FinVis: Visualising the relationship between news content and numeric equity data.

Submitted April 2024 in partial fulfilment of
the conditions for the award of the degree **BSc Computer Science.**

20363018
School of Computer Science
University of Nottingham

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

Signature: TJM

Date 28/04/2024

Abstract

The current global economic landscape is one of unprecedented inflation, impacted by post-pandemic market dynamics and political instability. Personal finances are being strained throughout the developed world, with individuals increasingly motivated to maximise savings and investments. The impact of news content and sentiment on share price movements has been extensively researched, as illustrated by Atkins and Niranian [1]. They go on to state the elegance that would come from presenting the relationship between news content and share price. However, academic work such as Nguyen et al. [32] has predominantly focused on machine learning and developing tools that seek to accurately predict the share price movement from analysing news content. The results have limited accuracy and remain inaccessible for individual investors. Existing web applications targeting individual investors do little to help the user extract meaning from news content and sentiment, to inform investments. The likes of Morning Star [27] and Motley Fool [28] amongst others simply provide list views of news content, at best presented on the same view as share price. A clear gap exists in the market to visualise news content and sentiment directly within the context of share price movement over time. This visualisation project seeks to provide individual investors with the mechanism to better understand news content and sentiment to inform their investment decisions. This is a novel project that forms part of a portfolio of information sources that can be used to improve investment outcomes for individuals.

Acknowledgements

The Author would like to specifically thank Prof. Robert S Laramee for his continued support and encouragement through the project. Feedback was always carefully considered and genuinely constructive at every stage. Furthermore, the papers 'Bob's Project Guidelines' [23] and 'Bob's Concise Coding Conventions' [22] have been hugely appreciated.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Project Goal and Benefits . . . 1
1.3 Challenges . . . 2
1.4 Structure . . . 2

2 Background **2**
2.1 Related Work . . . 2
2.2 Previous Systems . . . 3
2.2.1 Morning Star . . . 3
2.2.2 Motley Fool . . . 4
2.2.3 Seeking Alpha . . . 4
2.2.4 Google Finance . . . 5
2.2.5 Yahoo Finance . . . 5
2.3 Data Characteristics . . . 5
2.3.1 NewsCatcherAPI . . . 6
2.3.2 Polygon . . . 6

3 Project Specification **7**
3.1 Feature Specification . . . 7
3.1.1 Required Features . . . 8
3.1.2 Optional Features . . . 9
3.2 Technology Choices . . . 9
3.2.1 Developmental Tools . . . 9
3.2.2 Libraries . . . 10
3.2.3 Project Management Tools . . . 10

4 Project Design **10**
4.1 Frontend Component Overview . . . 10
4.2 Backend Class Overview . . . 12
4.3 Process Diagrams . . . 12

5 Project Plan and Gantt Charts **14**
5.1 Project Management . . . 14
5.2 Gantt Charts . . . 14

6 Implementation **16**
6.1 Required Features . . . 16
6.1.1 Login . . . 17
6.1.2 Run Search . . . 18
6.1.3 Retrieve Data . . . 19
6.1.4 Visualise Data . . . 22
6.2 Optional Features . . . 23
6.2.1 Run Search . . . 24
6.2.2 Retrieve Data . . . 25
6.2.3 Visualise Data . . . 26
6.2.4 Visualisation Interaction . . . 27
6.2.5 Save Visualisation . . . 27

7 Testing and Evaluation **28**
7.1 Automated Testing . . . 28
7.1.1 News Data . . . 28
7.1.2 Stock Data . . . 30
7.2 Manual Testing . . . 30
7.2.1 Visual Testing . . . 30
7.3 Benchmark Testing . . . 32
7.3.1 Backend Benchmarks . . . 32
7.3.2 Frontend Benchmarks . . . 33
7.4 Evaluation . . . 34
7.4.1 Case Study 1 - NVDA stock price correction April 2024 . . . 34

7.4.2 Case Study 2 - TSLA Q4 earnings report . . . 36
7.4.3 Case Study 3 - Federal Reserve Inflation Comments . . . 36

8 Conclusions **37**
8.1 Reflection . . . 37
8.2 LSEPI . . . 37

9 Future Work **38**

References **39**

1 Introduction

1.1 Motivation

The COVID pandemic and the period since have seen significant global challenges. Individuals and households in most of the developed world have seen finances stretched, in what has been termed the cost-of-living crisis [33]. Rising inflation has left many individuals keen to protect and maximise their savings and investments. An increasing number of people are turning to equity investments to achieve greater returns on their savings. The population in the United States of America is a good example of this mindset, with one of the highest propensities to 'play' the stock market on a personal level. Recent research [18] shows that 61% of Americans have some sort of personal equity investment portfolio. This is the highest proportion of the population since the financial crisis in 2008.

For individuals active in equity investing, achieving consistent positive returns can be challenging. A raft of internal and external factors influence equity prices. One such group of factors relates to market sentiment, both generally and specifically, which is reflected in the news content associated with the specific equity. As Grob-Klubmann [14] states there is a direct correlation between market reaction and news content. The scale of the information and the complexity of the investment decision places significant demand on the individual investor who can lack the time and resources to effectively analyse the available information. Atkins and Niranian [1] state the value that exists in the relationship between share price and news data, as shown in the following quote.

> "...the very idea of relating information extracted from news feeds to market movement via the intermediate step of characterizing sentiments is an elegant one. Overall emotions, both at the level of groups of traders and at the level of wider society, are bound to influence the behaviour of financial markets." [1]

The vast majority of the products and services designed for individual investors focus on presenting historical financial data. There is a gap in applications that seek to make sense of the investment news component that is an indicator of market sentiment. Subsequently, individual investors, are at a disadvantage, which this project seeks to rectify. This project is principly focused on using information visualisation techniques to empower the human user to more easily access meaningful investment insight.

Visualisation techniques are needed to ensure that the individual investor can identify themes and extract meaning from the news content. Keywords and frequency of use, are one of the ways in which evaluation and analysis can be carried out with news content. However, for there to be greater value in investment decision-making, this keyword analysis needs to be set within the context of the share price. By relating patterns in keyword frequency, with historic share price, individual investors should be better placed to factor sentiment into their investment decisions.

1.2 Project Goal and Benefits

This project goal is to develop an application that will support individual investors in gaining insight into invesment news content and its relationship to share price movements in set date ranges.

The application will visualise the frequency of news articles which include selected companies' tickers against their historic share price movements. Multiple companies and their associated news content frequency and share price are displayed on a single graph for easy comparison. The frequency visualised is the number of articles published per day, within this each article will also be given a sentiment score to try and predict if it is positive or negative news. This will provide individual investors with a new perspective that can act as part of a range of sources of insight to inform better investment decisions.

The benefits of this project are:

- This project provides a novel web app that will visualise the financial and news data of specified companies within set date ranges.
- The application will have the advantage of being interactive, including filtering data, changing display options, zooming for clarity and exploring news data for more details on specific articles or groups of articles.
- Individual investors will be the main beneficiaries of the project as it will contribute to better investment decisions.

- The novelty of the project comes from the visualisation of news data, frequency and sentiment, within the context of historic share price movement.

1.3 Challenges

In order to achieve the material benefits of this project, as outlined above, some challenges need to be overcome. The first area is that the data required, both financial and news sits behind paywalls that can make it prohibitively expensive. Given the academic nature of this project, the key is to identify sources of data that offer access at a significantly reduced rate. A further challenge to the data is making it usable for the application; this will require substantial data processing, which in turn comes with the technical challenge of making a subsystem to handle it effectively. A particularly challenging addition to general data processing is sentiment analysis. Assigning a value that indicates a degree of positivity or negativity to each news article is difficult.

Visualising the data is a general challenge to most information visualisation projects and this is no different. This challenge can be segmented into the choice of visualisaiton techniques and technical realisation. To access the benefits to the individual investor, who will use the application, the data must be presented in an easy-to-interpret way. A choice must also be made regarding how the visualisations will be realised, such as what libraries should be used if any.

The technical choices made from the outset of the project are material in delivering the benefits and ultimately the project goal. A language best suited to the project must be chosen to make the development process as efficient as possible. The architecture of the project must also be considered in order to make the code base easy to work with, especially if the project is to benefit from future work. How the data flows around the application will need to be carefully considered, particularly how subsystems interact with each other. Abstraction and decomposition are required to overcome the challenge of complexity within the project.

1.4 Structure

The following is a breakdown of how the remainder of the dissertation is organised this follows closely with the guidance of Robert S Laramee [23]:

- Section 2: Explores the academic literature on associated topics and evaluates related systems, as well as data characteristics.
- Section 3: Includes a breakdown of the project features and looks at the technology chosen throughout the implementation of the project.
- Section 4: Covers the architecture of the project, in addition to a breakdown of the data processing.
- Section 5: Outlines the project management deployed throughout this project and the associated timescales.
- Section 6: Details the technical realisation of the software application, including methodologies used.
- Section 7: Presents the testing done on the visualisations, in particular, the data processing and evaluation of the project.
- Section 8: Summarises the dissertation and offers reflective thoughts on the project.
- Section 9: Offers consideration of the potential for future work for the project.

2 Background

2.1 Related Work

The complexity in equity investing is well documented, with various assertions on the degree of predictability. First cited by Louis Bachelier in 1900 as 'The Random Walk Theory' and later refined by Malkiel [25] in 'The Efficient Market Hypothesis', the view was presented that the market factors all information into the share price, completely and immediately. There have been challenges to the idea that the markets are quite so efficient. The overreaction of investors is supported by the work of Veronesi (1999) who is cited by Feng and Fu (2022), stating that investors overreact to bad news in good times and conversely underreact to good news in bad times [8]. While not entirely aligned, Chan [5] undertook research into the monthly share price returns of numerous companies and cross-referenced with news content, both positive and negative, to understand the relationship. The idea that individual investors over-rely on sentiment when investing was cited by Frankel [10], who states that "most investors underperform the market because they let their emotions get the best of them…" For those seeking to invest and make good decisions, an understanding of news content and the resulting sentiment needs to be factored in.

The challenges associated with extracting indications of sentiment from news content, to predict share price is a further area of study. Grob-Klubmann (2011) explored the concept of automated text analysis interpreting news content [14]. While the paper confirmed that markets respond to the news, it also highlights the challenges associated with the scale and volume produced by contemporary news outlets. This position is illustrated in the following quote.

> "Markets react sensitively to textual information updates—so-called "news"—which are announced on a regular and irregular basis. However, due to the enormous amount of news continuously released by modern electronic communication media nowadays it becomes increasingly difficult to process all news related to a certain financial asset." [14]

Research into the relationship between news content, sentiment and share price, as outlined above, is extensive. Building on this study has sought to explore the effectiveness of using machine learning to interpret content, predict share price movement and inform investment decisions. Accuracy remains an issue, with a leading project by Nguyen et al. [32] delivering 54% accuracy in predicting overall market movement using Yahoo Finance messaging content. Extensive research has been undertaken; Gandmal and Kumar [11] examine 50 different research projects using a variety of methodologies to predict share price. This includes the Bayesian model, Fuzzy classifier, Artificial Neural Networks, and Support Vector Machine classifier amongst others. While there is merit in some, there are limits to all and there is currently no workable model.

Academic work to better understand the relationship between news content and sentiment, and its effect on share price movements are well documented. However, finding an automated or machine-learning based solution that has sufficient accuracy to inform good investment decisions remains an issue. This further validates the potential that effective visualisaiton could provide in offering a workable means of improving understanding of news content to help inform individual investors' decisions. Given that the focus of this project is individual investors there is certainly no viable tool that exists to support an accessible means of understanding and interpreting news content when making investment decisions.

A further area of research in undertaking this project has been that of information visualisation. While the primary goal of the project is not to develop the academic theory surrounding information visualisation, it does draw upon the theory to optimise the application. As illustrated in both the Survey of Surveys by McNabb and Laramee (2017) [26] and A Survey of Information Visualisation Books by Rees and Laramee (2019) [37], the work on the topic is extensive. These comprehensive surveys have provided a plethora of resources to inform and enhance the visualisation techniques adopted within the project.

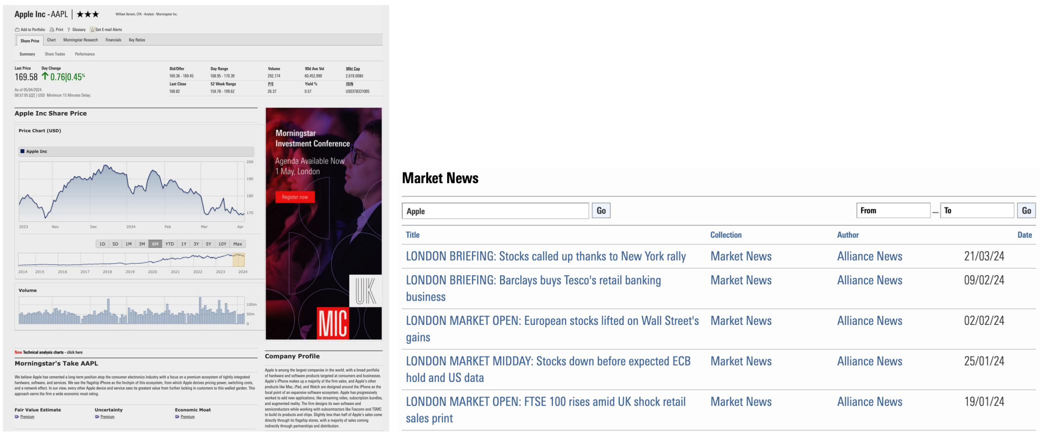
Visualisation of financial data is a widely considered topic as illustrated by Ko et al (2016), which explores a range of papers on the topic [21]. A key consideration is the use of line graphs or candle stick charts in the representation of different financial data sets. Interaction is an important aspect when it comes to visualising large amounts of data in order to abide by the quote "Overview first, filtering and selection and details-on-demand" as stated by Shneiderman (1996) [39]. Kiem (2002) [20] presents various important techniques to use when thinking about interaction in terms of information visualisation. In particular, for this project, interactive filtering and interactive zooming are both key to exploring subsets in further detail. Interactive techniques are relevant in managing large datasets such as financial and news data.

Liu et al (2022) [24] gather and review various tools in the space of visualisation. Colour is an important component of visualisation and this paper identifies and examines tools to support effective use of colour. This includes D3 Color Mapping [3] and Color Brewer [4]; both of which are offered to support effective use of colour. A further area of related work that has been explored is that of the use of glyphs in visualisation. As outlined by Borgo et al (2013) in [2] (identified from [26]), glyphs are an important visualisation technique when it comes to multivariate data as is the case in this project. The paper explores theories and application of glyph based visualisaiton techniques.

2.2 Previous Systems

2.2.1 Morning Star

Morning Star [27] is a web application that can be accessed by most modern devices on a web browser. While the site is promoted to both individual and institutional investors, it has a premium position with the market, due to its large paywall of up to $35 a month, with a 7-day free trial. The site has in-depth financial data and allows the user to access the majority of it through various visualisations, the rest being text-based. Presenting news data to the user is Morning Stars' biggest downfall; there is no directly linked news on the share view page and navigating to the page to find news content is difficult. Searching on the news pages displays a list of all articles related to the query showing the title, category, source and date. Financial reporting and advice is a key service offering of the site; given the costs, it is targeted more at serious individual investors.


Market News

Title	Collection	Author	Date
LONDON BRIEFING: Stocks called up thanks to New York rally	Market News	Alliance News	21/03/24
LONDON BRIEFING: Barclays buys Tesco's retail banking business	Market News	Alliance News	09/02/24
LONDON MARKET OPEN: European stocks lifted on Wall Street's gains	Market News	Alliance News	02/02/24
LONDON MARKET MIDDAY: Stocks down before expected ECB hold and US data	Market News	Alliance News	25/01/24
LONDON MARKET OPEN: FTSE 100 rises amid UK shock retail sales print	Market News	Alliance News	19/01/24

Figure 1: A screenshot of AAPL on Morning Star [27], as well as a screenshot of the news page

2.2.2 Motley Fool

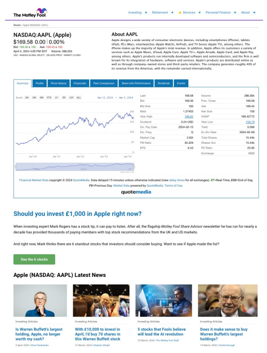


Figure 2: A screenshot of AAPL on Motley Fool [28]

Motley Fool [28] targets itself at individual investors and provides both free and premium content. Financial data on specific stocks is available within the free tier, as well as general news content. There is an attempt to link the news with individual stocks; however, the results showed a low level of correlation. The premium tiers start at £149 per year and go up to £999 per year. These services are investment advice targeted at individual investors with portfolios in excess of £25,000. A 30-day subscription refund guarantee is available on all premium tiers. Again, this is a web-based application that can be accessed by most modern devices on a web browser. It visualises its financial data in line graphs, candlestick graphs, bar charts and pie charts; however, each visualisation is used for specific data and is prescribed to the user.

