StockSentinel - An AI Powered Tool for Analyzing the Markets Perception of Stocks

Technical Reports: CL-2024-42, July 2024

Andreas Hecht, Linus Heise, Oliver Kneidl, Eva-Maria Maurer,
Christoph P. Neumann
CyberLytics-Lab at the Department of Electrical Engineering, Media, and Computer Science
Ostbayerische Technische Hochschule Amberg-Weiden
Amberg, Germany

***Abstract*—In the realm of investment, public sentiment towards stocks significantly influences the decision-making processes. Recognizing this, we have developed StockSentinel, an AI-powered tool designed to provide investors with insights into the sentiment surrounding various stocks. StockSentinel offers a comprehensive overview, including current stock prices, recent market performance, historical sentiment data, and average weekly sentiment. The base layer of the system is our Postgres database. Our backend is responsible for data acquisition and processing, and our frontend is utilized for visualization of the acquired data.**

***Index Terms*—SvelteKit, ExpressJS, NodeJS, TypeScript, ChartJS, PostgreSQL, Docker, NLTK-VADER, Web-Scraping, Yahoo! Finance, Google-News-RSS, Sentinemt Analysis.**

I. Introduction and Objectives

Basic psychology teaches us that opinions heavily influence our decisions. This is such a major factor in sales, that an entire field of research has developed over the years dedicated solely to understanding and changing the public perception towards certain goods and brands. Therefore, knowing the sentiment and trust towards certain stocks is essential for investors.

To help stay informed about these topics we have developed a tool powered by artificial intelligence (AI). We named this tool StockSentinel.

StockSentinel provides an overview over the sentiment for a predetermined list of stocks. Additionaly the user can get more detailed information about each stock, like the price at market close, the recent performance of the stock in the market, the historical sentiment towards the stock, as well as the average sentiment of the last week towards the stock.

II. Related Work

Because of the stock market's famous unpredictability, there has always been a lot of work and research trying to find ways to change that. Early research found that historical prices alone are highly unreliable in predicting stock prices[1]. So, researchers changed their approach to include sentiment analysis. This analysis can be based on a lot of different sources, the most prominent one being Twitter[2][3][4]. But there have also been works focusing on sources like Reddit[5] or Yahoo! Finance blog posts[6]. Like Kalyanaraman et al. (2014)[7] and Mohan et al. (2019)[8] we focus on articles as our sources instead, and use a machine learning model to analyze them. Because of the advancements in the field of AI, there has also been research into using neural networks to analze sentiment. For example, Jin et al. (2020)[9] used a Long Short-Term Memory (LSTM) model with Empirical Model Decomposition (EMD) to predict stock prices, which showed great potential. All these researchers agree with each other, that public sentiment can increase predictability of stock prices.

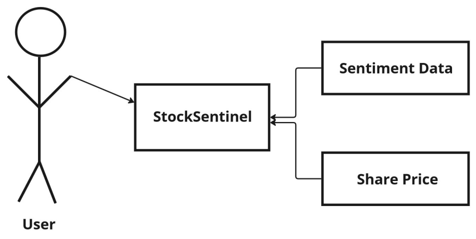


Figure 1: External System

III. Architectural Goals

Figure 1 shows StockSentinel's external systems and how the user interacts with them. There are no special technical preconditions to use our website. We want to provide an interactive and intuitive user interface, that is also aesthetically pleasing and functional. We primarily used Python for the backend as well as the Svelte framework for the frontend using TypeScript.

IV. Architecture of StockSentinel

A. Overall System

The base layer of the system is our Postgres database. Our backend is responsible for data acquisition and processing, and our frontend is utilized for visualization of the acquired data. The data importer consists of the news processing, which fetches pages and news from the Google RSS News Feed. It is followed by the sentiment process, which gets its data directly from the database. Lastly the finance process fetches the stock prices using the Yahoo! Finance API. Therefore we have a three layer architecture running on a single tier. Details about each module can be found in their corresponding section.

