The University of Nottingham | School of Computer Science | MSc Computer Science

FinVis: Visualizing the Complex Nature of Financial Markets

Submitted in September 2020, in partial fulfilment of the degree MSc Computer Science.

–

I hereby declare that this dissertation is all my own work, except as indicated in the text:

James Williams - 10/09/2020

–

Supervisor: Dr Robert Laramee

Abstract

Investors are constantly looking for insights through comparisons on stock market data to assist in the discovery of well-priced stocks. Modern stock market applications and research allow for a range of visualizations to be produced whilst enabling the viewing of complete datasets, however, these tools are often split, not providing queries and analysis in the same view, therefore, limiting the potential of the proposed solution. Within this thesis, a new system for performing analysis and comparison is proposed, a web-based application which is able to assist in the exploratory visual analysis and visualization generation of financial data. FinVis allows for over 500 stocks to be queried and 10,000 to be imported, before displaying interactive and customizable visualizations to assist in the discovery of bargains or good investments on the market. The application includes tools to save, store and load queries alongside a visual analytic display to enable customization and more complex individual views to be generated. A complete set of interactions are also provided to the user, whilst enabling all of this exploration to take place with real-time and historical data being provided through a financial API to the user's web browser. The thesis enables future work within the area of exploratory financial visualization to take place, providing core concepts in a comparative nature to potential researchers and investors alike.

Keywords: Financial Visualization; Visual Analysis; Information Visualization; Financial Data; Stock Markets; Finance Analytics.

Word Count: 19,998/20,000

Acknowledgements

I would like to thank my project supervisor Dr Robert Laramee for the support and advice throughout the project, he has assisted me in improving my research ability and knowledge within information visualization.

I would also like to thank my family for supporting me throughout the masters program.

Finally I would like to thank Amna Anwar for proofreading the document before submission.

Contents

1 Introduction

This thesis explores the research, design, implementation and evaluation of a finance visualization platform produced to assist in finding bargains on the stock market. Many tools exist which allow for market data to be viewed or analyzed, but lack features relating to custom searches and effective comparisons between different symbols. Throughout the thesis, novel research will be shared assisting in the potential for future business and research-oriented applications.

1.1 Motivation

When investigating stocks to buy, investors often use comparisons as a method of analysis. Modern applications such as Google Finance [Goo20], Yahoo Finance [Yah20b] and Seeking Alpha [Alp20] provide access to large datasets relating to the stock market, however lack effective query-based tools and provide limited options in relation to comparison of multiple stocks and their values. To assist investors in the discovery of undervalued or potentially profitable stocks, a visualization and visual analysis platform was produced which enabled exploratory investigation to take place into individual stocks in comparison to one another.

1.2 Research Problem

Financial visualization and analysis platforms often host and present large amounts of data and analysis tools, however limited tools and capabilities exist for comparing stocks within the market. Therefore the problem being researched is to explore better and improved methods of visualizing and querying large amounts of financial data and to discover if an exploratory visualization tool has capabilities when compared to existing use cases such as Google Finance and Seeking Alpha.

1.3 Aims and Objectives

The aim of the project is to produce a financial visualization and analysis system capable of both producing queries on stock meta data and providing comparisons of stock portfolios. The objectives of the project are:

- Develop a web-based system capable of visualizing large amounts of stock market financial data.
- Enable users to create and run queries within the application, to assist in finding specific stocks.
- Provide a large number of user options relating to the visual display of stock data, all of which are customizable by the user.
- Enable the quick access of previous queries through data storage, and provide analysis techniques to match.

1.4 Key Results

Through findings discovered within three case studies, the proposed solution is effective at querying and producing visualizations of stock market data. The application can effectively search and present analysis techniques specific to the users requirements. To further test the proposed solution, access to larger amounts of meta-data is needed and more focus on interaction with the line chart itself should be performed. In the future, work should focus on ensuring more techniques are implemented to analyse data directly in cases specific to real time and historical datasets.

1.5 Thesis Structure

The thesis is structured following the recommendations from Bob's Project Guidelines [Lar10b]. Following the introduction is a section on related work, which begins with an in-depth literature review and comparison to existing systems before exploring the characteristics of the chosen dataset. Section 3 introduces the project in the form of a project specification, providing the core and enhancement features and the choices of technology for implementation of the chosen functionality.

A complete project plan is presented in section 4, which explains how meeting notes were used and the progress on each core function of the system. Succeeding the project plan is section 5 which displays the design diagrams and design requirements of the system.

Presented after the design and project plan is the implementation phase, which is in chapter 6. The implementation initially focuses on showcasing the basic implementation and relevant enhancements before explaining the programming conventions used throughout the project. Following the implementation phase is the testing and evaluation which makes use of case studies and a performance analysis to explore the potential impact of the work.

Finally the conclusion is displayed within section 8, which explains the potential impact of the work and any key discoveries made throughout. This is followed by a short section on the future work that could be completed as part of the system in section 9.

2 Related Work

The related work section introduces literature alongside current developments within industry. The section aims to explore the existing developments within financial visualization, to ensure the novelty and originality of the research project. Content from multiple sources was gathered and compared, enabling a complete overview of existing systems to be presented, before the characteristics of the proposed data is explored.

2.1 Literature Review

Financial visualizations have developed significantly in recent years, the majority of visualization and financial analytic tools are used within business intelligence and market analysis software which are produced to allow traders to better understand markets [ID17; PM17]. Financial markets have relied heavily on technical innovation since the very start of the industry, many traders and companies use the latest technology to gain advantages against competitors. In modern cases this comes with an increase in usage of business intelligence systems, dashboards and algorithms to assist in the understanding of ever-increasing complexity of the data. These technological innovations are often implemented closely and provide insights to one another to assist in performing more successful trades. Financial systems in this regard can be directly judged against their performance in real-world markets, allowing for effective analysis to be performed.

The scope of the literature review is that the final application will be limited to companies floated on publicly available stock markets, whilst attempting to provide understanding into comparative visualizations and if they can be an effective tool. Throughout this section, research within the area of financial visualization will be explored to ensure a synthesis of understanding in the subject area. The section begins with understanding the content of financial visualization surveys, before moving into related papers discovered through scholarly databases.

2.1.1 Survey Literature

Work such as the Survey of Surveys by Liam McNabb and Robert Laramee [ML17] provides a synthesis of work within Information Visualization, included in this is a survey on

Visual Analysis Approaches for Financial Data by Sungahn Ko et al [Ko+16]. The survey by Ko et al categorises 24 different stock visualization papers alongside exploring the evaluation techniques and the interaction methods used in each. The papers focus ranges between financial-based line graphs and candlestick charts [EBM07] which often use data sources or statistical methods combining a wide range of multivariate data [Ko+16].

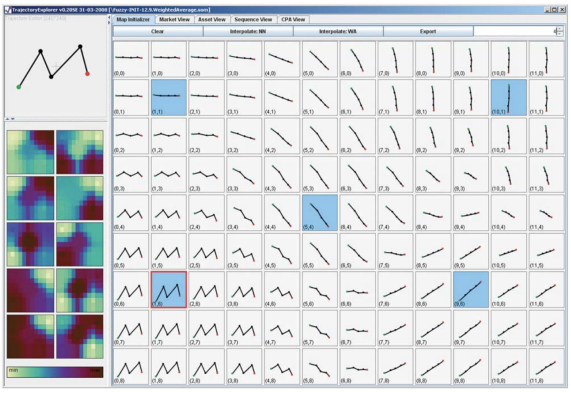


Figure 1: A screenshot of the user interface and visualizations produced through interactive Kohenen maps [Sch+09].

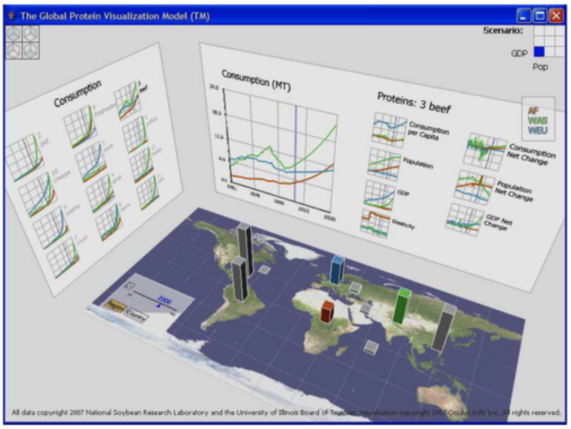


Figure 2: A screenshot of a mixed 2D/3D visualization for financial data, utilizing both columns and line graphs for visualization purposes [Mir+07].

The survey [Ko+16] combines papers from both industrial and research-based backgrounds, with a significant majority of these being within stock and fund trading paradigms. The paper presents sections relating to both methods of analysis and visualization, with section 7.1 presenting visualization methods. Findings suggest a large number of research elements focus on traditional 2D methods such as line graphs, pie charts and similar [Ko+16]. One such example of line charts being used in a novel way is proposed by Tobias Schreck et al [Sch+09], which explores the visualization of clustered trajectory data using interactive Kohenen maps. The paper allows the user to visualize a self-organizing map, whilst enabling control of the algorithm at a functional level of detail [Sch+09]. The produced visualizations of the clustering tasks can then be selected and compared such as seen in figure 1, allowing for a thorough analysis of the algorithms used to be performed using the framework. Researchers such as Dao et al [Dao+08] present methods of using wedges assigned to each stock to understand force indications within the NASDAQ stock market.

Another focus of the survey [Ko+16] is three-dimensional (3D) visualization techniques for financial analysis, the common method of visualizing such data as discovered in the

survey is focused on navigation-based 3D spaces with columns presenting the main view of discoverable data. Analysis for stock markets is explored in the survey through a visualization approach using point-of-view which was proposed by Lisa Strausfeld [Str95], the paper presents opportunities relating to point-of-view stock market analysis and investigates the usage of 3D space. Some of the 3D visualization papers surveyed focus on using surfaces to display line graphs [Gre+99] in the correct format, however the survey finds these often prevent the viewing of other line graphs through occlusion removing usability potential for the user. The line graph approach is then extended to include the presentation of price-time information [NB04], which did present finding supporting its usage possibilities. A key method of preventing the blocking of different line graph views can be seen in figure 2 produced by Barbara Mirel et al [Mir+07], through positioning core columns in the center and line graphs on the wall, the core visualizations are present to the user at all times whilst using the software, with a focus on exploring quick changes within commodity markets. Many of the 3D visualizations presented in the survey suffer from poor usability and have large amount of ambiguity in the method of which data is presented, making them poor at comparison and user-defined functionality.

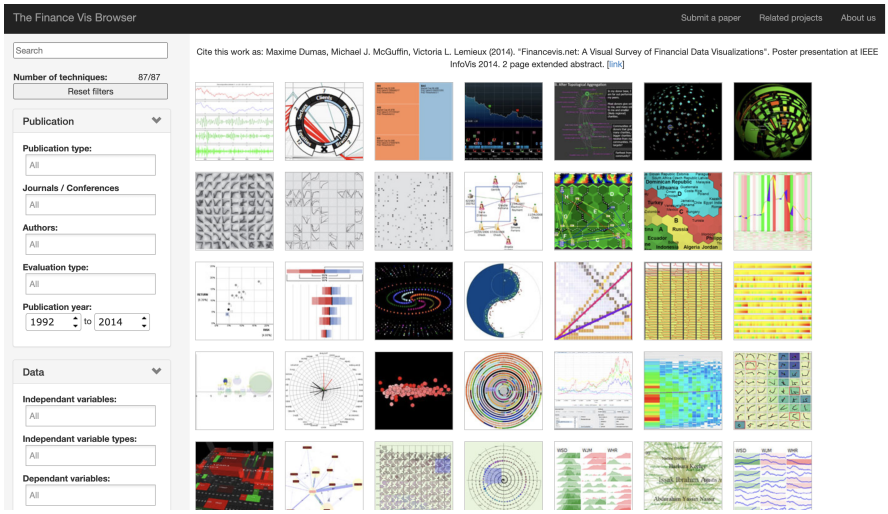


Figure 3: A screenshot of the financevis.net website, displaying query options and resulting research [DML14].

In addition to the survey produced by Ko et al [Ko+16], Maxime Dumas alongside a small team of researchers produced a visual survey of financial visualization and its applications [DML14]. The website for the survey is accessible at financevis.net and was produced alongside an extended abstract explaining the paper collection and categorization process [DML14]. The web application presents 87 visualization methods used within

financial data, allowing the researchers to then query and present results using a number of parameters. Upon changing the analytic task parameter on the web page, 61 papers are identified as matching this. Figure 3 displays the output of the system, which enables the rapid discovery of financial visualization techniques. In addition to the visual mappings, the researcher is able to identify different variable, dimensions and meta-data to assist in the discovery of literature.

A visual analytics system for financial time-series data by Lei and Zhang [LZ10] enables the analysis of stock market data, through producing visual encodings of ring-based visualizations seen in figure 4, the paper is able to provide investors a holistic view on market data in comparison with one another. The ring visualization proposed in the paper uses categorical variables to present arcs displaying the market cap size. Other elements of the ring visualization include color showing market sector, and the radius being used to display the performance of the stock. The paper additionally produces a stock price clustering technique which is produced using k-means clustering [LZ10], using both quantitative and time as variables, enabling a scatter plot of price changes in comparison to the individual value of each stock to be displayed. The produced visualizations as part of the paper were then evaluated through a user study, however this is out of scope for this project.

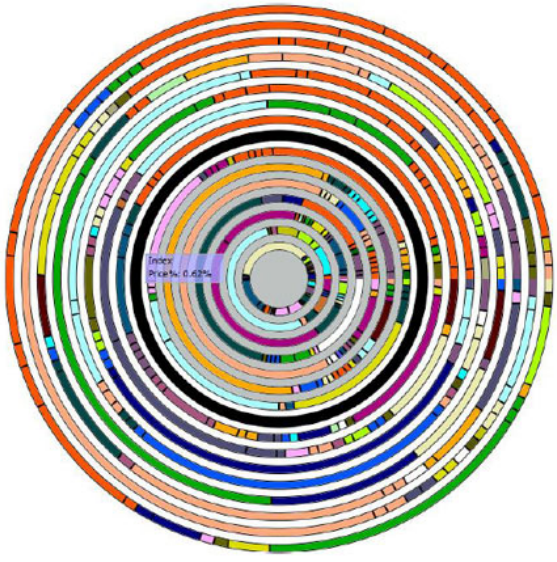


Figure 4: A screenshot of the ring visualization proposed by Lei and Zhang using radius, color and position to assist in stock market understanding [LZ10].

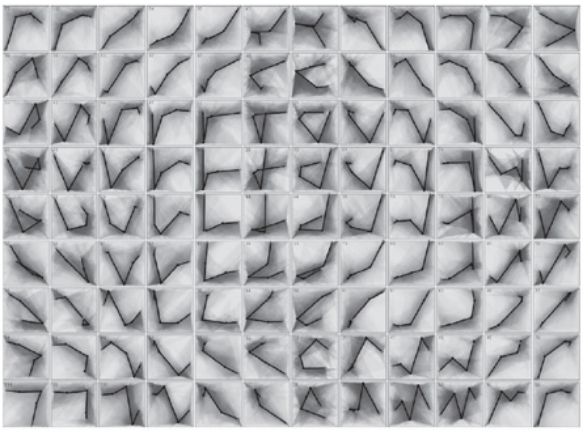


Figure 5: A screenshot of the Market View presented by Tobias Shreck et al using trajectory-based analysis of financial data [Sch+07].

Trajectory-based visual analysis of large financial time series data by Tobias Shreck et

al [Sch+07] is seen in figure 5, the paper aims to use visual analytics to enable automatic data analysis for large financial datasets. The paper proposes three separate visualizations including a market view, asset view and sequence view, seen in figure 5 is the market view, which enables exploratory discovery of market data. For the market view, the visualization uses a function of risk, enabling the analysis of 5,478 trajectories within the market; each of these 'groups' then being displayed in a grid. The asset view provides a similar view of trajectories, however due to the limited nature of these a large selection of spaces within the grid display no visual information. The sequence view also presented in the paper [Sch+07] makes use of a pixel static display, with each pixel representing an asset or stock, in this case as opposed to using trajectory line, the paper makes use of displaying trajectories in a pixel-color use case.

Other visualizations displayed on the browser [DML14] including visualizing the Ethiopian Commodity Market [Rog09], and NASDAQ Velocity and Forces [Dao+08]. Both papers present methods of visualizing stock or commodities, however are limited in the novel nature of the approach, using traditional polar area diagrams and line charts to present analysis, and no interactive tools being accessible at present for these. Both papers also present limited methods of visualizing stock market data, however the Ethiopian Commodity Market paper [Rog09] does make use of stock market symbols, its comparison techniques only enable limited future work as they are similar to existing applications and online tools and the publication date of 2009 does not encourage further work at present.

Both surveys [Ko+16; DML14] present numerous novel attempts at enabling financial visualizations to be more exploratory and analytical in nature. Within this section, each of these surveys have had the most relative and related papers analyzed for compatibility with the project, enabling various findings to be presented as part of this thesis. Following sections will therefore explore other tools, and research discovered through scholarly databases and online web platforms, enabling a greater comparison of potential work to take place.

2.1.2 Financial Visualization Techniques

Visualization techniques for the purpose of analysing financial markets can take multiple formats depending on the level of understanding needed, papers such as

“Visualizing the stock market” [Wat99] propose an improved treemap display for assisting in the understanding of financial markets, however whilst this approach does provide “an adequate overview over the financial market” [ZNK08] as explained by Ziegler et al this approach as it does not assist in the comparison of individual stocks or finances in temporal situations and “therefore cannot replace traditional line charts” [ZNK08]. The proposed treemap method can be further improved using interactivity as a tool for showing smaller stocks, however this once again suffers from a lack of comparison or temporal analysis being involved.

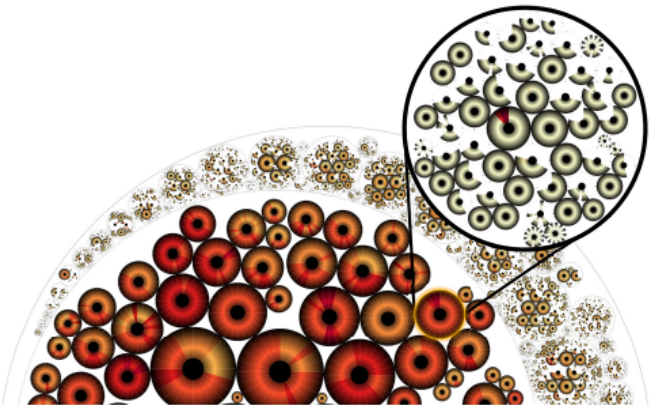


Figure 6: A screenshot of the resulting visualization produced as part of the ClockMap paper [FFM12].