2.2.3 Seeking Alpha

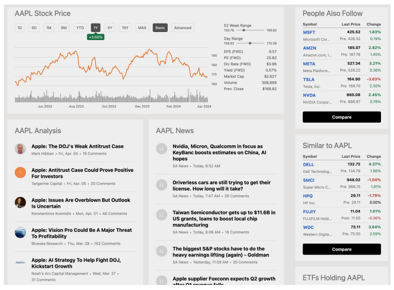
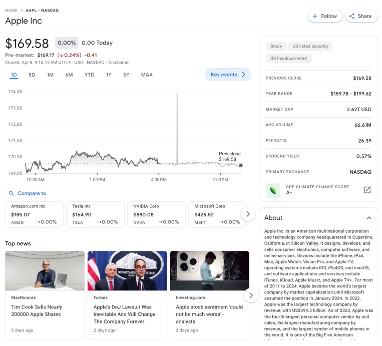


Figure 3: A screenshot of AAPL on Seeking Alpha [38]

Seeking Alpha [38], as with the other examples, is an investment site targeted at individual investors. This site has a greater emphasis on presenting aligned news content relevant to particular stocks. While it promotes the quality of the news that it accesses, it makes no effort to visualise the content to bring greater depth or meaning to the user. A two-tier model is employed with limited financial data and news content available for free and greater financial analysis available at a yearly cost of $239. The premium tier is offered with a trial month of around $5. Both tiers are accessed on a modern browser as a web-based application.

2.2.4 Google Finance



HOME > AAPL • NASDAQ

Apple Inc

$169.58 0.00% 0.00 Today

Pre-market: $169.17 (↓0.24%) -0.41

Closed: Apr 8, 9:16:12 AM UTC-4 · USD · NASDAQ · Disclaimer

PREVIOUS CLOSE	$169.58
YEAR RANGE	$159.78 - $199.62
MARKET CAP	2.62T USD
AVG VOLUME	66.61M
P/E RATIO	26.39
DIVIDEND YIELD	0.57%
PRIMARY EXCHANGE	NASDAQ
CDP CLIMATE CHANGE SCORE	A-

Amazon.com Inc	Tesla Inc	NVIDIA Corp	Microsoft Corp
$185.07	$164.90	$880.08	$425.52
AMZN →0.00%	TSLA →0.00%	NVDA →0.00%	MSFT →0.00%

Top news

MacRumors — Tim Cook Sells Nearly 200000 Apple Shares — 2 days ago

Forbes — Apple's DoJ Lawsuit Was Inevitable And Will Change The Company Forever — 2 days ago

Investing.com — Apple stock sentiment 'could not be much worse' - analysts — 3 days ago

About

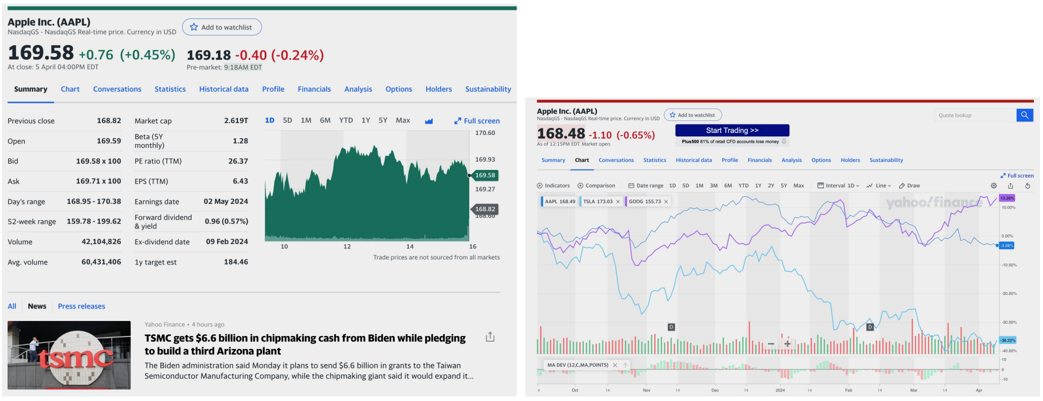
Apple Inc. is an American multinational corporation and technology company headquartered in Cupertino, California, in Silicon Valley. It designs, develops, and sells consumer electronics, computer software, and online services. Devices include the iPhone, iPad, Mac, Apple Watch, Vision Pro, and Apple TV; operating systems include iOS, iPadOS, and macOS; and software applications and services include iTunes, iCloud, Apple Music, and Apple TV+. For most of 2011 to 2024, Apple became the world's largest company by market capitalization until Microsoft assumed the position in January 2024. In 2022, Apple was the largest technology company by revenue, with US$394.3 billion. As of 2023, Apple was the fourth-largest personal computer vendor by unit sales, the largest manufacturing company by revenue, and the largest vendor of mobile phones in the world. It is one of the Big Five American

Figure 4: A screenshot of AAPL on Google Finance [13]

Google Finance [13] is provided as part of the portfolio of products offered by Google. It is a free web-based application accessible by modern browsers, combining simple financial data on a wide range of specific stocks and comprehensive aligned news content from a plethora of sources. There is no premium subscription or associated enhanced service offering. Google Finance is targeted at all investors or those with an investment interest. There are only two visualisations on each stock: the line chart showing share price over time and bar charts illustrating financial statements.

2.2.5 Yahoo Finance

Yahoo Finance [43] is another free web-based application. The site provides relevant news content but again only visualises the financial data. The level of functionality and interaction available with the financial data is extensive; this includes comparisons of multiple different stocks and indexes, the addition of indicators, choice of graph type and the ability to draw onto the graph. Again, due to the free nature of the software it is targeted at all investors. The level of sophistication possible with financial data visualisation means that this site could appeal to advanced individual investors opperating independently. However, the ability to simplify the visualisations keeps the audience broad.



Apple Inc. (AAPL)

NasdaqGS - NasdaqGS Real-time price. Currency in USD

169.58 +0.76 (+0.45%) At close: 5 April 04:00PM EDT

169.18 -0.40 (-0.24%) Pre-market: 9:18AM EDT

Previous close	168.82	Market cap	2.619T
Open	169.59	Beta (5Y monthly)	1.28
Bid	169.58 x 100	PE ratio (TTM)	26.37
Ask	169.71 x 100	EPS (TTM)	6.43
Day's range	168.95 - 170.38	Earnings date	02 May 2024
52-week range	159.78 - 199.62	Forward dividend & yield	0.96 (0.57%)
Volume	42,104,826	Ex-dividend date	09 Feb 2024
Avg. volume	60,431,406	1y target est	184.46

Trade prices are not sourced from all markets

Yahoo Finance • 4 hours ago

TSMC gets $6.6 billion in chipmaking cash from Biden while pledging to build a third Arizona plant

The Biden administration said Monday it plans to send $6.6 billion in grants to the Taiwan Semiconductor Manufacturing Company, while the chipmaking giant said it would expand it...

Apple Inc. (AAPL)

168.48 -1.10 (-0.65%)

Figure 5: A screenshot of AAPL on Yahoo Finance [43] and a screenshot of AAPL, GOOG and TSLA on Yahoo Finance Chart [43]

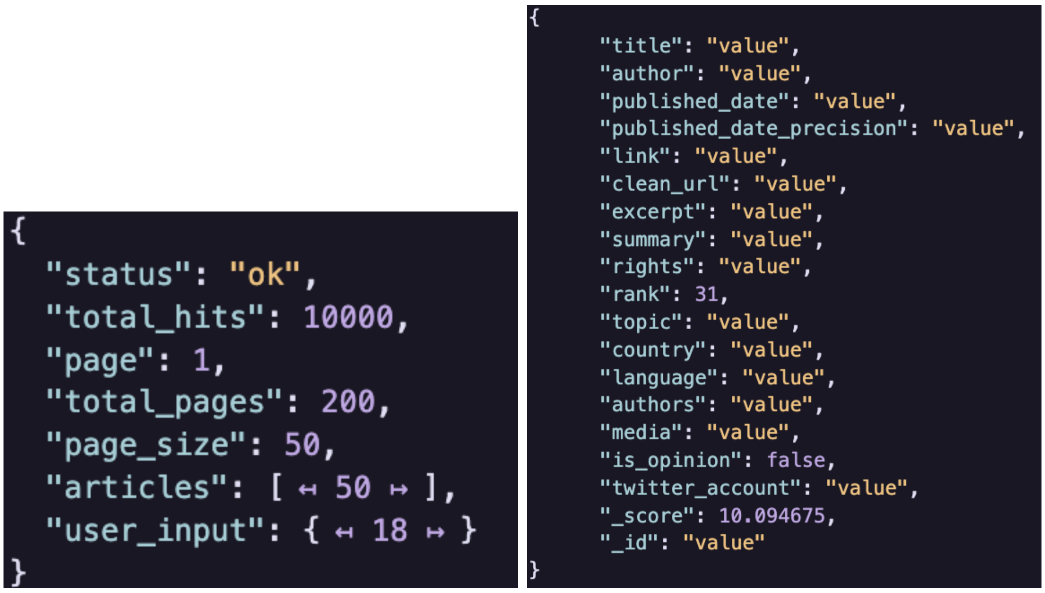
2.3 Data Characteristics

Within the final visualisation application, two key data sets will be needed: share price data of multiple stocks and news data related to those stocks. Due to the real-time and historic nature of the data required, an API is the best solution to this problem, as downloading historic data sets would prevent the real-time aspect.

The key requirements of the APIs are to contain historical data, to allow the visualisation to reflect an appropriate time range, and to allow for multiple stocks and news articles to be provided. Finally, the APIs need to be relatively cheap or ideally free due to the academic nature of this project. For news content, originally NewsAPI [30] was selected. However, later on in this project the restriction of 1-month historic data became a bigger drawback than initially anticipated, which led to the discovery of NewsCatcherAPI [31]. Requesting an educational discount NewsCatcherAPI [31] allowed premium access for this academic project, in turn granting historic data up to 6 years.

2.3.1 NewsCatcherAPI

NewsCatcherAPI [31] provides a REST-API that allows the developer to send a GET request to a specified URL along with their private key for authentication and query-related parameters in the URL to retrieve specified data. The specified data will be returned in JSON format, which allows for easy access and manipulation in JavaScript, which is the main language that will be used in this project. In the educational tier there is a size limit of 100 articles per request, meaning each of the responses will return 100 samples of data.



```
{
  "status": "ok",
  "total_hits": 10000,
  "page": 1,
  "total_pages": 200,
  "page_size": 50,
  "articles": [ ↤ 50 ↦ ],
  "user_input": { ↤ 18 ↦ }
}
```

```
{
      "title": "value",
      "author": "value",
      "published_date": "value",
      "published_date_precision": "value",
      "link": "value",
      "clean_url": "value",
      "excerpt": "value",
      "summary": "value",
      "rights": "value",
      "rank": 31,
      "topic": "value",
      "country": "value",
      "language": "value",
      "authors": "value",
      "media": "value",
      "is_opinion": false,
      "twitter_account": "value",
      "_score": 10.094675,
      "_id": "value"
}
```

Figure 6: News Catcher API sample JSON return structure [31] and News Catcher API sample article within JSON return with strings changed to ”value” to reduce space [31]

The above figures 6 show the formatting of the response from a GET request from the NewsCatcherAPI [31]; the examples shown above are requested as follows:

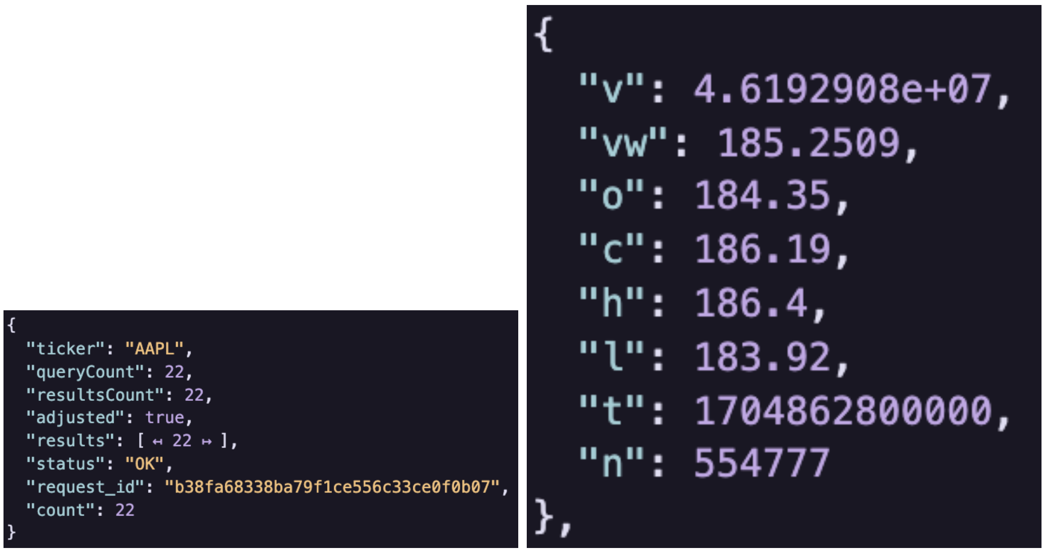
`GET: https://api.newscatcherapi.com/v2/search?q=apple&from=2020-11-12&to=2023-12-12&lang=en`

This can be broken down into sections to better understand how it works. The first part of the URL is NewsCatcherAPI’s domain where all of their API endpoints are stored. The URL then states the version of the API being requested, in this case ”v2”; after which the search is then specified using ”q=...”. This is where the user puts the query about the articles they would like to search for, in this case, apple. From and to dates are then specified to give time parameters for the searched articles and finally the language is selected, which in this case is ”en” for English.

2.3.2 Polygon

Polygon [34] is the chosen API for the financial data in this project. Like NewsCatcherAPI [31], it is a REST-API which also allows GET requests to be sent by developers to a URL to receive specified JSON formatted data. Unlike NewsCatcherAPI, there is no educational plan with this API; however, it does have a free tier. In the initial phases, the free tier providing 5 API calls per minute was sufficient for the demands of the project. However, in the latter stages, an upgrade to reduce the wait time between searches was required. This was because a feature enabling the viewing of multiple stocks was implemented increasing the calls per minute required by the application. Therefore the starter package was purchased which facilitated unlimited calls and reduced the time in between searches. The starter package costs $29 per month, which is relatively expensive compared to the free tier. However, this made a big difference when testing and presenting the product although not required for the core functionality.

The data used by this project mainly uses the free-tier restrictions and therefore does not have a reliance on the starter tier and its associated cost. Data requested still uses end-of-day data rather than the 15-minute delayed data available to the starter tier. The size of the data is dynamic but can be requested up to the previous two years which will return a limit of 730 samples of data per stock. Due to this being abstract financial data, the close price, which is what this project focuses on, has a minimum of 0 and no defined maximum value.



```
{
  "ticker": "AAPL",
  "queryCount": 22,
  "resultsCount": 22,
  "adjusted": true,
  "results": [ ↤ 22 ↦ ],
  "status": "OK",
  "request_id": "b38fa68338ba79f1ce556c33ce0f0b07",
  "count": 22
}
```

```
{
  "v": 4.6192908e+07,
  "vw": 185.2509,
  "o": 184.35,
  "c": 186.19,
  "h": 186.4,
  "l": 183.92,
  "t": 1704862800000,
  "n": 554777
},
```

Figure 7: Polygon sample JSON return for AAPL between 10th Jan 2024 and the 10th Feb 2024 [34] and a sample result

Figure 7 shows the results from a GET request to the polygon API. The following is an explanation of the returned keys in figure 7 as described in Polygon's documentation [34]:

- v: Trading volume
- vw: Volume weighted average price
- o: The open price
- c: The close price
- h: The highest price of the day
- l: The lowest price of the day
- t: The Unix Msec timestamp
- n: Number of transactions

3 Project Specification

3.1 Feature Specification

The primary goal of this project is to develop a web-based tool to concurrently visualize both news content and share price. Delivering on the project goal the specification can be segmented into smaller objectives. A key visualisation principle by Shneiderman (1996) [39] is ”Overview first, filtering and selection and details-on-demand”. This principle is key to the project as it will allow the user to see the overview of the data, filter the data to see specific subsets and then see the details of the data. This principle will be applied to the visualisation of the news content and share price.

The following is an outline of those smaller objectives:

- Visualise share price.
  This objective is not unique but is fundamental for the project, providing a comparative a contextual baseline for other aspects of the project.
- Quantify and visualise the sentiment of an article.
  Currently, where news content is incorporated with share price it is in a simple list format. The objective is to create a mechanism to define and contextualise news sentiment against share price movement.
- Visualse frequency of an article.
  A further objective to advance the current list view approach to news content is to visualise the number of articles published by day across specified date ranges again within the context of share prices.

- Comparison of multiple stocks.
  Currently, the most common view is single stocks; an objective of this project is to build a view of multiple stocks including both share price and news content.
- Enabling informed investments
  The current approach to presenting news content associated with share price makes the identification of sentiment laborious. The objective is to enable the user to draw meaning from news content as part of informed investment decisions.

The feature specification will deliver on the above objectives. To do this it is broken down into two sections: required features and optional features. The required features represent the minimum standard necessary to make the project useful and the optional features will provide more advanced functionality which separates this project from previous applications.

With the project adopting agile methodologies, there have been frequent iterations and prototypes. At each stage of the iterations, more requirements have come to light due to a greater understanding of the data and seeing which visualisaitons work and which do not, causing the requirements to vary since both the project proposal and interim report.

3.1.1 Required Features

Login

REQ-01 The homepage will have a log in button that will redirect the user to a Google login page.

REQ-02 The user will be able to log in using their Google account.

REQ-03 The system will allow the user to access the dashboard page once they have logged in.

