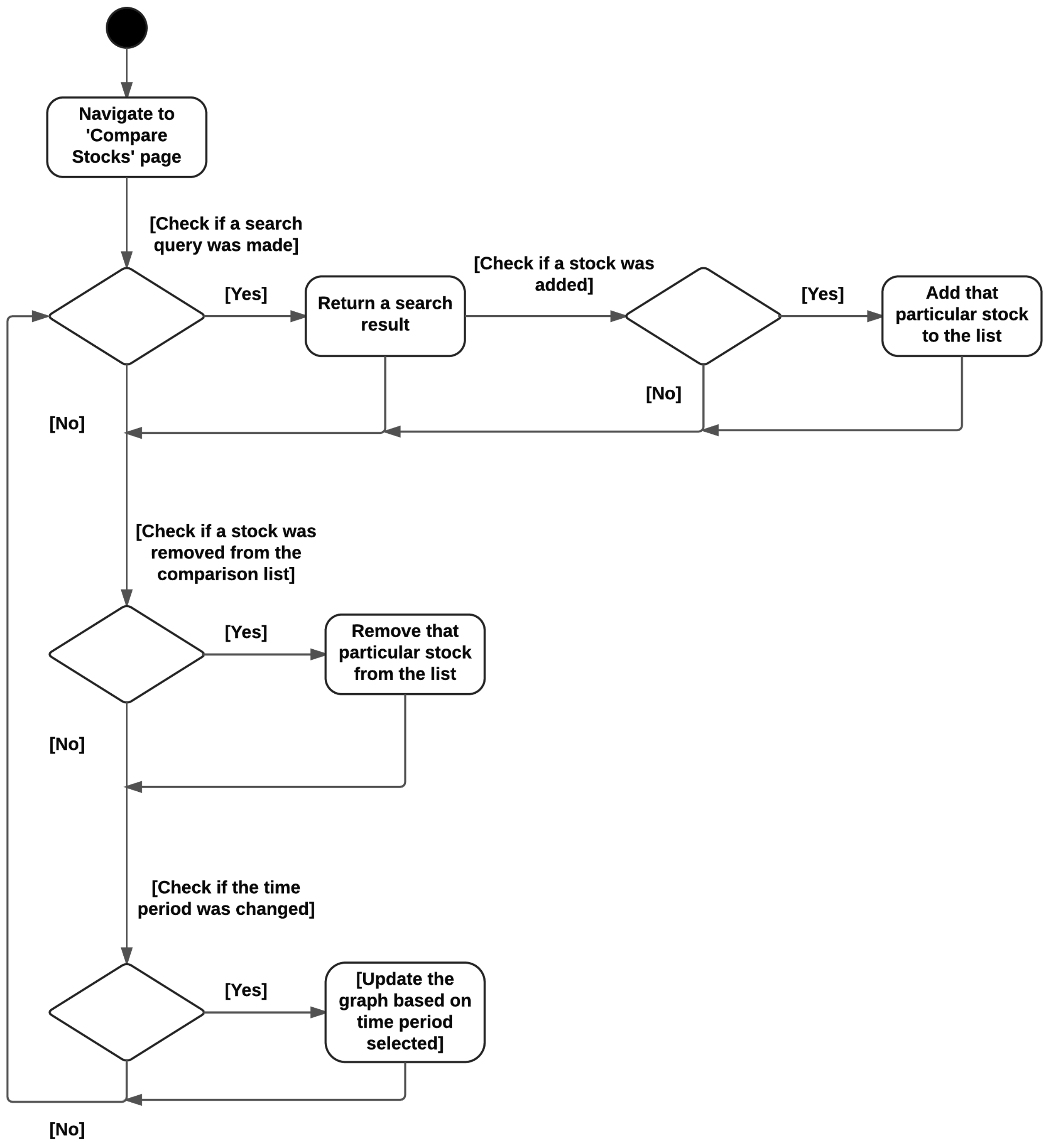

Navigate to 'Compare Stocks' page
[Check if a search query was made]
[Yes]
Return a search result
[Check if a stock was added]
[Yes]
Add that particular stock to the list
[No]
[No]
[Check if a stock was removed from the comparison list]
[Yes]
Remove that particular stock from the list
[No]
[Check if the time period was changed]
[Yes]
[Update the graph based on time period selected]
[No]

Figure 1.4: Activity diagram for comparing stocks

1.3 System Functions

The following list contain system functions that are core to the operation of the prototype.

Generating a ranking list based on time: As the user selects different time periods as shown in Figure 1.5, the system will collect data on the stock activity during these times, calculate the stock growth of each stock and then rank them in a list which is displayed on the webpage for the user to see.

Generating a ranking list based on industry: As the user selects an industry on the side button menu, the server will filter and display only those stocks which fall under that category. From this, the user can manipulate the list based on time period, for that select industry, the same way the user would be able to normally for the default stock list. This function was a suggested feature following the first project demonstration and thus, it is not included in the initial interface mock-ups below.

Expanded stock view: When the user clicks on a stock in the list, the item will expand as shown in Figure 1.6. The expanded view displays information relating to the stock such as growth over time, range, high and low values, etc. The information is displayed with respect to the chosen time period and updates accordingly when the user selects a different time period. In addition, the expanded view can be closed by clicking on the stock again.

Displaying news articles relevant to the stock: In conjunction to the chart, the expanded view features a news feed relating to the relevant stock. Using an API provided by WebHose.io we are able to populate a list to hold relevant news postings. Each posting features a thumbnail, a title and a link to the full article. We have done our best to keep it compact and un-intrusive yet easy to use. As with generating a ranking list based on industry, this feature was suggested in the feedback of the first project demonstration and hence, it is not included in the interface mock-ups.

Comparing Stocks: Figure 1.7 depicts the user interface that will be used when a user wishes to make a comparison against multiple stocks. The user can search for and add stocks to the graph, which also appear in a list for easy removal. Hovering over any data point of the graph will pop up information on all stocks at that point in time. Above the graph there are toggle buttons which allow the user to change the time period of the graph. Clicking these buttons will modify the graph's data plots and it's axis to reflect the new time period selected.

Searching stocks: The search bar of Figure 1.7 allows the user to be able to search for info for any particular stock they wish to see. The server should return the closest match to the search and display the stock's information below the search bar. In addition, this can be used in conjunction with our compare feature to compare multiple stocks of the user's liking.

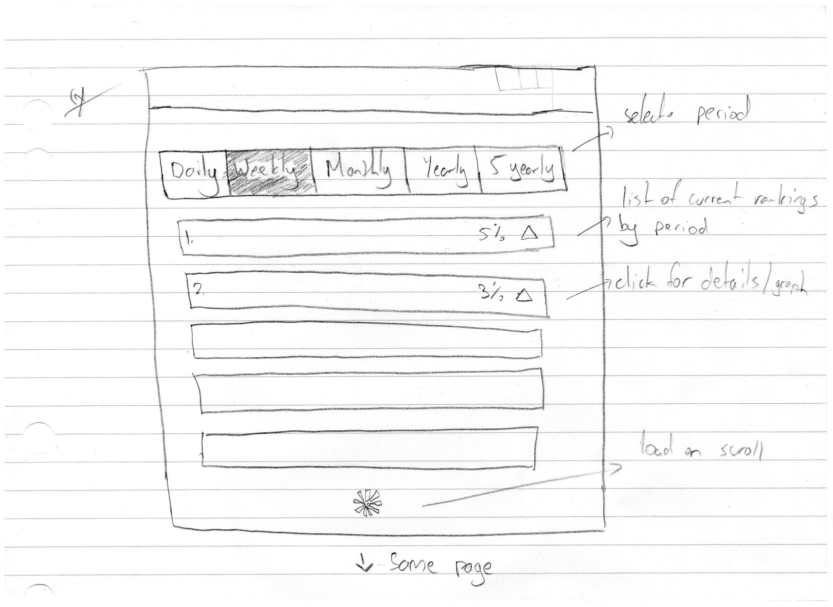


Figure 1.5: Displays the stock ranking list which functions as the homepage

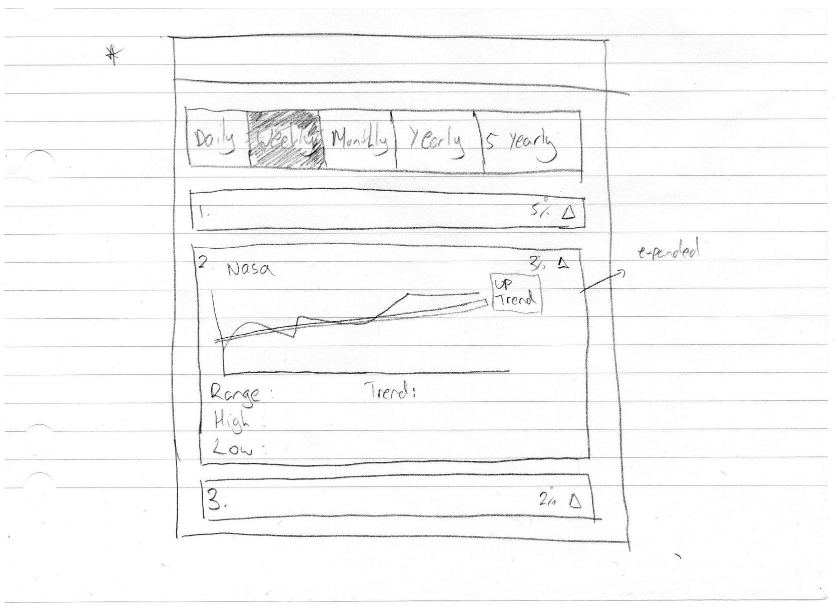


Figure 1.6: Displays the drop down graph after clicking on the list item

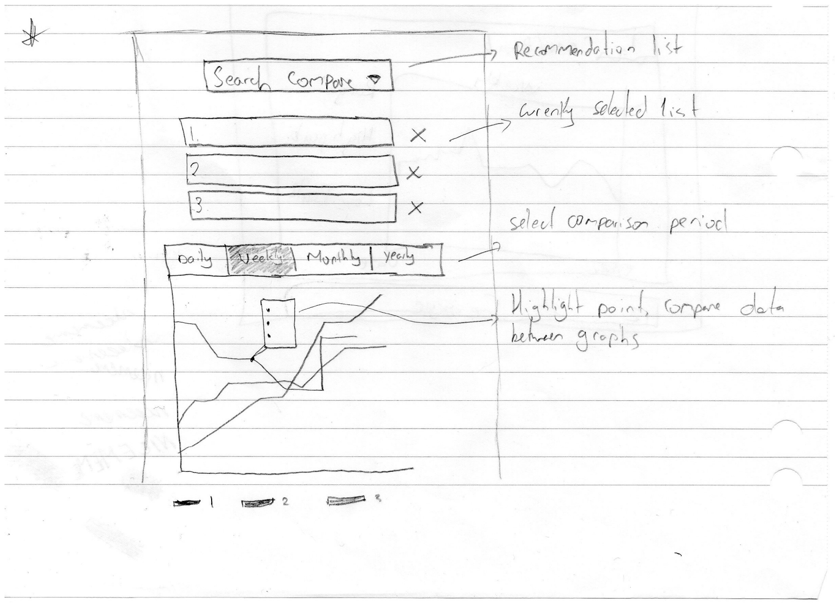


Figure 1.7: Displays a comparison of multiple stocks against one another

2 User Interface Design

2.1 Overview

The overall view and design of our website was based around a simple yet detailed navigation system. Featuring easy to browse and efficient point and click gestures, the user is presented with a highly maneuverable and customisable page. Navigation between timelines and different pages is non-intrusive and visually appealing. Every page is connected back to the original screen through the 'Home Button' which is very accessible but not distracting.