Building upon research by Wattenberg [Wat99], the ClockMap [FFM12] as seen in figure 6 proposes a solution which uses circular treemaps with temporal glyphs for usage with time-series data. The method combines a large number of glyphs whilst using size and colour to group these in a similar method to that of a treemap. Proposals do exist for this method to be used within financial markets [Zha+16] with one implementation comparing multiple temporal glyphs [Fuc+13], however this article removes the treemap setting and uses a multiple coordinated view approach instead. Whilst this method was found to be quite effective for a 24-hour timeframe it does make potential researchers aware of performance changes when changing this period [Fuc+13]. The solution also requires a larger amount of horizontal space in comparison to the ClockMap method due to the coordinated view approach, therefore making it unsuitable for large datasets such as a stock market due to the limited screen size on most monitors.

Candlestick, line and Opening, Highest, Lowest Closing (OHLC) graphs are commonly used for the visualization of individual stocks, highlighted in the thesis by Rui Ma [Ma09].

The thesis explores interactivity for the use of stock market comparison, however, suffers from the same issues as the majority of traditional mapping techniques due to the limited nature of comparison which can be performed. Methods of interaction for these encoding however are more developed than that of those seen in treemaps [Wat99] and similar methods due to the more common nature of presented techniques.

2.1.3 Stock Portfolio Monitoring

FundExplorer [Csa+03] further develops the concept of treemaps [Wat99] for financial analysis. FundExplorer is a visualization based majorly on the treemaps, where the technique is used as a "query device" [Shn94]. The paper for FundExplorer confirms it is mainly used as a method for investigation options, however only visualizes data relevant to the stock portfolio and once again misses the key elements such as temporal information in papers such as ClockMap [FFM12]. However, the paper proposes an interesting concept which concluded the "experiences with the FundExplorer are promising, and we believe that the concept of the Context Treemap can be applied to other domains as well" [Csa+03], Suggesting that the visualization techniques could be implemented in not only other financial systems but further interdisciplinary research too.

Another method of visualizing and discovering stock market portfolio data is through the use of clustering which allowed for a self-organising map to be produced by Joel Joseph and Indratmo [JI13]. This method assisted in exploratory finding for the SandP 100 stock market data, however only focused on discovery of stock market data. Due to the limited nature of the research, the research does not make note of the scalability the visualization is able to perform at. In comparison to the treemaps [Wat99] and ClockMap [FFM12] seen previously, the visualization groups stocks based on financial history as opposed to the active market allowing for a more comparative approach which automates the whole process whilst providing little to no user interaction throughout.

Tim Dwyer proposes a 3D visual model based upon movement in fund management [DE02], which was then expanded with a more scalable method for portfolios the following year [Dwy03] which can be seen in figure 7. The papers propose a "worm metaphor" [DE02] as a visual tool in presenting movement and changes in financial data, along with lines which are then formed to connect them. The author of the paper believes this was

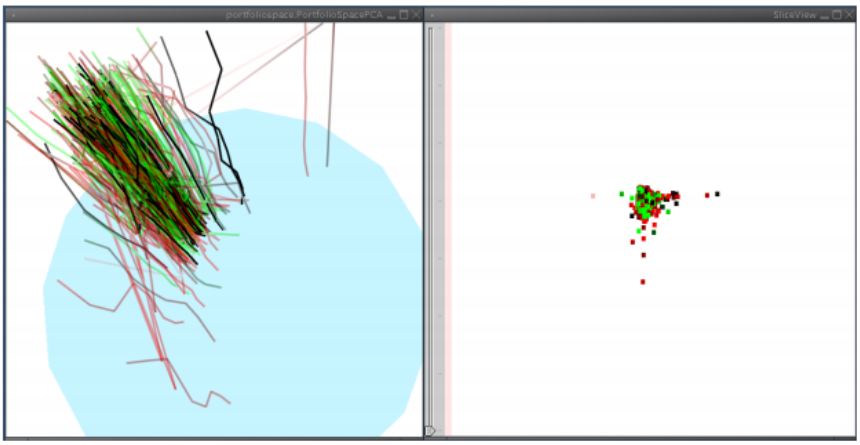


Figure 7: A screenshot showcasing the scalable method for visualizing movement in fund management by Tim Dwyer [Dwy03].

successful, however still had concerns regarding 3D visualization whilst a lack of clarity and textual content in comparison to treemaps [Wat99] and FundExplorer [Csa+03] means the visualization could potential lack clarity to the investor or researcher using the proposed system. Future work within 3D financial visualization could therefore attempt to solve the potential issues relating to the lack of understanding of the visualization, however these papers were published in 2002-2003 and are no longer the most modern implementation.

sPortfolio [Yue+19] is potentially a more recent implementation of visualizing stock portfolios and follows the findings regarding multiple coordinated views [Fuc+13] alongside the stratified analysis performed by Dwyer et al [DE02] and Csallner et al [Csa+03]. The research produced for sPortfolio is a successfull implementation of financial visualization research which is worked directly alongside a corporate entity and therefore it can be assumed the findings could be more “industry ready” and similar to the proposal of the Previous Systems section of the paper. The paper produces an effective visualization tool based upon the stratified analysis, whilst also removing the issues in regards to dimensions from previous research [DE02; Dwy03]. Through a user evaluation study, the conclusion was found that “sPortfolio performs efficiently” [Yue+19] in portfolio exploration and therefore was considered to be further developed as part of this research project.

2.1.4 Exploratory Financial Visualization Techniques

Exploratory is the process of producing visualization to discover new content or insights which would not be discoverable through methods such as traditional graphs. One such

method of this found in use of stock market is a narrative data visualization [Nai16], this method linked events and stock market data to assist in providing news-oriented insights and showing how these affected the market. This method is different to the treemaps explored previously [Wat99; FFM12] whilst also building on the interactivity of the line graphs seen in many interactive applications [Ma09]. The research did find the stocks were sensitive to non-economic traits and that these methods can assist in the general understanding of markets. However, further work is needed to explore larger and potentially live datasets to investigate the findings in real time.

VMap [KY18] is a visual comparison tool built for the web using Flash, it is a simplified tool built for comparison of stock market symbols implementing a grid-based structure. Research on VMap however is limited with little empirical testing having taken place, however the paper includes screenshots and reproducible code. However, the concept of the research allows for a total of 5,126 symbols to be displayed with an interactive search allowing for “Industries (that) are identified with the Sectors” [KY18]. The research aligns closely to that of SandP Discovery research [JI13] however provides a much larger scale of symbols and the relative values. The VMap concept is an illustration of work which could require further research, and as such was considered when producing visualization elements for individual stocks as part of the software.

2.2 Previous Systems

Financial visualizations appear in both a research and application-oriented formats, throughout this section, various software’s relating to the visualization and comparison of stocks will be explored and details of the systems and their architectures will be analysed. Therefore, this section will allow for a better understanding of the existing investments companies have within the subject area and what features are required and used in existing customer-focused applications.

Existing tools were mainly discovered through online searches, recommendation articles and scientific papers. Papers such as TechWare [ZK11] provide an overview tools and their positions within the market, such as FinViz [Fin20] being a Start-Up Visualization tool and Yahoo Finance [Yah20a] being an online financial data portal. Tools from a range of categories were therefore chosen, however the application reviews was limited only to

applications which allowed some form of free public access. Due to this, tools from larger companies such as Refinitiv [Ref20] and Bloomberg (which only provided delayed and closed access) were not part of the review.

2.2.1 Yahoo Finance

Yahoo Finance [Yah20a], as defined within TechWare [ZK11] is an online financial data portal produced by Yahoo, offered for free with a $349.99 annual subscription for in-depth analysis and research [Yah20b; Dav19]. The application is a web-based portal providing access to data from almost all large exchanges, such as those seen within figure 8. The application is web-based and therefore runs on all modern web-browsers, additionally the service used to provide a developer API which was used in large amounts of research such as Real-time Analytics [SCR16] and the Quality of Interactive Data [BN13], however this has since been deprecated.

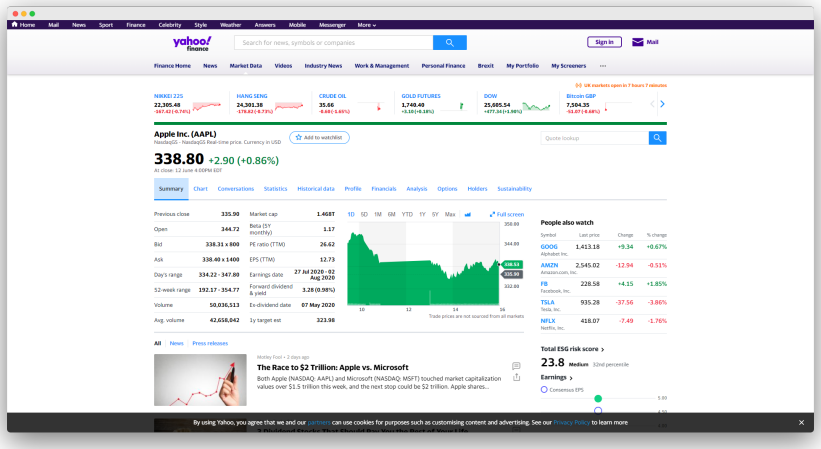


Figure 8: A screenshot of the AAPL stock on the Yahoo finance website [Yah20a], showcasing graphs, news and popular matches.

Yahoo Finance provides free, real time financial data from stocks through to cryptocurrencies. The data for each stock, as seen in figure 8, is mainly text-based with one singular temporal graph in the center, allowing for both in-depth and quick glances to be performed by the user. The application allows users to select data and how to display it, but does not provide any option to compare data away from organized lists and search queries. This search data however is quite in-depth and would allow for further analysis should it be imported into a different application.

The applications target audience is that of financial traders and people interested in financial news, which is aggregated and displayed through Yahoo's search results. The website has an estimated 70 million unique visitors each month [Fuh19], the largest of the financial web-based portals available on the market. The website is largely an analysis and data portal, and only has limited real time data capabilities.

2.2.2 Google Finance

Google Finance [Goo20] is another online financial data portal, produced by Google as part of their search engine offerings. The service is offered completely free, however is supported by relevant advertisements across the Google platform. The service is supported and used by searching a stock or "finance" into a Google search supported client; this is therefore supported by almost all platforms including Android, IoS and the Web. Figure 9 showcases a screenshot of the application, with related datasets and news sources being presented to the user.

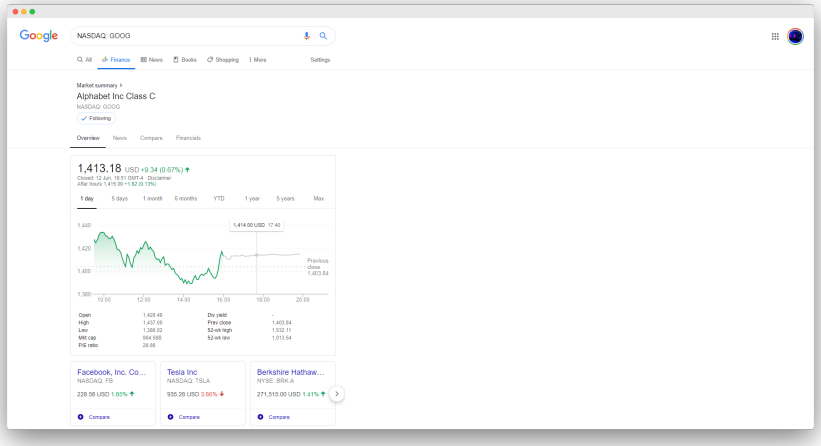


Figure 9: A screenshot of the GOOG stock on the Google Finance search page [Goo20], which showcases temporal graphs and news stories.

Google Finance can be used as a "at a glance" system, meaning a user would be able to quickly type a stock or company they are interested in, and receive instant results relating to this. The service is often used by more novice financial traders, or people who want to be aware of the market due to the lack of in-depth data or analysis that has taken place through the system. Google Finance, in comparison to Yahoo Finance emphasizes stock-charting [Fuh19] and notification capabilities [Goo20], enhancing its focus on data aggregation as opposed to exploring the market as a whole.

Google Finance has significant real time data offerings, whilst also providing historical data up to a maximum of the markets history. In addition to this, beneath the line graph shown the Market cap, previous close and 52 week high/low is provided in a textual format. In addition to this, related news sources and information is shown on the company however limited effort is made to compare symbols or to provide further research.

2.2.3 FinViz

FinViz [Fin20] is a start up financial market visualization and stock screening tool available as a website such as the screenshot in figure 10. The application allows users to visualize and produce charts about various datasets from different markets, stocks and commodities. The website offers both a free and premium feature set costing $24.96 each month. Premium features of the website include more advanced and interactive charts alongside real time data and correlation mapping.

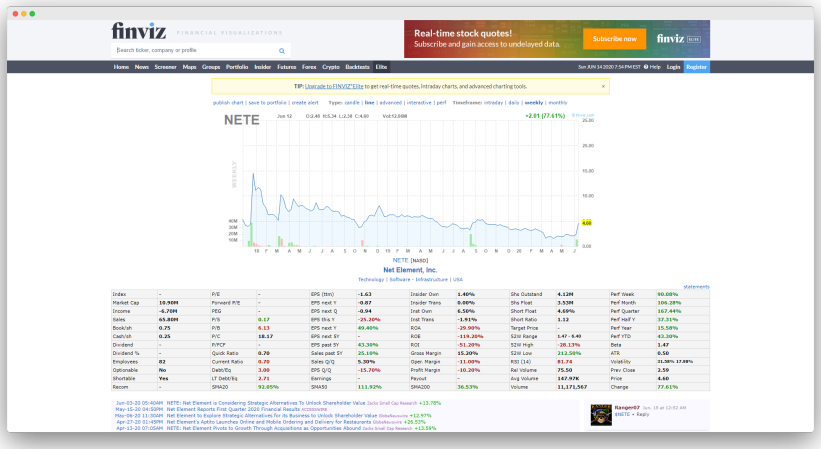


Figure 10: A screenshot of the NETE stock on FinViz [Fin20], which provides analysts with access to large datasets and complex graphs.

The application markets its users as being from companies such as HSCB and CreditSuisse and therefore it can be assumed it targets the professional and experience traders with its premium subscription features. The platform allows for significant amounts of data to be explored whilst also producing basic interactive visualizations, such as: line chart, OHLC and Treemaps. The aim of the project is to allow better visualizations to be produced quicker using the financial datasets available on the platform.

FinViz provides the most complex visualizations out of the applications explored as part

of the software review. The visualizations often combine multiple elements, whilst also allowing for changes such as 'advanced' graphs with notes and static visualizations to be produced. However, apart from the previously mentioned treemaps showing the sectors of the market, there is little comparison between the different stocks and portfolios a user is able to make.

2.2.4 Seeking Alpha

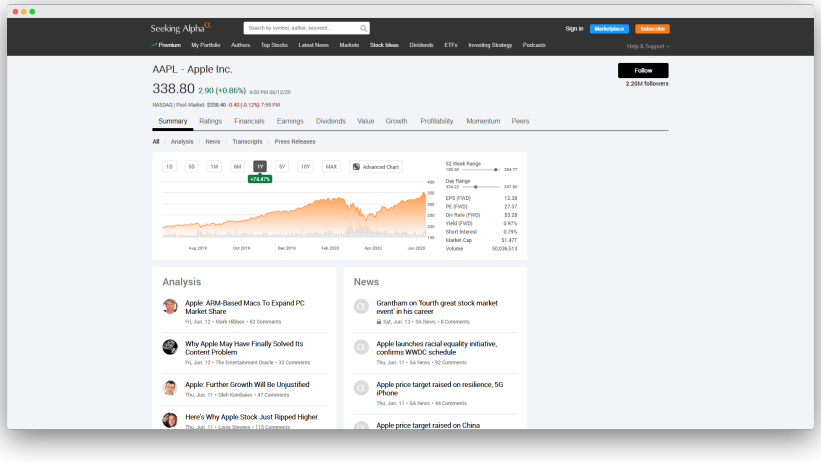


Figure 11: A screenshot of the AAPL stock on Seeking Alpha [Alp20], a tool which allows market traders to analyze and view large quantities of data.

Seeking Alpha [Alp20] is a stock market insights application, produced for the web. The main purposes of the website is to provide insights and knowledge through the financial data available to the company. The software has a free version with limited data, alongside a $19.99 per month subscription to premium data and services. The software provides large datasets to the users, whilst also allowing them to import stock portfolios and providing news and suggestions through aggregated articles on the website.

Seeking Alpha offer two payment modules, the premium features (as mentioned previously) and a Pro model (at $199.99 per month). The website therefore aims to market itself towards both a newcomer to the industry and a financial professional. Due to this the website provides a 14 day trial period of both account types. Content on the website promotes itself towards this target audience through in-depth analysis articles and more basic investment strategies for beginners, thus making it a platform for almost any type of investor.

Seeking Alpha does advertise its ability to compare data as part of its premium service, this is simply a datatable with customizable columns and does not implement visualizations at all. The website does provide data on individual stocks such as those seen in figure 11, however limits this to atomic financial data to the company selected such as current stock value, revenues and dividend growth history.

2.2.5 Morningstar

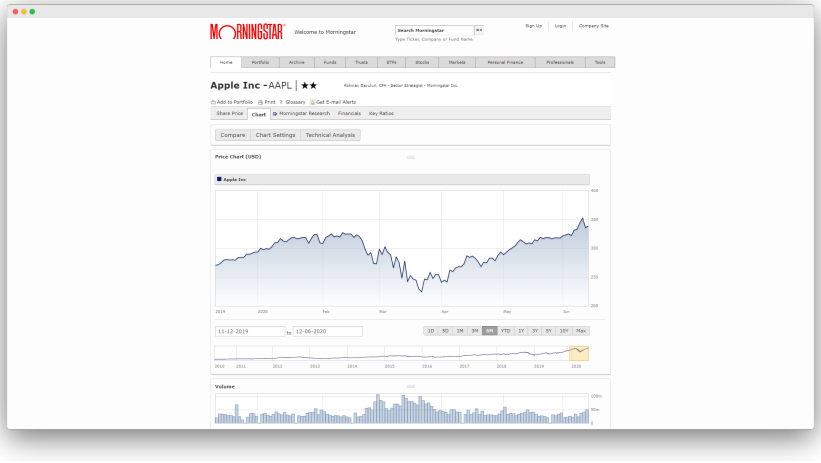


Figure 12: A screenshot of the AAPL stock as seen on the Morningstar Financial Research website [Mor20], a visualization and analysis dashboard mainly based around first-hand stock research.

Morningstar [Mor20] operates significantly as a stock market research company, providing reports to investors and financial trading companies. Morningstar offers both a free tier and premium subscription service costing 19 per month. Morningstar is mainly aimed towards providing investors with news and analysis reports, therefore these can be considered the target market. The application, as with all others discovered is web-based and therefore can be accessed through any up-to-date browser.

As seen in figure 12, Morningstar also provides access to data relating to each stock, mainly for the purposes of promoting the research charged for as part of the premium license. The website, much like most of the software reviewed does not provide a method of comparing multiple stocks and their growth. As Morningstar's main purpose is to provide research and analytics, these sections of content are pushed to the user throughout the webpages.