REQ-04 The system will not allow the user to access the dashboard page if they are not logged in.

REQ-05 The system will allow the user to log out.

Run Search

REQ-06 The dashboard page will have a dropdown menu for the user to select a stock ticker.

REQ-07 The system will allow the user to click a button to run the search.

REQ-08 The system will send the selected ticker to the backend requesting both historic share price for that ticker and any news related to it.

Retrieve Data

REQ-09 The system will retrieve the stock data from the Polygon API [34].

REQ-10 The system will retrieve the news data from the NewsCatcherAPI [31].

REQ-11 The system will process the stock data into a format that can be used by the front-end.

REQ-12 The system will process the news data into a format that can be used by the front-end.

REQ-13 The system will return the processed stock data to the front-end.

REQ-14 The system will return the processed news data to the front-end.

REQ-15 The system will return an error message if the stock data cannot be retrieved.

REQ-16 The system will return an error message if the news data cannot be retrieved.

Visualise Data

REQ-17 The system will display the stock data as a line chart.

REQ-18 The system will display the news articles as a list.

3.1.2 Optional Features

Run Search

OPT-01 The dashboard page will have a selector for glyph size.

OPT-02 The dashboard page will have a dropdown for glyph type.

OPT-03 The dashboard page will have a dropdown for date range.

OPT-04 The dashboard page will have a dropdown for colour mapping domain.

OPT-05 The dashboard page will have a dropdown for stock tickers that allows for multiple to be selected.

Retrieve Data

OPT-06 The backend will handle multiple stock tickers.

OPT-07 The backend will add sentiment to each article.

Visualise Data

OPT-08 The system will display multiple stocks on the line chart.

OPT-09 The system will display glyphs onto each line on the chart.

OPT-10 The system will map size of the glyphs to the number of articles in each day.

OPT-11 The system will map colour of the glyphs to the associated average sentiment of the articles in each day.

OPT-12 The system will display the glyphs as a pie chart with each segment's colour mapped to its individual sentiment.

Visualisation Interaction

OPT-13 The system will allow the user to click on a glyph to see the articles associated with that day.

OPT-14 The system will allow the user to zoom in on the line chart.

OPT-15 The system will allow the user to hover over a circular glyph and view the number of articles, average sentiment and date.

OPT-16 The system will allow the user to hover over a pie glyph segment and view the article title and sentiment score.

Save Visualisation

OPT-17 The system will allow the user to click a button which will save the visualisation.

OPT-18 The system will store the data for the visualisation in a database.

OPT-19 The system will allow the user to view their saved visualisations.

OPT-20 The system will not show other users' saved visualisations.

3.2 Technology Choices

3.2.1 Developmental Tools

- **JavaScript, HTML, CSS:** There are a few reasons why using a web stack is best for this project. The first is that I have experience with web technologies, therefore I can work quickly and efficiently. The second reason is that there are many libraries and frameworks that can be used to help build this project. Finally, a web-based approach makes the application accessible to the greatest amount of people.

- **React:** React [35] is a web framework that makes building reactive user interfaces simpler. This project seeks to deliver sophisticated visualisations and will therefore require a reactive user interface. Furthermore, React Native provides the future potential for the project to be exported to mobile devices.

- **React Router:** React Router [36] was used to handle routing, due to the main framework used being React [35]. This is because React [35] is designed to create single-page applications, so React Router [36] makes it easier to convert it into a multi-page application.

- **D3.js:** With visualisations being a key part of this project, a library that would allow flexibility to create effective and interactive visualisations was needed. D3 [6] was selected as it offers this functionality and is well documented.
- **Firebase:** Firebase [9] is a cloud service that was used to handle the backend of this project. It allows easy integration of users and authentication without having to handle storing the data independently. It also allows the running of backend functions without having to set up a server, as well as easily integrate a database and host the web application.
- **Tailwind:** TailwindCSS [40] is a library that speeds up styling html elements. It does this by providing pre-built classes for general styling, which can be combined to create more complex styles. Saving you from having to write the CSS yourself.
- **DaisyUI:** DaisyUI [7] is a plugin for TailwindCSS [40]. It provides more complex pre-built classes that speed up styling even further as well as providing style guides for common components.
- **JSDoc**: JSDoc [19] is a tool that is used to document the project's code base. It allows comments to be written in the code that are then turned into a documentation website. This is important for future work as it provides a more comprehensive understanding of the code.

3.2.2 Libraries

- **multiselect-react-dropdown** multiselect-react-dropdown [29] is a library containing a react component that lets the user select multiple items in a dropdown list rather than the default HTML dropdown which only allows for one. Using this component saves time as a custom component does not need to be created to handle this functionality.
- **vader-sentiment** The vader sentiment [41] library allows the import of the vader-sentiment model into the backend code to generate a sentiment value for each article title. This avoids having to implement the model from scratch as well as providing a good alternative to the costly cloud-based solutions or training a machine learning algorithm.

3.2.3 Project Management Tools

- **GitLab:** GitLab [12] is a web-based Git repository manager that is used to store the code. It allows for version control, issue tracking, and tagging. Version control is important as it allows for tracking of changes to the code and the ability to revert to previous versions if needed. Issue tracking is important as GitLab allows issues to be viewed on a Kanban board, which makes it easier to keep track of what needs to be done. Tagging is important as it allows marking of certain commits as a milestone, which makes it easier to keep track of progress.
- **Agile Methodology:** An agile methodology has been used to manage this project. It is a good fit for this project as it asks for an iterative approach, which allows for the production of a number of prototypes as feedback is given to each one.
- **Sprints:** One-week sprints have been used throughout this project. This aligns well with the weekly meetings with the project supervisor, where a sprint review takes place, talking about the previous sprint and planning the next one.

4 Project Design

4.1 Frontend Component Overview

Due to this project being written using React [35], a component-oriented approach has been taken. Structuring in this way has close similarities to object-oriented programming. However, the methodology slightly changed from OOP with the focus being on interchangeable components rather than functionality-defined objects. This allows the code to be more maintainable, as well as making it easier to swap in and out components when needed.

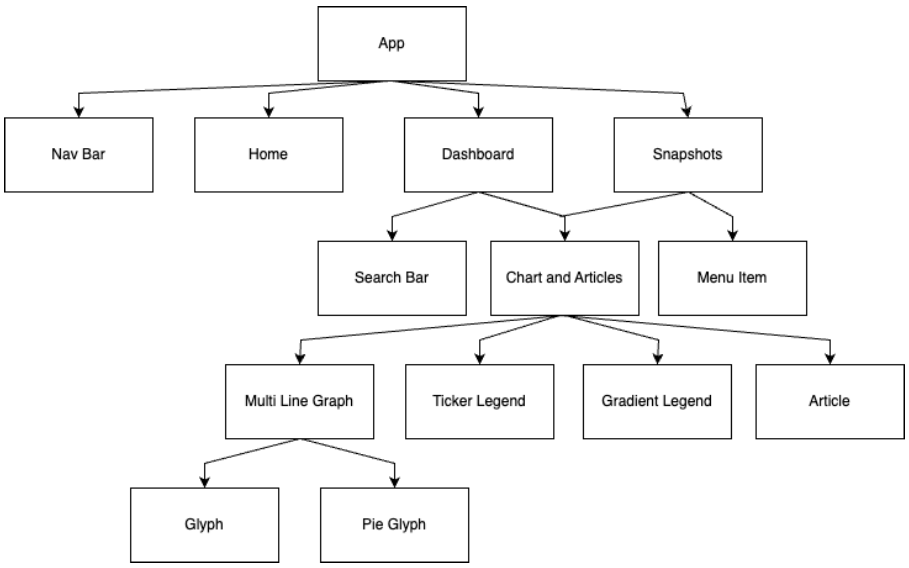


Figure 8: A component overview of the frontend of the project showing the heihrachy of components i.e which components are children of others

- **App:** The App component is the root component of the project. It contains the routing for the project and the navigation bar allowing the user to navigate easily between the different pages.
- **Nav Bar:** This component holds the navigation bar that is displayed at the top of the page. It contains the dropdown menu that allows the user to choose which page they want to go to, as well as the log in and log out buttons.
- **Home:** The home component contains a brief description of the project and an arrow pointing towards the login button if the user is not logged in. This is the only page that is accessible to the user if they are not logged in.
- **Dashboard:** The dashboard component is the main page of the project. It contains the search bar and the component which holds the visualisations as well as the articles. It also contains most of the "states" which control the flow of the project.
- **Snapshots:** The snapshots component is a page that allows the user to view their saved visualisations. It also contains the chart and articles components which display the visualisations and articles. However, it does not contain the searchbar and instead presents a list of saved visualisations.
- **SearchBar:** The search bar component holds all of the various select elements which control the data retrieved for the visualisation, as well as a few options for the visualisation itself. It also contains the button that runs the search.
- **Chart and Articles:** This component displays the visualisations and the articles. This is so it is easy to add both functionalities to different pages. It also holds the logic for which articles need to be displayed and which visualisation choices have been made.
- **Menu Item:** The menu item component is a simple component that purely handles the display of each item in the list of saved visualisations.
- **Multi Line Graph:** This component contains the D3 [6] code that creates the line graph. It also handles interaction with the graph such as zooming. This also handles the creation of the glyphs on the graph. The tooltips for each glyph are also created in this component.
- **Ticker Legend:** This component handles displaying the legend for each line on the graph.
- **Gradient Legend:** This component handles displaying the gradient line showing the sentiment of the articles.
- **Article:** This component displays each article, as well as containing a button that will take the user to said article.
- **Glyph:** This component handles the displaying of a days worth of articles as a glyph.

- **Pie Glyph:** This component handles the displaying of a days worth of articles; however each article is a segment of the pie chart.

4.2 Backend Class Overview

The backend of the project is handled by Firebase [9] and follows a more traditional OOP approach. This is because the backend is more focused on handling data and processing it, rather than displaying it. The backend is split into three main classes: one for each of the external APIs being used and one for handling the saving of snapshots. The classes are as follows:

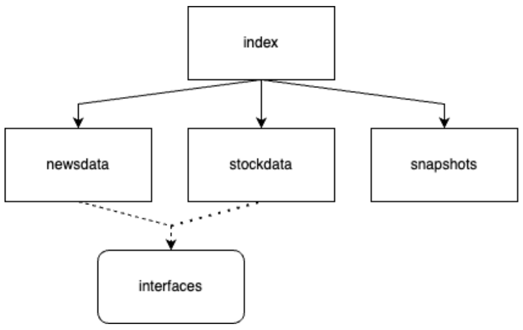


Figure 9: A class overview of the backend of the project showing the heihrachy of classes i.e which classes are children of others. The rounded rectangle represents a class that only contains interfaces and the rectangle represents a class that contains functions.

- **Index:** The index class is the main class of the backend. It exclusively handles exporting the functions from the other classes to Firebase [9].
- **News Data:** The news data class handles the API checks for calling the NewsCatcherAPI [31]. It retrieves the data from the API and formats the data, including the removal of data not needed by the application. Additionally, it also handles running sentiment analysis on the article titles.
- **Stock Data:** This class functions similarly to the news data class, however dealing with financial data. It handles API checks making sure the data is correct to be sent to Polygon [34]. It retrieves the data from the API and formats the data into a more usable format, including filling in missing days with estimated values.
- **Snapshots:** This class handles the saving of snapshots. It takes data from the front end and saves it to the Firebase [9] database. It also retrieves the data from the database and sends it back to the front end.
- **Interfaces:** On its own this class provides no functionality; however, it contains the interfaces that are used by the other classes. This is so that shared types can be used across the other classes.

4.3 Process Diagrams

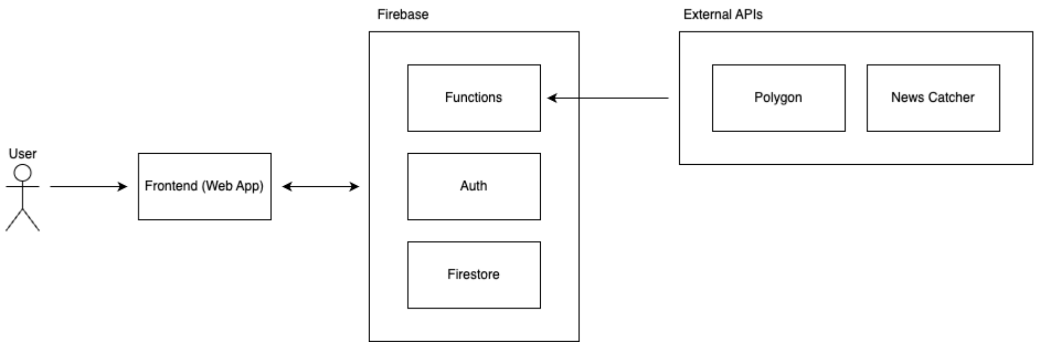


Figure 10: A high-level overview of the project showing how the largest sections of the project interact with each other.

As shown in figure 10, the project is split into three distinct sections: the front-end web app, Firebase [9], and the external APIs. The front-end web app provides the user interface and will be where the user engages with the application. The front-end will display the visualizations and allow the user to interact with them, such as selecting specific stocks.

Firebase [9] has been selected as it provides an effective backend for the application, supporting objectives. There are three services of Firebase [9] being used in this project. Authentication allows user verification, confirming they are who they say they are and managing the sign-in process. Firestore (NoSQL database) provides an easy and scalable database that integrates well with the other Firebase services. Functions remove the requirement of a dedicated server whilst still allowing backend code to be run.

The external APIs are used to retrieve live and historic data, both share price and news articles. Polygon [34] provides access to stock data for a particular company in a given range of dates. NewsCatcherAPI [31] is used to get the top 100 most popular news articles for each selected stock in a given date range that can be aligned to the stock data.

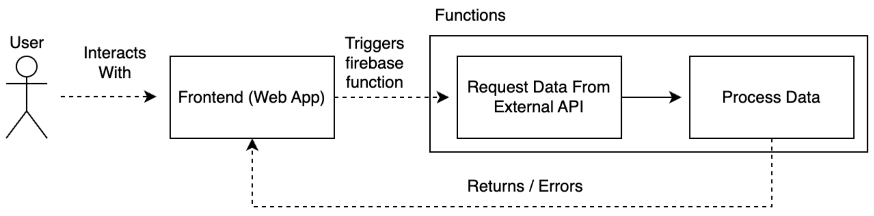


Figure 11: This is a high-level overview of the functions section showing the generalised flow of interaction and data between the user, frontend and internal functions.

The function section of the project is broken down further in figure 11. This section takes a request for data from the front-end, retrieves it from the external APIs, and then processes it before returning it to the front-end. This is so the front-end does not waste resources processing data into an easier format, as the user's device may not be powerful enough to do so.

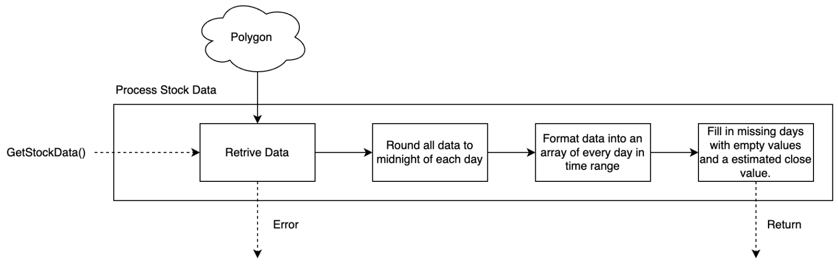

Polygon
Process Stock Data
GetStockData()
Retrive Data
Round all data to midnight of each day
Format data into an array of every day in time range
Fill in missing days with empty values and a estimated close value.
Error
Return

Figure 12: A more in-depth look into the stock data processing section of the functions.

The processing part of the functions can be again broken down into two parts; the first of which is outlined in figure 12. The process starts by retrieving the data from the Polygon API [34], then re-formats the data into an array of objects, one object for each day. It then fills in the missing days with the previous day's data before finally returning the data to the front end.

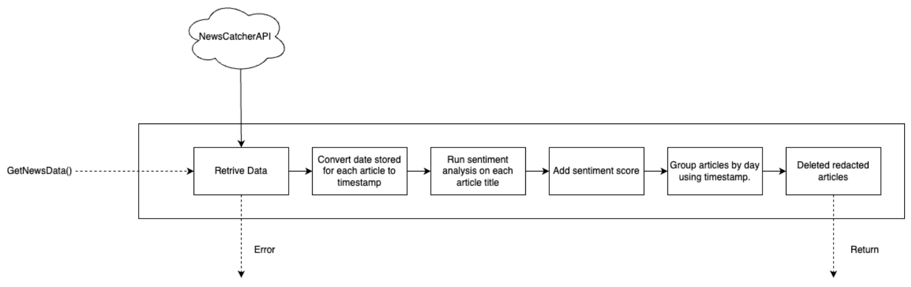

NewsCatcherAPI
GetNewsData()
Retrive Data
Convert date stored for each article to timestamp
Run sentiment analysis on each article title
Add sentiment score
Group articles by day using timestamp.
Deleted redacted articles
Error
Return

Figure 13: A more detailed overview of the news data processing section of the functions.

The second part of the processing overview is figure 13, which retrieves the data from the NewsCatcherAPI [31] then converts the date format to a timestamp to match the stock data. It groups the articles by timestamp to make it easier to visualize before running sentiment analysis on the article titles to get a sentiment value. Then it removes any redacted articles that may cause errors in the visualizations. Finally, it returns the data to the front end if there have been no errors along the way.