The design and flow of the ranking list page is simple and elegant. It enables the user to clearly ascertain the order of each company's stock growth in the list as well readily access more information with the simple search and scroll bar features. The use of infinite scrolling drastically reduces load time and is more appealing as it eliminates the need to refresh the page each time the end of the page is reached. The simplistic nature of the design adheres to the clarity design principle in order for the user to obtain and infer information easily.

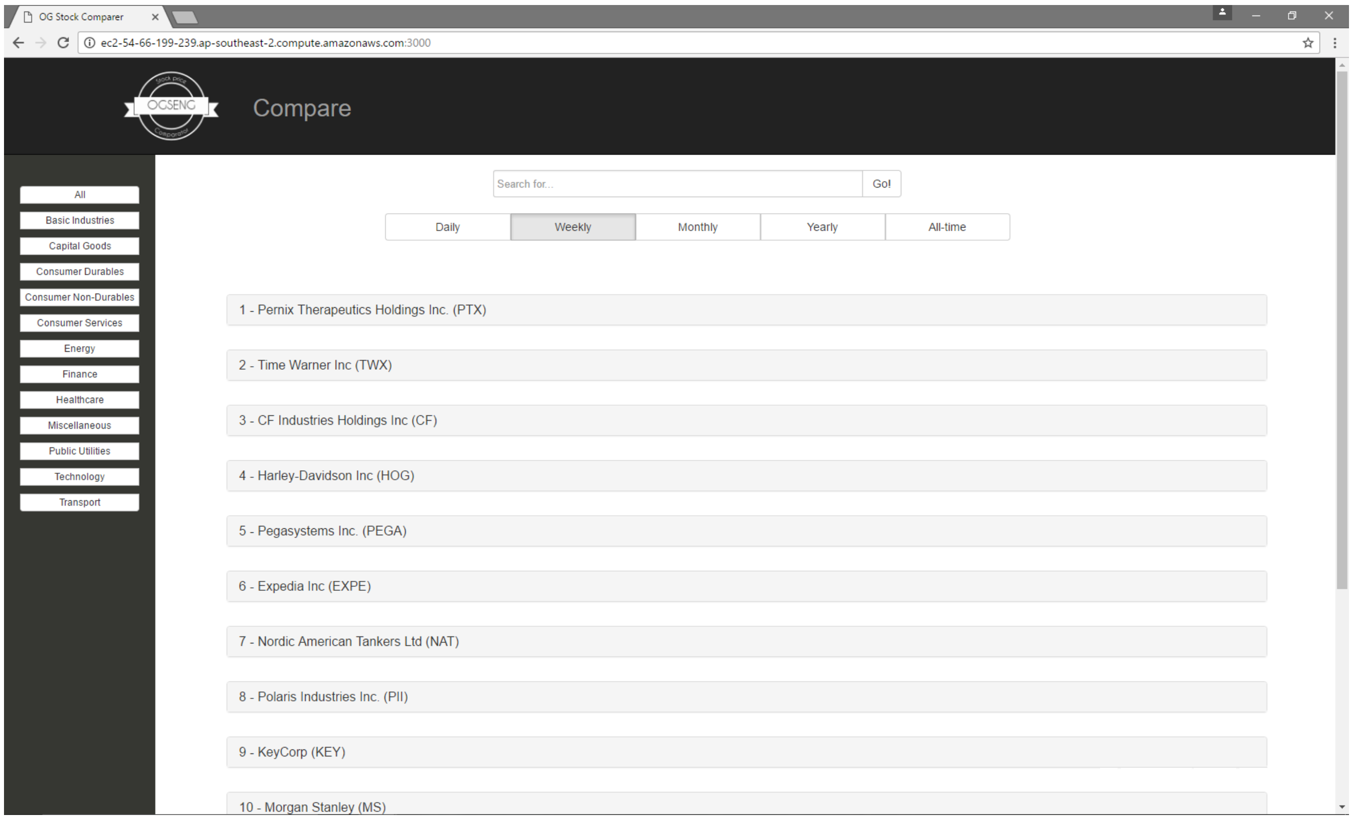
We have choosen to use the side bar column to hold our Stock Grouping buttons instead of a drop down menu to reduce clutter and improve visibility of the Ranking List. This adheres to the 'three-click-rule' navigation principle and reduces the amount of clicking and searching needed to filter by group.

This simplistic design is seen throughout the user interface which adheres to the consistency design principle as well. A rigid "column like" structure of the ranking list and buttons, was chosen to give off the impression of order and succinctness to the user, a trait that is critical to investors. The simplistic colour palette combined with the large buttons/icons were purposefully designed as to simplify navigation and not overwhelm the user with barbaric colour schemes.

2.2 Home Page Interface Design

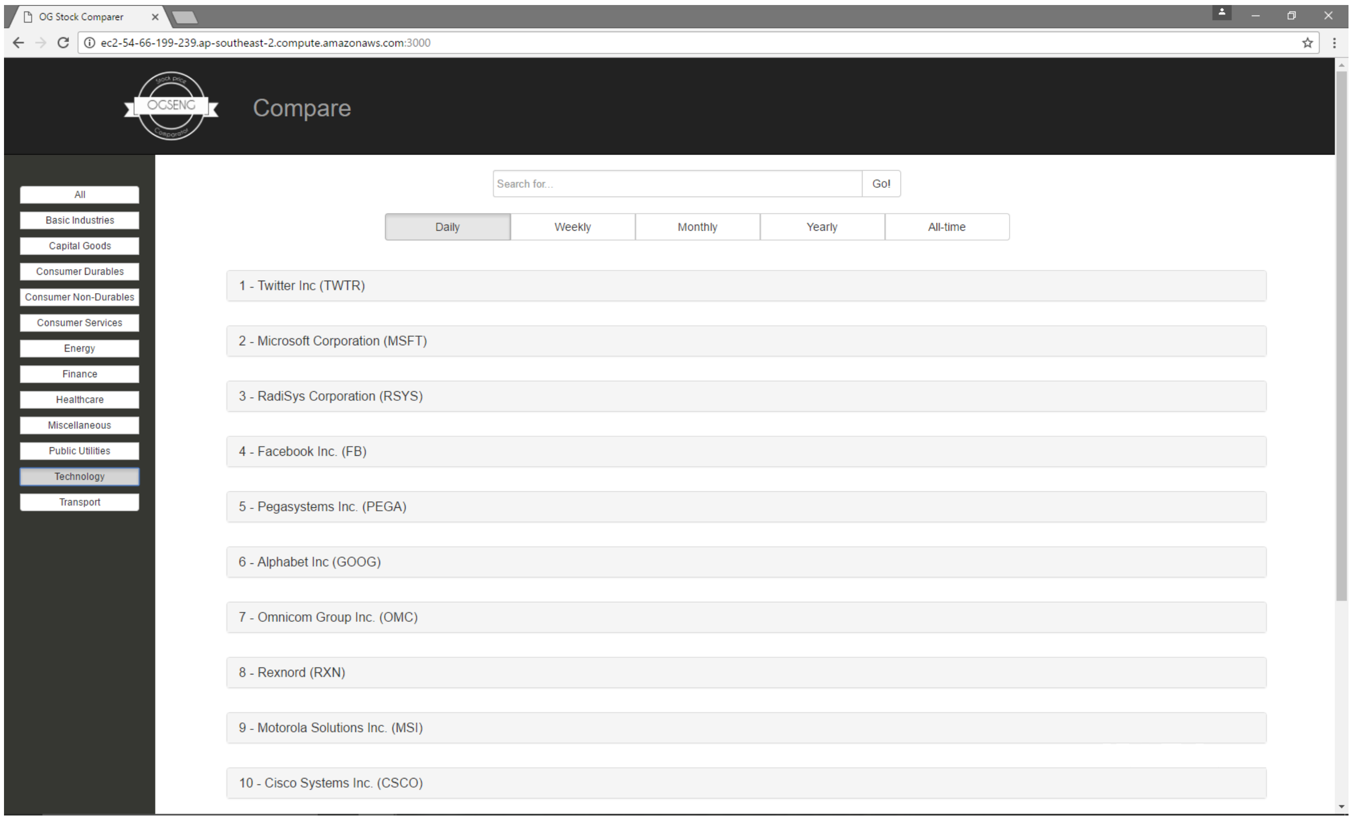
Our web page is based around the ‘Ranking list’ homepage. This is set up as an interactive list of scrollable elements that show the company name, its rank in comparison to other companies and its growth respective to the time period selected. When the user displays the home page (shown in Figure 2.1), they’ll see the first top fifteen companies with the greatest stock price growth for the last week.

From here, the user is also able to switch time periods and/or industry to display a new ranking list of stock growth (shown in Figure 2.2). In addition to displaying the first fifteen items, the user can scroll down to generate the next items in the list (similar to how Facebook will generate new posts once you scroll to the bottom), or alternatively search for a specific company in the list. Furthermore, the industry tags are located on the side of the home page as a list of small buttons,each of which contain a specific industry that the user can select to filter stocks.