Morningstar uses an analyst's estimate to produce and communicate datasets relating

to markets, stocks and commodities trading. Due to this, a significant number of the elements and downloadable content on the website comes in the formats of reports, the website also contains a map-based visualization which can be used to compare the analysts estimates of the markets from each country.

2.3 Potential Contribution

Through the literature review and previous systems sections, various methods of visualizing the financial markets have been discovered. Whilst the research performed explored the basics such as treemaps [Wat99] and interactive traditional graphs [Ma09] it was found that these methods often lacked immediate clarity or were not able to cover the entire spectrum of data (such as temporal, value and large-scale comparison). Whilst modern web applications such as Google Finance [Goo20], Seeking Alpha [Alp20] or Yahoo Finance [Yah20b] are able to effectively convey large amounts of information, they are not capable and producing comparative stock market graphs and usually do not feature any method of being able to produce and run queries on datasets. Therefore the focus on research throughout the project was to follow and build upon existing visualization approaches and develop them to include queries and assist in the visual understanding on the large and readily available data.

Papers such as vMap [KY18], ClockMap [FFM12] and sPortfolio [Yue+19] provided a purpose for the research area, assisting in understanding the potential of a financial visualization system which can be further developed to aid the discovery of unknown or undervalued stocks. Elements such as the size of results in vMap [KY18], glyph display capabilities in ClockMap [FFM12] and the completeness of data presented using sPortfolio [Yue+19] are able to aid understanding and promote the potential of visual analysis systems for large scale queries and stock market research.

2.4 Data Characteristics

The proposed financial visualization platform will be capable of comparing and querying data that is both up-to-date and historical in nature, to allow investors to investigate and explore potential portfolios or investment options. Therefore a potential real-time API was needed to allow the querying of such financial data, in both present and historical formats.

Options relating to only downloading historical data were therefore not viable as real time pricing and market availability was needed to match the capabilities of applications such as Seeking Alpha [Alp20] and Google Finance [Alp20].

2.4.1 Data Sources and Description

	Details		Time Series				Forex/Currency			Company Info			Misc
	Access	Cost	Real Time	Intraday	Day	Historical	Real time	Historical	Cryptocurrency	Profiles	Search	News	
IEX Cloud	REST	Freemium	X	X	X	X	X	X	X	X	X	X	
Alpha Vantage	REST	Freemium		X	X	X	X	X	X				Sector' performance
World Trading Data	REST	Freemium	X	X	X	X	X	X			X		
Morning Star	REST	Freemium			X	X				X			In-depth balance sheets etc
Finnhub	REST	Freemium	X				X		X	X		X	Built in analysis
StockFluence	REST	Freemium				X							Sentiment analysis
polygon io	RESTful, WebSockets	Premium	X	X	X	X	X	X	X				
Intrinio	REST	Premium		X	X	X				X		X	Large datasets, unknown accuracy

Figure 13: A feature comparison table was made comparing the differences and usage requirements of each financial API discovered.

A comparison of the various financial APIs was performed, which can be seen in figure 13. The comparison table allowed for a complete synthesis of potential options to be found whilst also allowing for it to be matched against the specifications of the application. Tools such as the IEX Cloud [Clo20], Alpha Vantage and World Trading Data performed well when matched against options and also provided free accounts. Upon further research of these results, The IEX Cloud offered significantly more (50,000) free results than that of any competitor such as Alpha Vantage's 15,000 requests and World Trading Data's 7,500 daily API requests. Therefore combining this accessibility of data, and the fact it provides profiles, historical and real time data the IEX Cloud was chosen as the API provider for the project.

The IEX Cloud is produced and offered by the IEX to provide financial data as a service, the dataset covers all elements of the financial markets including profiles alongside real time and historical stock pricing results. The dataset was produced by the IEX to allow financial products and portfolio tools to be producing using their API and market services. The resolution of the dataset matches that of the IEX's current offerings within the market space, of which a complete list can be found on the eligible symbols section of the main website [Clo20]. Characteristics of the dataset vary depending on what is being queried, however time-dependent abstract data is returned in the majority of cases, with the corresponding values relating to a stocks symbol.

In some cases data as part of the IEX Cloud [Clo20] is prevented from being used due to the limitations of a free account, should a developed want to integrate queries relating to income details or other advanced meta-data a full paid account would be required for use.

FinVis therefore is not able to integrate advanced meta-data into search queries, but should a full account be purchased, this data is available.

2.4.2 Accessing Data

The service is provided through an online cloud platform, which shows usage statistics and provides access to change and modify application permissions. To query the data, the company uses a REST-API which means a developer can use a URL followed by specific parameters to return results. Due to this, a returned object does not have a specific size, but can be presumed to be very lightweight due to each result needing a different query alongside being returned in a JSON format.

Figure 14: An example of the JSON object which would be returned as part of the GET request.

The data can be returned using a GET request, such as: `GET /stock/symbol/book`. The request would then return data in a JSON format such as that seen within figure 14. More samples of the potential responses and the data which is returned in each can be seen within the IEX Cloud documentation [Clo20]. The IEX Cloud documentation provides a complete overview of all queries, parameters and returning data and can be referenced when integrating the platform into a software development project.

Data can be returned in one of two formats from the IEX Cloud, one of which is as a JSON object as seen in figure 14, the second is as a JSON array. Within a JSON array, multiple objects are used and stored within one response, this is common for largely temporal or historical queries, such as stock prices over a time period or intraday pricing. The IEX Cloud also enables users to export data as CSV files, which is potentially useful for long-term storage or creating a secondary data cache system for an application.

3 Project Specification

3.1 Feature Specification

Following from recommendations made within Bob's Project Guidelines [Lar11], a feature specification was generated to better understand the required features and potential enhancements that could be made to the system. The system can therefore be broken down into a list of feature requirements, which can assist in guiding the project to completion.

3.1.1 Basic Features

The basic features are those which must be implemented to give core functionality to the system, they are a requirement for completing the project successfully. The requirements were gathered via the literature review which took place, whilst also involving the analysis of previously completed applications. The basic features cover the scope of the initial product and the planned application which was to be built.

1. **Importing Data.** The system should be able to import (load) a large number of financial stocks and should be able to interact with such data through JSON.
2. **Web-based UI.** The system should be delivered through a web-based user interface.
3. **Stock Selection.** The system should allow individual stocks to be selected, through a search query on the webpage.
4. **Stock Collection.** The system should be able to have multiple stocks be selected, allowing for them to be stored within 'groups' or portfolios.
5. **Line Graph.** The system should have a line graph as a core element on the page, showcasing multiple stock lines.
6. **Stock Queries.** The system should allow meta-data such as market cap beta to be searched, enabling these to then be displayed on the line graph.
7. **Stock Query Groups.** Queries should be developed allowing them to be combined for the purposes of enhancing the potential analytics produced in the system.

8. **Temporal Analysis.** The user should be able to select a time span for which the data is displayed.
9. **Data Selection and Details.** Upon hovering a point in the line graph, further data should be displayed to the user.
10. **Color Map Selections.** The application should allow users to select from pre-defined color maps, depending on personal preference.

3.1.2 Enhancements

Enhancements are elements of the work which extend and build upon the existing features mentioned previously. Enhancements are defined by adding extra features to that of the visualization or the overall application, thus increasing the potential for effective results to generated. Enhancements are specifically mentioned in the papers produced by Dr Robert Laramee [Lar11], similar to that of the core requirements. In some cases, enhancements may be updated to existing work whilst in other this could be a complete new feature being implemented.

1. **Preset Options.** A selection of presets should be made available to the user, to allow for rapid examples to be displayed.
2. **User Saved Stock Groups.** The user should be able to save custom stock groups, through queries or custom searches.
3. **Merge Stock Groups.** The system should be able to merge queries and/or stocks with those from a different group.
4. **Line Graph Interaction.** The system should be able to pan, zoom and be selected by the mouse.
5. **Custom Color Maps.** Color maps for individual companies should allow custom overrides to be enabled through the UI.
6. **Custom Query Manipulation.** Saved queries should allow modification and deletion after being created.
7. **On Click Comparisons.** Upon a data point being selected, a comparison of historical to present pricing should be displayed.

8. **Portfolios Save Queries.** Alongside the previously mentioned stocks, queries should be saved as part of portfolios, allowing the user to select a choice through the system.

9. **Last Portfolio Date Displayed.** A glyph should be displayed on the line graph of the date the portfolio was last saved/created.

10. **Points as Volume.** Individual data points should be able to be selected to display the volume of the present time scale.

11. **Point Glyphs as Company Logos.** The logos of the stock companies should be displayed as points on the graph.

12. **Complete User Options.** Each core element and enhancement should be assigned relevant user options on the User Interface. Note: In some cases, this requirement is linked with previous sections (such as enabling logo glyphs through user options), however this assumes more options are necessary for line graph or other display settings.

3.2 Technology Choices

Technology choices refers to the software, tools and methods used to produce the final product. Significant thought should be placed on which technology is chosen, such as whether it is open source, standard of its community and the accessibility of the product. Many technology choices can vary depending on developer choice [PAN18], system requirements or subject knowledge [PAN18]. Open source refers to projects of which source code is public [VV06], additionally in some cases this software is licensed as re-usable such as public domain or works licensed under MIT [MUS12; VV06].

3.2.1 Programming Languages

The programming language is a significant choice to make when producing visualizations, each deliverable must be considered against the positives and negatives of the selected programming languages. Bob's project guidelines [Lar11] suggests using tools such as the Java Swing Library, which relies on using Java to integrate and produce visualizations, however as the project is based towards a more application-oriented approach this is not as suitable within this use case.

Other implementation options for programming languages would be Python and R, both heavily statistical in nature [Ozg+17] whilst focusing less on interactivity. Python has interface libraries such as Qt, however this must be installed as an executable on the end users desktop and therefore is not practical for a financial analysis software. The main focus of R to to process and analyze data, however this is often static despite libraries such as Plotly existing. Therefore to assist in determining which software to implement, previous systems were investigated [Yah20b; Goo20; Fin20] all of which were web-based and implemented mainly in JavaScript and other web-based technologies which was therefore chosen for this project.

Hypertext Markup Language and Cascading Stylesheets will provide structure and style the website, whilst JavaScript will be used to interact with the elements and server-side systems. For data storage the project will make REST-API calls, and will store these within a SQL database through PHP scripts as these can be easily queried through AJAX based requests. This was chosen above JSON-based tools such as MongoDB due to the simplicity of implementation and availability of locally-hosted server software such as PHPMyAdmin.

3.2.2 Libraries

3.2.2.1 User Interface

CSS Framework Alternatives [ASP] explores Graphical User Interface (GUI) libraries and CSS Frameworks with relevant project examples. The article defines Bootstrap, Materialize and Foundation as the largest modern web development frameworks [ASP], however provides lightweight alternatives in the form of Milligram [SP18b], Skeleton [SP18a] and Ulkit [SP18c]. Figures 15 and 16 provide examples of the Materialize and Skeleton project example pages, showcasing the different stylistic guidelines between a GUI and boilerplate CSS framework.

Due to the nature of the application, and the time limit assigned the project will use a complete GUI library meaning the three main choices would be Bootstrap, Materialize or Foundation. Materialize integrates Google's Material Design Language, and the majority of elements have been developed from these guidelines [PS16]. Bootstrap is produced by Twitter and provides a large amount of initial elements for developers to work from. Foundation is a mobile-first platform, meaning that it scales from mobile to desktop.

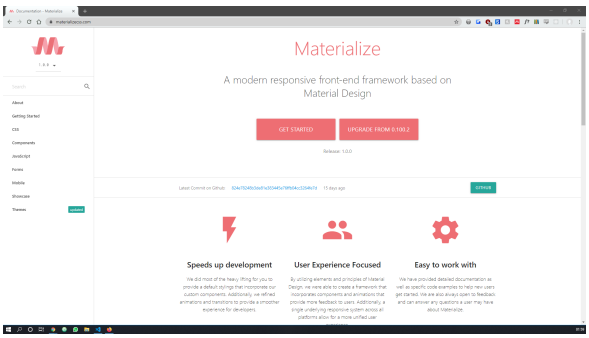


Figure 15: A screenshot of the homepage of Materialize, showcasing the elements included in the framework. Taken from: https://materializecss.com/

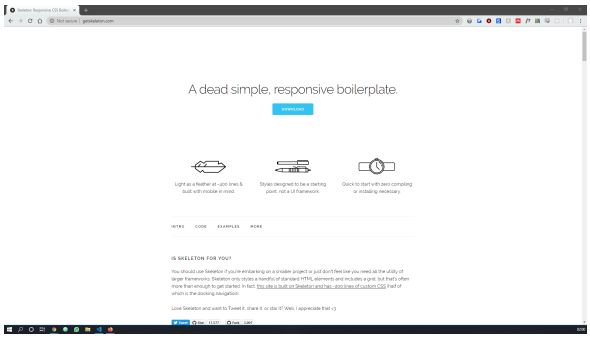


Figure 16: A screenshot of the homepage of the Skeleton website, showcasing the elements of a boilerplate. Taken from: http://getskeleton.com/

All three projects are open source, and are freely available to download from Github. Each library provides a responsive grid, meaning websites can be scaled between desktop, tablet and mobile devices. Bootstrap has the largest community out of the software packages, however has recently transitioned to the fourth iteration of its development versions meaning a large amount of documentation is no longer accurate. For the purposes of the project, Materialize 1.0 will be selected due to the inclusion of sortable data tables and interactive elements being packaged within the project.

3.2.2.2 Visualization

There is a number of visualization libraries produced using JavaScript, the most popular of which is D3.js. Data Driven Documents (D3) uses a custom syntax to produce complex charts built using JavaScript. The library offers functionality for the production of custom visualizations but does come with a significant overhead. D3 has a significant learning curve, but does support completely customizable visualization tools.

Chart.js and Google Charts are more lightweight and simplified libaries, both providing developers with tools that allow for the rapid production of graphs. Chart.js in comparison to D3.js and Google Charts supports the use of Canvas elements, which can significantly reduce the number of elements being loaded into the document therefore decreasing load times. Due to the nature of this project, Chart.js was used as it provided customizable functions, canvas support and has a large open source community supporting it.

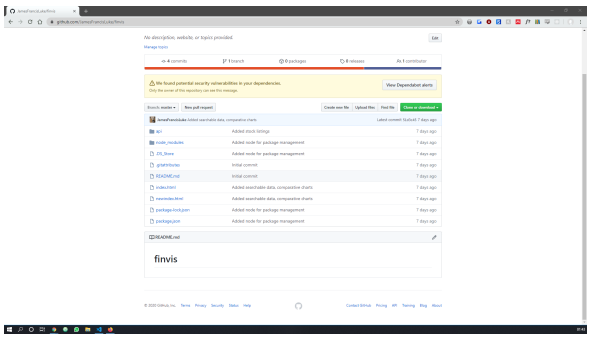


Figure 17: A screenshot of the Github repository for the project.

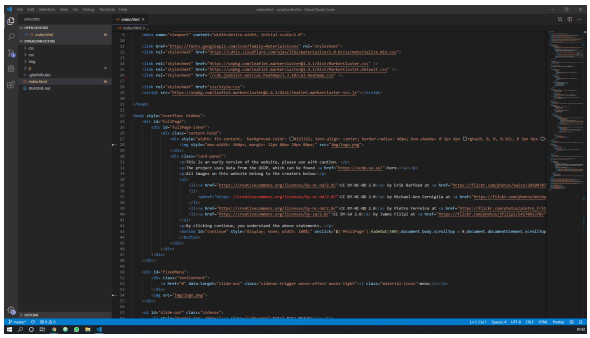


Figure 18: A screenshot of Visual Studio Code alongside plugins.

3.2.3 Other Software

3.2.3.1 Version Control

Version control applications allow for code to be placed within an accessible repository, Git is the most popular example of this. Git is a command line executable for managing versions, code and errors throughout the lifespan of a project. Various applications exist for interacting and storing code in a Git format, the most popular of which is Github as seen in figure 17. Github enables the online storage of programming resources, such as files and package management data alongside allowing historical versions to be saved and stored permanently and changes being labelled.

Throughout the project, the Github student license was used to produce, create and store data within a private repository. Backup's could therefore occur when needed, and the project could be effectively managed for dependency errors and security flaws through recent integration of package.json scanning.

3.2.3.2 Package Management

Package management has traditionally been completed by installing packages through content delivery networks (CDNs) or downloading them locally. This method has significant issues in keeping up to date with security flaws and overall project management. Node Package Manager (NPM) 6.13 will therefore be used to install, update and monitor packages used with the project. NPM has a significant number of packages available, all of which can be installed through the command line interface provided. As an example, should a user want to install Materialize they could just type: `npm i materialize` and the package would be installed to the workspace.

3.2.3.3 Programming Environments

The list of web-based IDE's is extensive, and significantly based upon user opinion. Tools such as VIM, Notepad++ and Brackets offer simplicity and accessibility throughout, however lack plugins built-in features available with improved applications. Environments such as Atom, Visual Studio Code and Sublime Text all have large communities and community-driven content and therefore it is the option of the individual developer as to what they want to use.

For the FinVis project, Visual Studio Code was used as seen in figure 18. The platform has built in functionality for supporting the majority of web-based languages, with downloadable content being supported for extensions. In addition to this, the platform has built in Git and console support, meaning it can be used in conjunction with the previously mentioned project and package management tools.

4 Project Plan and Timetable

As part of the project, a plan, software development life cycle and tracking mechanism was produced to ensure that each element of the work was successfully completed. Meeting notes are first presented, providing details of the format and examples of what these look like. This is followed by the project planning chart, which follows Bob's Project Guidelines [Lar11], however have been modified to better suit the nature of the project.

4.1 Meeting Notes

A comprehensive set of meeting notes were completed as part of the project, these were noted by the project supervisor, before being re-written in a digital format to be placed within the portfolio document. The minutes of Meeting Protocol [Lar10b] presents the format of the meeting notes, alongside advising of the importance of correct notes for future reference and work.

4.2 Project Plan

As defined within Bob's Project Guidelines [Lar11], the project was planned to ensure all core elements could be added within the defined time frame. The project took place over 3 months, and therefore each task was provided with a corresponding completion date. Seen in table 1, is a complete list of expected completion dates (each aligned to a meeting for the project) alongside the actual date of completion. The table presents details relating to both enhancements and core features of the application.