5 Project Plan and Gantt Charts

5.1 Project Management

Project management is a key aspect of any software engineering project. It is important to stay organised and keep track of progress when it comes to working on a large project. Many tools and methodologies have helped with this. The methodology that has been the biggest help is agile. The agile manifesto [15] states the following:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

These principles have been followed throughout the project where possible and meaning has been interpreted to suit the nature of this project.

Following these principles, the project has been broken into one-week sprints. This corresponds to the weekly meetings with the project supervisor. In these meetings, the previous week's work is discussed and the next week's work is planned, as well as reflection on the current iteration of the project to understand what is working and what is not.

To keep track of the project and to maintain version control, Gitlab [12] has been used. As well as just storing the code, GitLab allows for the tagging of commits, which has been used at every major milestone of the project. This allows for easy rollback to previous versions if needed and makes it easier to view the progress of the project. GitLab also has an issue tracker with an integrated Kanban board. This has been useful to break down the weekly tasks into smaller more manageable tasks and visualise the progress of the sprint.

More details of the tools used can be found in the technology choices section of this report. 3.2.3

5.2 Gantt Charts

As part of the initial project proposal, a Gantt chart was created with a breakdown of the key components of the project and estimated time frames. This was a helpful exercise and has formed a reference point in the regular reviews of progress. The original Gantt chart is shown in figure 14.

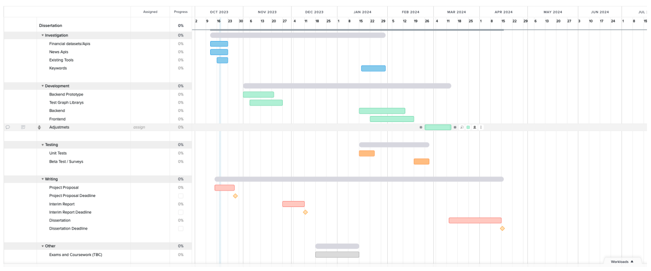


Figure 14: Initial outline of timeframes for the project created for the project proposal.

At the time of the interim report, a revised Gantt chart was created, figure 15, to reflect the progress made and the changes required. This was useful to understand the progress made and reflect on the causes of the changes. Some examples of these changes include the removal of the keyword research as this was replaced with a query-based approach. Time required on frontend development was increased, leading to a reduction in the time allocated to backend development. Increasing the time allowed for writing the final dissertation, due to the interim report taking longer than expected. Throughout this initial stage of the project, the project Gantt chart has been updated to reflect the progress made and the changes required.

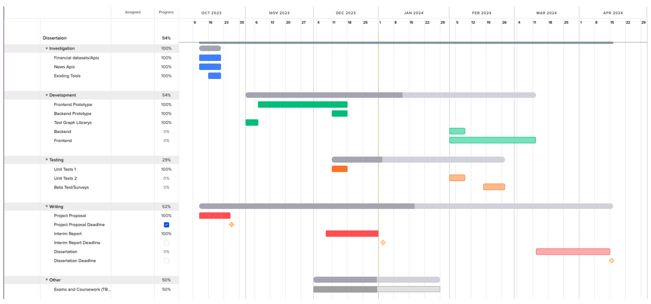


Figure 15: Revised Gantt chart showing the changes made to the timeframes during the writing of the interim report.

A few changes have been made to the timeframes since the interim report. The final Gantt chart is shown in figure 16. The biggest changes were to the timeframes for the backend and frontend development; this was mainly due to the addition of both the Pie Glyph and the Multi-Line Graph visualisations. Due to the complexity of these visualisations, a lot of work was required to implement them correctly and maintain previous functionality. An extension of the dissertation deadline was also allowed giving more time to write the final report. However, the overall estimation of time frames in the interim report was reasonably accurate.

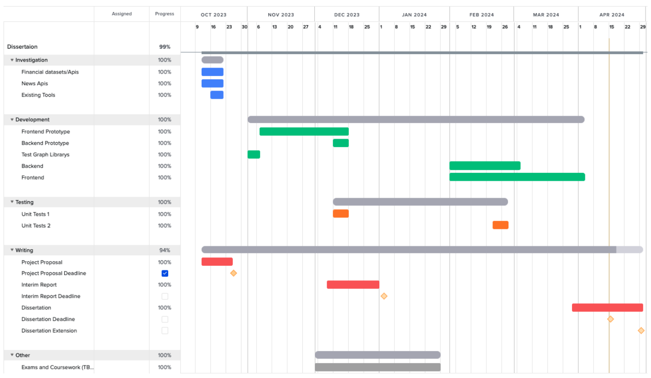
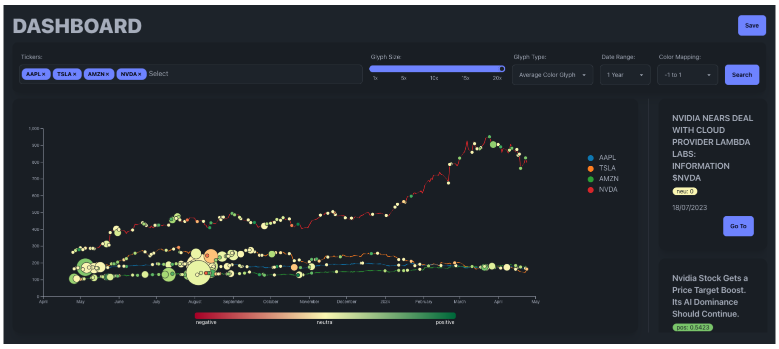


Figure 16: The final Gantt chart showing the timeframes used at the end of the project.

6 Implementation

The implementation phase provides a detailed overview of the development process that has been undertaken to create the final web application. This is broken down into two sections: required features and optional features. Throughout the development phase of this project, reference has been made to the guidance contained within Bob's Concise Coding Conventions[22] by R. S. Laramee. It should be noted that the conventions have been adapted to suit the language and framework, as this project has used JavaScript and React.

The implementation section provides clarity on establishing the web application outlined in figure 17. This screenshot provides an overview of the full functionality of the site with the exception of some of the interactive elements.



Tickers: AAPL × TSLA × AMZN × NVDA × Select

Glyph Size: 1x 5x 10x 15x 20x

Glyph Type: Average Color Glyph

Date Range: 1 Year

Color Mapping: -1 to 1

NVIDIA NEARS DEAL WITH CLOUD PROVIDER LAMBDA LABS: INFORMATION $NVDA

neu: 0

18/07/2023

Nvidia Stock Gets a Price Target Boost. Its AI Dominance Should Continue.

pos: 0.5423

Figure 17: The final version of the web application showing AAPL, TSLA, AMZN, NVDA from 25th Apr 2023 to 25th Apr 2024.

6.1 Required Features

The first phase of implementation focuses on the required features of the project outlined in section 3.1.1. Table 1 below shows the required features, their associated IDs and their current status. It is segmented into four sections: Login, Run Search, Retrieve Data, and Visualise Data. This structure will be used throughout this section to describe the implementation phase.

Required Features		
ID	**Description**	**Status**
	Login	
REQ-01	Log-in button on homepage	Completed
REQ-02	Log-in with Google	Completed
REQ-03	Access to dashboard on log-in	Completed
REQ-04	No access to dashboard without log-in	Completed
REQ-05	Log-out button	Completed
	Run Search	
REQ-06	Dropdown menu for stock ticker	Completed
REQ-07	Button to run search	Completed
REQ-08	Send selected ticker to backend	Completed
	Retrieve Data	
REQ-09	Retrieve stock data from Polygon	Completed
REQ-10	Retrieve news data from NewsCatcherAPI	Completed
REQ-11	Process stock data	Completed
REQ-12	Process news data	Completed
REQ-13	Return processed stock data	Completed
REQ-14	Return processed news data	Completed
REQ-15	Error message if stock data cannot be retrieved	Completed
REQ-16	Error message if news data cannot be retrieved	Completed
	Visualise Data	
REQ-17	Display stock data as a line chart	Completed
REQ-18	Display news articles as a list	Completed

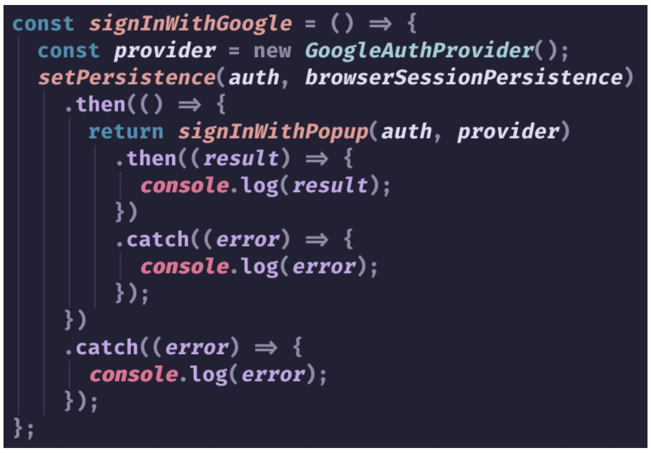
Table 1: A table showing the required features of the project and their current status.

6.1.1 Login

Having login functionality comes with a few benefits. The first being that it enables the addition of user-specific saved visualisations. The second is that, when combined with the blocking of the dashboard page without login, it provides a level of security, stopping bots from accessing the dashboard. This would cause strain on the backend functions and potentially cost money.

The login features have been implemented using Firebase [9] authentication. This made the process of developing this section more simple than it otherwise would have been. To do this, a firebase project was created and the Firebase [9] package was installed into the project.

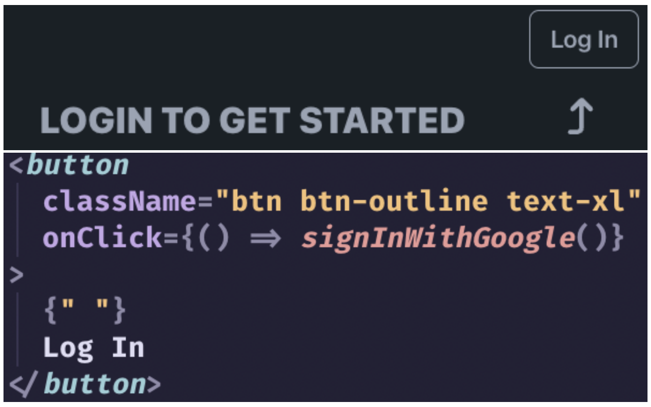
Once the Firebase [9] package was installed a sign-in function was than created that would open up the Google sign-in page, as well as correctly handling authentication. This function can be seen in figure 18.



```
const signInWithGoogle = () ⇒ {
  const provider = new GoogleAuthProvider();
  setPersistence(auth, browserSessionPersistence)
    .then(() ⇒ {
      return signInWithPopup(auth, provider)
        .then((result) ⇒ {
          console.log(result);
        })
        .catch((error) ⇒ {
          console.log(error);
        });
    })
    .catch((error) ⇒ {
      console.log(error);
    });
};
```

Figure 18: The login function that handles the Google sign-in process.

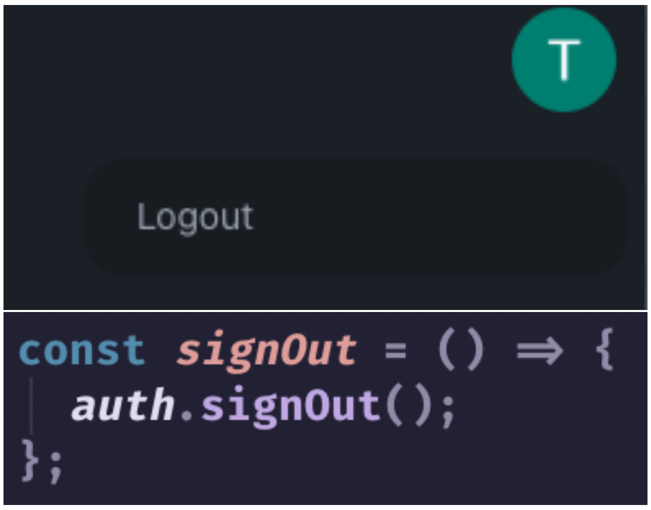
This was then added to a button on the Nav Bar component, shown in figure 19. Now the user is able to click a button and be taken to the Google sign-in page.



```
<button
  className="btn btn-outline text-xl"
  onClick={() ⇒ signInWithGoogle()}
>
  {" "}
  Log In
</button>
```

Figure 19: The login button that opens the Google sign-in page and the associated code.

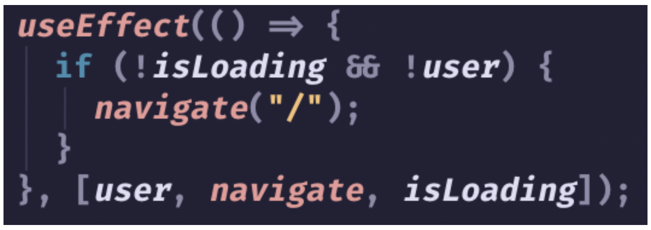
With the user now able to login, the next step was to change the login button to the user's google profile picture and add a way for the user to sign out.



```
const signOut = () => {
  auth.signOut();
};
```

Figure 20: The profile picture and logout button that replaces the login button and the code that handles the logout process.

The final step was to make sure that the user cannot access the dashboard page without being logged in. To do this a function was implemented that redirects the user to the homepage if they are not logged in; this function can be seen in figure 21. It has been implemented using the useEffect hook provided by React [35].



```
useEffect(() => {
  if (!isLoading && !user) {
    navigate("/");
  }
}, [user, navigate, isLoading]);
```

Figure 21: The function that redirects the user to the homepage if they are not logged in.

6.1.2 Run Search

The run search features are some of the most important features in the project, without them the user would not be able to request the data for the visualisations. To implement them, first a searchbar component was created. This is so the functionality of the search can be easily added to other pages if needed in future. The search bar originally contained a dropdown menu for the stock ticker and a button to run the search; however it was later updated to include more options for the user to select from (See 6.2).

SearchBar.jsx

```
const SearchBar = forwardRef((props,ref) =>
    const {handleSearch} = props
    const {tickerSelectRef} = ref
    ...
    return(
        <div className="searchbar">
            <select ref={tickerSelectRef}>
                <option>AAPL</option>
                <option>GOOGL</option>
                ...
            </select>
            <button onClick={() => handleSearch}> Run Search </button>
        </div>
    )
)
```

Dashboard.jsx

```
const Dashboard = () => {
    const [ticker, setTicker] = useState("AAPL")
    const [stockData, setStockData] = useState([])
    const [newsData, setNewsData] = useState([])
    const tickerSelectRef = useRef()
    ...
    const handleSearch = () => {
        setTicker(tickerSelectRef.current.value)
    }
    ...
    useEffect(() => {
        ...
        getMultipleStockData({ticker}).then((data) => {
            setStockData(data)
        })
        ...
        getMultipleNewsData({ticker}).then((data) => {
            setNewsData(data)
        })
        ...
    },[ticker])
    ...
    return(
        <div>
            <SearchBar ref={{tickerSelectRef}} handleSearch={handleSearch}/>
            ...
        </div>
    )
}
```

The above code shows how this functionality would be implemented if it had not been updated to include more options. The search bar component contains a dropdown menu for the user to select a ticker, which has a ”ref” passed to it from the parent component. There is also a button on the searchbar that, when clicked, runs the function passed to the component by the parent component.

In the parent component, Dashboard, the ticker is set to AAPL; this is done using the useState hook provided by React [35] and is used to store the ticker that the user has selected. useState is used to enable the value to be stored so it does not get lost when the component re-renders. The useState hook is also used to store the stock data and news data so it can be used later in the visualisations. A reference is created for the ticker dropdown menu so that the value can be accessed later. The handleSearch function is then created so that it can be passed into the search bar component and run when the button is clicked. The handleSearch function sets the ticker to the value selected in the dropdown. A useEffect hook is then used to run the requests to the backend using firebase functions to get the stock and news data which will be epanded on in 6.1.3. The useEffect hook waits for the ticker value to change and when it does, it calls the code inside the hook, therefore retrieving the data and saving it to the newsData and stockData values respectively.

6.1.3 Retrieve Data

Retrieving data is fundamental as it is the process in which we get the data needed for the visualisations. The way that this has been implemented is by using Firebase [9] functions to call the external APIs, as well as processing them. Using Firebase allows for the backend to be run without the need for a dedicated server or running the functions on the front-end. This is important as it allows for the data to be processed in a more powerful environment than the user’s computer in turn making the application more efficient. Additions will be made in section 6.2 to increase the functionality of these features.

The first thing required to enable the Firebase functions is to add the functions service to the Firebase project; this does require swapping from the free tier to the pay-as-you-go tier. However, in the context of this project, the costs are negligible and remain well within the credits provided by Google Cloud. As of writing this, the project has spent £0.01 on server time and the credits from Google Cloud allow up to £300.

The backend has been split up into four main classes; within this section the focus will be on the main class, Index, and the two classes that handle the external APIs, News Data and Stock Data.