1 - Pernix Therapeutics Holdings Inc. (PTX)
2 - Time Warner Inc (TWX)
3 - CF Industries Holdings Inc (CF)
4 - Harley-Davidson Inc (HOG)
5 - Pegasystems Inc. (PEGA)
6 - Expedia Inc (EXPE)
7 - Nordic American Tankers Ltd (NAT)
8 - Polaris Industries Inc. (PII)
9 - KeyCorp (KEY)
10 - Morgan Stanley (MS)

Figure 2.1: The home page of the website

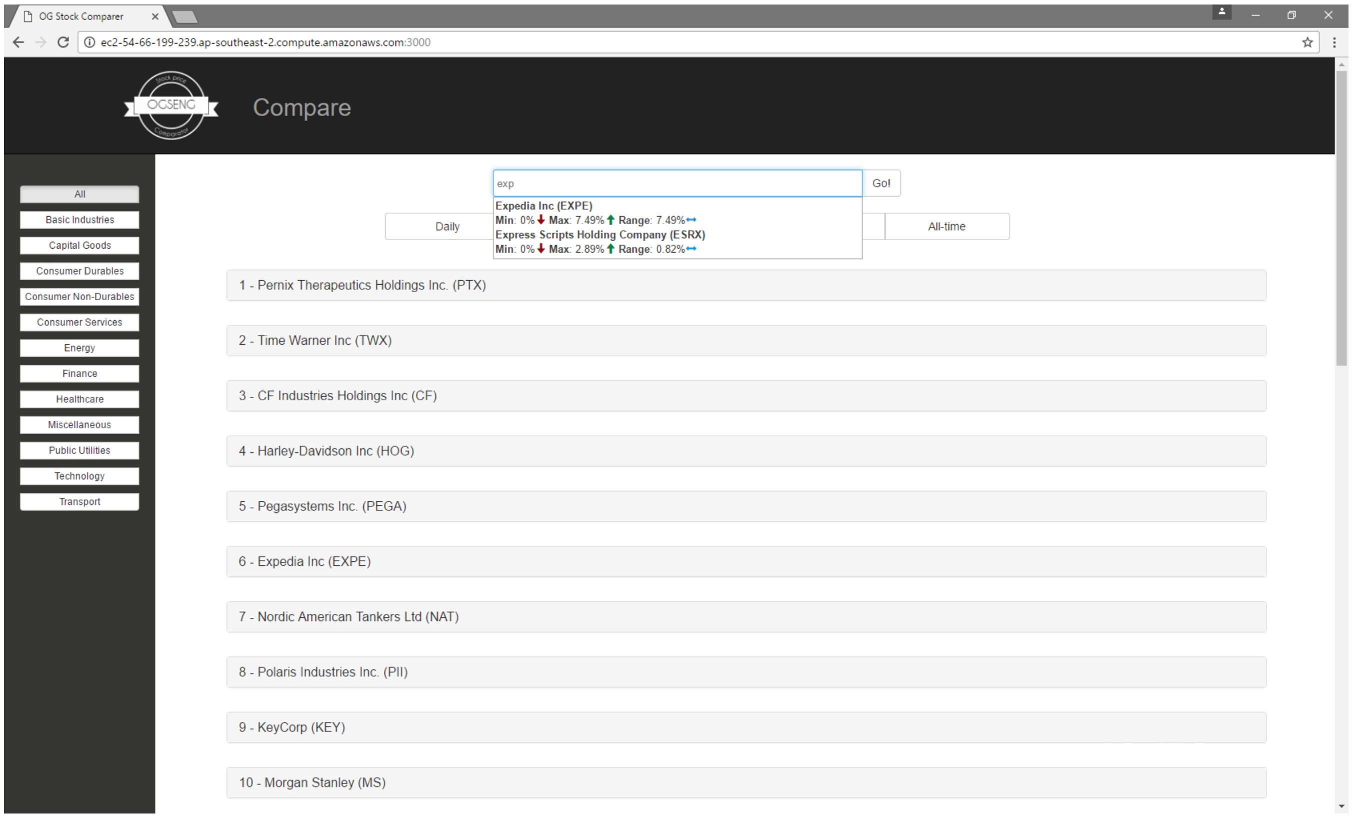


1 - Twitter Inc (TWTR)
2 - Microsoft Corporation (MSFT)
3 - RadiSys Corporation (RSYS)
4 - Facebook Inc. (FB)
5 - Pegasystems Inc. (PEGA)
6 - Alphabet Inc (GOOG)
7 - Omnicom Group Inc. (OMC)
8 - Rexnord (RXN)
9 - Motorola Solutions Inc. (MSI)
10 - Cisco Systems Inc. (CSCO)

Figure 2.2: Shows a new ranking list as the user selects the technology and daily buttons

2.3 Stock Search

As mentioned previously, the search bar allows the user to search for any given stock in the list for a given time period. The search bar incorporates dynamic searching (shown in Figure 2.3), similar to Google's autofill searches, which allows the user to differentiate companies with a similar name as well as to alert the user of the position the particular company is in the list, without even having to go through with the search query. Once the user selects an item in the search menu, the list will snap down to that item in the list, displaying the company stock growth for the given company as well as the subsequent companies below it (shown in Figure 2.4).

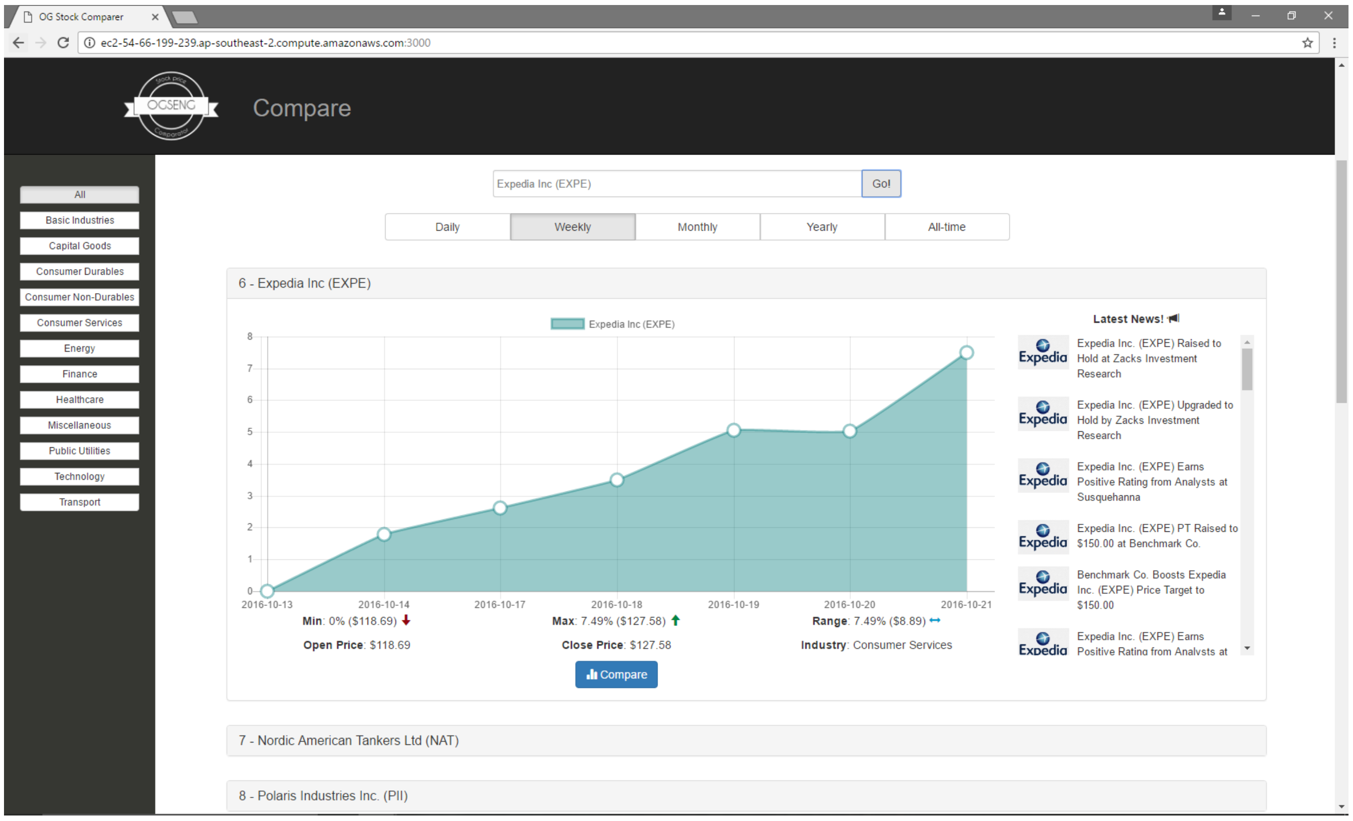


exp

Expedia Inc (EXPE)
Min: 0% **Max**: 7.49% **Range**: 7.49%
Express Scripts Holding Company (ESRX)
Min: 0% **Max**: 2.89% **Range**: 0.82%

1 - Pernix Therapeutics Holdings Inc. (PTX)
2 - Time Warner Inc (TWX)
3 - CF Industries Holdings Inc (CF)
4 - Harley-Davidson Inc (HOG)
5 - Pegasystems Inc. (PEGA)
6 - Expedia Inc (EXPE)
7 - Nordic American Tankers Ltd (NAT)
8 - Polaris Industries Inc. (PII)
9 - KeyCorp (KEY)
10 - Morgan Stanley (MS)

Figure 2.3: Auto-filling dynamic searching



6 - Expedia Inc (EXPE)

Min: 0% ($118.69)
Max: 7.49% ($127.58)
Range: 7.49% ($8.89)
Open Price: $118.69
Close Price: $127.58
Industry: Consumer Services

Latest News!

Expedia Inc. (EXPE) Raised to Hold at Zacks Investment Research

Expedia Inc. (EXPE) Upgraded to Hold by Zacks Investment Research

Expedia Inc. (EXPE) Earns Positive Rating from Analysts at Susquehanna

Expedia Inc. (EXPE) PT Raised to $150.00 at Benchmark Co.