Task	Expected Completion	Actual Completion	Additional Notes
1. Importing Data	June 4th	June 4th	Data was imported through the IEX Cloud
2. Web-based UI	June 11th	June 11th	The web-based UI was implemented using Materialize
3. Stock Selection	July 2nd	July 2nd	
4. Stock Collection	July 2nd	July 2nd	
5. Line Graph	June 25th	June 25th	Line graph initial implementation through Chart.js
6. Stock Queries	July 9th	July 9th	
7. Stock Query Groups	July 16th	July 16th	
8. Temporal Analysis	July 23rd	July 23rd	
9. Data Selection	July 30th	July 30th	
10. Color Map Selections	August 6th	August 6th	Color maps generated through Colorgorical.
1. Preset Options	August 6th	September 3rd	Presets were added at the end to ensure all functions used.
2. User Saved Stock Groups	August 13th	August 13th	
3. Merge Stock Groups	August 27th	August 27th	
4. Line Graph Interaction	July 30th	July 30th	Pan and zoom.
5. Custom Color Maps	August 6th	August 6th	Added at the same time.
6. Custom Query Manipulation	August 13th	August 27th	
7. On Click Comparisons	August 13th	August 13th	
8. Portfolios Save Queries	August 13th	August 20th	
9. Last Portfolio Date	August 27th	August 27th	
10. Points as Volume	August 27th	August 27th	
11. Point Glyphs as Logos	August 27th	August 27th	
12. Complete User Options	August 6th	August 13th	

Table 1: The expected completion dates for the core functionality of the application, followed by the time frames proposed for the possible enhancements.

4.3 Software Development Life Cycle

The project progressed similar to that of a software development application, meaning that throughout the project a Software Development Life Cycle (SDLC) needed to be chosen, to ensure that development was efficient and effective. Software Development Lifecycle Models by Nayan Ruparelia [Rup10] explores the potential methodologies which could be followed throughout a project. The paper [Rup10] presents methods such as Agile, Waterfall and Lean Development. For the focus of the project, an Agile development methodology was followed, which enabled a rapid prototyping stage to then gain feedback from the project supervisor. The iterative nature of Agile development meant meetings with the project tutor could be used to then revise or change existing features from the set of core requirements in the application. Developing the software through this methodology allowed for features to be partially implemented before receiving feedback for future development and changes that needed to be tested or made.

Agile is often positioned within software development as a team-focused task [CH01], however this significantly limits the potential capabilities when using it for a independent research project, as advisors and other students are also present to provide feedback and comments of functionality and work. Agile is heavily showcased within the book by James Shore [Sho+07], and was used for the project management as it allowed for rapid prototyping to implemented to then encourage feedback. Agile is seen as useful for projects where scope changes [Rup10], in the case of FinVis, features and functionality were constantly changing in requirements and therefore using the methodology enabled these to assist in the future development of the project as opposed to slowing the process.

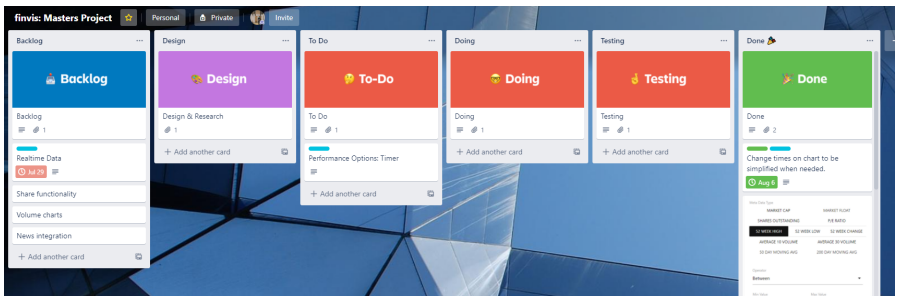


Figure 19: A screenshot displaying the project as a set of cards on a Trello board [Atl].

In addition to the advantages mentioned previously, a Trello board was implemented to monitor and measure how far each feature was to completion. Trello is a project

management tool which assists in the organization of large applications and other team tasks. Figure 19 displays the project as a set of cards on the Trello website [Atl]. Trello worked well as part of the Agile methodology as the progress of each individual component or feature can be monitored through this, it also enabled quicker responses when screenshots of specific features or functionality of the application is requested as this can be connected as a file through the UI.

5 Design

Before building the visualization platform, designs in regards to the system and its capabilities were produced. Due to the iterative nature of the Agile methodology, designs were re-designed and changed throughout the project, enabling changes to be made to both design and implementation depending on the functionality required. The designed system is capable of implementing the features mentioned within the project specification and therefore enabled the enhancement features to also be included.

The section initially introduces the visualization pipeline, before focusing on the individual designs produced as part of the project. Designs for the project include the API and the web-based platform with the external API being referenced as an external source of data (where necessary). The diagrams produced build upon those suggested as part of Bob's Project Guidelines [Lar11] and are presented alongside a short explanation of how the system or function works.

5.1 Visualization Pipeline

The visualization pipeline can be seen in figure 20 [End+17], it describes the process of creating visual representations of data and begins with the source data to the final views. The visualization pipeline can be used to based custom pipelines relating to visualizations of domain-specific data. In the case of FinVis, the visualization pipeline displayed by Endert et al [End+17] matches quite well and further diagrams and understanding can be gained from it.

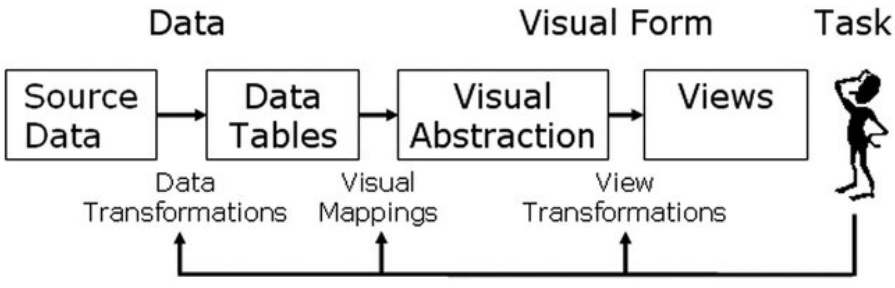


Figure 20: The visualization pipeline describing the process of creating visual representations of data as displayed by Endert et al [End+17].

5.2 Process Diagram

Bob's Project Guidelines [Lar11] suggests using a process diagram within the design stage of the visualization. The diagram in figure 21 presents a schema of a basic overview of the visualization process, which allows a custom diagram to be produced based upon this understanding presented by Ware et al [War19]. The schema presented in figure 21 mainly displays how a human is able to interact with the system through data exploration, and through the manipulation of data with this being linked to a data source and preprocessing of the data and transforming to match the desired format.

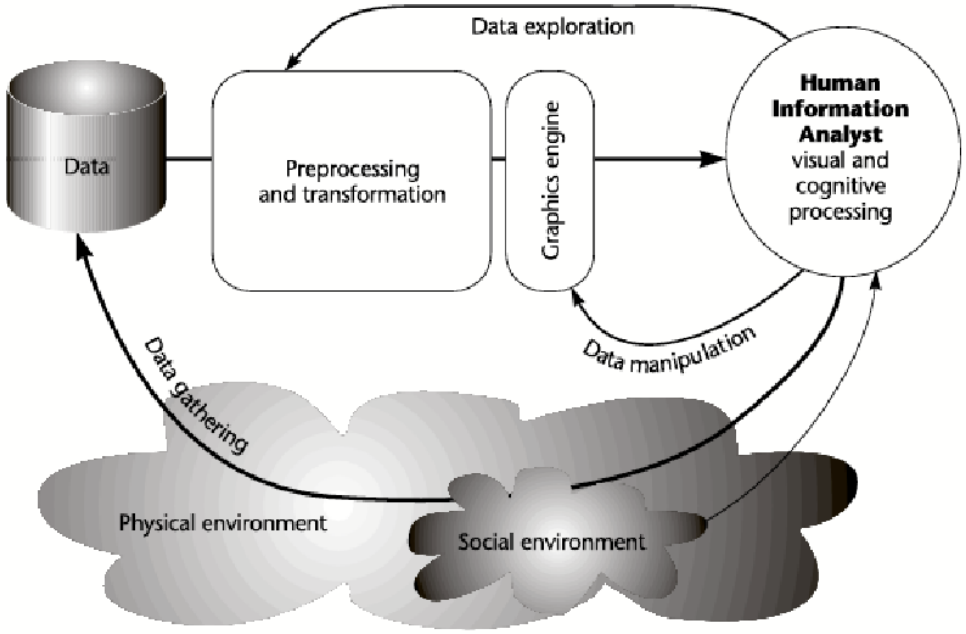


Figure 21: A schema describing a basic overview of the visualization process presented by Ware et al [War19].

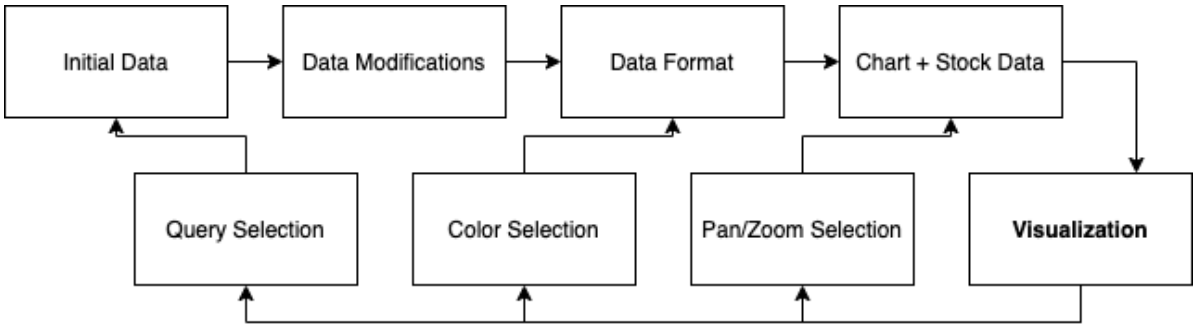


Figure 22: A process diagram produced for the software, displaying how the iterative nature of interaction can allow multiple visualization generations.

Displayed in figure 22 is the process diagram for the application, it begins with the initial data, before allowing data modifications, formatting and chart and stock data to be

displayed. The process only generates an output at the end which is when it reaches the visualization process before then allowing user modifications and data filtering to take place. Initial data refers to the data gathered and returned from the IEX Cloud API, whilst data modifications includes the changes made to labels to format them as dates and changes made to the order of symbols should none be returned. The data format section re-writes the data as a new object, allowing it to then be parsed within the lineChart namespace. The chart and stock data is then the final lineChart ready to be displayed, combined with the stock data being generated within a categorical line chart.

Upon the visualization being displayed to the user, they are able to make modifications relating initially to pan/zoom selection which simply updates the x,y coordinates of the chart. The user is also able to modify user options relating to data such as the color selection, which simply changes the format of the data to match the new color scheme. Finally, the user is able to write and edit queries relating to the dataset, which changes the initial data being returned to the application. Therefore the process diagram, presents the entire process a potential investor would have whilst interacting with the system, there is obviously a much larger set of subsystem (or in the case of JavaScript namespaces) and these will be displayed in 'System Diagrams' below.

5.3 System Diagrams

Figure 23 presents the system diagram produced as part of the design of the application, the diagram showcases the various subsystems in use and how they interact with one another. It should be notes that smaller and less important systems are not displayed on the diagram (such as user options etc). The software subsystem diagram enables a schematic display of the overall architecture of the system to be presented, enabling potential developers to understand how the different components of the application work together. Within each subsystem, key attributes and functionality is also shown, which assists in the understanding of the different elements and the planned location within the code.

The systems displayed in figure 23 begins with the user interface, which links to the query manager, portfolio manager and the line chart itself. The user interface is made up of components allowing interaction with the software from the end user. The portfolio manager

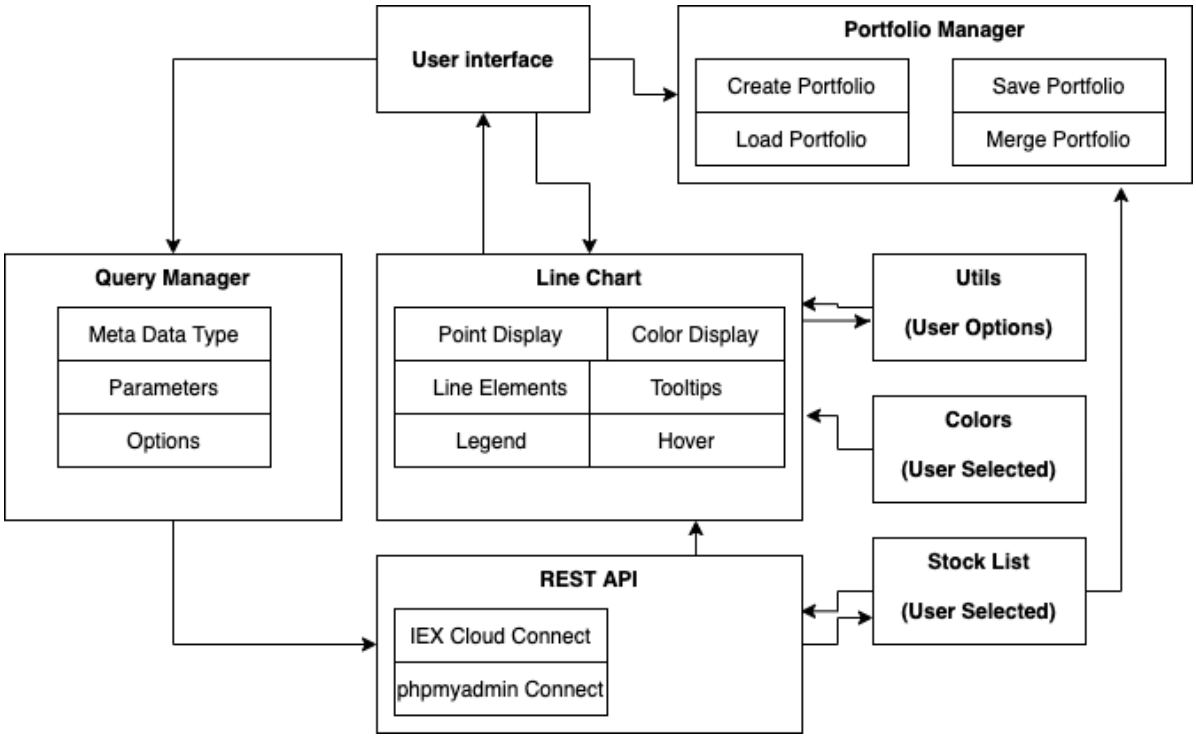


Figure 23: A schematic display of the software systems displaying the key subsystems and how they interact with each other.

enables the user to create, load, save and merge stock portfolios within the system, the user is able to activate the portfolio manager from the user interface and it does not have any interactions other than accessing the current stock list (before being saved to local storage). The query manager enables the user to create queries based around the different types of meta data available within the system, it first combines the various parameters the user has selected before sending this as a request to the REST API system. The line chart system enables the display and interaction of the line chart by the user, which is why it has a two-way arrow. The line chart system interacts with almost all the user options and receives data from the rest API.

The REST API system enables interaction between the users computer and the databases which are storing the financial data, which includes the IEX cloud and the phpmyadmin SQL databases. The REST API receives this data before then sending it to the users line chart, and storing a list of resulting symbols within the stock list. The stock list enables a set of stock symbols to be stored without having to query data again or store it in a format related to the line chart. The Line Chart system also interacts with the user options, which in the diagram shown are the utils and colors systems (however, depending on enhancements required this can be extended). The systems interacting with the line chart

enable various user options to be changed and then displayed to the user.

The diagram in figure 23 presents the novel implementation design of the FinVis system, displaying how the user interacts with each system through the user interface and how each system is designed to interact with one another. The design produced may be further improved and expanded through adding other user options or other modifications, such as the enhancement 'points as volume' which would require extra point display modifications to take place before being returned to the user. Other modifications such as hover events and functionality would occur within the user interface itself, and therefore have also been left off the diagram.

5.4 Data Interaction Diagram

Figure 24 presents the data interaction diagram used to plan how data is queried and returned from the data source (IEX Cloud) and then modified through a REST API before being returned to the user. The diagram presented showcases this data from the user interface to the REST API before displaying the two separate storage systems used to keep stock and meta data. The IEX Cloud API will be used to return stock prices and historical prices relating to the core data of the system, whilst the SQL database will be used to store meta data relating to the user customized parameters available within the application.

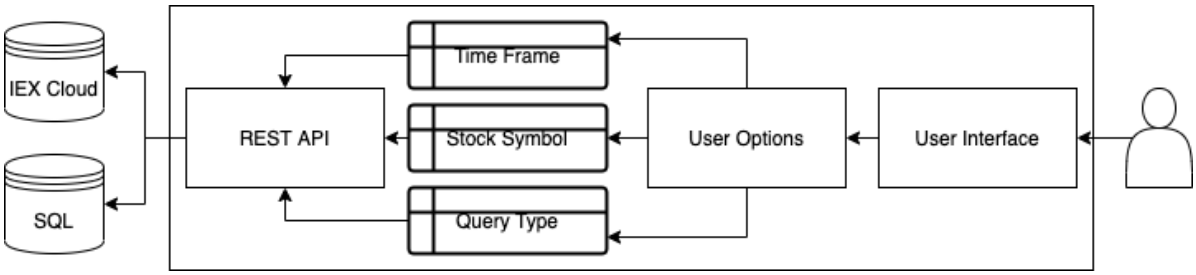


Figure 24: A schematic flow chart displaying how the users interactions change the data which is being queried and returned from the database.

The user interface is first to be interacted with by the end user, in which they have multiple options relating to user options: time frame, stock symbols and query types (other user options only change visual elements and therefore do not require a connection to the server to be made). The time frame data flow describes the time selected for the data being returned in a string format such as "1d" for one day or "6m" for six months. The stock symbol is the stock type required for the server to search for, in this case it will always be a

string and would be formatted in a similar way to ”APPL”. Finally the query type is an SQL statement which is sent to the server, which can be defined through its requirements of having to be sent to the SQL database instead.

The REST API then decides where each query will be sent, and whether any other server side interaction is needed. In the majoirty of cases (such as real time and historical prices) data is sent to the IEX Cloud and returned directly to the user, however in the case of the meta-data or user defined queries data is first returned from the IEX Cloud before being stored in the database (should it not exist) and should an existing copy be found, this is instead sent to the user to prevent rate limiting on the IEX Cloud.

The data interaction diagram presents a key concept that should be considered when interacting with the various databases used in financial data, specifically data should be managed with potential caching built-in, which is what the proposed REST API enables.

6 Implementation

The implementation phase was completed using web-based technologies and the tools defined within the technology choices section. As defined within Bob's Project Guidelines [Lar11], this section outlines the core knowledge needed to re-create the visualization platform. The implementation section covers all elements mentioned as complete within table 1, however in some cases these have been combined to present the implementation in a more understandable format. Each sub section will explore the features and functionality, data structures and any issues discovered throughout the projects development.