Index.ts

```
const getStockData = require('./stockData')
exports.getStockData = getStockData.getStockData
...
const getNewsData = require('./newsData')
exports.getNewsData = getNewsData.getNewsData
...
```

The above code shows the index file for the functions; this file is used to get the function from the other classes and export them so that Firebase [9] can use them. This is done by requiring the other classes and then exporting the functions that return the processed and retrieved data.

stockData.ts

```
const isStockDataParams = (data: any){
    return(
        typeof data.ticker === "string" &&
        typeof data.from === "string" &&
        typeof data.to === "string"
    )
}
...
const mutateStockData = ({data, from, to}: any) => {
    const fromDate = new Date(from)
    const toDate = new Date(to)

    fromDate.setHours(0,0,0,0)
    toDate.setHours(0,0,0,0)

    const stockData = data

    const stockDataMap = stockData.reduce(
        (acc: any, item: {t: number, c:number}) => {
            const date = new Date(item.t)
            date.setHours(0,0,0,0)
            acc[date.getTime()] = item.c
            return acc
        }, {}
    )

    let lastNonNullClose = 0;

    const mutatedStockData = []
    for(let d = fromDate ; d <= toDate: d.setDate(d.getDate + 1)){
        const timestamp = d.getTime()

        mutatedStockData.push({
            t: timestamp,
            c: stockDataMap[timestamp] || null,
            ec: stockDataMap[timestamp] ? null : lastNonNullClose
        })

        if(stockDataMap[timestamp]){
            lastNonNullClose = stockDataMap[timestamp]
        }
    }

    return mutatedStockData
}
...
const getExternalStockData = async ({ticker, from, to}: StockDataParams) => {
    const url = "... + ticker + ... + from + ... + to + ..."
    try{
        const response = fetch(url)
        if(!response.ok) throw new Error(`Failed to retrieve data! status: ${response.status}`)
        const data = await response.json()
        return mutateStockData({data, from, to});
    }
}
...
exports.getStockData = onCall (
    {
        cors:  [...]
        region: "..."
    },
    async (request) => {
        if(!stockDataParams(request.data)) throw new Error("Invalid Parameters");
        const [ticker, from, to] = [request.data.ticker, request.data.from, request.data.to]
        return await getExternalStockData({ticker, from, to});
    }
)
```

The above code shows the stockData class. The first thing that happens when the getStockData function is called from Firebase [9] is the sent-in parameters are checked to see if they match what is expected. This is so when the inputs are passed to the external API they are guaranteed to be in the correct format, reducing the chance of errors. The request is then sent to the next function, getExternalStockData, which retrieves the data from the external API and returns it, after passing it to the mutateStockData function.

Inside the getExternalStockData function, the URL is created with the ticker, from, and to parameters. The URL is then fetched using the inbuilt JavaScript fetch function. The response is then checked to see if it is ok, if not an error is thrown. Assuming that the response is ok the data is the sent to the mutateStockData function alongside the from and to parameters, before being returned.

The mutateStockData function is used to process the data. The first thing it does is normalise the from and to values to midnight of the day. It then creates a map of the stock data with the timestamp ”t” being normalised to midnight, as well as getting the closing price ”c”. The map is then used to fill in any missing days with a new property ”ec” which is an estimated close price. In practice, this is the last close price before the missing data. This is used to make it easier for the line chart to draw the line without any gaps.

newsData.ts

```
...
const isNewsDataParams = (data: any){
    return(
        typeof data.ticker === "string" &&
        typeof data.from === "string" &&
        typeof data.to === "string"
    )
}

const mutateNewsData = (articles: Article[]) => {
    const articlesWithTimestamps = articles.map((article: Article) => {
        const articleDate = new Date(article.published_date)
        articleDate.setHours(0,0,0,0)
        return {...article, t: articleDate.getTime()}
    })

    const groupedArticles = articlesWithTimestamps.reduce((grouped: any, article: any) => {
        (grouped[article.t] = grouped[article.t] || []).push(article)
        return grouped;
    }, {})

    delete groupedArticles[0]

    return groupedArticles;
}

const getExternalNewsData = async ({ticker, from, to key}: NewsDataParams) => {
    const url = "... + ticker + ... + from + ... + to + ..."
    const options = {method: "GET", headers: {"x-api-key": key}}
    try{
        const response = await fetch(url, options);
        if(!response.ok) throw new Error(‘Failed to retrieve data! status: ${response.status}‘)
```

```
            const data = await response.json()
            return mutateNewsData(data.articles)
        }
        catch{
            throw new Error("Failed to retrieve data!")
        }

    }

    exports.getNewsData = onCall (
        {
            cors: [...]
            region: "..."
            secrets: ["NEWS_API_KEY"]
        },
        async (request) => {
            if (!isNewsDataParams(request.data)) throw new Error("Invalid Parameters");
            if(!process.env.NEWS_API_KEY) throw new Error("API Key not found");
            const key: string = process.env.NEWS_API_KEY
            const [ticker, from, to] = [request.data.ticker, request.data.from, request.data.to]
            return await getExternalNewsData({ticker, from, to, key});
        }
    )
```

Within the newsData class, there are some similarities with the stockData class. The getNewsData function is the entry function called by Firebase [9]. Like the stockData class, it first checks the parameters to make sure they are in the correct format. However, it then also checks that an API key is present in the env file. Storing the API key in the env file is good practice as it stops the API key from being hardcoded in, reducing the chances of it accidentally being published to a public repository. Once the parameters have been checked, the function then calls the getExternalNewsData function.

The getExternalNewsData function is almost identical to the getExternalStockData function. The only differences are that the URL is different and that options have to be specified including the API key. Once the data has been retrieved, it then sends the gathered data to the mutateNewsData function alongside the ticker.

The mutateNewsData function is the biggest difference from the stockData class as the two different types of data require different processing or "mutating". It first converts the published date of the article to a timestamp, alongside normalising it to midnight so the dates all align. It then groups the articles by day so that they can be easily visualised. Then index 0 is removed as it contains redacted articles.

6.1.4 Visualise Data

The visualisation of the data is done using D3 [6] and React [35]. The visualisations this section will focus on are the line chart and the list of articles. A line chart has been chosen for stock data as it will be the most familiar to the user and therefore is the most effective.

To create this visualisation, x and y scales have been created using the D3 [6] scaleLinear function, as well as the D3 scaleTime function. These scales are functions that take in a value and will map it against the domain and range specifiecd. The domain is the minimum and maximum values of the data and the range is the minimum and maximum values of the chart. The scales are then used to create the line, which is done using the D3 line function. A second line is also created to show the estimated close and fill in the gaps in the original line. To differentiate, a stroke is used to create a dashed effect. This function takes in the data and the x and y scales to produce a line that can be drawn on the chart. The x and y axis are then created using the D3 axisLeft and axisBottom. The resulting chart is shown in figure 22.

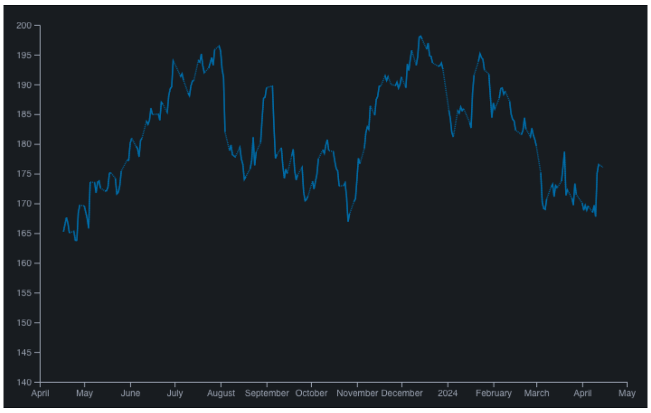
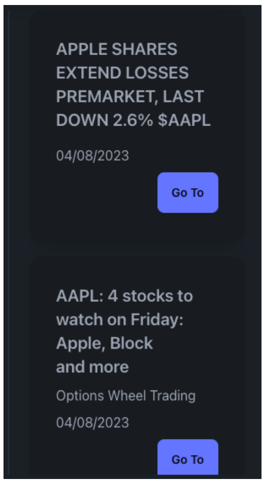


Figure 22: The line chart visualisation of AAPL from 2023-04-15 to 2024-04-15.

The list of articles was created by using the JS map function to loop through all the articles in the retrieved data and create a new Article component for each one. The Article component contains the title, date and a button to take the user to the article. An example of this can be seen in figure 23.



APPLE SHARES EXTEND LOSSES PREMARKET, LAST DOWN 2.6% $AAPL

04/08/2023

AAPL: 4 stocks to watch on Friday: Apple, Block and more

Options Wheel Trading

04/08/2023

Figure 23: 2 articles for AAPL published within 2023-04-15 and 2024-04-15.

6.2 Optional Features

The following section outlines the implementation of the optional features previously discussed in section 3.1.2. Table 2 shows the optional features and their current status. The optional features are split into three sections: Run Search, Visualise Data, and Save Visualisation.

Optional Features		
ID	**Description**	**Status**
	Run Search	
OPT-01	Add glyph size selection	Completed
OPT-02	Add glyph type selection	Completed
OPT-03	Add date range selection	Completed
OPT-04	Add colour mapping selection for sentiment	Completed
OPT-05	Add multiple stock selection	Completed
	Retrieve Data	
OPT-06	Handle multiple stock tickers	Completed
OPT-07	Add sentiment analysis to news data	Completed
	Visualise Data	
OPT-08	Add multiple lines to the line chart	Completed
OPT-09	Add glyphs to the line chart	Completed
OPT-10	Map size of glyphs to the number of articles each day	Completed
OPT-11	Add circular glyphs to the line chart	Completed
OPT-12	Add pie glyphs to the line chart	Completed
	Visualisation Interaction	
OPT-13	Show articles associated with glyph when clicked	Completed
OPT-14	Zooming on line chart	Completed
OPT-15	Circular glyph hover tooltip	Completed
OPT-16	Pie glyph hover tooltip	Completed
	Save Visualisation	
OPT-17	Add a save button to the dashboard	Completed
OPT-18	Save visualisation to Firebase	Completed
OPT-19	Add a page to view the saved visualisations	Completed
OPT-20	Make sure only the user can see their saved visualisations	Completed

Table 2: A table showing the optional features of the project and their current status.

6.2.1 Run Search

The run search features have been expanded upon from section 6.1. This is to allow the user to be able to control the new features added in the rest of section 6.2.

The features in the updated search bar build on the original search bar. They all get passed their own ”ref” which is used in the handleSearch function sent from the parent component, Dashboard, to the searchBar component. These refs are then used to get the values from the inputs and store them in state variables created by the useState hook. As before the state variables are then watched by the useEffect hook and when they change the requests are sent to the backend to get the data.

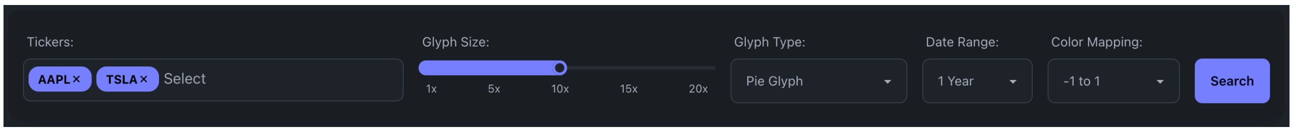
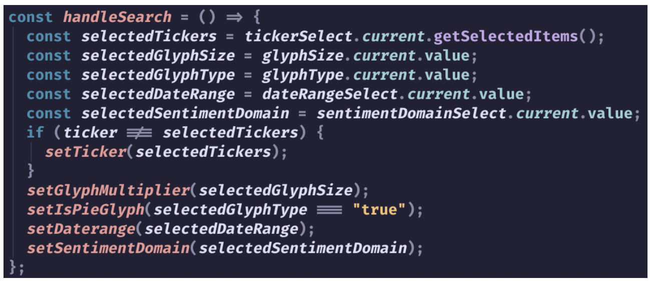


Figure 24: The up-to-date searchBar component with all optional features added.

The dropdown menu has been replaced with a multi-select dropwdown menu [29]; this allows the user to select more than one ticker without using a separate dropdown menu for each one. The state variable holding the ticker selection has been changed from a single string to an array to accommodate this.

To handle the new additions to the search bar, the handleSearch function has been updated and is shown in figure 25.



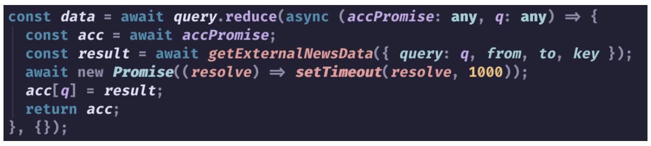
```
const handleSearch = () => {
  const selectedTickers = tickerSelect.current.getSelectedItems();
  const selectedGlyphSize = glyphSize.current.value;
  const selectedGlyphType = glyphType.current.value;
  const selectedDateRange = dateRangeSelect.current.value;
  const selectedSentimentDomain = sentimentDomainSelect.current.value;
  if (ticker !== selectedTickers) {
    setTicker(selectedTickers);
  }
  setGlyphMultiplier(selectedGlyphSize);
  setIsPieGlyph(selectedGlyphType === "true");
  setDaterange(selectedDateRange);
  setSentimentDomain(selectedSentimentDomain);
};
```

Figure 25: The updated handleSearch function that handles the new inputs from the search bar.

6.2.2 Retrieve Data

Slight changes have had to be made to the backend functions in order to allow for the multiple stock tickers to be handled.

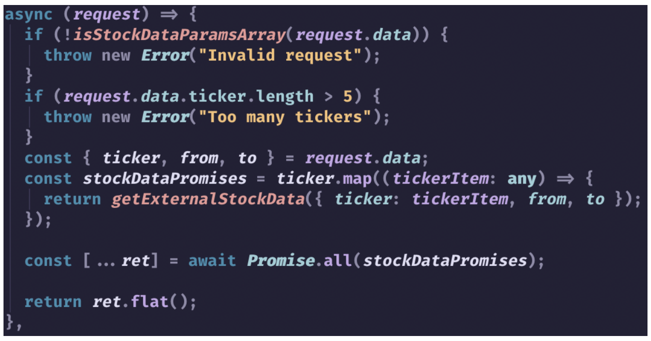
Within the entry function of the news data class, multiple tickers are now accepted. This is done, first by changing the parameters to accept an array of tickers instead of a single ticker. The function then loops through the tickers and calls the getExternalNewsData function for each one. A one second wait is added to prevent a 429 error (too many requests) from the NewsCatcherAPI [31], as it has a rate limit of 100 articles per second. The loop within this function is shown in figure 26.



```
const data = await query.reduce(async (accPromise: any, q: any) => {
  const acc = await accPromise;
  const result = await getExternalNewsData({ query: q, from, to, key });
  await new Promise((resolve) => setTimeout(resolve, 1000));
  acc[q] = result;
  return acc;
}, {});
```

Figure 26: The loop that handles multiple tickers in the newsData class.

A similar change to that made above is also made to the stock data class. Again, the parameter checks have been changed to accept an array of tickers; however a new check has also been introduced to make sure the array is no longer than 5 tickers. This is to allow the backend to use the free tier of the Polygon API [34], even though it currently uses the starter tier to reduce the wait time. The Polygon free-tier has a limit of 5 requests per minute, which in turn would require restriction on the frontend to prevent the user from searching more than once per minute. This is the main reason the starter tier has been purchased. The updated function can be seen in figure 27.



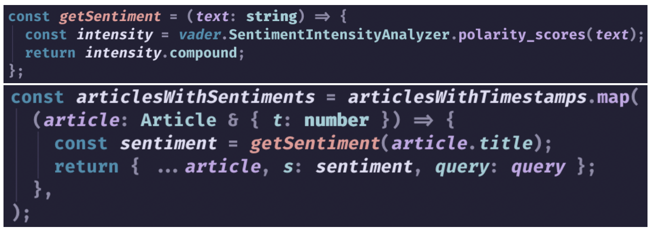
```
async (request) => {
  if (!isStockDataParamsArray(request.data)) {
    throw new Error("Invalid request");
  }
  if (request.data.ticker.length > 5) {
    throw new Error("Too many tickers");
  }
  const { ticker, from, to } = request.data;
  const stockDataPromises = ticker.map((tickerItem: any) => {
    return getExternalStockData({ ticker: tickerItem, from, to });
  });

  const [ ...ret] = await Promise.all(stockDataPromises);

  return ret.flat();
},
```

Figure 27: The updated getStockData function that handles multiple tickers.

The final change that needs to be made to the newsData class is to add in the sentiment analysis. This is done using the Vader Sentiment Analysis library [41]. The library is used to analyse the title of the article and return an estimated sentiment score. This score is then added to the article object and returned to the frontend. The code for this can be seen in figure 28.



```
const getSentiment = (text: string) => {
  const intensity = vader.SentimentIntensityAnalyzer.polarity_scores(text);
  return intensity.compound;
};
```

```
const articlesWithSentiments = articlesWithTimestamps.map(
  (article: Article & { t: number }) => {
    const sentiment = getSentiment(article.title);
    return { ...article, s: sentiment, query: query };
  },
);
```

Figure 28: The getSentiment function and the loop added to the mutateNewsData function.

6.2.3 Visualise Data

The visualisation aspect of the project has been greatly expanded up on in this section, starting with adding multiple lines to the line chart. This was done by first using the D3 [6] group function to group the data by its ticker. This is necessary for the data to be in a format that is handled well by D3. The data is then looped through, adding a line for each colour and changing the colour of the line to differentiate them. Finally, a legend is added to display the ticker alongside the associated colour of the line. The rest of the line chart code is negligibly different from the single line chart. An example of the multiple line chart can be seen in figure 29.