Benchmark Co. Boosts Expedia Inc. (EXPE) Price Target to $150.00

Expedia Inc. (EXPE) Earns Positive Rating from Analysts at

7 - Nordic American Tankers Ltd (NAT)

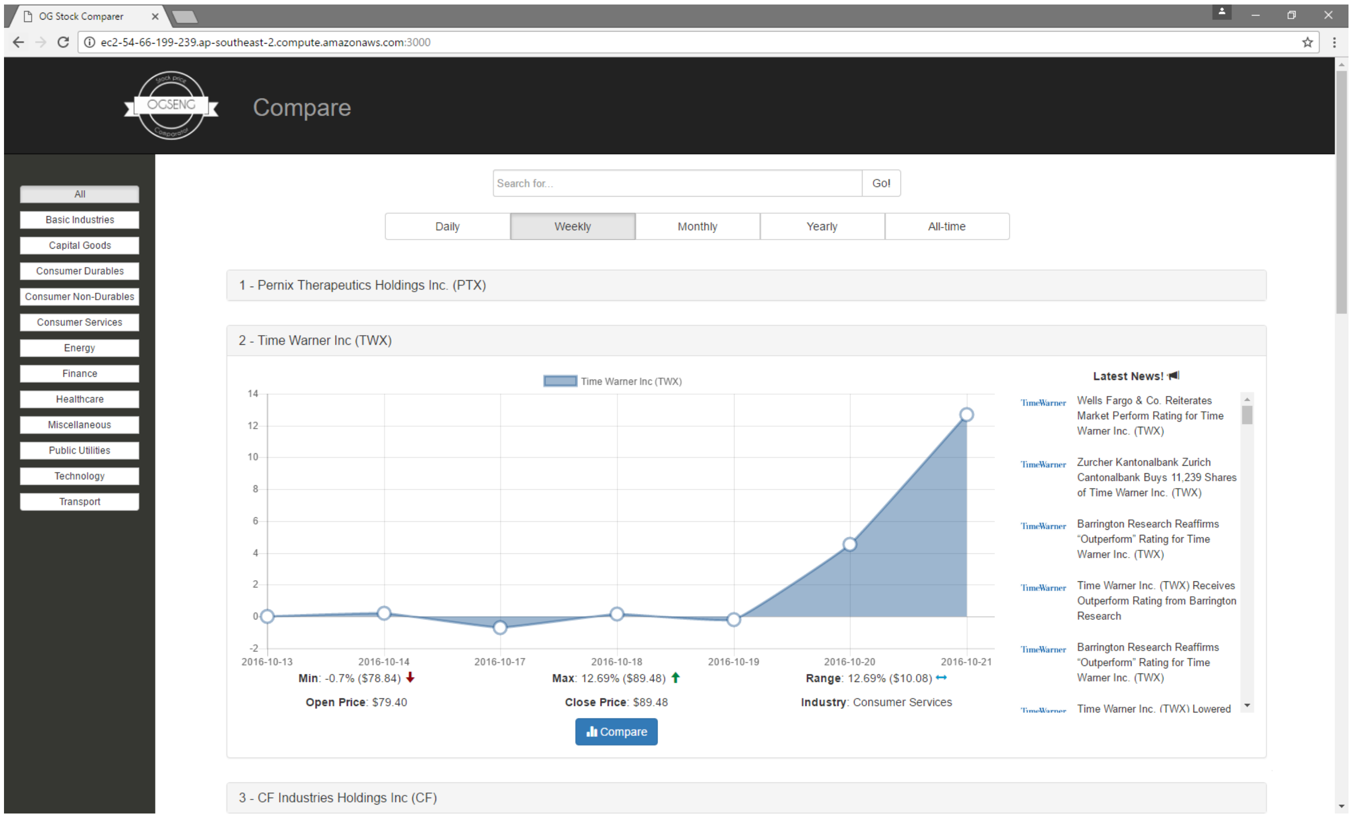
8 - Polaris Industries Inc. (PII)

Figure 2.4: Snapping down to selected stock

2.4 Expanding Stocks

While browsing the Ranking List the user has the option to view more information about a given stock. Clicking on a stock item will initiate a smooth dropdown animation as the element is expanded. This expansion presents the user with an interactable graph, useful stock price information, an option to compare this stock against others (shown in Figure 2.5), as well as news articles relating to the particular stock.

The drop down animation automatically extends the list, sliding some elements out of view, gives the user a rewarding experience in return for their action. In accordance with responsive design, hovering over data-plots loads a tooltip which enlarges the value of that datapoint and other information the price. Clicking the top of the stock element again will snap the element back into its simplified form. Located to the side of the graph is a scrollable column containing several news articles, each of which have an image, a title and subsequent text underneath. Located near the bottom of the graph is the large "Compare" button. When clicked, the user will be sent to the stock comparison page (shown in Figure 2.6).



1 - Pernix Therapeutics Holdings Inc. (PTX)

2 - Time Warner Inc (TWX)

Min: -0.7% ($78.84)
Max: 12.69% ($89.48)
Range: 12.69% ($10.08)
Open Price: $79.40
Close Price: $89.48
Industry: Consumer Services

Wells Fargo & Co. Reiterates Market Perform Rating for Time Warner Inc. (TWX)

Zurcher Kantonalbank Zurich Cantonalbank Buys 11,239 Shares of Time Warner Inc. (TWX)

Barrington Research Reaffirms "Outperform" Rating for Time Warner Inc. (TWX)

Time Warner Inc. (TWX) Receives Outperform Rating from Barrington Research

Barrington Research Reaffirms "Outperform" Rating for Time Warner Inc. (TWX)

Time Warner Inc. (TWX) Lowered

3 - CF Industries Holdings Inc (CF)