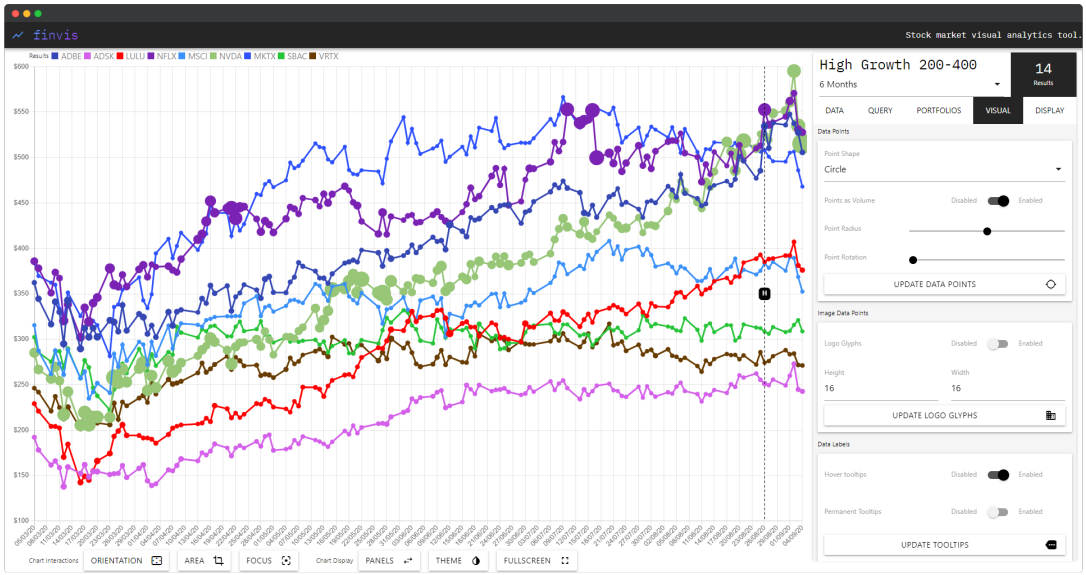


Figure 25: A screenshot of the completed application, with the visual options tab selected.

Figure 25 presents a completed screenshot of the application, with a previous query loaded. The implementation of the project enabled for an advanced financial visualization system to be produced. Within the following section, individual elements of the applications development will be focused on enabling for a complex understanding of how the systems components work. The presented screenshot is an example query of companies which match the parameters of recent growth and a stock price of between 200-400 (at the initial time of query). The completed system therefore combines functionality and requirements from both the basic and enhancement implementations.

6.1 Basic Implementation and Relevant Enhancements

The basic implementation section covers the core functionality produced as part of the project, each element is key to enable succeeding in meeting the projects initial goals. The application is accessible through any locally hosted web server, should all the files be installed correctly. In addition to this, a valid API key for the IEX Cloud is required for enabling access to both real-time and historical data. However, throughout the development of the project a sandbox key was discovered which enabled unlimited API requests for slightly randomised data, using this allowed for significantly more testing to take place as the rate limit no longer applied.

6.1.1 Data Import

The initial step in producing the visualization platform was to create a method of receiving real time and historical data from the IEX Cloud API. For this, an ajax request was used which allows for PHP pages to be called using JavaScript. Within the PHP page is a query relating to the symbol required and the time frame of which the data should cover. Displayed in figure 26 is the code used to perform this task. Data can therefore be imported for any stock on the IEX cloud and any time frame (up to 5 years) can be selected to be displayed.

```
$.ajax({
  type: "GET",
  url: "api/getDailyData.php?symbol=" + value + "&time=" + getData.time,
  success: function (data) {
    try {
      var JsonResponse = $.parseJSON(data);
    } catch (e) {
      getData.checkComplete(1);
      return;
    }
```

Figure 26: The GET request implemented to return data from the web server which handles both historical and real time data.

Figure 26 makes use of a REST API call, which has two main callbacks: success and failure. Making use of the callbacks enables the platform to detect if data has been returned or similar opportunities relating to this. However, issues were found when stocks did not exist as they would return empty JSON objects, meaning the application would display a

results that was not valid. The method implemented to prevent this was to check if there was an error whilst parsing the object using a catch() statement, as a null result could then cancel the entire function and add one to the completed count of symbols.

```
label: value,
borderColor: getData.getColor(value, getData.loops),
fill: false,
backgroundColor: getData.getColor(value, getData.loops),
data: datas,
datalabels: {
  color: "white",
  align: "start",
  anchor: "start",
},
pointStyle: point,
```

Figure 27: The format of the data stored for the line chart, which is part of the getData.generateObject() function.

Within the getData.runLoop function (which activates the previously mentioned GET API call), data is returned, parsed and then re-stored in a format which matches that required for the line chart. Figure 27 displays the JavaScript object format used to store data in the format needed. Each variable (such as label or data) is a specific requirement of the line chart and is passed as a parameter into the getData.generateObject() function. As part of this, any parameters which the user has stored can also be automatically updated such as the pointStyle being set as the point variable.

The functionality of the getData namespace is to request and parse the data that is returned from the API, presented within figure 28 are both the public variables and JSDoc comments for the section. Storing the core data such as symbols, time or labels enables other namespaces to interact and receive such elements enabling easier and more accessible programming. As the data was being returned and parsed correctly, the data import functionality was complete.

```
var getData = {
 /** * Holds the symbols URL parameters. */
 symbols: symbols,
 /** * Stores the time as selected by the user. */
 time: time,
 /** * Holds the produced datasets for the line chart */
 datasets: [],
 /** * Stores a array of labels. */
 labels: [],
 /** * Counts the number of loops. */
 loops: 0,
 /** * Runs the loop to get each symbols data returned from the database. */
 extras: {
   open: {},
   high: {},
   low: {},
   close: {},
   volume: {},
},
```

Figure 28: The code used to request data from the web server and API in real time.

```
<small>Chart Interactions</small>
<a
  id="zoom-button"
  class="waves-effect waves-black btn white black-text tooltipped"
  data-position="top"
  data-tooltip="Change the zoom/pan orientation."
  onclick="toggleZoom(this)"
  href="#"
>
  <i class="material-icons right">settings_overscan</i>
  orientation
</a>
<a
  id="crop-button"
  class="waves-effect waves-black btn white black-text tooltipped"
  data-position="top"
  data-tooltip="Toggle area cropping."
  onclick="toggleSelect()"
  href="#"
  ><i class="material-icons right"> crop </i>
  area
</a>
```

Figure 29: A group of HTML elements for interacting with the line chart. The screenshot only shows a subset of those available in the application.

6.1.2 Web-Based UI

A significant requirement of the project was that it was accessible via the internet, through any modern web browser. Whilst this functionality was defined within the technical specifications, an enhancement feature was also included to provide an extensive list of user options for modifying the produced visualizations within runtime. To display the complex set of options a two panel UI was produced, positioning the line chart to the left and panel of controls to the right. Additionally other options were displayed including a user interaction bar along the bottom, and a list of stocks at the top of the screen. Displayed in

figure 29 is a short example of tooltips alongside the information stored as part of the display of options and the application layout.

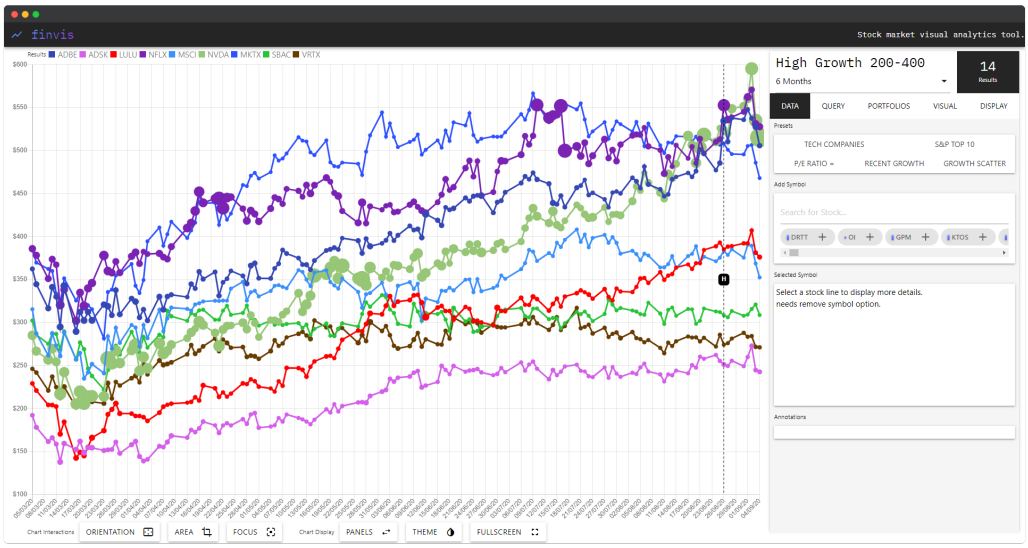


Figure 30: A screenshot of the Web-Based UI as seen on a 1920x1080 Google Chrome window.

Displayed in figure 30 is a screenshot of the final UI, with the five tabs displayed on the right side of the page. The name, count of results and temporal setting are permanent between views whilst the tabs themselves switch between data, query, portfolio, visual and display. This display had been iterated and improved multiple times, with the initial functionality all being displayed on one panel before being deemed that this was too little space to display the complex set of user options available to the user. Therefore, each tab was organised relating to the content and visualization settings available in the application. The grouping of options for the tabs are defined as follows:

- **Data.** The data tab displays options and information relating to the currently selected data or dataset.
- **Query.** The query tab presents options relating to the execution of queries within the system.
- **Portfolio.** The portfolio tab presents the currently saved portfolios alongside options to save the current data.
- **Visual.** The visual tab presents options such as color selection and point display options (e.g. points as volume and points as company logos).

- **Display.** The display tab presents options relating to the line chart and other glyphs displayed within view.

```
/** * Store performance related variables for re-setting lineChart */
  performance: {
    linesActive: true,
    bezierCurves: 0,
    lineSpanGaps: true,
    animation: 1000,
    pointRadius: 3,
    pointRotation: 0,
    pointShape: "circle",
    logoGlyph: false,
    iconHeight: 16,
    iconWidth: 16,
    useVolume: true,
  },
```

Figure 31: A screenshot of the utils.performance options which are stored within the browsers cache.

Options have been provided for almost all use cases of the system, through using the tabs at the top of the right panel options such as: point size, color map selection, display lines and animation can be changed and customised to the users needs. Options are presented as HTML form elements, meaning that each element is displayed as either a drop down menu, number input, switch or range slider. Upon completion of selecting the dropdown elements, the selected data is stored in the utils namespace such as the performance options seen in figure 31 of which it can be included upon the next re-generation of the line graph (which in normal cases occurs immediately after the storing of the new options).

6.1.3 Line Graph

The line graph was mainly implemented using a library Chart.js, which enables a canvas-based element to be displayed and interacted with by the user. The data being displayed was temporal (time) in comparison with stock price. Each line chart had multiple categorical groups, each of which was a stock, which is also to be displayed on the line chart. Each dataset is stored as a JavaScript object which includes strings and arrays (for data and labels), upon being loaded into the Chart.js element, the features are iterated

through and displayed on screen to the user.

```
this.chart = new Chart(ctx, {
      // The type of chart.
      type: "line",
      data: {
        labels: getData.labels,
        datasets: getData.datasets,
      },

      options: {
        showLines: utils.performance.linesActive,
        legend: {
          display: false,
        }
        ...
```

Figure 32: A screenshot of the code used to generate and display a chart to the user through Chart.js.

In figure 32 the object and code for producing and updating the settings of the chart is displayed. As previously seen, the figure includes options relating to whether lines are visible and similar performance-based settings are controlled through this function within lineChart.generateChart(). Upon requesting the generateChart() function, data received within getData.runLoop is included within the data object, which beigns to process the and display the selected visualization.

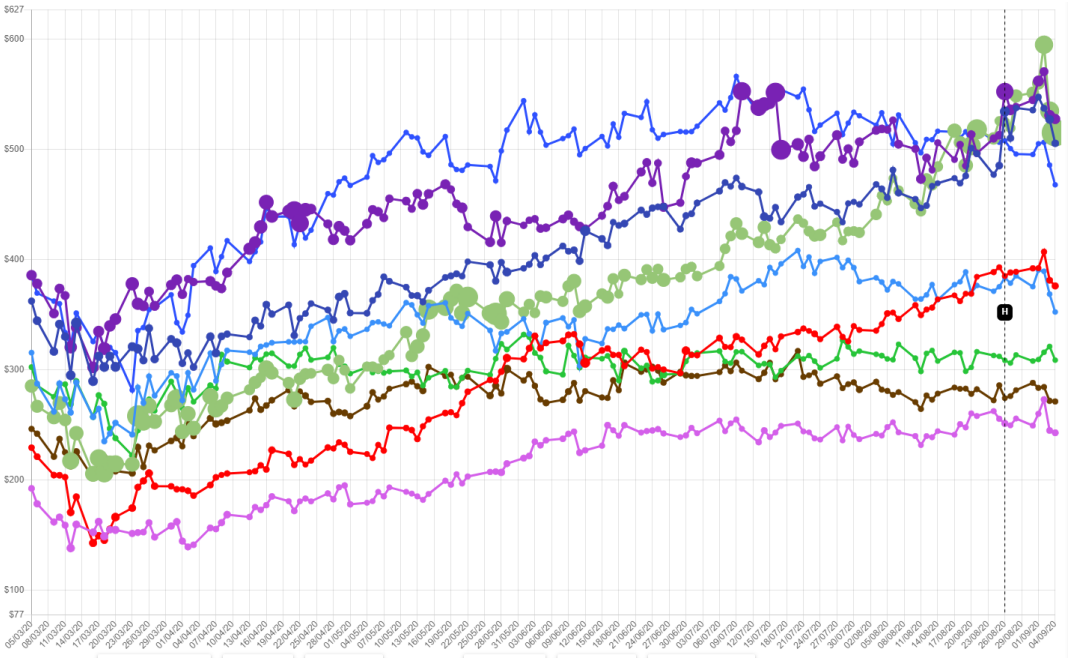
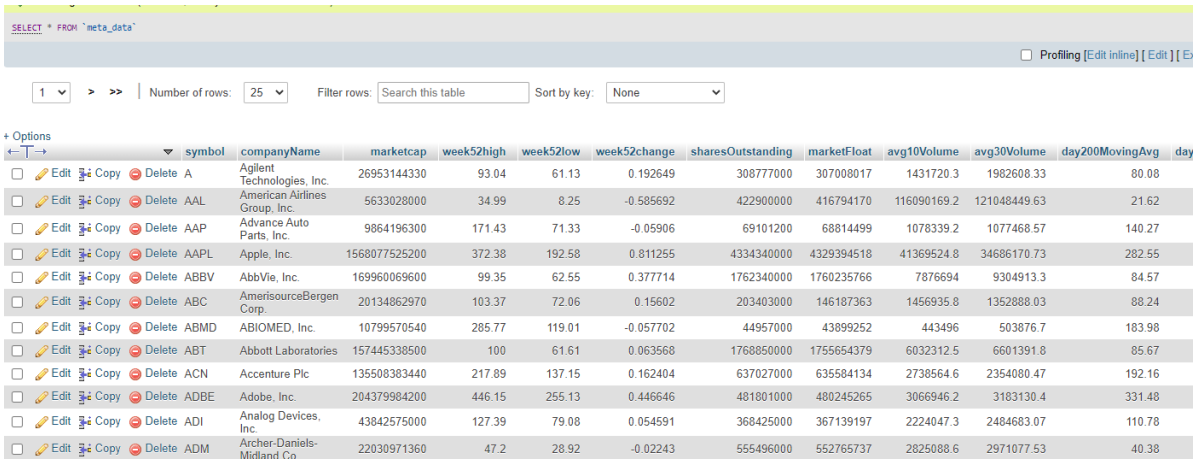


Figure 33: A screenshot of line chart visualization within the web browser, displaying all content within the canvas element.

Figure 33 displays the output visualization within the canvas element, the x axis displays dates through the ISO 8601 format, which enables the visualization to dynamically change dates - specifically useful when changing between day and month formats. The y axis displays the price of stocks using a decimal format, before appending a dollar sign to the start of the number. Upon completion of processing for the lineChart instance, the linechart.createLegend() function is ran, which iterates through all valid stocks and displays them to the user (allowing them to hide/show specific stock choices).

6.1.4 Queries



SELECT * FROM `meta_data`

Profiling [Edit inline] [Edit] [E

1 > >> | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	symbol	companyName	marketcap	week52high	week52low	week52change	sharesOutstanding	marketFloat	avg10Volume	avg30Volume	day200MovingAvg	day
Edit Copy Delete	A	Agilent Technologies, Inc.	26953144330	93.04	61.13	0.192649	308777000	307008017	1431720.3	1982608.33	80.08	
Edit Copy Delete	AAL	American Airlines Group, Inc.	5633028000	34.99	8.25	-0.585692	422900000	416794170	116090169.2	121048449.63	21.62	
Edit Copy Delete	AAP	Advance Auto Parts, Inc.	9864196300	171.43	71.33	-0.05906	69101200	68814499	1078339.2	1077468.57	140.27	
Edit Copy Delete	AAPL	Apple, Inc.	1568077525200	372.38	192.58	0.811255	4334340000	4329394518	41369524.8	34686170.73	282.55	
Edit Copy Delete	ABBV	AbbVie, Inc.	169960069600	99.35	62.55	0.377714	1762340000	1760235766	7876694	9304913.3	84.57	
Edit Copy Delete	ABC	AmerisourceBergen Corp.	20134862970	103.37	72.06	0.15602	203403000	146187363	1456935.8	1352888.03	88.24	
Edit Copy Delete	ABMD	ABIOMED, Inc.	10799570540	285.77	119.01	-0.057702	44957000	43899252	443496	503876.7	183.98	
Edit Copy Delete	ABT	Abbott Laboratories	157445338500	100	61.61	0.063568	1768850000	1755654379	6032312.5	6601391.8	85.67	
Edit Copy Delete	ACN	Accenture Plc	135508383440	217.89	137.15	0.162404	637027000	635584134	2738564.6	2354080.47	192.16	
Edit Copy Delete	ADBE	Adobe, Inc.	204379984200	446.15	255.13	0.446646	481801000	480245265	3066946.2	3183130.4	331.48	
Edit Copy Delete	ADI	Analog Devices, Inc.	43842575000	127.39	79.08	0.054591	368425000	367139197	2224047.3	2484683.07	110.78	
Edit Copy Delete	ADM	Archer-Daniels-Midland Co	22030971360	47.2	28.92	-0.02243	555496000	552765737	2825088.6	2971077.53	40.38	

Figure 34: A screenshot displaying the rows and columns of the meta data table within php-myadmin.

The screenshot displayed within figure 34 is of the tool phpmyadmin, which was used within the project to store data relating to key statistics and meta data (for use with the core queries within the application). Meta data was stored in this way as rate limits for the IEX Cloud API would not allow multiple requests for stocks and the key statistics relating to them. The solution was to produce a iterative code element which could be ran using a server function every morning/night (depending on the future requirements of the application). In addition to this, storing the data within an SQL database allowed more complex queries to be formed, enabling the view seen in figure 35.

Figure 35 displays the various user options enabled through the SQL database, which upon implementation can combine, merge and join multiple queries. Using the SQL-based database for this task enables users to query any data stored in the database in the format matching the parameters required. Upon selecting or choosing a meta-data group the user is also presented with options relating to removing or changing the current queries.