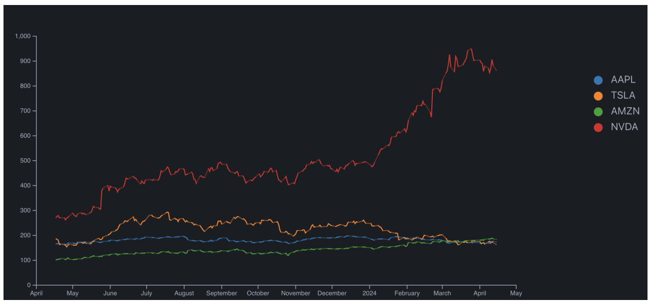


Figure 29: The line chart visualisation of AAPL, TSLA, AMZN and NVDA from 2023-04-16 to 2024-04-16.

Another addition to the visualisations is the glyphs. These glyphs are used to depict both the frequency of articles on a given day but also the sentiment of said articles. In the final project, there are two types of glyphs: circular glyphs, colour-mapped to the average sentiment of the articles, and pie glyphs, with segments mapped to the individual sentiment of the articles. The circular glyphs are created exclusively using the in-built svg circle element with the colour being set to the average sentiment of the articles and the colour scale being ColorBrewer's RdYlGn scale [4]. The radius is then set using the following formula:

```
radius =
    number of articles in the day ÷ maximum number of articles
    × width of a day ÷ 2 × user selected multiplier
```

After the glyph component has been created it is then added to the line chart by looping through all of the article groups and adding a glyph element to an array which is stored in a state variable. The state variable is attached to the return of the linechart component this way the glyphs are automatically rendered when the array is changed. An example of the line charts with circular glyphs can be seen in figure 30.

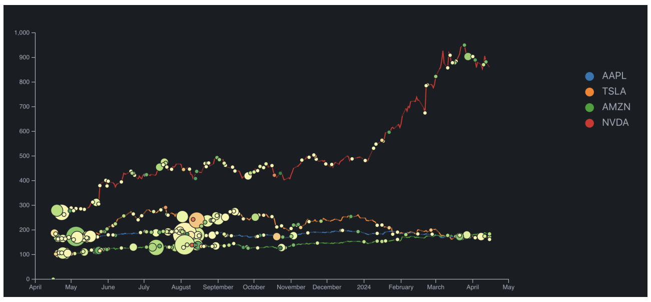


Figure 30: The line chart visualisation of AAPL, TSLA, AMZN and NVDA from 2023-04-16 to 2024-04-16 with circular glyphs at 20 times multiplier for better clarity.

The second type of glyph, the pie glyph, is built using the D3 [6] pie and arc functions. First, a filler array is created to ensure all of the segments of the pie are the same size. This is done by creating an array the same size as the number of articles in a day and filling it with 1s. The pie function is then called on the filler array and the arc function is called on the result of the pie function. The colour is then mapped using the same colour scale as the circular glyphs, and the same radius formula. However, instead of using the average sentiment for the colour, each segment's article sentiment is used. An example of a pie glyph can be seen in figure 31.

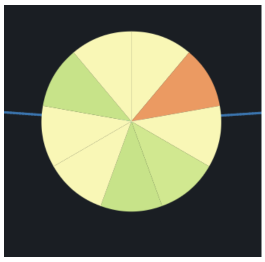


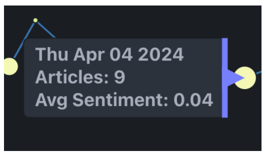
Figure 31: The pie glyph for AAPL on 2024-04-04.

6.2.4 Visualisation Interaction

There are 4 different interactions that have been added to the line chart: glyphs on click to show relevant articles along side the chart, zooming on the line chart and two different tooltips for the two glyph types. The first interaction, the glyph on click, is done by adding an onClick listener onto the glyph component. This listener then calls a function that sets a state variable to the articles associated with the glyph. The state variable is then used to display an article component for each article in the state variable array.

The zooming on the line chart is done using the brush function provided by D3 [6], alongside updating the x functions inputted domain to be that of what is selected by the brush. After which the linechart is rerendered showing the selected area. To go back to the original view of the chart, the user can double click resetting the x domain to the original value.

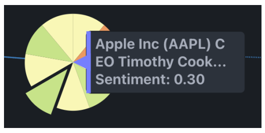
Tooltips have been created for both the circular and pie glyphs. The circular glyph tooltip is created by adding an onMouseOver listener to the circle element. A Tooltip is then created and displayed when hovering. This tooltip is removed by an onMouseLeave listener. The tooltip flips to the other side of the glyph depending on where the glyph is on the chart. This is done by working out the midpoint of the graph. An example of a circular glyph tooltip can be seen in figure 32.



Thu Apr 04 2024
Articles: 9
Avg Sentiment: 0.04

Figure 32: The circular glyph tooltip for AAPL on 2024-04-04.

The pie glyph tooltip is created using a lot of the functionality from the circular glyph tooltip. The only difference is the text. The text is created by splitting the article title by how many characters fit on a line. If the second line is too long, its last 3 characters are replaced with "...". Additionally, the specific article's sentiment score is displayed in place of the average sentiment. On top of showing the tooltip the, hovered over segment is also highlighted by moving it outwards slightly. This can be seen in figure 33



Apple Inc (AAPL) C
EO Timothy Cook...
Sentiment: 0.30

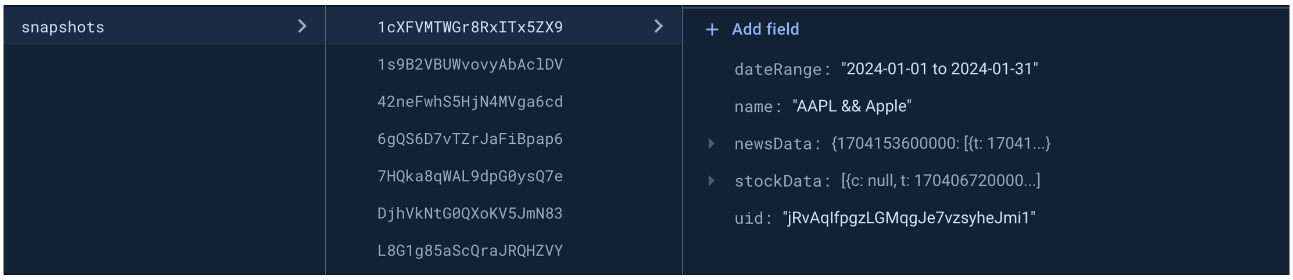
Figure 33: The pie glyph tooltip for AAPL on 2024-04-04.

6.2.5 Save Visualisation

Saving visualisations allows the user to save the selections they have made and revisit them at a later date. This is done by adding a save button to the dashboard page, which is linked to a function that pushes the currently loaded news and stock data to the Firebase [9] database. This data is then retrieved from Firebase and displayed on the snapshots page.

Within the backend, the snapshots class has been created to handle the saving and retrieving of the snapshots. Again this is done using Firebase [9] functions. The first entry point ”add” is used to add a snapshot to the database. First, the parameters are sent in to make sure everything required is present and is of the correct types. The data is then added to the snapshots collection in the database. The required fields are uid (the user id), name (the name of the snapshot), newsData, and stockData.

The second entry point list is used to retrieve all snapshots for a user. This is done by first checking a uid has been sent in via the request and that it is a string. The snapshots are then searched for in the database using the uid. Finally the snapshots are returned to the frontend. A screenshot of the database is shown in figure 34.


snapshots

1cXFVMTWGr8RxITx5ZX9
1s9B2VBUWvovyAbAclDV
42neFwhS5HjN4MVga6cd
6gQS6D7vTZrJaFiBpap6
7HQka8qWAL9dpG0ysQ7e
DjhVkNtG0QXoKV5JmN83
L8G1g85aScQraJRQHZVY

+ Add field
dateRange: "2024-01-01 to 2024-01-31"
name: "AAPL && Apple"
newsData: {1704153600000: [{t: 17041...}
stockData: [{c: null, t: 170406720000...]
uid: "jRvAqIfpgzLGMqgJe7vzsyheJmi1"

Figure 34: The Firebase database showing the snapshots collection.

7 Testing and Evaluation

Testing and debugging are key to having a successful project. This section will outline the testing that has been done on the project and the evaluation of the project as a whole. The testing of this project has been conducted using a combination of manual and automated testing. Throughout the project, testing at each stage has also been done to ensure each feature works as expected.

7.1 Automated Testing

The automated testing has been done on the backend to ensure each function works as expected. To do this the Jest testing framework [17] has been used. The tests are run using the npm test command.

Automated Tests	
Test	**Pass / Fail**
Mutate News Data	Pass
Mutate Stock Data	Pass

Table 3: A table showing the automated tests conducted and their results.

7.1.1 News Data

The first automated test is checking the mutateNewsData function to check that it processes the data correctly.

Input
The following is an array of articles from NewsCatcherAPI [31] with the following structure:

```
[
    {
        title: "Test Article",
        author: "Test Author",
        published_date: "2024-04-04T00:00:00Z",
        published_date_precision: "day",
        link: "https://test.com"
        clean_url: "test.com"
```

```
            excerpt: "This is a test article"
            summary: "This is a test article"
            rights: "test.com"
            rank: 1
            topic: "test"
            country: "test"
            language: "test"
            authors: "test"
            media: "test"
            is_opinion: false
            twitter_account: "test"
            _score: 1
            _id: "test"
        }
    ]
```

Expected Output
The expected output should be a JSON object with the following structure:

```
    {
        timestamp: [
            {
              ... Input Article,
              s: sentiment score
              t: timestamp
            },
            ...
        ]
    }
```

The output should be a JSON object with the timestamp as the key and an array of articles as the value. Each article should be exactly the same as in the input but with the sentiment score and timestamp added.

Output
The output of the test was as expected and the test passed. The following is an example of the output:

```
    {
        "1670150400000": [
            {
                title: "Test Article",
                author: "Test Author",
                published_date: "2024-04-04T00:00:00Z",
                published_date_precision: "day",
                link: "https://test.com"
                clean_url: "test.com"
                excerpt: "This is a test article"
                summary: "This is a test article"
                rights: "test.com"
                rank: 1
                topic: "test"
                country: "test"
                language: "test"
                authors: "test"
                media: "test"
                is_opinion: false
                twitter_account: "test"
                _score: 1
                _id: "test"
                t: 1670150400000
                s: 0
            }
        ]
    }
```

7.1.2 Stock Data

The second automated test is checking the mutate stock data function to check that it processes stock data correctly.

Input
An array of stock data from the Polygon API [34] with the following structure:

```
[
    {
                                "v": 4.6192908e+07,
                                "vw": 185.2509,
                                "o": 184.35,
                                "c": 186.19,
                                "h": 186.4,
                                "l": 183.92,
                                "t": 1704862800000,
                                "n": 554777
    }
]
```

Expected Output
The expected output should be an array of JSON objects with the following structure:

```
[
    {
        t: timestamp normalised to midnight,
        c: close price or null,
        ec: estimated close price or null
    }
]
```

The output should be an array of objects with a normalised timestamp, the close price, and the estimated close price. The close price or estimated close price can be null but not both. The estimated close price should be the last close price before the missing data.

Output
The output of the test was as expected and the test passed. The following is an example of the output:

```
[
    {
        t: 1704844800000,
        c: 186.19,
        ec: null
    }
]
```

7.2 Manual Testing

Manual testing has been done across the entire project to ensure that the user experience is as expected.

7.2.1 Visual Testing

Visual Tests	
Test	**Pass / Fail**
Check line chart displays correct data	Pass
Check all search bar options work	Pass
Check articles are correctly displayed on glyph click	Pass
Check average glyph tooltip displays correctly	Pass
Check pie glyph tooltip displays correctly	Pass

Table 4: A table showing the visual tests that have been conducted and their results.

The first test was to check that the line chart was displaying the correct data.

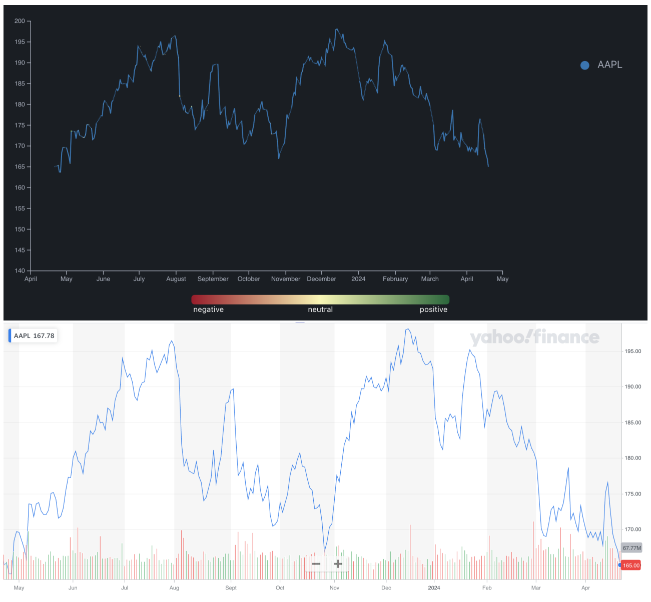
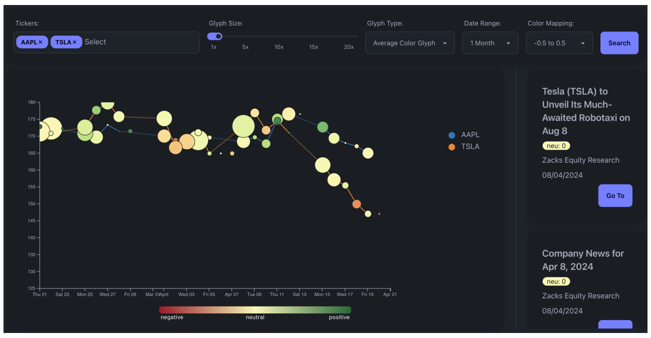


Figure 35: The line chart visualisation of AAPL from 2023-04-21 to 2024-04-21 on this projects visualisation and yahoo finance [43].

As shown in figure 35, the line chart displays the correct data. The data is the same as that on Yahoo Finance [43]. The only difference is the estimated close price, which is not displayed on Yahoo Finance [43] and this is expected.

After the line chart was confirmed to be displaying the correct data, the next test was to check that the search bar was working as expected. To check this each part of the search bar was set to a different value and the data was checked to see if it was correct.


Tickers: AAPL × TSLA × Select
Glyph Size: 1x 5x 10x 15x 20x
Glyph Type: Average Color Glyph
Date Range: 1 Month
Color Mapping: -0.5 to 0.5
Search

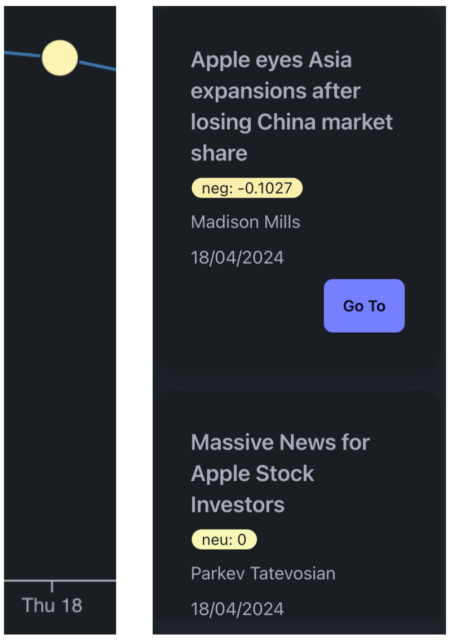
Tesla (TSLA) to Unveil Its Much-Awaited Robotaxi on Aug 8
neu: 0
Zacks Equity Research
08/04/2024
Go To

Company News for Apr 8, 2024
neu: 0
Zacks Equity Research
08/04/2024

Figure 36: Search bar test with the ticker set to AAPL and TSLA, the glyph size set to 2x, the glyph type set to the average colour the date range set to 2024-03-21 to 2024-04-21 and the colour mapping set to -0.5 to 0.5.

All the data was shown as expected and the test passed. The two lines were displayed on the chart with the correct colours. The correct type of glyphs were shown with the increased size and the correct colour mapping. The articles were also displayed correctly.

The next check is to see if the articles appear correctly when a glyph is clicked. This is done by clicking on a glyph to see if the articles appear and then checking that they are all from the correct date and related to the correct ticker.


Apple eyes Asia expansions after losing China market share

neg: -0.1027

Madison Mills

18/04/2024

Massive News for Apple Stock Investors

neu: 0

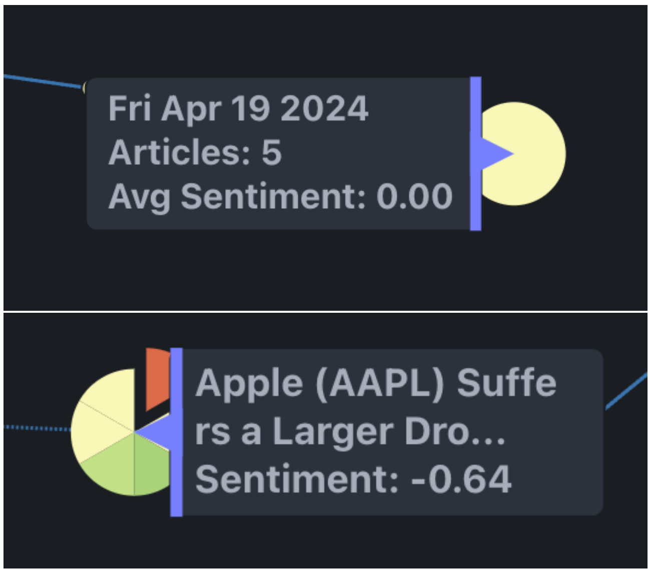
Parkev Tatevosian

18/04/2024

Figure 37: The glyph clicked and the associated articles displayed.