Figure 2.5: Expanding a stock item

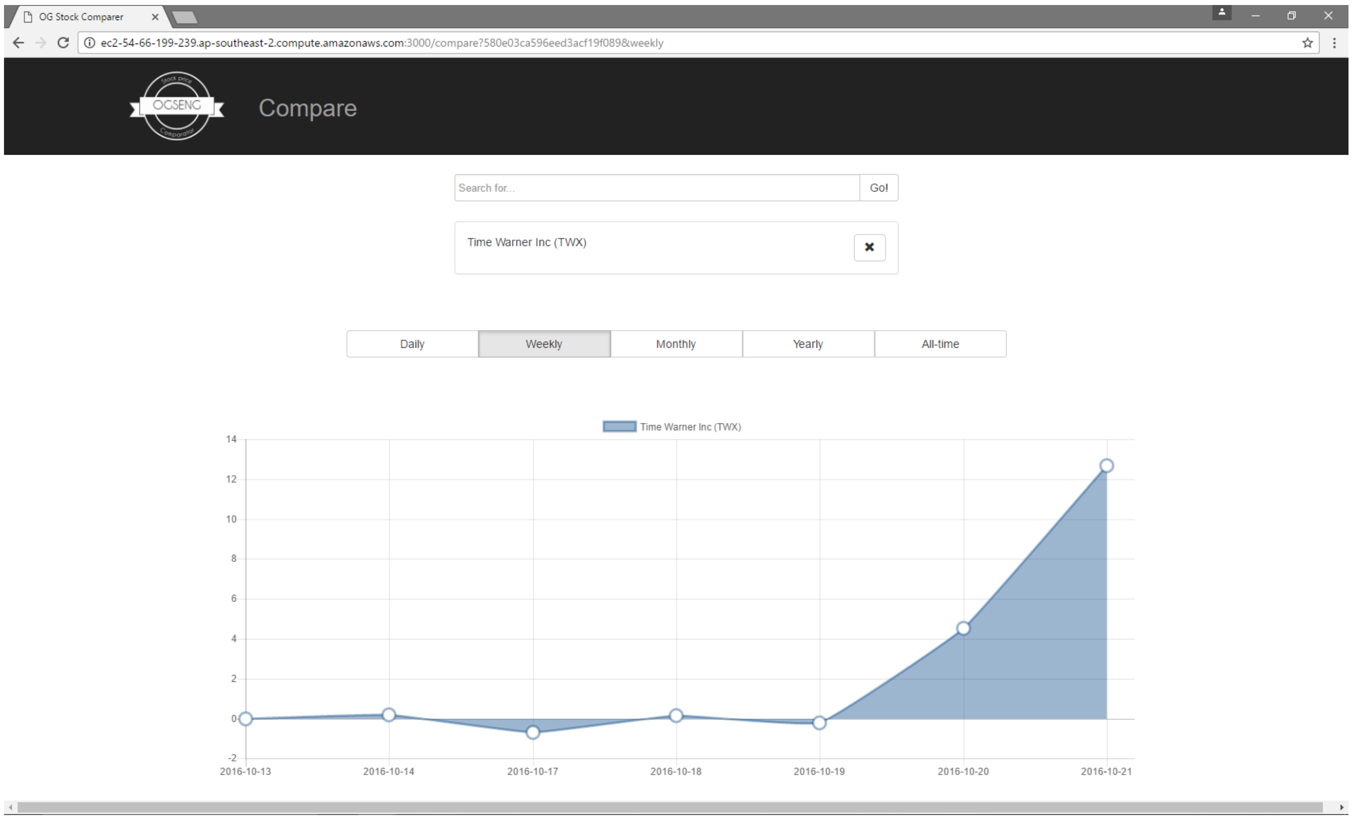


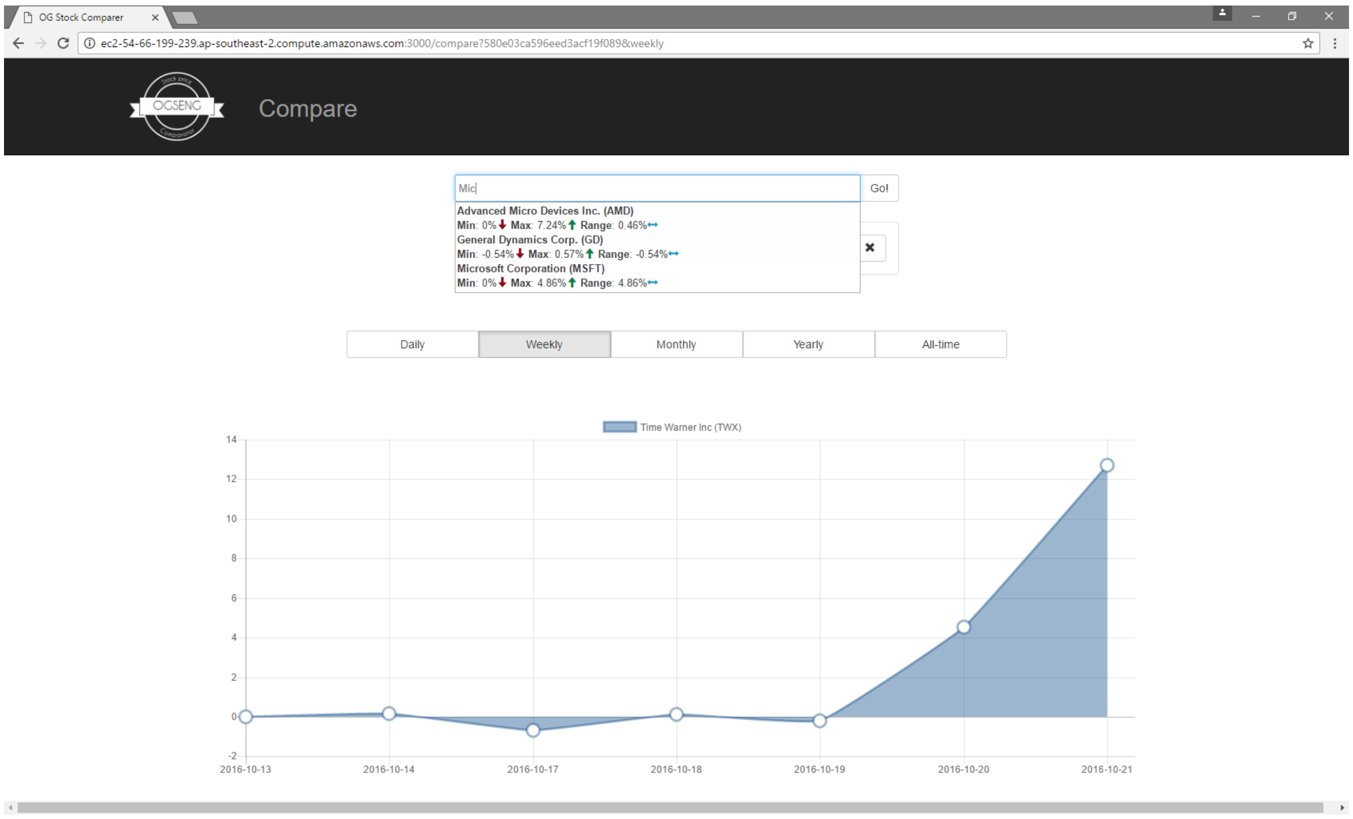
Figure 2.6: Switching to compare page

2.5 Stock Comparison

The stock comparison page is used to compare numerous stocks. Users search for particular stocks and add them by clicking on the desired stock from a list of results. Stocks may also be removed by clicking the remove icon beside the stock. Adding and removing to the list of stocks automatically updates the graph at the bottom of the page. Users select the time period of the comparison using the buttons just above the graph. The following mockups demonstrate the roles and functionality of the user interface elements.

2.5.1 Adding Stock

Suppose that the user wishes to add Microsoft to the list. The user will input a term into the search bar and a drop-down menu will automatically appear with the relevant stocks, as shown in Figure 2.7. When the user clicks on a stock in the drop-down menu, it is added to the list. Figure 2.8 shows Microsoft added to the bottom of the list after being clicked on. Upon adding the stock, the graph updates accordingly.



Advanced Micro Devices Inc. (AMD)
Min: 0% **Max**: 7.24% **Range**: 0.46%
General Dynamics Corp. (GD)
Min: -0.54% **Max**: 0.57% **Range**: -0.54%
Microsoft Corporation (MSFT)
Min: 0% **Max**: 4.86% **Range**: 4.86%

Figure 2.7: Search for stock

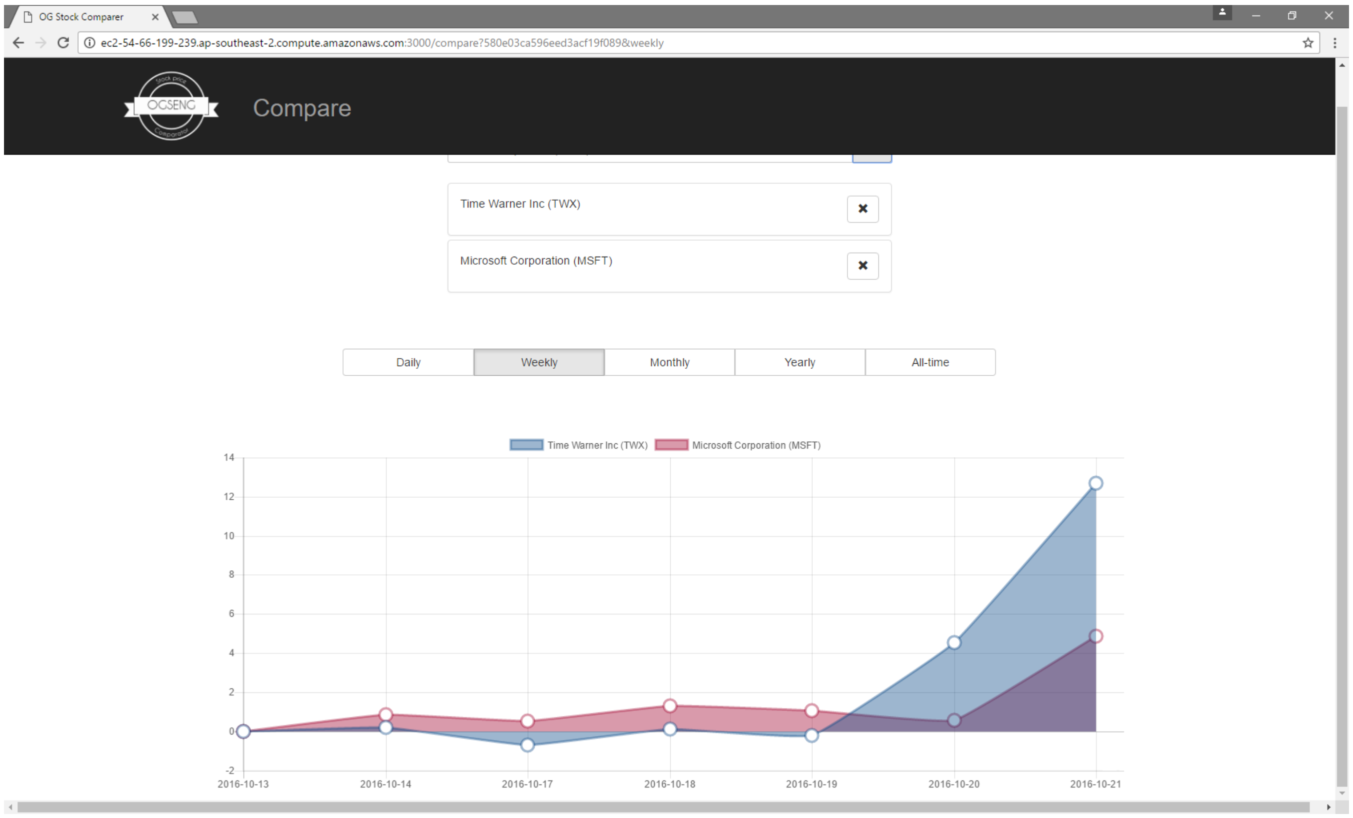


Figure 2.8: Stock added

2.5.2 Changing Compare Period

Users can change the comparison time period by selecting the desired time as depicted in Figure 2.9.

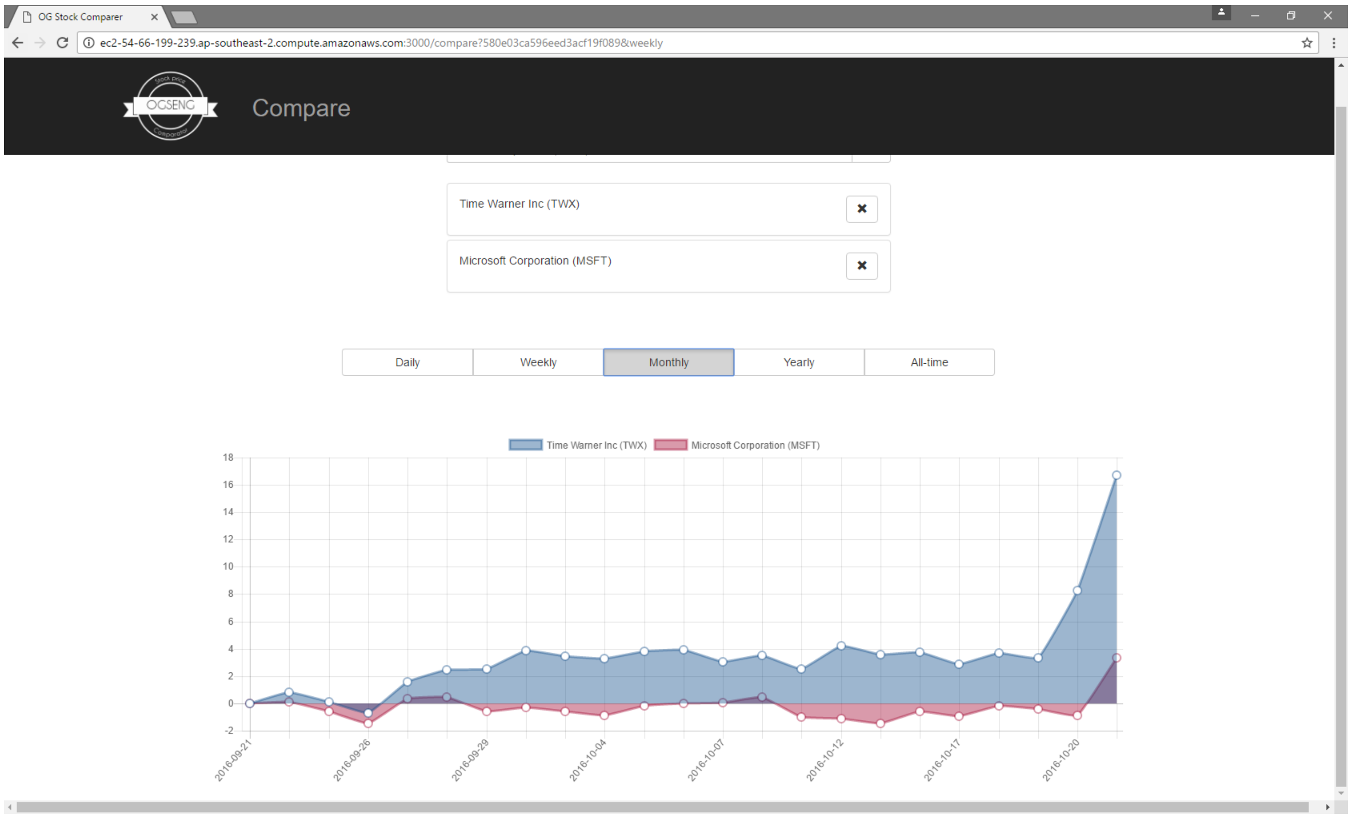


Figure 2.9: Changing time period

2.5.3 Removing Stock

Now suppose the user wishes to remove Microsoft. The user clicks the 'X' icon beside the Microsoft stock which removes it updates the graph accordingly, as seen in Figure 2.10.