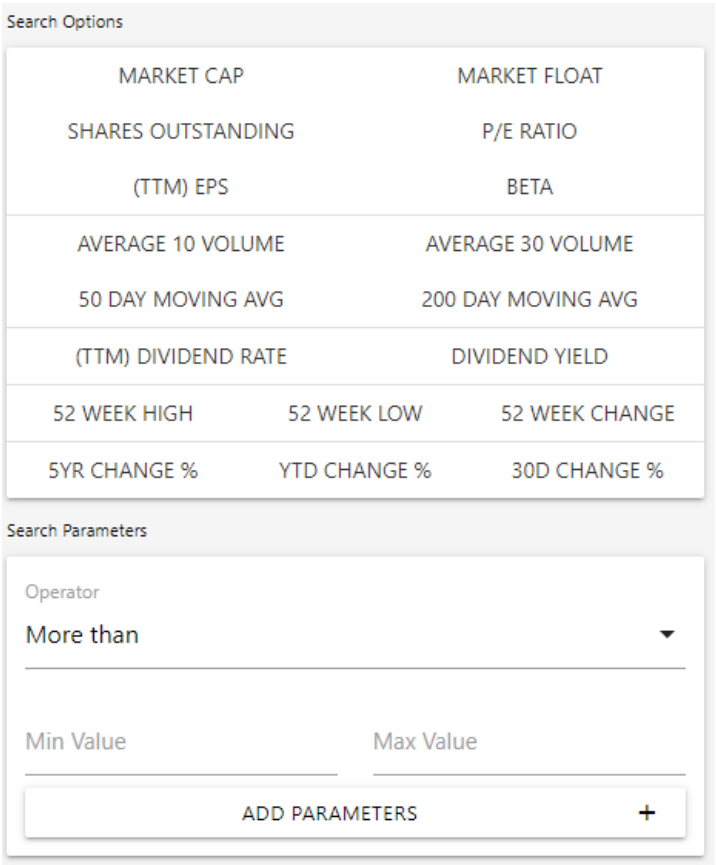


Figure 35: A screenshot of the query tab, displaying the relevant meta data options and the user selectable parameters.

6.1.5 Portfolios

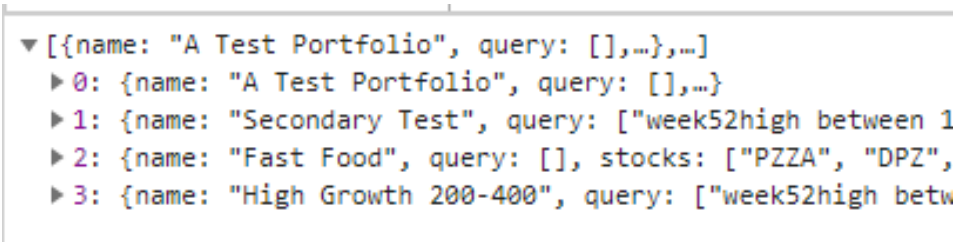


Figure 36: A screenshot of the application tab within the Chrome Developer tools, displaying the currently saved portfolios.

Figure 36 displays how and the format of which local storage data objects are used to store the users selected portfolios of stocks and queries. Storing the data within HTML5 local storage enables for quick access within the code without the need for an external service to be made write-able by the user. Each object is then iterated through and displayed to the user in the format seen in figure 37. Issues relating to having no queries or stocks were solved through only displaying an arrays contents should it actually contain a result or query.

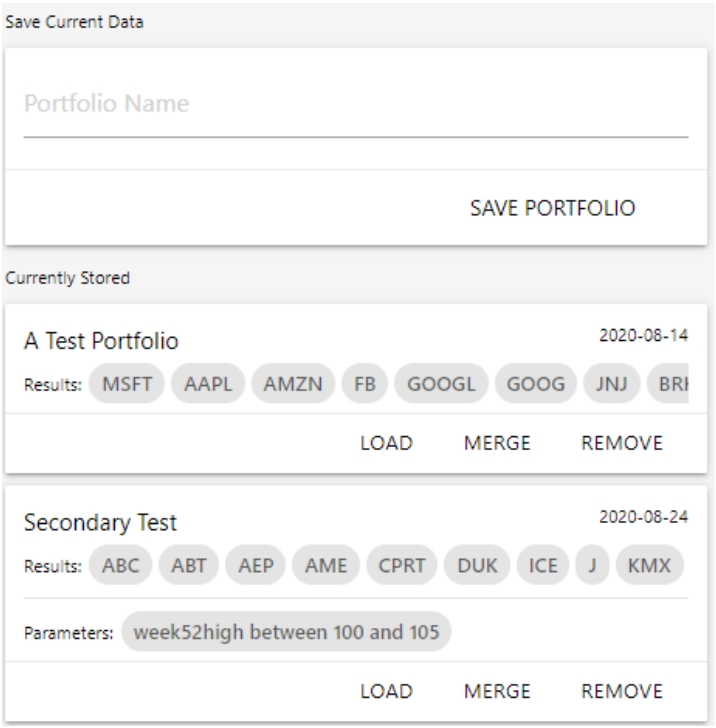


Figure 37: A screenshot of the portfolios tab on the website, presenting the users current portfolios.

Portfolios can be loaded into the visualization platform dynamically, using either stocks or the query parameters chosen. Portfolios may also be merged, enabling novel visual analytics approaches such as comparing high/low stock groups against one another. The portfolio save option also stored the last access time of the query, meaning that a time-based glyph as seen in figure 38 can be displayed to the user enabling them to understand what has changed since the last time they had opened the platform.

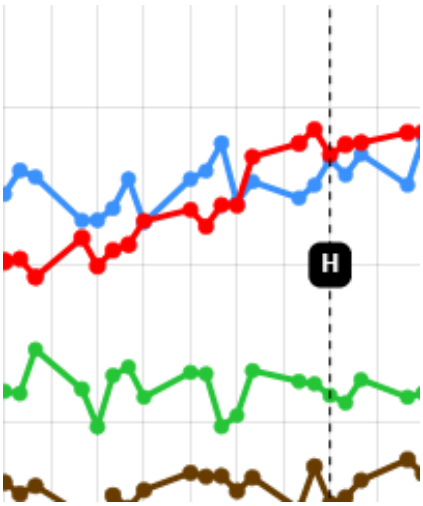


Figure 38: A screenshot of historical time glyph being displayed within the line graph canvas.

6.1.6 Colour Selection

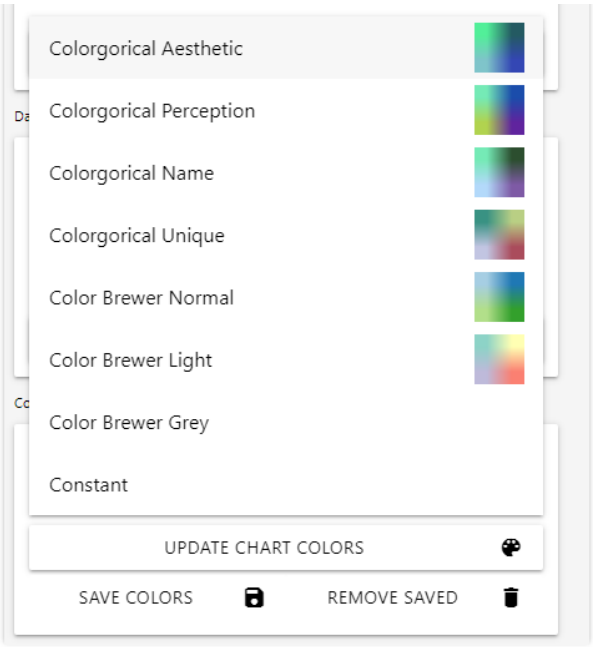


Figure 39: A screenshot of the color map selection tool, which can be used alongside completely custom colors.

Figure 40: Individual screenshots of the four color palettes produced using Colorgorical, displayed left to right as a, b, c and d.

Colorgorical by Gramazio et al [GLS17] is able to produce aesthetically preferable categorical color palettes [GLS17]. Each color is assigned a weight, and through using the web-based tool the perceptual difference, name difference, pair preference and name uniqueness was able to be changed. Figure 40 showcases the variety of palettes produced for the system whilst figure 39 presents how the user is able to select them, which implements both colorgorical [GLS17] and color brewer [HB03]. A limitation with both tools is that is requires the outputting system to have a limited number of potential categories. Therefore a user option is embedded within the drop down menu, to allow settings relating to constant coloring or the predefined palettes to be used. Constant coloring allows for colors to be constant across different generations of the graph, however limits the potential aesthetic quality of the overall web application.

6.2 Other Enhancements

Enhancements have been spoken about as part of 6.1 (in each case where an enhancement builds upon existing work), however in cases where no existing functionality exists for the enhancement, the extended work will be mentioned within the following section.

6.2.1 Preset Options

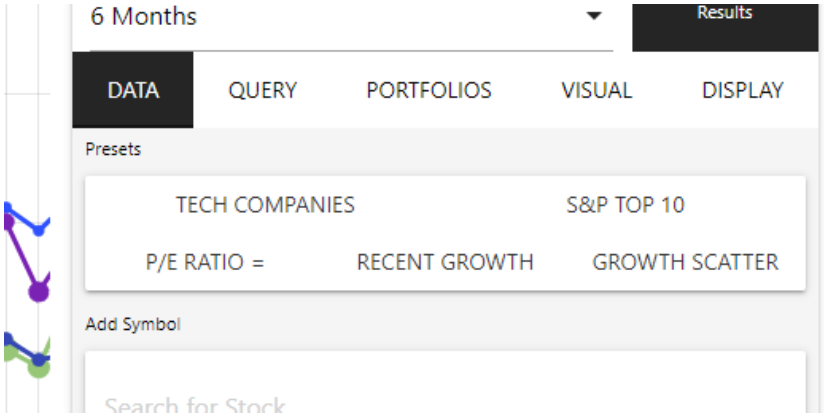


Figure 41: A screenshot of the presets displayed as part of the system, which make it easier to run initial queries.

Figure 41 displays a set of presets which the user can immediately access within the application. The idea behind the concept is to enable quick exploration should a user only want to view the visualizations for a couple of minutes. The presets enable rapid exploration to be performed, whilst also allowing the full functionality of the application to be displayed in a quick and understandable format.

Some of the presets are displayed as case studies within the testing and evaluation section of this thesis.

6.2.2 Line Graph Interactions

Adding interactivity to the line graph was a key element of the application, as movement, panning and zooming would enable greater exploratory capabilities of the platform to be used. Displayed in figure 42 is a screenshot of all the interaction and display user options available through the system. The line graph interactivity initially had issues relating to zoom and panning boundaries, as constraining both edges resulted in zoom and

panning to not work correctly. Therefore, the right side of the graph was left without constraints resulting in the visualization being easier to use and move around.

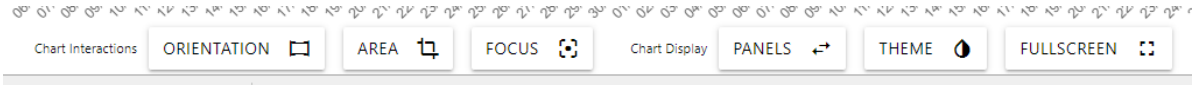


Figure 42: A screenshot of the interaction toolbar placed at the bottom on the line chart.

The visualization interaction options enabled: tool tips, pan and zooming to be implemented into the chart. Each option was enabled by the user options visible beneath the graph. Users could select whether to lock movement horizontally or vertically and could also use an area selection tool to zoom into a specific area of the visualization results. The interaction options overall enabled a variety of exploratory techniques to be implemented, and combined with the on click and on hover event comparisons investors are able to explore large areas and small areas of the market concurrently. The image in figure 43 displays a user selecting a group of stocks to discover more information. In addition to the interaction improving accessibility, the chart axes are able to scale meaning results can be instantly compared to the hover line and other stocks visible.

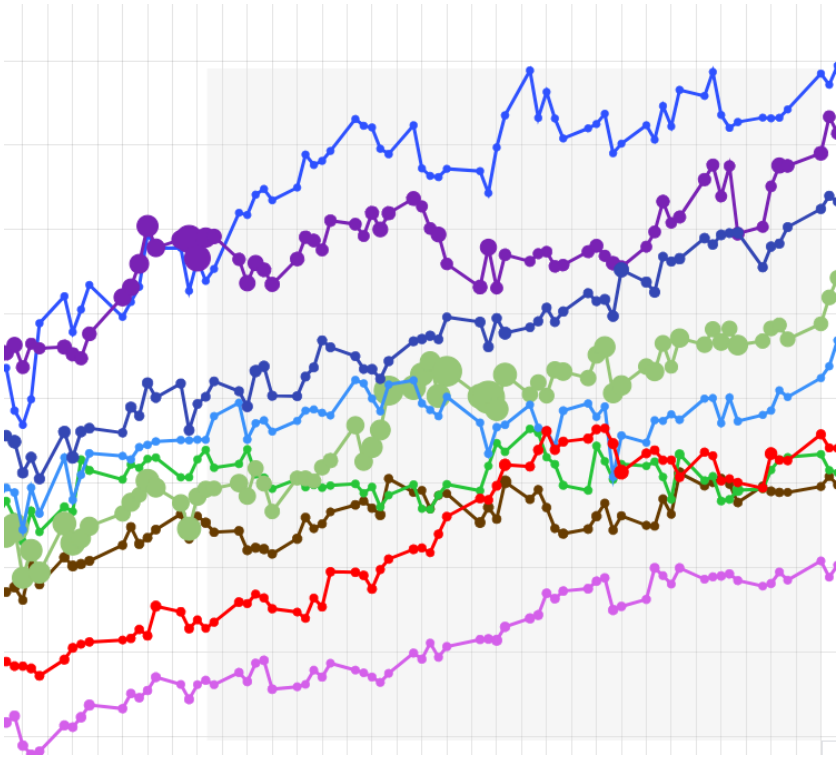


Figure 43: A screenshot of the user selection box being dragged around a group of stocks for further analysis.

6.2.3 Points Display

Individual data points are able to convey a large amount of information, in addition to the position on the graph; size, color and other visual variables are able to share information about specific points and elements. For the purposes of FinVis, the visual variables include; shape, size, color and pattern (in the sense of a company logo). These options were implemented as they assisted greatly in the exploratory capabilities of the systems whilst also using data which was already available and stored through the getData namespace. The changing values of points is performed before the final object is generated and the code for such actions can be seen in figure 44.

```
if (getData.time === "1d") {
  pointValue = volumePoint + item.volume / 1200;
}

if (pointValue <= utils.performance.pointRadius) {
  pointValue = utils.performance.pointRadius;
} else if (pointValue >= 34) {
  pointValue = 34;
}
```

Figure 44: A screenshot of the user selection box being dragged around a group of stocks for further analysis.

The code enabling the changing of points can be accessed through the 'visual' tab on the user interface. The displayed user options are able to change the decimal size of points, alongside activating options such as 'points as volume' and 'points as logo' which both change the visual appearance of the points to integrate other data about the presented companies. Point initially were not as complex, however to enable the application to contain more data at a glance, the points as volume option was produced - which changes the size of a point into a volume. However this volume setting is constrained to be capable of a max size of 34 pixels as in some case point would overload and occlude other elements on the chart. Occlusion is a significant issue within the application as in some cases points may overlap and prevent other information from being displayed. Through comprehensive testing of the system, it was found that a maximum size of 34px enabled visualization to present the data without overwriting other information on the screen.

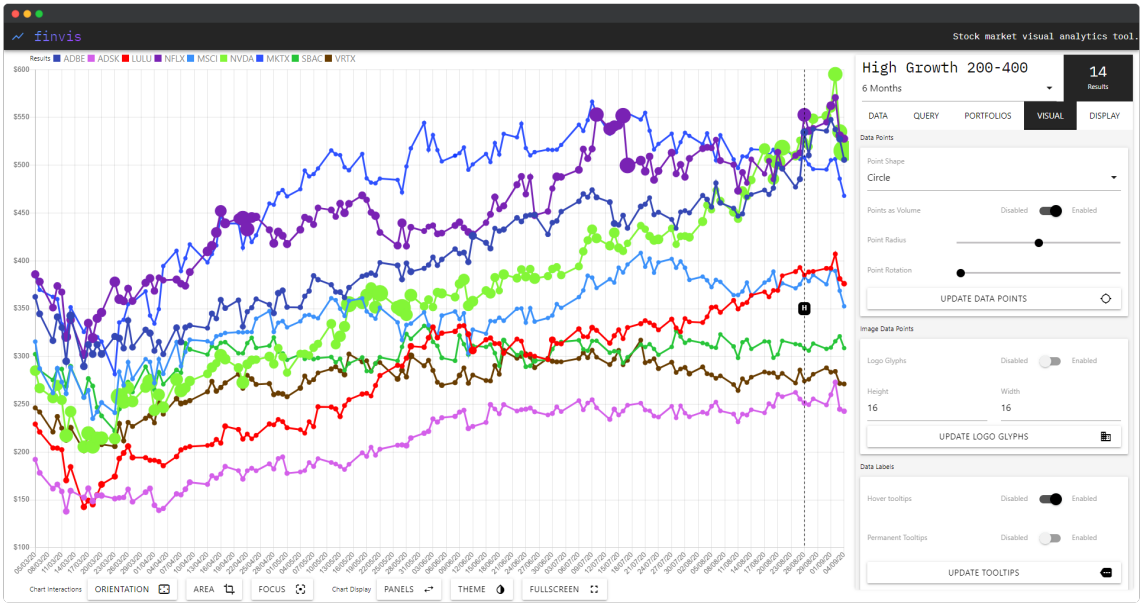


Figure 45: A screenshot displaying the user option of 'points as volume' which changes size of data points depending on the volume of the stock in the specified time frame.