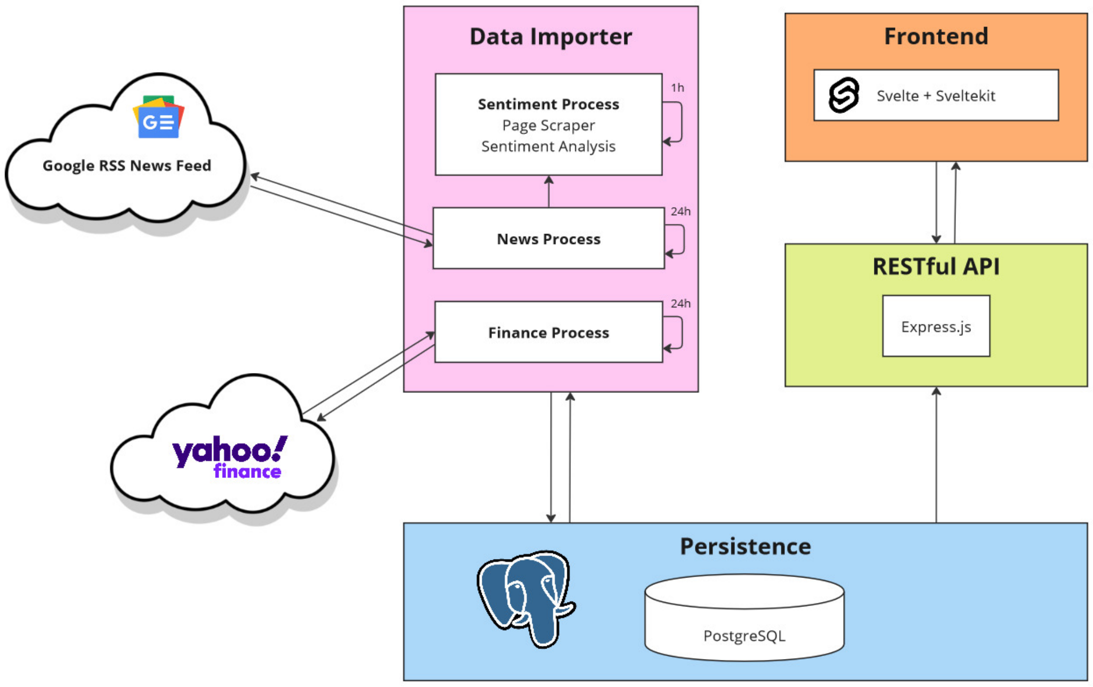


Figure 2: Overview of the Architecture

B. Frontend

The frontend of StockSentinel was built using Sveltekit and consists of two primary pages: The Overview Page and the Dashboard Page. Additionally, we created a layout component, which is shared between all pages, to ensure consistent navigation across the site.

1) Overview Page: The Overview Page is simultaneously the homepage of our website. It presents the user with a list of stocks, in the form of a table with the following features:

- Each row shows a specific stock with its name, ticker and aggregated sentiment of the last week.
- The user can pick, which stock they want more information on. Each stock name and ticker in the table is a clickable link, which navigates the user to the corresponding dashboard for detailed information. For this routing, we are using the built-in routing system of Sveltekit.
- The sentiment has both a numerical and graphical representation in the form of a little icon. Depending on the sentiment score, the icon is either a green thumbs up, a red thumbs down or a black thumb pointing to the right.
- The table has an integrated search bar, allowing users to quickly locate specific stocks by typing its name. The search bar automatically filters out all stocks, that do not match the user input. To create this feature, we used the preexisting table called TableSearch from the Flowbite-Svelte library.

2) Dashboard Page: The Dashboard page is dynamically generated based on the selected stock from the Overview page. We utilize slugs to identify the clicked stock and fetch the relevant data from the database. This approach allows us to construct a single Dashboard page with placeholders, eliminating the need to create multiple pages for each stock. The Dashboard is composed of several Svelte components:

- Price Component: Displays the stock's price at market close.
- Sentiment Component: Shows the aggregated sentiment score for the past week.
- Graph Component: Utilizes Chart.js to render a line graph displaying both sentiment and price data from the beginning of the year up to the present. The sentiment and price data share an x-axis but have separate y-axes. This way the user can easily compare them and identify trends and correlations over time.
- Sources Component: Lists the sources of the sentiment data in a table. It includes the website name, the number of scraped articles and the average sentiment score for each source.

The layout of the Dashboard is mainly structured into three sections. At the top we have the Price and Sentiment component, which are displayed side-by-side in a flex container. Right underneath, the Graph component spans almost the width of the page. The Sources table is positioned below the graph, summarizing the sentiment data sources.

3) Layout Component: In addition to each page's own components, we have created a layout component, which is shared between all pages, even new ones we might create in the future. We included our header in this component, so it is consistently displayed at the top of the website. The header displays the StockSentinel name, logo, and the navigation bar with a button that links to the Overview page.