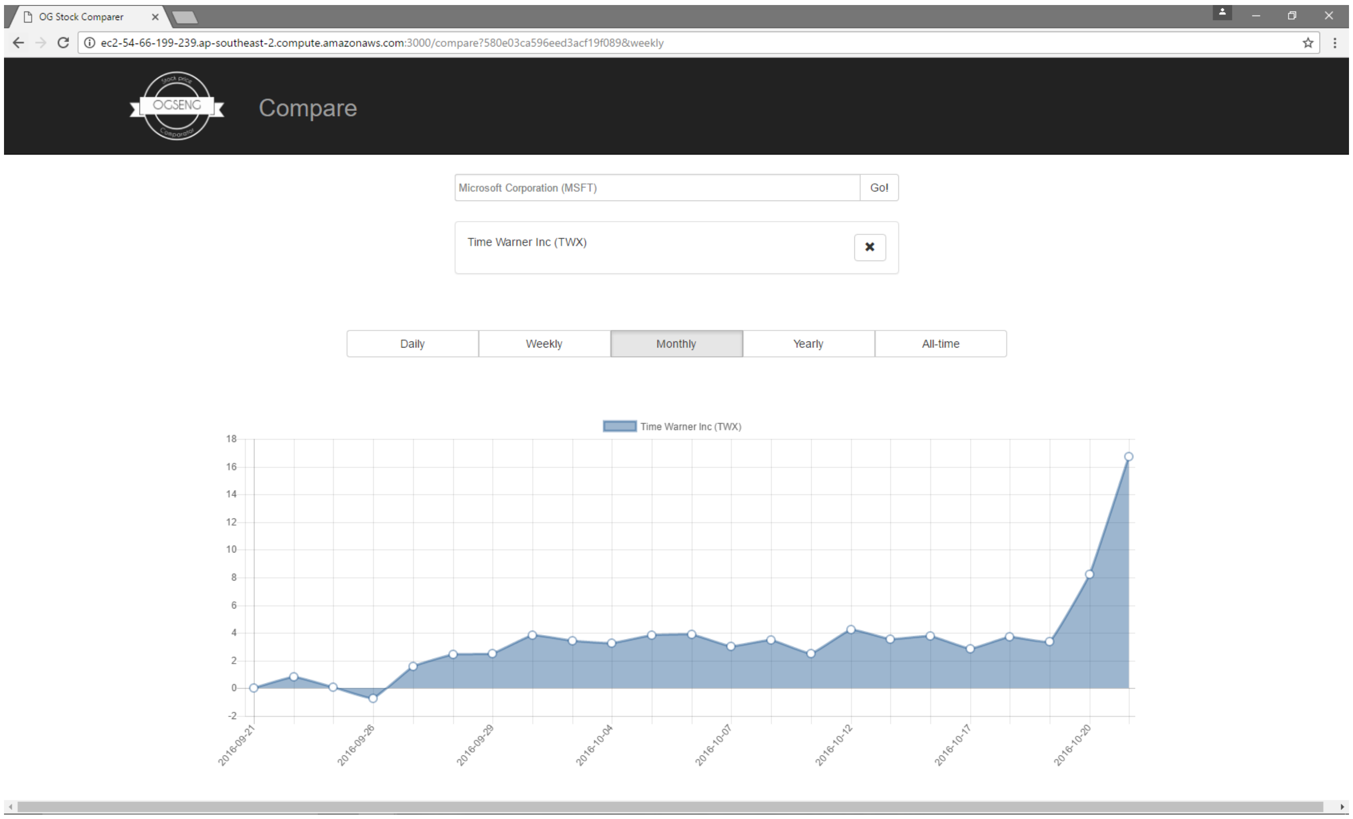


Figure 2.10: Stock removed

2.6 Brief Summary

We have included a simple storyboard to summarise the content in this report. The storyboard explains the interactions between the elements in each page and between pages.

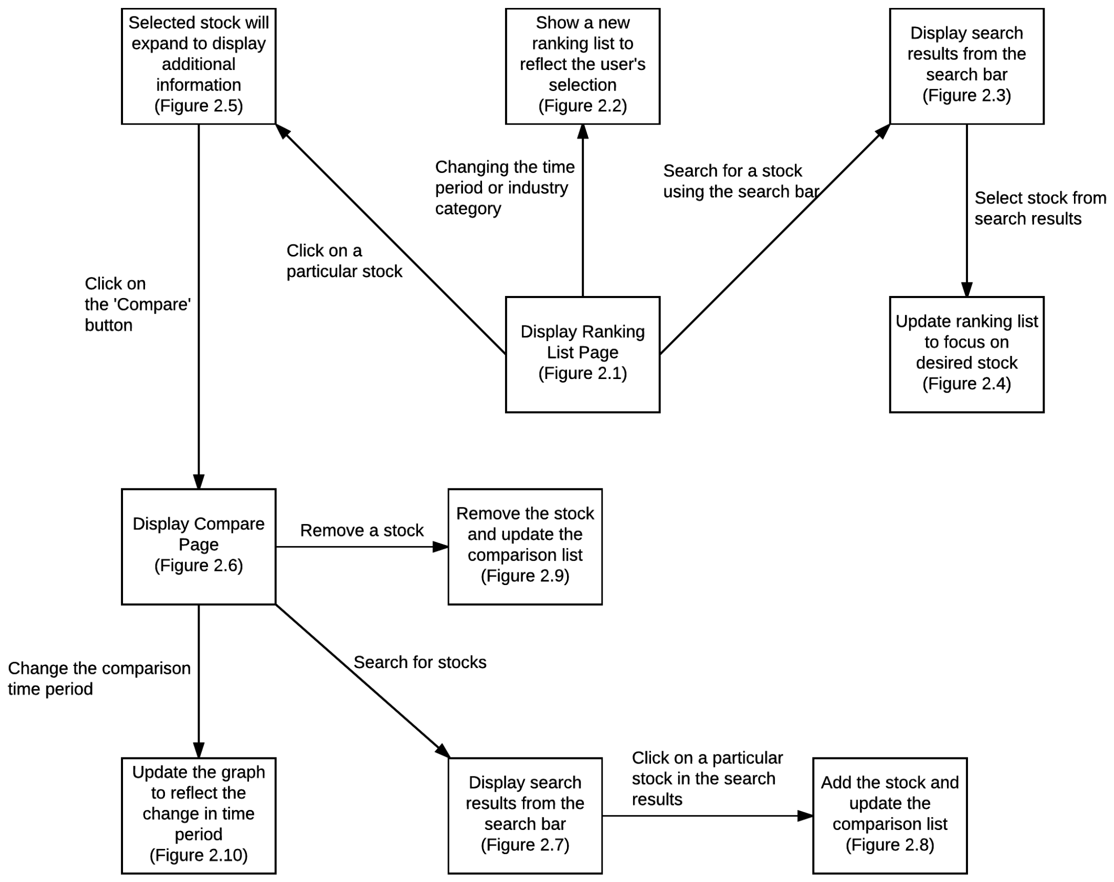

Selected stock will expand to display additional information (Figure 2.5)
Show a new ranking list to reflect the user's selection (Figure 2.2)
Display search results from the search bar (Figure 2.3)
Changing the time period or industry category
Search for a stock using the search bar
Select stock from search results
Click on a particular stock
Click on the 'Compare' button
Display Ranking List Page (Figure 2.1)
Update ranking list to focus on desired stock (Figure 2.4)
Display Compare Page (Figure 2.6)
Remove a stock
Remove the stock and update the comparison list (Figure 2.9)
Change the comparison time period
Search for stocks
Update the graph to reflect the change in time period (Figure 2.10)
Display search results from the search bar (Figure 2.7)
Click on a particular stock in the search results
Add the stock and update the comparison list (Figure 2.8)

Figure 2.11: User Interface Story Board

3 Sequence Diagrams

3.1 Ranking List Use Case

All use cases under the ranking list use case follows the basic overview in the sequence diagram below.