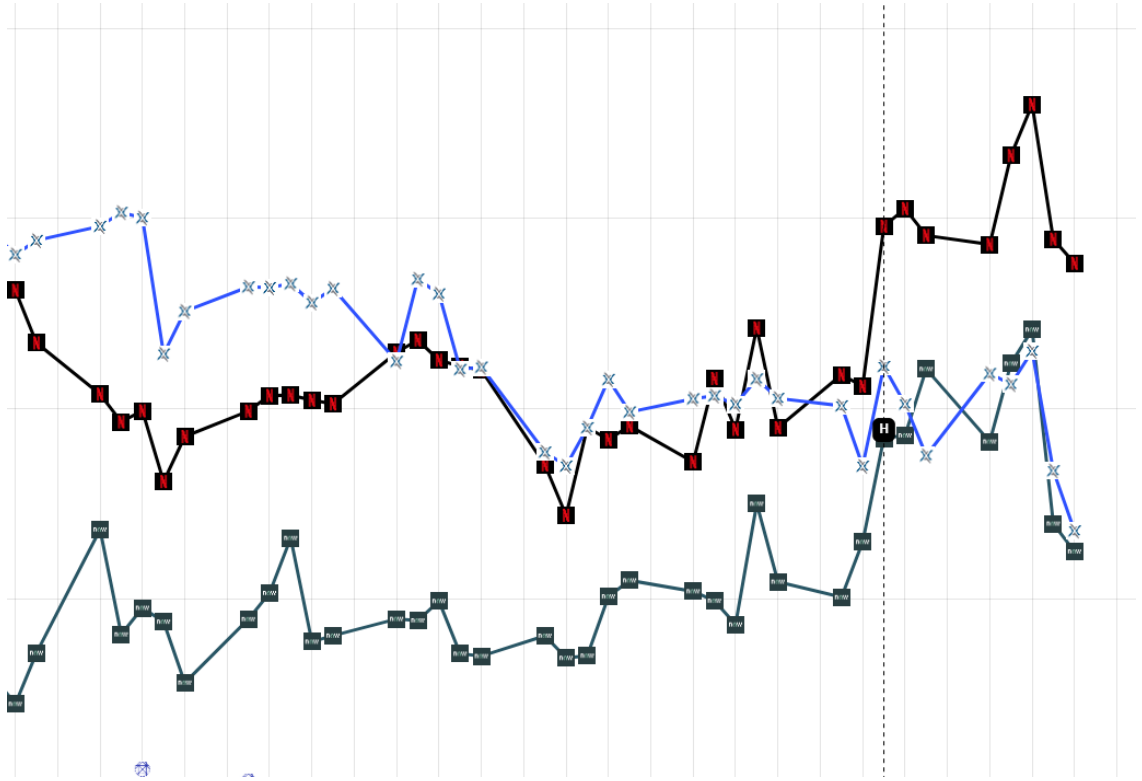


Figure 46: A screenshot of the user option 'points as company logos' which presents the individual data points as a company logo.

The results relating to the points as volume user option are displayed in figure 45 whilst the individual stock logos are presented in figure 46. Both user options enable other exploratory methods to be used by investors when interacting with the application. Point as

volume uses the attached volume array which is returned within the getData namespace, and then modified to match sizing requirements through the lineChart namespace.

6.3 Code Guidelines

6.3.1 Code Commenting

As defined within Bob's Project Guidelines [Lar10b], JSDoc was included. Installed through NPM using the `npm i jsdoc` command, documentation can be rapidly produced and created for each JavaScript element of the website. Whilst the guideline document suggests using a Java documentation or C documentation library [Lar10b] an equivalent was found for the JavasScript language and therefore was implemented throughout. Potential usages of comments included each individual namespace, function and parameter however full documentation on how to use the tool is available on the JSDoc website.

Presented as part of the portfolio submitted alongside this document is a HTML website in the `out` folder of the submission. The website generated through JSDoc enables potential future researchers or other developers to understand the parameters, namespaces and functions used within the development of the project. The output website enables exploration of the code base through interactive means, and was updated continuously throughout the project.

6.3.2 Coding Conventions

Bob's Project Guidelines [Lar11] suggests using the coding guidelines mentioned in Bob's Concise Coding Conventions (C3) [Lar10a]. The guidelines mentioned [Lar10a] were followed throughout the production of the code where possible, however due to the nature of JavaScript additional guidelines were followed. The book JavaScript Patterns [Ste10] was followed as a key reference throughout the project, which aligns closely to the guidelines suggested within the C3 paper [Lar10a], however focuses on JavaScript implementation.

Notable conventions followed throughout the programming focused on ensuring methods were short and concise [Lar10a], methods should not take more than five parameters [Lar10a] and the implementation of camelcase [Bin+09] (with a notable exception being the imported functions from external packages). Through using the coding

conventions mentioned throughout the documentation [Lar10a; Bin+09; Ste10] an application in which another developer or researcher is able to build upon has been produced.

7 Testing and Evaluation

The testing and evaluation section presents the results of both a case studies and a performance analysis. Comprehensive testing of each element took place throughout implementation of the project, and the results of these tests can be seen through the usable and resulting visualizations produced. The case studies presented are also available as presets within the application, whilst the performance analysis assists in understanding the potential of the user settings and any constraints linked to too many elements within the canvas.

In some web based projects it may be possible to implement testing frameworks, however this is out of scope for this project as it usually requires large teams of developers and a significant amount of time. A testing table was not used, however individual element tests did take place. A method of monitoring console events and checking data results means that individual UI testing is not needed as all results can be displayed within the Chrome developer tools. It may also be suggested that in the future a comprehensive user test takes place; however due to the current COVID-19 situation, and the lack of time to produce a rigorous testing framework the visualizations produced will be used as results and enable evaluation through the exploratory methods and visual images produced.

7.1 Results

To enable results generation and testing of the application, a range of case studies were generated (A-C). Each case study displays the novel approaches available to investors within the produced application. For each case study, the method of generation and the results of queries/functions are displayed in addition to this a list of stock symbols is supplied. Each section also combines short notes on why certain functionality and display options were chosen and how these might assist in an exploratory analysis of the stock market.

7.1.1 Case Study A `Bargain Stocks`

The purpose of Case Study A is to discover and find bargain stocks through using and interacting with the queries and visualization options available on the web based platform. In

the case of bargain or undervalued stock discovery, the P/E ratio and year to date change meta-data options will be used. The parameters for such will focus on stocks with a low P/E ratio (below the average of the stock market in question) and a year to date growth rate which is positive. Through querying the two parameters, results will then be presented and displayed to the user and an analysis of such results can be performed using the tools available within the application.

The data being tested in returned from the IEX Cloud and is a relatively large dataset, depending on the time period selected data for each point ranges between 200-400 elements in length with that then being multiplied by the number of stocks (in this case five). The dataset ranges from the users determined time selection, which in the case of the presented case study is the past six months. The data is structured in format, as it is presented and received in a predictable format. Data that does not exist (should a stock not be traded at that time) is returned as a null value, meaning no point is placed on the visualization system.

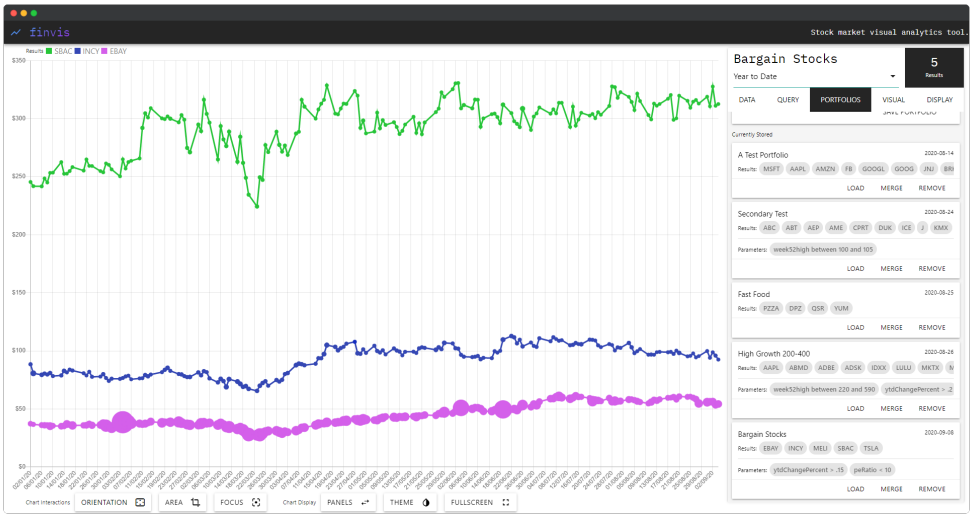


Figure 47: A screenshot of the resulting visualization produced as part of Case Study A, displaying three of the five stocks as both MELI and TSLA traded at above $800.

The specific parameters in use will be: `year to date change percent > 15%` and `P/E Ratio < 10`. Combining the two queries is possible through the user interface elements on the query tab of the right panel. Upon running the query, five stocks are returned which include: `SBAC, INCY, EBAY, MELI and TSLA`.

Figure 47 presents the results from the proposed queries, overall five different stocks were returned however both TSLA and MELI were filtered by the user as they were trading

at above $800 meaning they would significantly effect the view of SBAC, INCY and EBAY. The presented stocks can then be explored through the built-in visualization interaction options, specifically in the case of figure 48 the visualization can explore the slight market drops which occurred during the initial stages of COVID-19. Using the provided interaction tools, stocks within the time period can be compared to discover which had high/low growth during the March period. The tool also allows for the visualizations to be explored at current and past prices, meaning daily or a five year view can be selected - should it be required.

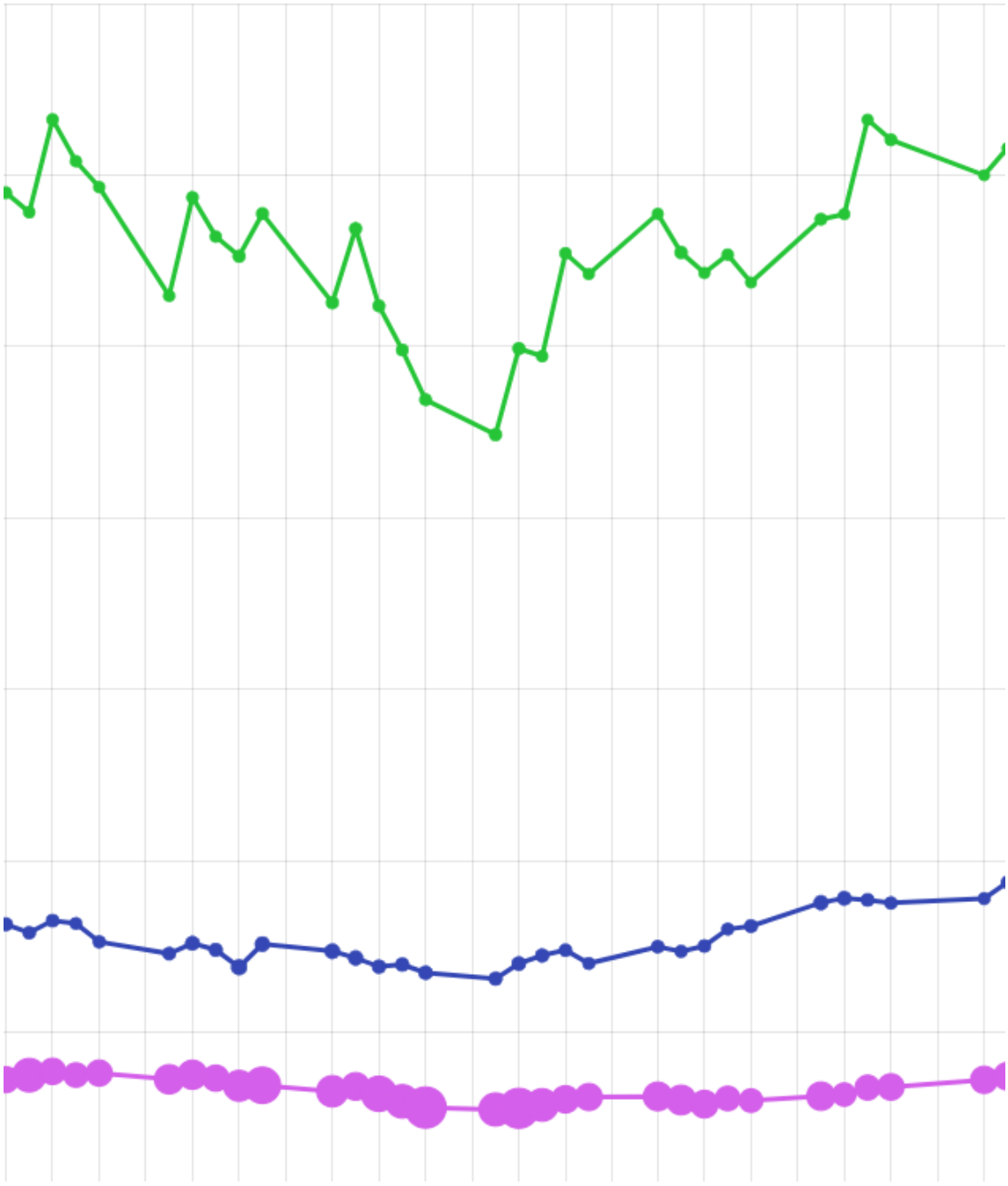


Figure 48: A screenshot of the resulting visualization produced as part of Case Study A, displaying a zoomed in section of the line graph during March 2020.

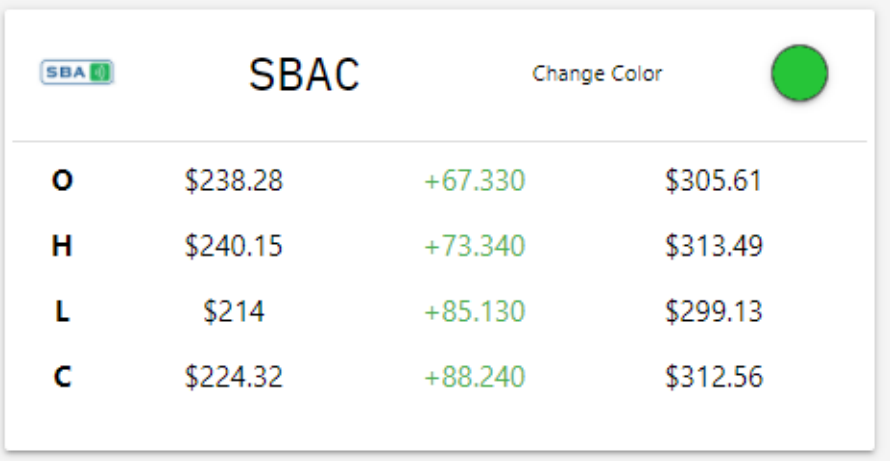


Figure 49: A screenshot of a comparison between the SBAC stock in March compared to the present day price. The visualization tools provided enable investors to explore stock prices in comparison to each other.

Figure 49 presents a comparative analysis which can be performed within the system, using the 'data' tab. The comparison can be generated using historical and current stock data of any symbol currently displayed on the screen. The table provides a price increase/decrease and enables a quicker understanding of the market presented as part of the visualization. Specifically in the case of SBAC, the large growth is displayed from March to the 7th September. In addition to this, to enable specific stocks to stand out, the SBAC stock was changed to a green color as this was the same as the companies logo.

7.1.2 Case Study B `Technology Portfolio`

Case study B presents a method of visualizing an existing stock portfolio, in this case the user is looking to display a comparison of prices within the technology industry. Using either URL parameters or the visual search tool in the data tab seen in figure 50, the user is able to generate and modify visualizations relating to the query. Upon generating the query, it is possible for the user to remove the line and also enable logo glyphs meaning stock prices can be compared in a more general form, the user is once again able to use chart interactions to zoom and customize the visual experience. The data for the technology portfolio search is the same as that defined for case study A, however case study b returns nine results instead.

The visualizations produced for case study B are displayed within figure 51, the produced visualization enables a visually appealing exploratory analysis to be performed of

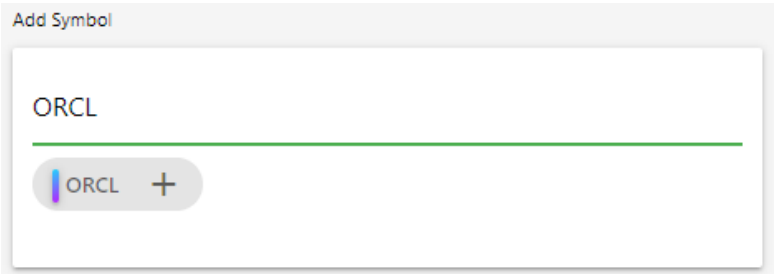


Figure 50: A screenshot of the visual search displaying the user wishing to add ORCL to the visualization, all stock symbols can be added/removed from this tool.

the users custom technology stock portfolio. Using user options available through the visual and display tabs, it is possible to modify the visualization to present data in multiple formats within the FinVis system. Case study B also showcases the systems potential usage as a tool for news websites, where a collection of stocks can be stored as URL parameters to share market updates at specific times through a link. Each generation of the visualization changes the URL, meaning that should a investor want to share a portfolio it is possible to do so through the system.

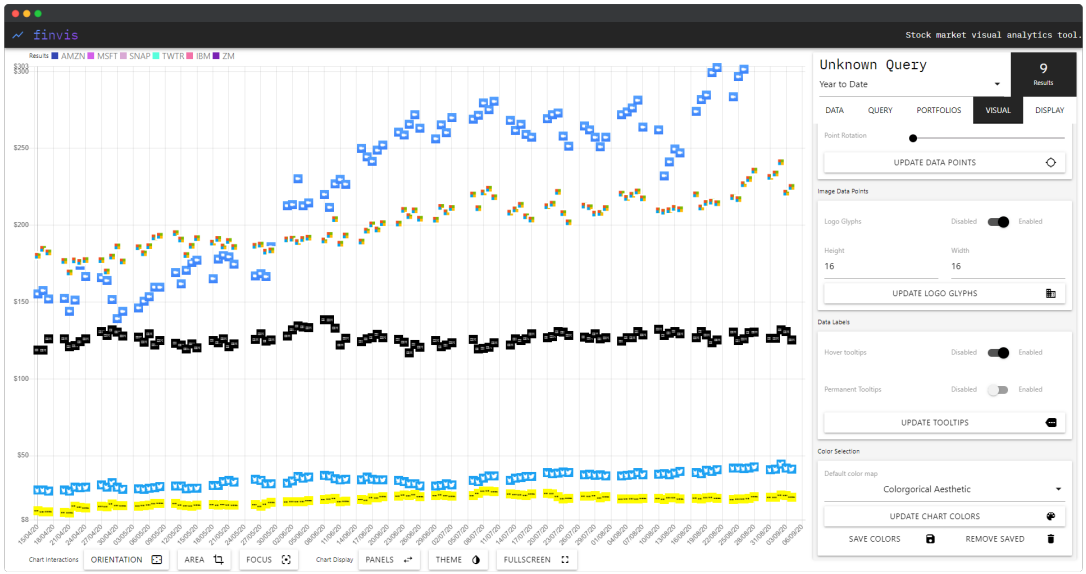


Figure 51: A screenshot displaying the produced visualization of the users custom 'technology' stock portfolio, each dataset is shown using a logo glyph as opposed to a line.

7.1.3 Case Study C `COVID-19`

Case study C attempts to present opportunities relating to analysis of market downturns or improvements. When saving a portfolio on the FinVis system, the date and time is also saved within the portfolio object. Using this data, it is possible to present the user with a

previous view of the system enabling potential analysis improvements relating to last known prices in comparison to current. Using the recent March 2020 downturn, it is possible to present a previous stock portfolio and showcase the features possible for investor analysis. For the purpose of the case study, the last access date was changed to the 14th March and was then opened again on the 7th September.