4) Backend Integration: The backend was implemented as a Node.js server. For this the express web framework was utilized. Our API implements the Representation State Transfer (REST) principles, since it is a stateless API and all calls to the backend can be made without any dependency on another call. All calls contain the entire information required to make the call. The API uses soley GET endpoints. The calls are made by HTTP request and return HTTP Status codes, like 404 and 200 etc. The API route is accessible via /api. The backend enables the frontend to load the necessary data from the database. The data is loaded with a SQL query which is sent to the database. The raw data is then passed to the frontend and integrated into the components. Once the frontend sends a request to the backend, the data is loaded from the database according to the needs of the frontend. For example, the sentiments of several sources are aggregated for each stock and returned in a JSON format, which contains the precise data necessary to implement the visualizations on the frontend Through this preaggregation of data, the frontend can work faster, and reacts almost immediately to user interaction. Most data is pretransformed for the specific use-case of the frontend component, that it is meant to fill with data. The backend was developed to accommodate the specific needs of the frontend, and therefore has GET endpoints that are utilized only once on the website. All GET endpoints, that get stock specific data from the database, have a parameter, in which the name of the stock is passed, and the data is filtered directly at the database, to simplify the frontend code base and improve responsiveness.

C. Persistence

For persistence we use a relational Postgres database. For our core application we want to store stocks and their historic market price, as well as trusted news sources and their published articles with their calculated sentiment. The resulting database schema is depicted in Figure 3. We also considered using the Timescale DB extension, however due to a relatively coarse granularity of time data, our acquired data volume was too low to make proper use of it.

D. Data Acquisition

1) News-Crawler: The News-Crawler uses the Google-News-RSS service (https://news.google.com/rss), which provides an aggregated search for news by topics, URLs, time and language. This API has several advantages over the Google News UI such as structured data (XML), a higher rate limit as well as up to 100 results per request, instead of only 10. For the implementation we use the python library requests. The crawler acquires the URL of news articles for all of our stocks, aggregated by time and news sources. After that, the retrieved information gets stored in the database. A problem for the actual page request is that the provided links are redirections from Google to the actual page, which would make the scraping process significantly more difficult. However as of today, the links can be decoded with base64 to get the real link of the desired website.

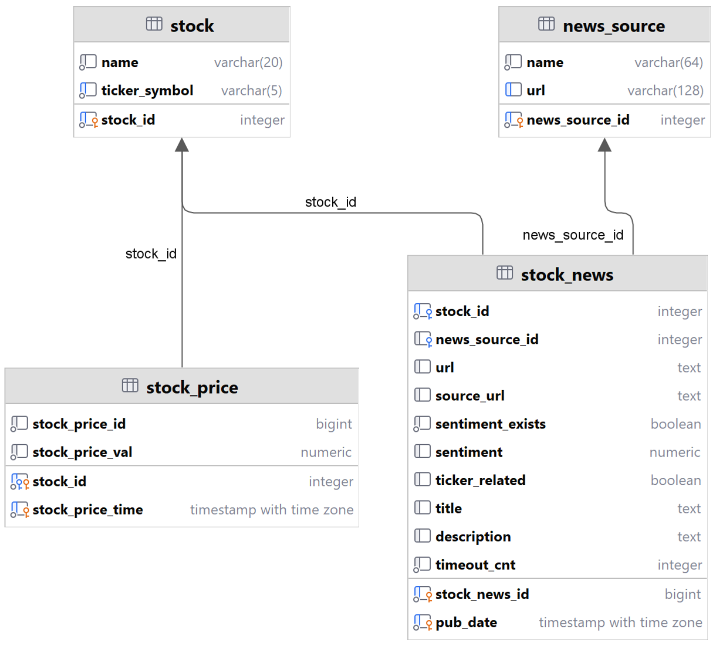


Figure 3: Database

2) Page-Scraper: The News-Crawler only stores URLs and certain page meta data from Google, but for the sentiment analysis we need actual text information to work with. In order to do that we use the Page-Scraper which, at it's core uses the requests and BeautifulSoup libraries of Python to get the HTML of an article. Because most sites have a certain rate limit, we have to restrict the amount of requests to a website per cycle. The process then gets restarted periodically. Pages from different websites can be loaded concurrently. Thus the program repeatedly loads 10 pages per source and starts new threads for each source to fetch the HTML data. When a page request is successful the scraper saves the text data and resolves meta data from JSON-LD to get the description of a page. In order get a cleaner result for the sentiment analysis, the text is filtered by removing common sentences. This happens by storing the sentences for each source in a hashmap and counting their frequency. The Process also sets a lock flag on the pages it is currently working on. This enables running multiple instances of the Page-Scraper simultaneously. When the process is finished or gets shut down the lock flag is removed again.