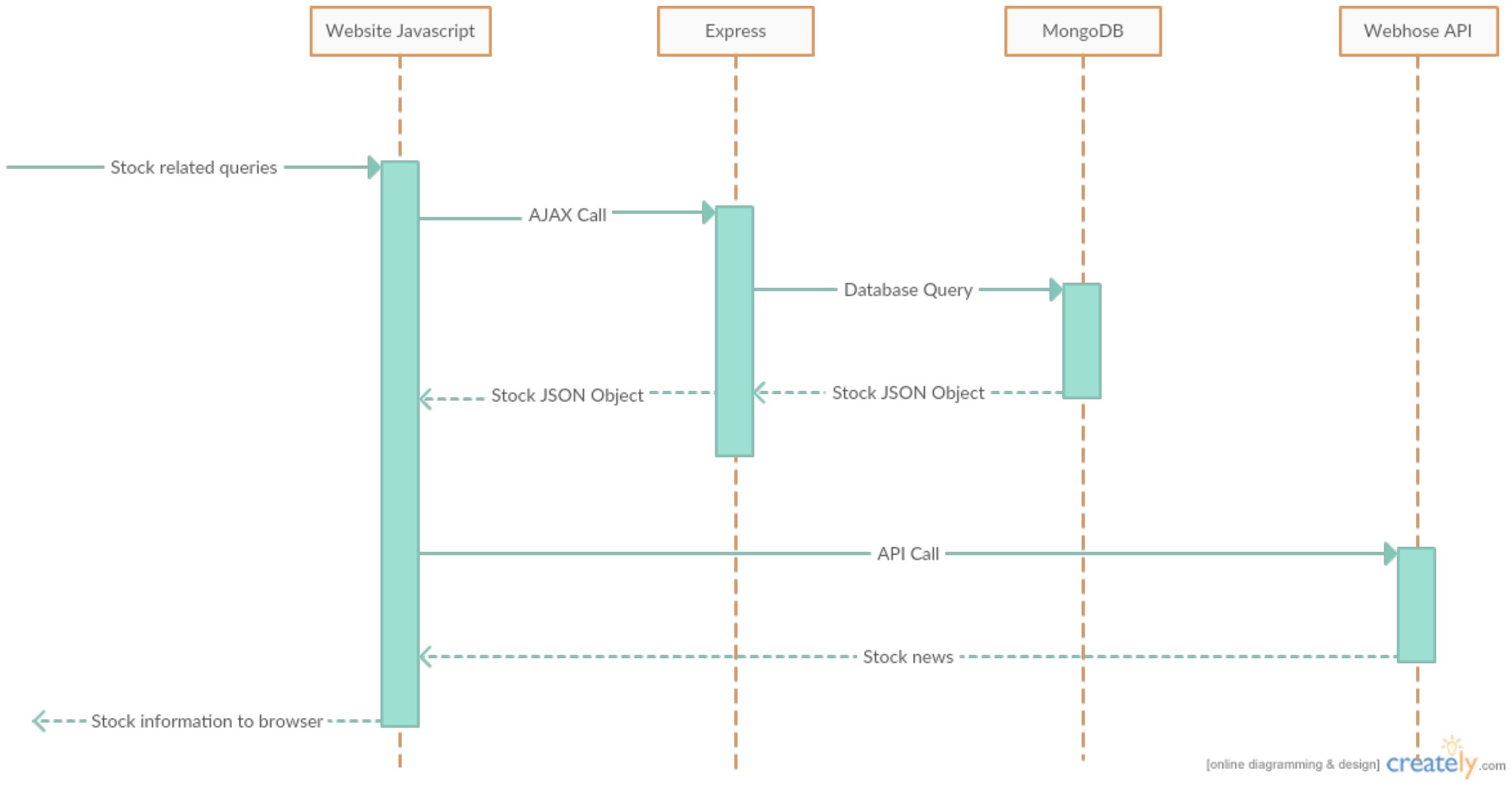


Figure 3.1: Sequence diagram for ranking list use case

3.2 Comparing Stocks Use Case

All use cases under the comparing stocks use case with the exception of the removing stock use case follows the basic overview in the sequence diagram below.

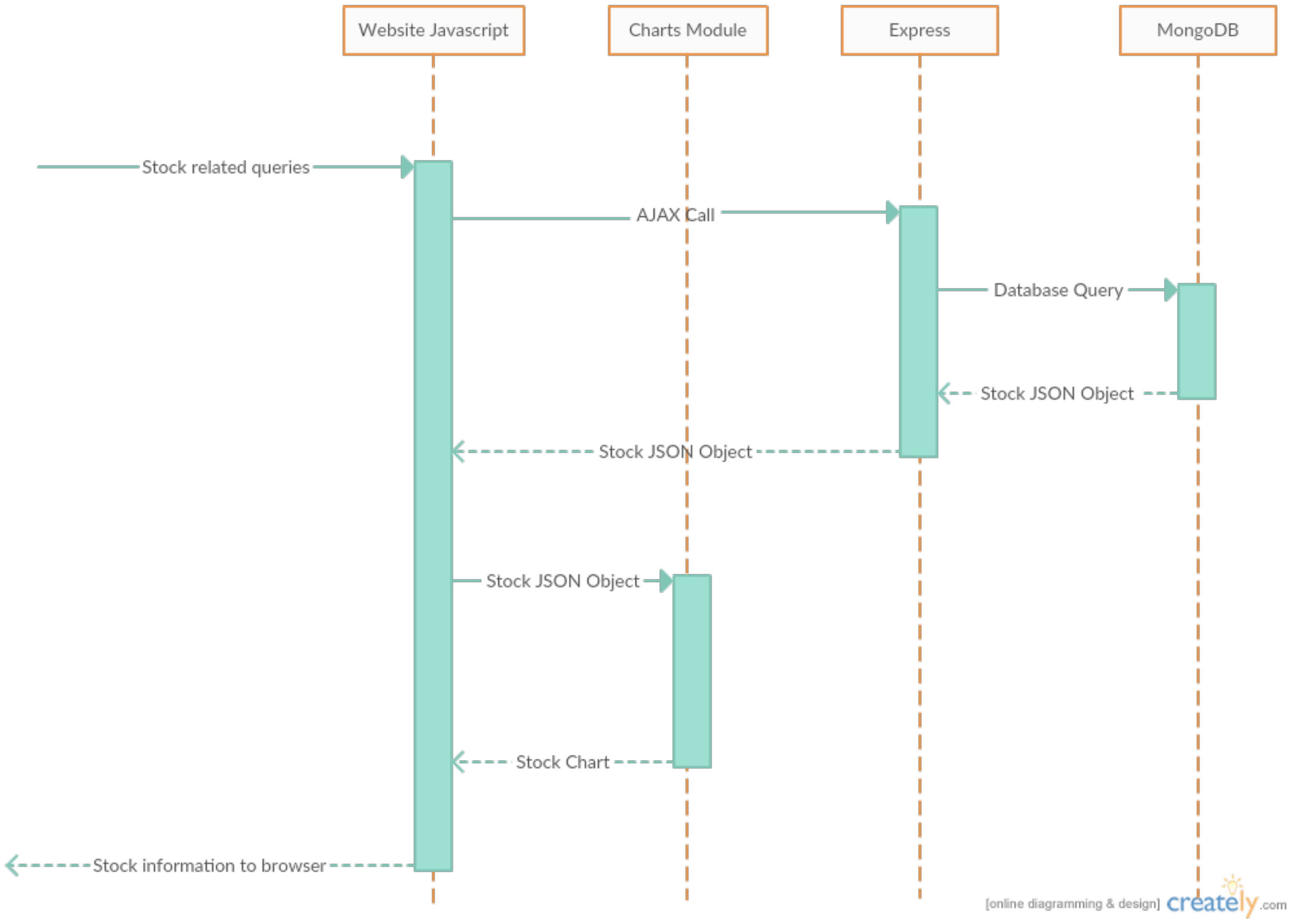


Figure 3.2: Sequence diagram for ranking list use case

The removing stocks use case is unique to the other use cases and is as follows:

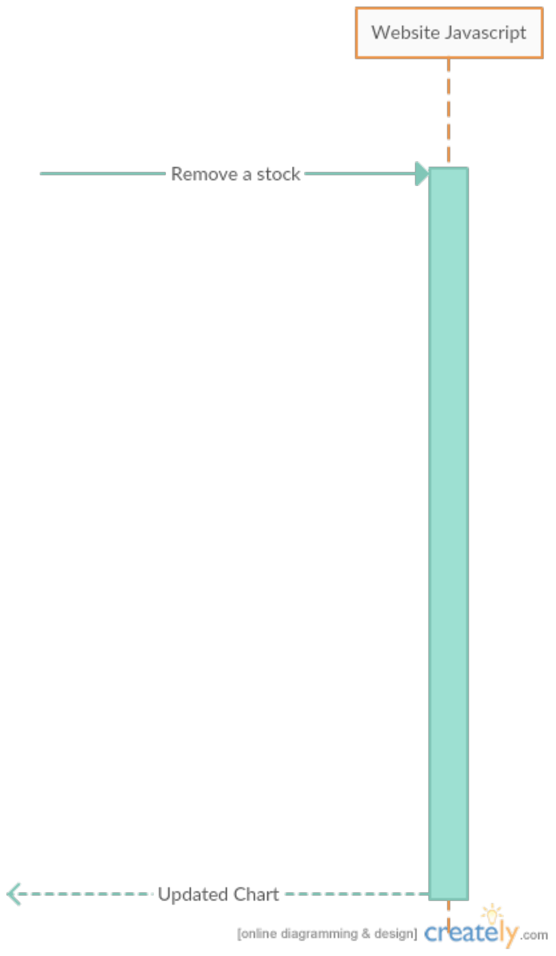


Figure 3.3: Sequence diagram for ranking list use case

4 Software Architecture

4.1 Software Components

Node.js: Node.js is the run-time environment which executes our back-end Javascript code. This is effectively the backbone of our back-end implementation.

Express.js: As users browse the website, the front-end components retrieve necessary information from the database using HTTP requests. Express.js is the back-end framework used to handle such requests. For example, when a user searches for a stock, the front-end passes on the query to the web server's back-end. Express.js handles this query and returns the relevant stocks as a JSON object. Express.js forms the majority of our back-end implementation.

MongoDB: MongoDB is the database we use to store stocks on our web server. MongoDB is a NoSQL database which stores JSON-like objects.

Bootstrap and Jquery: The front-end of our system is comprised of a combination of Bootstrap and Jquery. Bootstrap is a front-end framework that is used for designing websites through HTML and CSS templates. jQuery is a Javascript library which runs on the front-end of the system and provides an interactive user interface.

Chart.js: Charts.js is a powerful Javascript graphing library which we utilise to deliver simplistic and interactive stock data to the user. The library is flexible enough for us to simply pump data straight from our database into the chart and manipulate the labels to show the time scale. Its flexibility extends to the aesthetics of the graph. We were able to easily add animations, unique colour schemes, custom line tensions, and tooltips.

WebHose.io: In order to retrieve the news for our news feeds we use an API provided by WebHose.io. This API connects us to a vast database of news articles, blog posts and forum threads that have been freshly scraped from the Internet. We are also able to ensure quality and relevancy on the news by placing restrictions on our query such as, date, high social media presence and matching by keywords.

Quandl: The driving force behind our solution is a means to be able to grab both up-to date and historical financial records for companies. Quandl is the end point for hundreds of financial databases, the information from the databases can be viewed on their website, exported or, most importantly accessed through their REST API. Our solution requires being able to acquire daily stock prices of companies across a given time line. Quandl allows us to do this very easily, we can get the daily closing stock price of Facebook with a few simple parameters.

Company: FB

Start Date: start_date=2014-01-01

End Date: start_date=2014-12-31

Collapse: collapse=daily

To produce the request:

https://www.quandl.com/api/v3/datasets/WIKI/FB.json?column_index=4&start_date=2014-01-01&end_date=2014-12-31&collapse=daily&api_key=(6Bc43JXNTrFsXTqtgY3D)

We get a JSON response which has a closing price for each day of 2014.

For demonstration purposes, we are using a free Quandl account. Free accounts have limited features and are subjected to a limit of 2000 API calls per 10 minutes. We feel that the services available for free accounts are more than sufficient for demonstration purposes.