As shown in figure 37, the articles are displayed correctly and are all from the correct date and ticker.

The next two tests are to check that the tooltips are displayed correctly. The average sentiment glyph should have a tooltip with the date, the number of articles, and the average sentiment. The pie glyph should have a tooltip with the article title and the sentiment of the article.


Fri Apr 19 2024
Articles: 5
Avg Sentiment: 0.00

Apple (AAPL) Suffe
rs a Larger Dro...
Sentiment: -0.64

Figure 38: Both the average glyph tooltip and the pie glyph tooltip displayed.

Both tooltips correctly display the information required, with the data displayed matching both the data on the chart and the data contained within the articles and the state variables within the application.

7.3 Benchmark Testing

Benchmark testing has been conducted to compare the performance of the application, both backend and frontend. This is to better understand the how the application performs under differing number of stock tickers.

7.3.1 Backend Benchmarks

Backend Benchmarks				
No. Tickers	**From Date**	**To Date**	**Time**	**Size**
Stock Data				
1	2024-04-19	2024-04-19	120ms	66B
2	2024-04-19	2024-04-19	170ms	123B
5	2024-04-19	2024-04-19	143ms	291B
1	2024-03-19	2024-04-19	183ms	1830B
2	2024-03-19	2024-04-19	134ms	3.6KB
5	2024-03-19	2024-04-19	162ms	8.9KB
1	2023-10-19	2024-04-19	316ms	10.2KB
2	2023-10-19	2024-04-19	355ms	20.4KB
5	2023-10-19	2024-04-19	570ms	51.1KB
1	2023-04-19	2024-04-19	384ms	20.4KB
2	2023-04-19	2024-04-19	460ms	40.8KB
5	2023-04-19	2024-04-19	513ms	101.9KB
News Data				
1	2024-04-19	2024-04-19	1.24s	13.5KB
2	2024-04-19	2024-04-19	2.57s	31.6KB
5	2024-04-19	2024-04-19	6.25s	52KB
1	2024-03-19	2024-04-19	2.36s	318.2KB
2	2024-03-19	2024-04-19	4.65s	572.3KB
5	2024-03-19	2024-04-19	11.5s	1809.1KB
1	2023-10-19	2024-04-19	2.19s	187.3KB
2	2023-10-19	2024-04-19	4.64s	375.5KB
5	2023-10-19	2024-04-19	11.3s	1236.3KB
1	2023-04-19	2024-04-19	2.95s	211.6KB
2	2023-04-19	2024-04-19	5.11s	387.2KB
5	2023-04-19	2024-04-19	12.6s	1336.9KB

Table 5: A table showing the benchmarks done on the backend. The tickers used are APPL, TSLA, MSFT, GOOG and NVDA, with the 1 and 2 tickers being the first and first two tickers in the list respectively. The size is the size of the data returned from the backend. The time is the response time from the backend. This has been tested using Insomnia [16].

The results from the backend benchmarking are as expected. When it comes to the stock data tests, the size increased linearly with increasing date range and increasing number of tickers. The time taken also slightly increased; however, the biggest factor for this was the slight latency in the network connection.

News data was different, this is because there is a hardcoded time increase with the more tickers that are added, 1 second per ticker. Additionally, the size of the data is greatly affected by which articles are returned; this is more random so the size of the data is more sporadic than the stock data. The time taken increased more with the number of tickers than the date range, which should be the case due to the number of articles per ticker being the same no matter the date range.

7.3.2 Frontend Benchmarks

Frontend Benchmarks		
No. Tickers	**Date Range**	**Time**
1	1 Months	2.1s
1	3 Months	2.83s
1	6 Months	3.24s
1	1 Year	2.88s
2	1 Months	5.67s
2	3 Months	5.61s
2	6 Months	5.74s
2	1 Year	6.42s
5	1 Months	12.49s
5	3 Months	13.75s
5	6 Months	13.68s
5	1 Year	12.97s

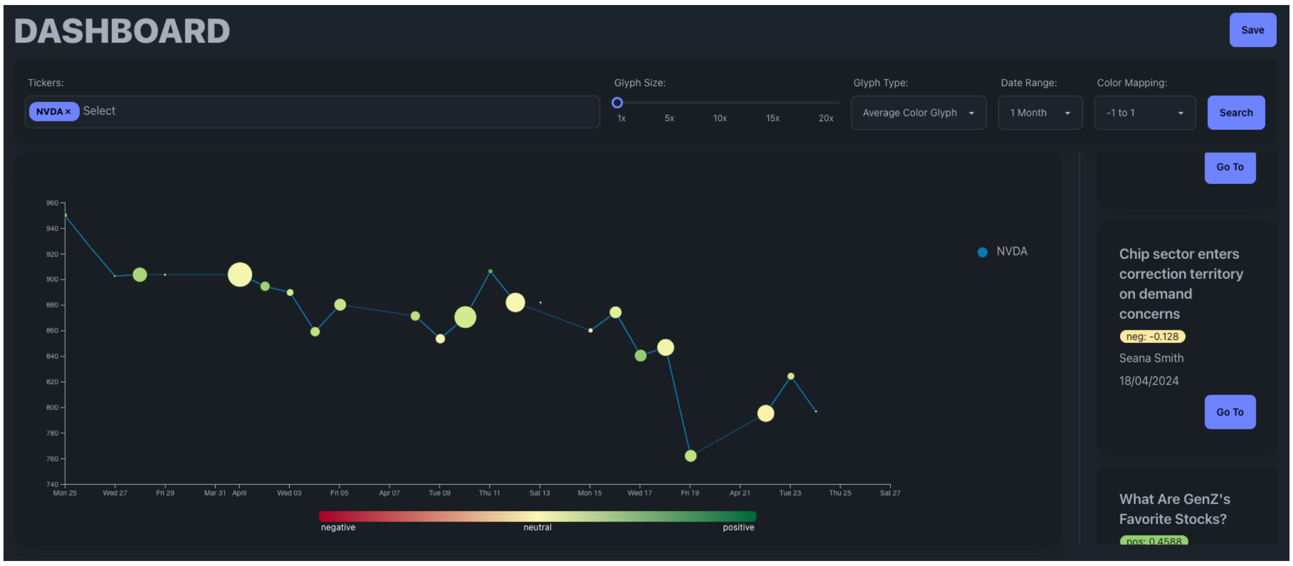
Table 6: A table showing the benchmarks done on the front end. This has been tested by clicking the search button and timing how long it takes for the data to be displayed. Slight human error is expected.

The frontend benchmarks line up very similarly to the backend benchmarks, with the limiting factor being the news data. The time taken for the frontend to visualise the data seems to have an extremely minimal impact on the time taken even with the larger sample sizes.

7.4 Evaluation

7.4.1 Case Study 1 - NVDA stock price correction April 2024

Nvidia is a Californian headquartered technology business that designs and manufactures graphics processing units (GPUs) and is listed on the NASDAQ stock exchange. The business is closely aligned to the artificial intelligence market, due to GPUs being highly used in machine learning and AI. Nvidia is extremely well positioned, given that its technology is currently the only truly viable option for AI. The share price has benefited significantly over the last 12 months, seeing an increase of over 200%. However, the stock price has seen a correction in April 2024, with a decline of 10% between the 18th and 19th of April. This provides an interesting case study in using this project to support the understanding of share price and market sentiment through its visualisation.


DASHBOARD

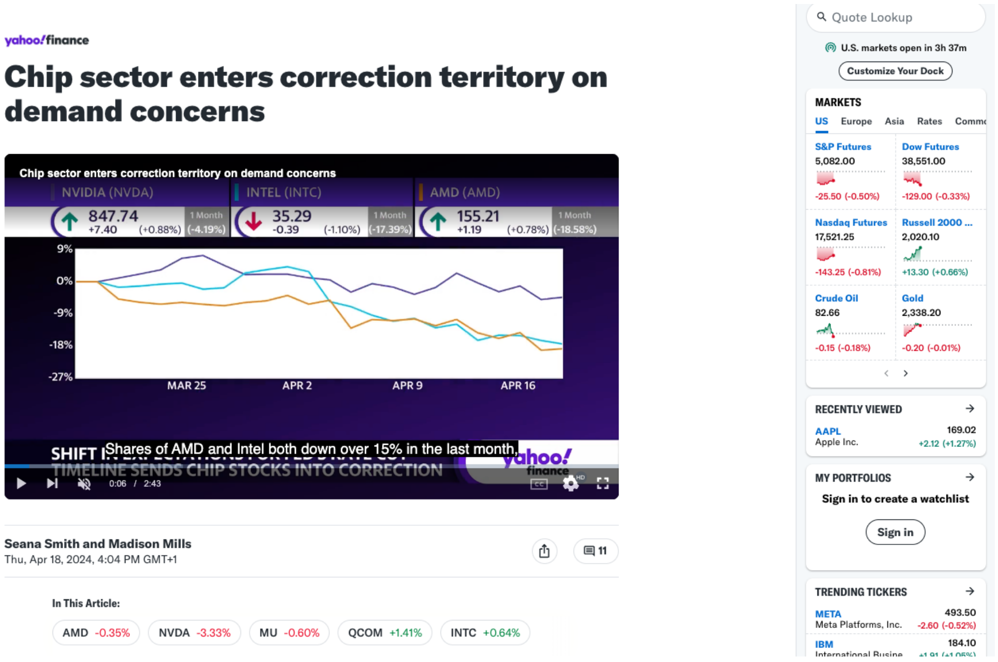
Tickers: NVDA ×
Glyph Size: 1x
Glyph Type: Average Color Glyph
Date Range: 1 Month
Color Mapping: -1 to 1

Chip sector enters correction territory on demand concerns
neg: -0.128
Seana Smith
18/04/2024

What Are GenZ's Favorite Stocks?
pos: 0.4588

Figure 39: The visualisation of NVDA from 2024-03-25 to 2024-04-25

Figure 39 shows share price movements of Nvidia across late March and April 2024. There is a general downward trend but the most significant movement was the 10% drop on the 19th of April. Preceding this drop, there is a glyph showing news coverage with average sentiment that is neutral. Over the last 12 months, Nvidia has typically had a positive sentiment, so this period shows a decline in sentiment. Reviewing the specific news content in the period directly preceding the market correction, articles include the one outlined in figure 40, which has been accessed through the application.


yahoo!finance

Chip sector enters correction territory on demand concerns

Chip sector enters correction territory on demand concerns
NVIDIA (NVDA) 847.74 +7.40 (+0.88%) 1 Month (-4.19%)
INTEL (INTC) 35.29 -0.39 (-1.10%) 1 Month (-17.39%)
AMD (AMD) 155.21 +1.19 (+0.78%) 1 Month (-18.58%)
9%
0%
-9%
-18%
-27%
MAR 25
APR 2
APR 9
APR 16
Shares of AMD and Intel both down over 15% in the last month.
TIMELINE SENDS CHIP STOCKS INTO CORRECTION
yahoo! finance
0:06 / 2:43

Seana Smith and Madison Mills
Thu, Apr 18, 2024, 4:04 PM GMT+1

In This Article:

AMD -0.35% NVDA -3.33% MU -0.60% QCOM +1.41% INTC +0.64%

Quote Lookup

U.S. markets open in 3h 37m

Customize Your Dock

MARKETS

US Europe Asia Rates Commo

S&P Futures 5,082.00 -25.50 (-0.50%)	Dow Futures 38,551.00 -129.00 (-0.33%)
Nasdaq Futures 17,521.25 -143.25 (-0.81%)	Russell 2000 ... 2,020.10 +13.30 (+0.66%)
Crude Oil 82.66 -0.15 (-0.18%)	Gold 2,338.20 -0.20 (-0.01%)

RECENTLY VIEWED

AAPL Apple Inc. 169.02 +2.12 (+1.27%)

MY PORTFOLIOS

Sign in to create a watchlist

Sign in

TRENDING TICKERS

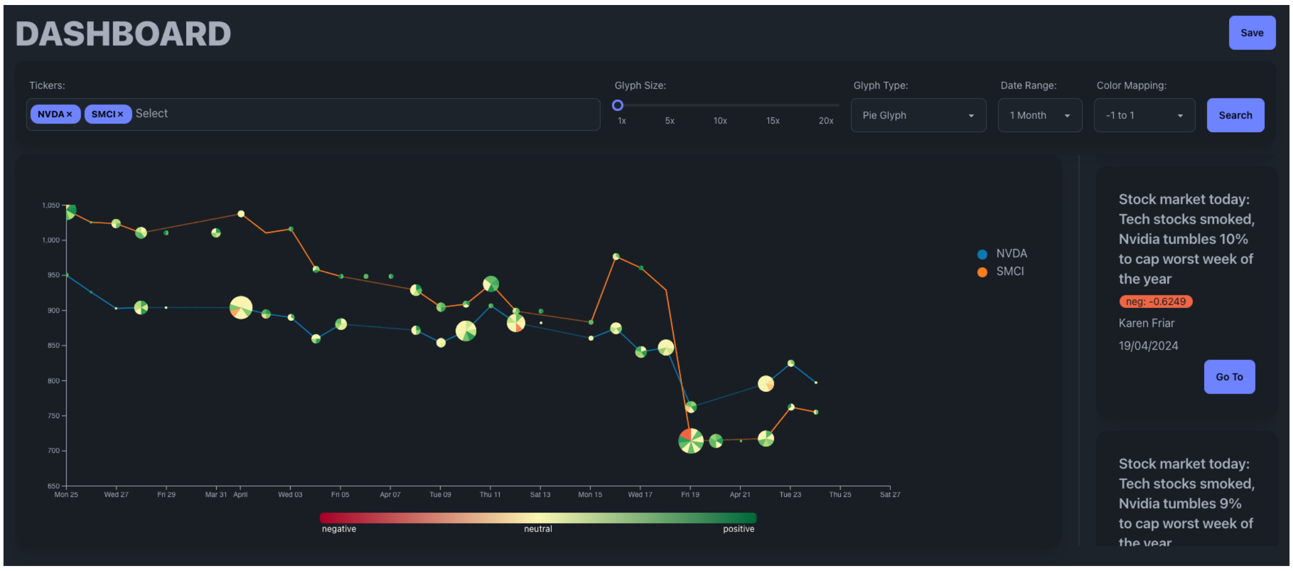
META Meta Platforms, Inc. 493.50 -2.60 (-0.52%)

IBM International Busine 184.10 +1.91 (+1.05%)

Figure 40: A yahoo finance article on Nvidia stock correction

The news article from Yahoo Finance, which has a sentiment rating of -0.128, indicates that the cause of the share price drop relates to sentiment regarding the broader technology hardware market that Nvidia operates within. The application in this case study enables the user to better understand share price movement within the context of market sentiment. However, as the news articles are based on the search of the stock ticker term, news regarding broader market sentiment is marginally less clear in the glyph visualisation.

The news articles regarding the Nvidia share price drop, flag that the trigger for the correction was another business Super Micro Computer Inc. (SMCI), who broke tradition and provided no preliminary indication prior to the scheduled earnings report at the end of April. This caused a ripple effect across the technology hardware market and significantly impacted the share prices of NVDA and SMCI. This application is designed to enable individual investors to undertake an iterative process to inform better investment decisions, an example of which is shown in figure 41, showing the same time period as figure 39 but with NVDA and SMCI in a comparative view.