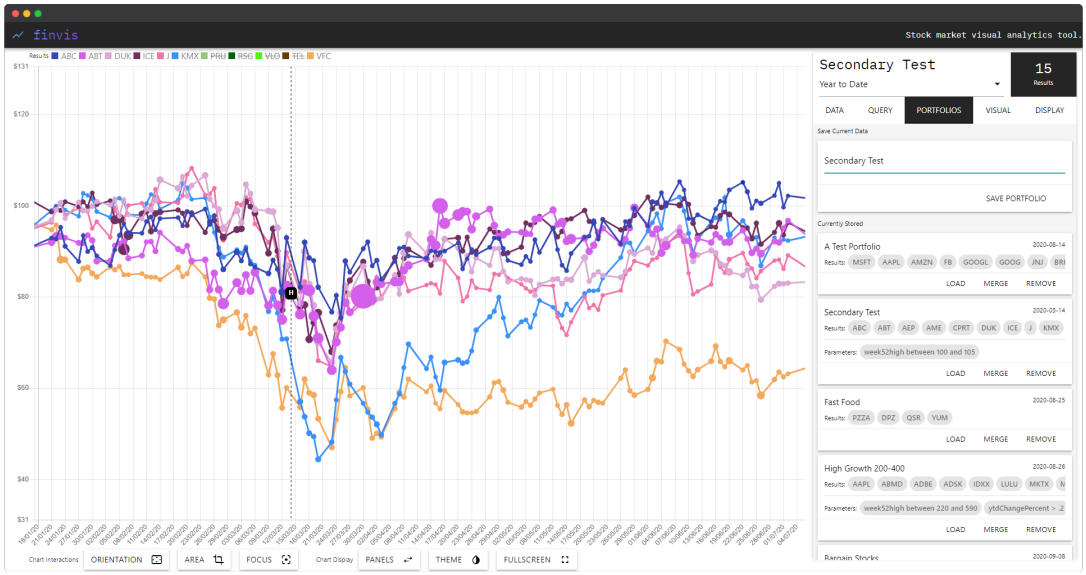


Figure 52: A screenshot displaying the visualization produced as part of case study C, showing the H glyph at the date of last access.

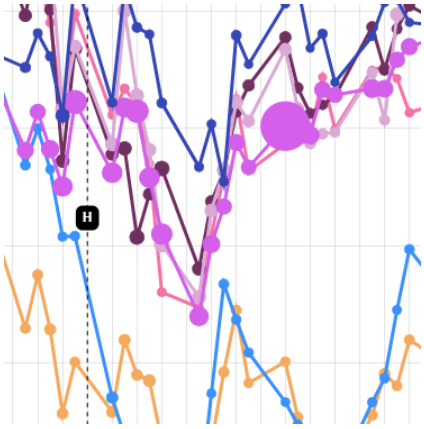


Figure 53: A close-up screenshot of the H glyph as part of case study C, which is displayed on the 14th March.

Figure 52 displays the produced visualization upon the loading of a previously saved dataset, the visualization is able to load queries and results or merge the query or results with the existing symbols. Through loading the results, a historical H glyph is displayed, showing the price and value of the data the previous time a user loaded it in the application. This enables many novel uses, such as allowing the monitoring of stocks over a long period

of time or in this case enable comparison to past data. Through the visualization in figure 53, it is possible to see that the ABT stock had significantly higher volumes traded during the March period in 2020.

7.2 Performance Analysis

7.2.1 Data Retrieval Analysis

Number of Stocks	Loading Time (ms) Any Errors
5	1110 ms
10	1562 ms
15	2140 ms
20	3098 ms
25	3284 ms
30	3859 ms
35	4311 ms
40	4612 ms

Table 2: The loading times (in milliseconds) for returning data from the REST API through the IEX Cloud.

The data retrieval analysis is calculated from the initial GET request being sent, to the completion of all stocks. A console.log command is inserted to display the timestamp at both the start and end, which then allows the extraction of the loading time to be displayed in milliseconds. The number of stocks significantly changes the possible loading time of the system, as each stock means more data is being returned through the API.

The data displayed in table 2 presents the number of stocks in comparison to the loading time. The findings suggest that the larger number of stocks being queried, the longer it takes for data to be requested and then received from the web server. However, results for this analysis may change depending on internet speeds meaning a larger amount of tests would need to be ran over a longer period of time from various connected devices. Further testing is required based around data retrieval speeds as it significantly depends on the API performance at the time and testing devices connection speed. It may be possible within future work to dynamically load stocks into the visualization, meaning that once data is returned from the API, it is placed within an interactive interface using a synchronous connection.

Option Changed	Loading Time (ms)	Any Errors
Logos as Points	1982 ms	
Volume as Points	1734 ms	
Color Map	1450 ms	
Animation Removed	784 ms	
Animation Increased	2108 ms	
Lines Removed	1389 ms	
Data Labels Added	2304 ms	

Table 3: The loading times (in milliseconds) for changing different user options within the application.

7.2.2 Visual Performance

The table presented within table 3 showcases how different user options change the overall speed of the applications load. Each test was set up using the same stocks, amount of data (5 stocks over 1 month) and through the same Chromium-based web browser (Google Chrome 85.0). The performance analysis of such data was taken through placing console.log() functions within the run time code, allowing for an analysis to be performed of each individual user option.

Findings suggest that it is possible to significantly improve the speed of canvas generation through removing animations from the element. This however removes aesthetic quality from the end product, meaning presenting it as a user option enables end users to decide which technique to choose. Significantly the second largest loading time was extending the animation speed, as this requires more frames to be displayed to the user, which means the browser needs to process more elements over time. It may also be noted from the results that points as logos and volume slightly increase the average load time, meaning that the increased amount of data slightly increases the potential processing power required.

As with the previous performance analysis, further testing through end user browsers would enable better and more complete results to be gained. Using web-based analytics tools may assist in finding slow or elements which lack optimization presenting potential for further development work to increase these.

8 Conclusion

The thesis and development work enabled the production of a visual analysis system for financial stock markets. The tool produced enabled multiple queries and statistics to be queried, whilst also producing exploratory visualizations capable of presenting the data in a line graph format. Through the case studies which took place, a large amount of opportunities exist within the application for assisting in the exploration and analysis of financial markets. Results that are specifically effective include those that used queries to find stocks that match specific queries, or importing an existing portfolio of stocks to produce a comparative visualization from.

The application provided a large amount of interactive visual analysis, which can assist in the exploration of the stock market. Whilst also provided methods of visualizing stock data in the following formats:

- View large datasets as a scatter graph, without any lines.
- Visualize specific and closely related stocks, whilst also comparing volume.
- Produce aesthetically pleasing visualizations of stocks within specific industries.
- Display logos of companies as individual data points, enabling larger displays.

Whilst the application may not be a complete system yet in comparison to Google Finance or Seeking Alpha, the produced visualizations and potential queries offer a complexity that is not available within the current set of tools. To further improve the application access to more advanced and complete datasets is required, to allow for more exploratory queries to be produced.

Overall the research provided a novel method of visualizing stock market data, alongside a complex collection of user options accessible without a download through a web browser. To further develop the work, a larger amount of data access is required to assist in providing further analysis features.

9 Future Work

Future work may explore options relating to the investigation of the use of micro-visualization to present portfolio statistics without having to click on each individual collection. Further development could also provide better comparative analysis of stocks, through interactive tools comparing different rows of data from the returned selection. Work within financial visualization has been significantly developing in recent years, and therefore other visualization options could also be explored.

In addition to having a larger amount of data to work with, future work within the subject area should look towards providing better visual analysis of specific meta data types - and may look to include these as either glyphs or icons within the line chart. Future researchers may also investigate methods of tracing or drawing desired stock lines on the graph, before querying the database to present these to the investor.

References

[Alp20] Seeking Alpha. *Stock Picks, Stock Market Investing*. 2020. URL: https://seekingalpha.com/symbol/AAPL.

[ASP] Explore Five Lightweight Alternatives, Aravind Shenoy, and Anirudh Prabhu. "CSS Framework Alternatives". In: ().

[Atl] Atlassian. URL: https://trello.com/.

[Bin+09] Dave Binkley et al. "To camelcase or under_score". In: *2009 IEEE 17th International Conference on Program Comprehension*. IEEE. 2009, pp. 158–167.

[BN13] J Efrim Boritz and Won Gyun No. "The quality of interactive data: XBRL versus Compustat, Yahoo Finance, and Google Finance". In: *Yahoo Finance, and Google Finance (April 18, 2013)* (2013).

[CH01] Alistair Cockburn and Jim Highsmith. "Agile software development, the people factor". In: *Computer* 34.11 (2001), pp. 131–133.

[Clo20] IEX Cloud. *IEX Cloud: Financial Data Infrastructure*. 2020. URL: https://iexcloud.io/.

[Csa+03] Christoph Csallner et al. "Fundexplorer: Supporting the diversification of mutual fund portfolios using context treemaps". In: *IEEE Symposium on Information Visualization 2003 (IEEE Cat. No. 03TH8714)*. IEEE. 2003, pp. 203–208.

[Dao+08] Huyen Tue Dao et al. "NASDAQ Velocity and Forces: An Interactive Visualization of Activity and Change." In: *J. UCS* 14.9 (2008), pp. 1391–1410.

[Dav19] Dave. *Yahoo Finance Premium Review - Is It Worth Paying For?* Sept. 2019. URL: https://daytradereview.com/yahoo-finance-premium-review/.

[DE02] Tim Dwyer and Peter Eades. "Visualising a fund manager flow graph with columns and worms". In: *Proceedings Sixth International Conference on Information Visualisation*. IEEE. 2002, pp. 147–152.

[DML14] Maxime Dumas, Michael J McGuffin, and Victoria L Lemieux. "Financevis. net-a visual survey of financial data visualizations". In: *Poster Abstracts of IEEE Conference on Visualization*. Vol. 2. 2014, p. 8.

[Dwy03] Tim Dwyer. “A scalable method for visualising changes in portfolio data”. In: *Proceedings of the Asia-Pacific symposium on Information visualisation-Volume 24*. 2003, pp. 17–25.

[EBM07] Robert D Edwards, WHC Bassetti, and John Magee. *Technical analysis of stock trends*. CRC press, 2007.

[End+17] Alex Endert et al. “The state of the art in integrating machine learning into visual analytics”. In: *Computer Graphics Forum*. Vol. 36. 8. Wiley Online Library. 2017, pp. 458–486.

[FFM12] Fabian Fischer, Johannes Fuchs, and Florian Mansmann. “ClockMap: Enhancing Circular Treemaps with Temporal Glyphs for Time-Series Data.” In: *EuroVis (Short Papers)*. 2012.

[Fin20] FinViz. *FINVIZ.com - Stock Screener*. 2020. URL: https://finviz.com/.

[Fuc+13] Johannes Fuchs et al. “Evaluation of alternative glyph designs for time series data in a small multiple setting”. In: *Proceedings of the SIGCHI conference on human factors in computing systems*. 2013, pp. 3237–3246.

[Fuh19] Ryan Fuhrmann. *Yahoo! Finance vs. Google Finance: Which Is Better for Investors?* Apr. 2019. URL: https://www.investopedia.com/articles/investing/081314/yahoo-finance-vs-google-finance-which-should-you-use.asp.

[GLS17] Connor C. Gramazio, David H. Laidlaw, and Karen B. Schloss. “Colorgorical: creating discriminable and preferable color palettes for information visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* (2017).

[Goo20] Google. *Finance - Google Search*. 2020. URL: https://www.google.co.uk/finance.

[Gre+99] Donna L Gresh et al. “An interactive framework for visualizing foreign currency exchange options”. In: *Proceedings Visualization’99 (Cat. No. 99CB37067)*. IEEE. 1999, pp. 453–562.

[HB03] Mark Harrower and Cynthia A Brewer. “ColorBrewer. org: an online tool for selecting colour schemes for maps”. In: *The Cartographic Journal* 40.1 (2003), pp. 27–37.

[ID17] Vaishali Ingle and Sachin Deshmukh. “Live news streams extraction for visualization of stock market trends”. In: *Proceedings of the International Conference on Signal, Networks, Computing, and Systems*. Springer. 2017, pp. 297–301.

[JI13] Joel Joseph and Indratmo Indratmo. “Visualizing stock market data with self-organizing map”. In: *The Twenty-Sixth International FLAIRS Conference*. 2013.

[Ko+16] Sungahn Ko et al. “A survey on visual analysis approaches for financial data”. In: *Computer Graphics Forum*. Vol. 35. 3. Wiley Online Library. 2016, pp. 599–617.

[KY18] Zura Kakushadze and Willie Yu. “Stock Market Visualization”. In: *arXiv preprint arXiv:1802.05264* (2018).

[Lar10a] Robert S Laramee. “Bob’s concise coding conventions (c3)”. In: *Advances in Computer Science and Engineering (ACSE)* (2010).

[Lar10b] Robert S Laramee. *Bob’s Minutes of Meeting Protocol: Incentive and a Description*. Mar. 2010.

[Lar11] Robert S Laramee. “Bob’s project guidelines: Writing a dissertation for a BSc. in computer science”. In: *Innovation in Teaching and Learning in Information and Computer Sciences* 10.1 (2011), pp. 43–54.

[LZ10] Su Te Lei and Kang Zhang. “A visual analytics system for financial time-series data”. In: *Proceedings of the 3rd International Symposium on Visual Information Communication*. 2010, pp. 1–9.

[Ma09] Rui Ma. *Designing interactive visualization methods for comparing multivariate stock market data over time*. na, 2009.

[Mir+07] Barbara Mirel et al. “Visual analytics for model-based policy analysis: exploring rapid changes in commodities markets”. In: *Proceedings of the 8th annual international conference on Digital government research: bridging disciplines & domains*. 2007, pp. 312–313.

[ML17] Liam McNabb and Robert S Laramee. “Survey of Surveys (SoS)-Mapping The Landscape of Survey Papers in Information Visualization”. In: *Computer Graphics Forum*. Vol. 36. 3. Wiley Online Library. 2017, pp. 589–617.

[Mor20] Morningstar. *Morningstar Financial Research, Analysis, Data and News*. 2020. URL: https://www.morningstar.co.uk/uk/.

[MUS12] Andrew Morin, Jennifer Urban, and Piotr Sliz. “A quick guide to software licensing for the scientist-programmer”. In: *PLoS Comput Biol* 8.7 (2012), e1002598.

[Nai16] Suchismita Naik. “A Narrative Data Visualization Of The Indian Stock Market”. In: *Proceedings of the 8th Indian Conference on Human Computer Interaction*. 2016, pp. 162–177.

[NB04] Keith V Nesbitt and Stephen Barrass. “Finding trading patterns in stock market data”. In: *IEEE Computer Graphics and Applications* 24.5 (2004), pp. 45–55.

[Ozg+17] Ceyhun Ozgur et al. “MatLab vs. Python vs. R”. In: *Journal of Data Science* 15.3 (2017), pp. 355–372.

[PAN18] Aaron Pang, Craig Anslow, and James Noble. “What programming languages do developers use? a theory of static vs dynamic language choice”. In: *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE. 2018, pp. 239–247.

[PM17] Shyam Prabhakar and Larry Maves. “Big Data Analytics and Visualization: Finance”. In: *Big Data and Visual Analytics*. Springer, 2017, pp. 219–229.

[PS16] Anirudh Prabhu and Aravind Shenoy. “Introducing Materialize”. In: *Introducing Materialize*. Springer, 2016, pp. 1–9.

[Ref20] Refinitiv. *Financial Technology, Data, and Expertise*. 2020. URL: https://www.refinitiv.com/en.

[Rog09] Jakob Rogstadius. *Visualizing the Ethiopian Commodity Market*. 2009.

[Rup10] Nayan B Ruparelia. “Software development lifecycle models”. In: *ACM SIGSOFT Software Engineering Notes* 35.3 (2010), pp. 8–13.

[Sch+07] Tobias Schreck et al. “Trajectory-based visual analysis of large financial time series data”. In: *ACM SIGKDD Explorations Newsletter* 9.2 (2007), pp. 30–37.

[Sch+09] Tobias Schreck et al. “Visual cluster analysis of trajectory data with interactive kohonen maps”. In: *Information Visualization* 8.1 (2009), pp. 14–29.

[SCR16] Brian Steele, John Chandler, and Swarna Reddy. “Real-time Analytics”. In: *Algorithms for Data Science*. Springer, 2016, pp. 381–401.

[Shn94] Ben Shneiderman. “Dynamic queries for visual information seeking”. In: *IEEE software* 11.6 (1994), pp. 70–77.

[Sho+07] James Shore et al. *The Art of Agile Development: Pragmatic guide to agile software development*. ” O’Reilly Media, Inc.”, 2007.

[SP18a] Aravind Shenoy and Anirudh Prabhu. “Building a Landing Page with Skeleton”. In: *CSS Framework Alternatives*. Springer, 2018, pp. 15–40.

[SP18b] Aravind Shenoy and Anirudh Prabhu. “Building a Product Page with Milligram”. In: *CSS Framework Alternatives*. Springer, 2018, pp. 41–68.

[SP18c] Aravind Shenoy and Anirudh Prabhu. “Introducing UIkit”. In: *CSS Framework Alternatives*. Springer, 2018, pp. 69–106.

[Ste10] Stoyan Stefanov. *JavaScript Patterns: Build Better Applications with Coding and Design Patterns*. ” O’Reilly Media, Inc.”, 2010.

[Str95] Lisa Strausfeld. “Financial Viewpoints: using point-of-view to enable understanding of information”. In: *Conference companion on Human factors in computing systems*. 1995, pp. 208–209.

[VV06] Georg Von Krogh and Eric Von Hippel. “The promise of research on open source software”. In: *Management science* 52.7 (2006), pp. 975–983.

[War19] Colin Ware. *Information visualization: perception for design*. Morgan Kaufmann, 2019.

[Wat99] Martin Wattenberg. “Visualizing the stock market”. In: *CHI’99 extended abstracts on Human factors in computing systems*. 1999, pp. 188–189.

[Yah20a] Yahoo. *Yahoo Finance – stock market live, quotes, business finance news*. 2020. URL: `https://uk.finance.yahoo.com/`.

[Yah20b] Yahoo. *Yahoo Finance Premium*. 2020. URL: `https://finance.yahoo.com/premium-marketing/`.

[Yue+19] Xuanwu Yue et al. “sPortfolio: Stratified Visual Analysis of Stock Portfolios”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2019), pp. 601–610.

[Zha+16] Kaiyu Zhao et al. “MaVis: Machine Learning Aided Multi-Model Framework for Time Series Visual Analytics”. In: *Electronic Imaging* 2016.1 (2016), pp. 1–10.

[ZK11] Xiao-ping Zhang and David Kedmey. “TechWare: Financial Data and Analytic Resources [Best of the Web]”. In: *IEEE Signal Processing Magazine* 28.5 (2011), pp. 138–141.

[ZNK08] Hartmut Ziegler, Tilo Nietzschmann, and Daniel A Keim. “Visual analytics on the financial market: Pixel-based analysis and comparison of long-term investments”. In: *2008 12th International Conference Information Visualisation*. IEEE. 2008, pp. 287–295.