4.2 Relationships Between Components

Our back end will feature the ability to grab and compare the growth and demand of different companies from NASDAQ. This data will be fetched and processed by our Node.js backend, utilising Quandl's API to retrieve the information in a JSON response and then using a JSON parser to extract the information. The extracted information is then passed to our MongoDB database to be stored until the user requests for it. When the user expands a stock item, the Javascript function responsible for stock item expansions will initialize an AJAX call to the Express framework which fetches data from the database. This data is then loaded into the charts module which displays the data in a graphical fashion. The same Javascript function which handles stock item expansions also loads the latest news about the stock. This is done through a asynchronous HTTP GET request to the Webhose.io API. When the data returns from the API call, we parse it through a JSON parser and display it on the screen.

5 Implementation Considerations

5.1 Choice of Web Technology/Framework

Initially we had decided to use the MEAN stack or: MongoDB, Express, AngularJS and Node.js. However, due to various reasons such as problems with AngularJS and advice from our mentors, we transitioned to a variation of the MEAN stack whereby we swapped AngularJS with Bootstrap and JQuery. This has not affected the operation of our stack in any adverse way.

For the front end, as specified above, we are now using a combination of Bootstrap and JQuery. We have decided on using these technologies as we have found them easy to use to achieve the functionality and design of the website. Bootstrap readily provides a CSS file loaded with a variety of element designs which we can use to put together a website easily. JQuery allows us to embed interactive elements into the website to provide for a more interesting and pleasing experience.

As for the back end, it consists of MongoDB, Express and Node.js. This is due to the versatility of JavaScript and the ability of Node.js to handle an event-driven environment. Express is necessary and also complements Node.js as it provides the request handling needed for us to build our web application. Although we only have to enough data to satisfy the requirements of a prototype, MongoDB is useful in that the records are stored as JSON-like documents. This is allows ease in converting data from collected from web APIs to records to be stored in our database.

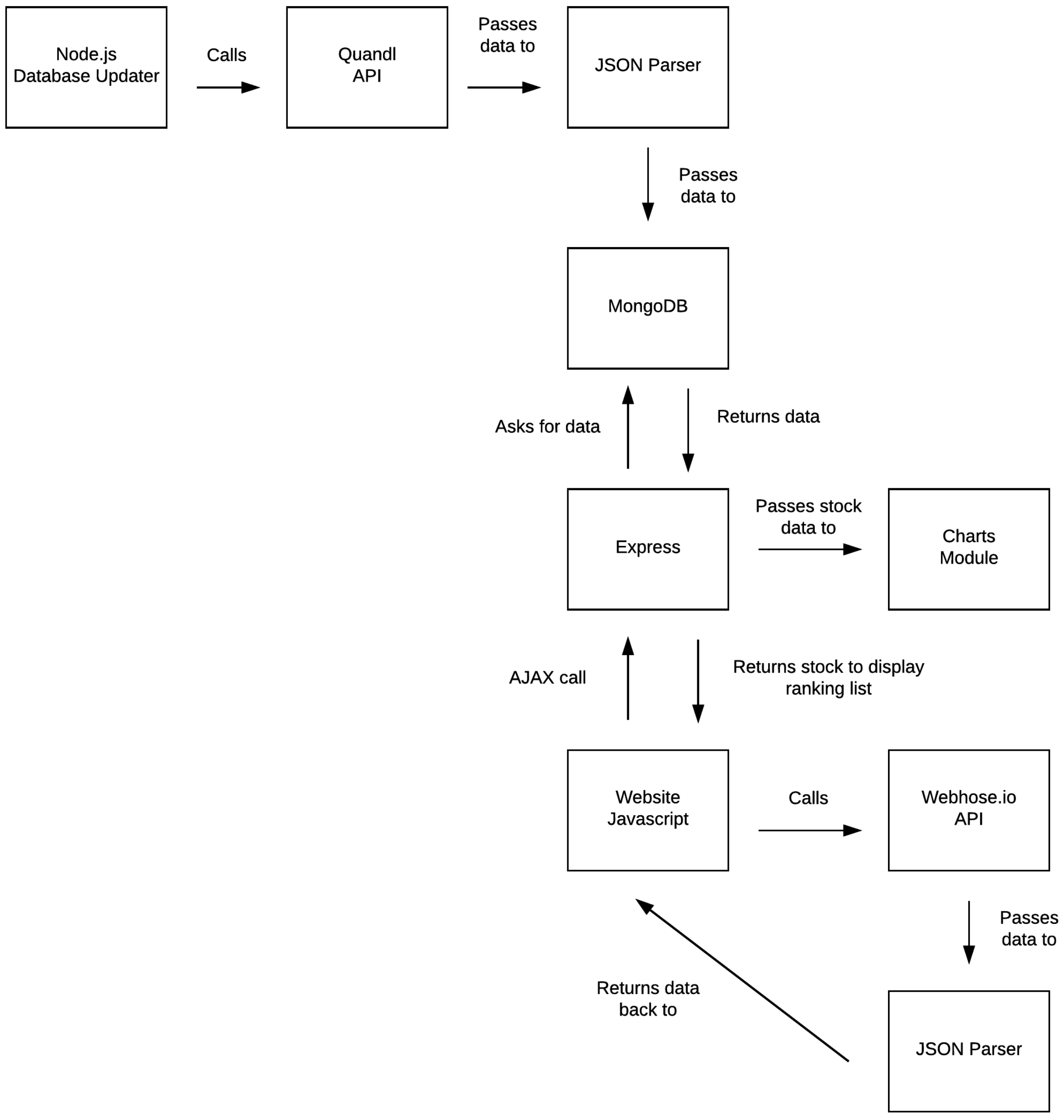


Figure 4.1: Software Components Relationship Diagram

5.2 Relating Choices to Components

As explained in the section above, we have decided to use a variation of the MEAN stack where we swap AngularJS with Bootstrap and JQuery. The reason we chose to use this particular stack was because we wanted to explore innovative and exciting options. This course was our first try at developing a web based application that incorporates the use of web APIs. Although we thought it might have been a bit difficult using such an advanced stack for our first web application, we believe it would be worth the effort to learn something which the industry will be using frequently in the near future.

Aside from our own personal reasons, we thought that the MEAN stack was advantageous in several ways. The stack heavily relies on Javascript, which is widely believed to be a better web language than its counterpart, PHP. Javascript could do more in far fewer lines and has more functionality than PHP. It is also widely supported by many browsers and there are many free open source libraries which offer additional functionality.

MongoDB uses NoSQL and is the alternative to SQL and its variations. There are many advantageous of MongoDB which include:

- Data is stored in a JSON format which allows for easy retrieval and parsing.
- Easy to setup.
- Small in size and uses little system resources.

Node.js was used instead of and not with Apache or any other web server technology due to the nature of our web application. It is a interactive application where the page will change dynamically based on user input. As stated before, Javascript is a more promising technology compared to the combination of Apache and PHP. Express was added due to the excellent synergy with Node.js. It provides extended features on top of Node.js and helps manage routing between front end and back end elements. Lastly,

Bootstrap and JQuery was used instead of other alternatives such as AngularJS or React as they were easier to use and had extensive documentation available. JQuery also provides us with additional functionality such as auto-complete and chart plotting libraries.

5.3 Choice of Platform

Our web application will be run on an Linux machine utilizing Node.js as its web server software. This is due to Linux being lightweight and familiar. We will be employing the help of Amazon Web Services to deploy our website on their machines. Specifically, the website will be running 24/7 on a EC2 instance. As for clients, we intend to have the web application usable by all sorts of browsers that support Javascript and obviously HTML.

5.4 Licensing of Components

When choosing the components to use in our system, we also considered the licensing aspect of project design. We ensured that all components used were distributed under either an open source license or public domain.

Node.js, Express.js, Bootstrap, jQuery and Chart.js are all open source components that are distributed under the MIT License. Under this license, we are able use, copy, modify, merge, publish, distribute, sub-license, and/or sell copies of the component without restriction.

MongoDB is also open source and is distributed under the GNU Affero General Public License. The license requires that modifications to the source code are made publicly available. We weren't required to make any modifications to MongoDB, so this is not an issue for us.

The information provided by Quandl is Public Domain. We may copy, distribute, disseminate or include the data in other products for commercial and/or noncommercial purposes.

6 Conclusion (Team Organisation/Appraisal)

6.1 Responsibilities of the Team

6.1.1 General Team Roles

For the duration of the course, the team utilised the scrum process to develop the project. Kaan took on the role as scrum master and facilitated meetings as well as assigned deadlines of certain tasks. Damon took on the role of product owner and thus all design ideas were discussed with Damon before Damon made the ultimate decision on a particular design. The rest of the team were appointed as scrum members.

6.1.2 Project Specific Roles

In terms project specific roles, the team worked on different parts of the prototype as follows:

- Damon was in charge of the design of the prototype, focusing his attention on developing the layout of UI and interaction between elements
- Orion majorly focused on the developing the backend, this included routing between pages as well as transferring data from the backend database and displaying it on the frontend
- Simon worked on displaying the interactive graphs in our UI as well as implementing the news article API next to the graphs.
- Kaan worked on processing the data from the Qandle API and storing it into the database, as well as styling of CSS and bootstrap elements.
- Kevin worked generally on all areas of the project mainly testing and debugging code, as well as worked on a majority of the reports

6.2 Organisation

Scrum meetings were held twice a week (one in person and one via Skype) to discuss the tasks that were completed, tasks that needed to be completed in the foreseeable future, as well as any factors that might hinder our ability to complete the tasks assigned by the given deadline. In addition a Gantt Chart was prepared to help ensure that all team members were on the same page and that tasks were completed by the given deadline. Several Google documents were created to work collaboratively on the reports as well as note down several ideas that team members had during meetings.

6.3 How Did the Project Go?

In the teams unanimous opinion, we felt that although time constraints were a little heavy for the big project we set out to develop, we ultimately are satisfied with the prototype that we have developed. The team feels that this experience has bestowed many valuable lessons, not only gaining experience on working collaboratively within a group environment but also gaining exposure to the many languages needed to develop the prototype for this project.

6.4 Issues

The issues encountered weren't necessarily related to team cohesion or arguments among team members but more so related to time constraints on accomplishing certain tasks by a given deadline. All team members had taken a networking course (COMP3331) and a software construction course (COMP2041) during this semester, both of which released assignments and several lab exercises which harshly added to the already demanding workload of the SENG2021 course. This coupled with the fact that the team needed to learn several languages and become familiar with software such as MongoDB presented some issues to the team.

6.5 Conclusion

Looking back on the project and the events that transpired, starting from developing draft UI designs to finishing a working prototype, the team is satisfied with their performance and thus wouldn't change a thing given the opportunity to start over.