DASHBOARD

Tickers: NVDA × SMCI × Select

Glyph Size: 1x

Glyph Type: Pie Glyph

Date Range: 1 Month

Color Mapping: -1 to 1

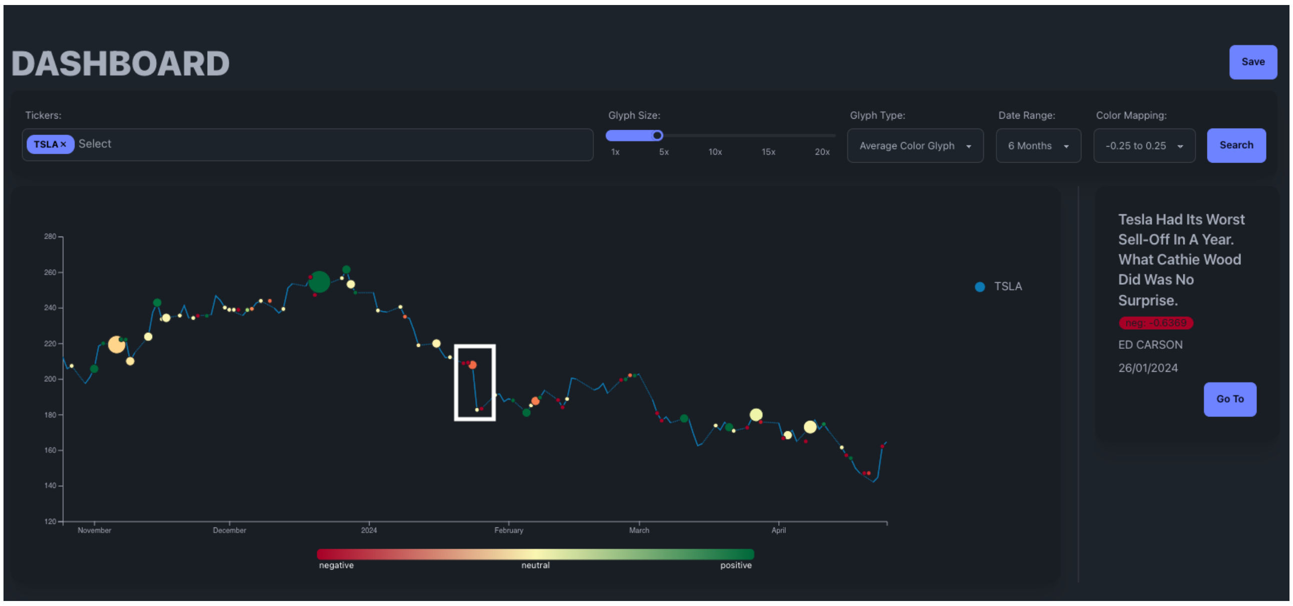
Stock market today: Tech stocks smoked, Nvidia tumbles 10% to cap worst week of the year
neg: -0.6249
Karen Friar
19/04/2024
Go To

Stock market today: Tech stocks smoked, Nvidia tumbles 9% to cap worst week of the year

Figure 41: The visualisation of NVDA and SMCI from 2024-04-01 to 2024-04-25

7.4.2 Case Study 2 - TSLA Q4 earnings report

A further case study relates to where individual investors would seek to evaluate share price movements in respect to news content and sentiment over a specific time period. This example relates to Tesla, a Californian-based electric vehicle manufacturer listed on the NASDAQ stock exchange. The period under review is the last six months to date, which has seen significant volatility in the share price. Individual investors can use the application to review news content and evaluate sentiment at key price movements. Figure 42 shows the Tesla share price and glyphs for the period.


DASHBOARD

Tickers: TSLA × Select

Glyph Size: 1x 5x 10x 15x 20x

Glyph Type: Average Color Glyph

Date Range: 6 Months

Color Mapping: -0.25 to 0.25

Tesla Had Its Worst Sell-Off In A Year. What Cathie Wood Did Was No Surprise.

neg: -0.6369

ED CARSON

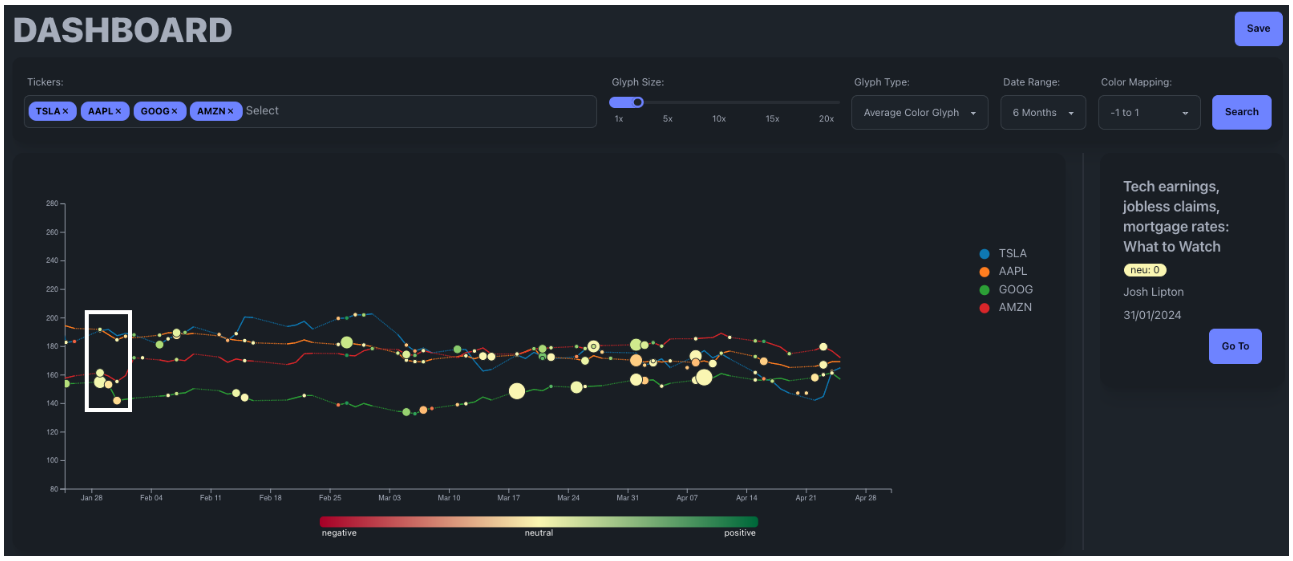
26/01/2024

Figure 42: The visualisation of TSLA from 2023-10-25 to 2024-04-25, with glyph colour mapping lowered to show higher definition.

The screenshot above in figure 42 also highlights a specific time period in late January 2024. The line graph indicates a significant drop in the share price and the glyphs indicate a corresponding negative sentiment in the news content. The article indicated to the right has a sentiment score of -0.6, which shows a significant negative bias. Further review of the content shows that Tesla reported poor earnings for Q4 2023, which resulted in a decline in the share price. This microeconomic perspective is well served by the application, as the search functionality is preset for the specific share price. Where factors influencing share price movements are company-specific and not general market factors, the application provides meaningful insight to the individual investor.

7.4.3 Case Study 3 - Federal Reserve Inflation Comments

Individual investors seeking to make sound decisions will want to understand both microeconomic factors relating to specific companies, as well as macroeconomic factors such as the broader market. One of the biggest macroeconomic factors impacting share price is the actions and comments of central banks such as the Federal Reserve in the United States. The screenshot shown in figure 43 shows the share price of significant US tech stocks: Tesla, Apple, Google and Amazon. Specific focus is given to the 31st January 2024 which shows a decline in the share price of all four stocks.



Tickers: TSLA× AAPL× GOOG× AMZN× Select

Glyph Size: 1x 5x 10x 15x 20x

Glyph Type: Average Color Glyph

Date Range: 6 Months

Color Mapping: -1 to 1

Tech earnings, jobless claims, mortgage rates: What to Watch

neu: 0

Josh Lipton

31/01/2024

Figure 43: The visualisation of TSLA, AAPL, GOOG and AMZN from 2024-01-27 to 2024-04-25

The Federal Reserve chairman Jerome Powell made comments on the 31st January 2024 that inflation was more persistent than expected and interest rate cuts could be delayed [42]. After reviewing the news articles associated with the glyphs on the graph in figure 43, there is no evidence the application has picked up this significant macroeconomic factor, assuming that the Federal Reserve's influence on the market has caused the decline. This suggests there are limitations to the evaluation possible for individual investors using this application. The search functionality is based around predefined terms, using the specific stock ticker; this makes it more effective in understanding microeconomic factors on share price movements.

8 Conclusions

8.1 Reflection

This project has had a successful outcome, delivering on all the requirements as listed in the Project Specification section 3: visualisation of stock data, news content and sentiment within an accessible user interface. There is no current product available to individual investors that helps to extract meaning and sentiment from the vast array of investment news content.

The main goals of this project were in developing visualisation techniques to present the data effectively and creating interactive elements that further enhance understanding. This has been achieved and the application offers individual investors a way to extract meaning from news content. The frequency of news content related to a specific share offers insight into how the volume of news impacts share price. Presenting a visualisation of the degree of positive or negative bias in the content offers an efficient means of extracting understanding. The interactive aspects of the visualisation provide the user the opportunity to focus on specific shares, time frames and views that deepen understanding. While the application does not instruct investment decisions, it enhances the information available to the individual investor, which in turn supports better returns.

This project has produced a web application that fulfils a gap in the market for individual investors. Currently while news is clearly important and included by other systems, no effort is made to use visualisation to enhance understanding. This project provides a unique visualisation-based perspective, that offers genuinley improved understanding of market context for individual investors to seek to improve investment descisions.

8.2 LSEPI

This project has been undertaken in consideration of the research framework: Legal, Social, Ethical and Professional Issues (LSEPI). The nature of this project means the implications are limited but full reference has been made to ensure the work is compliant.

There are several legal legislations and standards that the project has considered, most notably the General Data Protection Regulation (GDPR). Conscious of the implications, this was a further advantage of using Firebase [9] which outsources the responsibility and avoids the retention of data that would cause the regulation to apply. The social aspects of the research are limited, with no primary research involving direct interaction, thus this is not a significant concern. The project has no bias to a particular author, academic or work and as such, ethical concerns are limited. The investment nature of the application does raise ethical considerations but at no point is the application encouraging investments or dictating specific investments. The author has completed the research mindful of the British Computer Society (BCS) and its code of conduct. Further professional considerations have included a careful approach to research and adhering to the Copyright, Designs and Patents Act 1988, with accurate referencing.

Consideration has also been given to the United Nations (UN) 'Sustainable Development Goals' and the 'Big Picture' that the project operates within. The goal of 'Decent Work and Economic Growth' is an aspect of the project. The project goal is to support individual investors in achieving better returns through a greater understanding of news content and sentiment against share price. The project delivers on this goal and should encourage economic growth through improved investments. A further UN goal is the reduction in inequalities. This project seeks to provide greater understanding and access to information for all, rather than the current disadvantage individuals have with limited access to information.

9 Future Work

The application offers a viable tool to support individual investors in understanding the context of share price movements. However, there are opportunities for further development and enhancement, including sentiment accuracy, additional search functionality, additional quantitive data, additional media sources, and improved data processing times.

The sentiment analysis is the main drawback of the project and can be improved. An example of its current limitation is that the sentiment analysis has no context of the stock the article is about. This means that if an article mentions the selected stock ticker and is negative about a competitor, the sentiment will be negative for the selected stock. This could be improved by developing a machine learning model that can understand the context of the article and also know which stock the sentiment should be for. This would be a large undertaking and would likely be a project in itself.

As indicated in the case studies, one of the challenges of the application has relates to its ability to provide insight relating to macroeconomic factors. This is due to the search functionality being restricted to the stock ticker term. An area of further work would be to broaden the search functionality to include the ability to pickup key events in the broader market.

Not all share price movements are directly related to news content, this could be improved upon by adding more quantitive data to the application. Factors such as trade volume, moving averages and other technical indicators could have their own visualisations. This would give greater context to the user and allow for a more comprehensive view of the stock.

The application only focuses on news content, this could lead to bias in the sentiment analysis. The inclusion of other media sources such as social media could provide more differing opinions on stocks and greater accuracy in sentiment analysis. This would require a similar machine learning model to the one mentioned above, as well as an additional API to retrieve data from these sources.

The wait times for the data processing could be improved. This is mainly a limitation on the API tiers that are being used. Replacing the current APIs with higher tiers could improve the speed by allowing more requests to be made at once. Another solution is to make the application independent of the APIs by web scraping the data; however, that would be a much larger task.

References

[1] A. Atkins, M. Niranian, and E. Gerding. Financial news predicts stock market volatility better than close price. *International Review of Financial Analysis*, 4(2):120–137, 2018.

[2] R. Borgo, J. Kehrer, D. H. S. Chung, E. Maguire, R. S. Laramee, H. Hauser, M. Ward, and M. Chen. Glyph-base visualization: Foundations, design guidelines, techniques and applications. *Eurographics State of the Art Reports*, pages 39–63, 2013.

[3] M. Bostock, P. Riviere, and K. A. Github - d3/d3-scale-chromatic: Sequential, diverging and categorical color scales. `https://github.com/d3/d3-scale-chromatic`, 2021. [Online; accessed 2024-04-07].

[4] Brewer and C. A. Colorbrewer: Color advice for maps. `https://colorbrewer2.org/`, 2016. [Online; accessed 2024-04-07].

[5] W. S. Chan. Stock price reaction to news and no-news: Drift and reversal after headlines. *Journal of Financial Economics*, 70(2):223–260, 2003.

[6] D3. D3. `http:d3js.org`, 2023. [Online; accessed 2023-12-29].

[7] DaisyUI. Daisyui. `http:daisyui.com`, 2023. [Online; accessed 2023-12-29].

[8] L. Feng, T. Fu, and Y. Shi. How does news sentiment affect the state of japanese stock return volatility? *International Review of Financial Analysis*, 84, 2022.

[9] Firebase. Firebase. `http:firebase.google.com`, 2023. [Online; accessed 2023-12-29].

[10] M. Frankel. 3 reasons why the average investor actually stinks at investing. `https://www.fool.com/the-ascent/buying-stocks/articles/3-reasons-why-the-average-person-actually-stinks-at-investing/#:~:text=The%2520bottom%2520line,a%2520combination%2520of%2520the%2520three/`, 2023. [Online; accessed 2023-10-15].

[11] D. P. Gandhma and K. Kumar. Systematic analysis and review of stock market prediction techniques. *Computer Science Review*, 34:100–190, 2019.

[12] GitLab. Gitlab. `http:gitlab.com`, 2023. [Online; accessed 2023-12-29].

[13] Google. Google finance. `http:google.com/finance`, 2023. [Online; accessed 2024-04-07].

[14] A. Grob-Klubmann and N. Hautsch. When machines read the news: Using automated text analytics to quantify high frequency news-implied market reactions. *International Review of Financial Analysis*, 18(2), 2011.

[15] J. Highsmith and F. M. The agile manifesto. *Software development*, 9(8):28–35, 2001.

[16] Insomnia. Insomnia. `http:insomnia.rest`, 2023. [Online; accessed 2023-12-29].

[17] Jest. Jest. `http:jestjs.io`, 2023. [Online; accessed 2023-12-29].

[18] J. M. Jones. What percentage of americans owns stock? `"https://news.gallup.com/poll/266807/percentage-americans-owns-stock.aspx"`, 2023. [Online; accessed 2023-10-14].

[19] JSDoc. Jsdoc. `http:jsdoc.app`, 2023. [Online; accessed 2023-12-29].

[20] D. A. Keim. Information visualizatioin and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

[21] S. Ko, I. Cho, S. Afzal, C. Yau, J. Chae, A. Malik, K. Beck, Y. Jang, W. Ribarsky, and D. S. Ebert. A survey on visual analysis approaches for financial data. *Computer Graphics Forum*, 35(3):599–617, 2016.

[22] R. S. Laramee. *Advances in Computer Science and Engineering*, 4(1):23–36, 2010.

[23] R. S. Laramee. Bob's project guidelines: Writing a dissertation for a bsc. in computer science. *Innovation in Teaching and Learning in Information and Computer Science*, 10(1):43–54, 2011.

[24] X. Liu, M. Alharbi, J. Chen, A. Diehl, E. E. Firat, D. Rees, Q. Wang, and R. S. Laramee. Visualization resources: A survey. *Information Visualisation*, 22(1):3–30, 2022.

[25] B. G. Malkiel. The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, 17(1):59–82, 2003.

[26] L. McNabb and R. S. Laramee. Survey of surveys (sos) - mapping the landscape of survey papers in information visualization. *Computer Graphics Forum*, 36(3):589–617, 2017.

[27] MorningStar. Morning star. `http:morningstar.com`, 2023. [Online; accessed 2023-10-15].

[28] MotleyFool. Motley fool. `http:motleyfool.com`, 2023. [Online; accessed 2023-10-15].

[29] Multiselect-React-Dropdown. Multiselect-react-dropdown. `https://github.com/srigar/multiselect-react-dropdown`, 2023. [Online; accessed 2023-12-29].

[30] NewsAPI. Newsapi.org. `http:newsapi.org`, 2023. [Online; accessed 2023-12-29].

[31] NewsCatcherAPI. Newscatcherapi.com. `https://www.newscatcherapi.com/`. [Online; accessed 2023-04-09].

[32] T. H. Nguyen, K. Shirai, and J. Velcin. Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications*, 42(24):9603–9611, 2015.

[33] R. Patrick and K. Pybus. Cost of living crisis: we cannot ignore the human cost of living in poverty. *British Medical Journal*, (377), 2022.

[34] Polygon. Polygon.io. `http:polygon.io`, 2023. [Online; accessed 2023-12-29].

[35] React. React. `http:reactjs.org`, 2023. [Online; accessed 2023-12-29].

[36] ReactRouter. React router. `http:reactrouter.com`, 2023. [Online; accessed 2023-12-29].

[37] D. Rees and R. S. Laramee. A survey of information visualization books. *Computer Graphics Forum*, 38(1):610–646, 2019.

[38] SeekingAlpha. Seeking alpha. `http:seekingalpha.com`, 2023. [Online; accessed 2023-10-15].

[39] B. Shniederman. The eyes have it: a task by data type taxonomy for information visualizations. *IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[40] Tailwind. Tailwind css. `http:tailwindcss.com`, 2023. [Online; accessed 2023-12-29].

[41] Vader-Sentiment. Vader-sentiment. `https://github.com/vaderSentiment/vaderSentiment-js#readme`, 2023. [Online; accessed 2023-12-29].

[42] R. Wile. Federal reserve holds interest rates stead, as consumer confidence improves and inflation slows. `https://www.nbcnews.com/business/economy/federal-reserve-interest-rate-decision-january-2024-increase-decrease-rcna136429`, 2024. [Online; accessed 2024-04-25].

[43] Yahoo. Yahoo finance. `https://uk.finance.yahoo.com/`. [Online; accessed 2024-04-07].