3) Finance-Scraper: In order to always have up-to-date stockprices, we programmed a scraper using the Yahoo! Finance API. To scrape only the data that's needed, the method expects a start date of every stock. This date is obtained by a database request. The scraper then aggregates the data of interest into a dataframe, which is then inserted into the database by a different function. To prevent a long runtime of the overall system, the finance-scraper is being run as a thread.

4) Sentiment Analyzer: The Sentiment Analyzer is based on the Vader-model from the nltk library. It's a Natural Language Processing (NLP) model trained on social media posts and therefore on short sentences. This was our first challenge, as we want to analyze articles that are longer and less emotional than social media posts. Our first trick was to run the sentiment analysis not on the entire article but on every sentence of the article seperately. At the end of the article the mean of the aggregated sentiment is being calculated. We also gave the headline and description of the article a greater weight than the average sentence. The returned sentiment is a list consisting of four floating point values. The first one being the negative sentiment, the second being the neutral sentiment, the third being the positive sentiment and at last the fourth value being a compound value. For our purposes we exclusively use the latter.

E. Docker

The different components of StockSentinel run in Docker-Containers. Docker-Compose is used to bundle the different services and start them in the correct order. It also sets the environment variables required for each service. When our Data-Importer processes finish, the container is also shut down. The Data Importer consists of the finance-, news- and sentiment-container. We dont want them to always restart so we use dedicated services, that run in their own container, to restart the Data-Importer containers periodically. We deployed our docker containers on a Hetzner CX22 server.

V. Lessons Learned

Even though the website is functional and fullfills all aesthetic purposes, it has some room for improvement. These improvements were not realized due to the short project time. The main area of improvement would be the training of a NLP model, that is better fitted for longer and professional articles, specifically for a stock and finance context. The difference between the use case Vader NLTK was developed for, and our use case is readily apparent. Stock and finance journals as well as news websites use financial jargon and tend to be less polarized and rather neutral. This resulted in most articles being characterized as neutral, and only few articles were labelled as strongly negative or positive as they actually are. Taking this obvious problem into account, Vader NLTK still performed well, and achieved good results in most cases. Our number of sources is limited, because of several restrictions, such as paywalls. In contrast, the number of stocks is easily scaleable. Another area of improvement is the lack of a login page, as well as lacking customizability of the stock view. The original idea was to include a personalized page, which enabled users to exclude certain sources from the presented information. In general, the website has achieved its goals in our opinion, but as always there is room for improvement.

VI. Conclusion and Future Work

The website shows a certain level of correlation between our calculated public sentiment towards a stock and the price of the stock. This correlation would probably be higher, if the sentiment analysis was more accurate. Even if the analysis was better, more information is required to make an educated investment in the stock market. The sentiment of our sources usually takes most of this lacking information into account, but not every factor that is relevant. With more information, and an improvement of the model, the website would most certainly be a useful tool for stock investors. The main benefit in its current form, is to get a quick overview of the Sentiment towards a certain stock, and to see how the price has developed over time. To enhance the utility of the website, we could incorporate an analysis of quarterly reports alongside other critical metrics such as the P/E ratio, fair market value, and EBITDA. This comprehensive approach would provide more robust insights for investment decisions. However, it's essential to acknowledge that no amount of data can fully predict the stock market, as it is a complex system with numerous interrelated and unpredictable factors. Another factor to consider is the overall market sentiment towards the economy, elections, inflation, wars and other geo-political and natural events that can not be predicted with any degree of accuracy.

References

[1] Eugene F. Fama. "The Behavior of Stock-Market Prices". In: *The Journal of Business* 38.1 (1965), pp. 34–105. ISSN: 00219398, 15375374. URL: http://www.jstor.org/stable/2350752 (visited on 06/28/2024).

[2] Venkata Sasank Pagolu, Kamal Nayan Reddy, Ganapati Panda, and Babita Majhi. "Sentiment analysis of Twitter data for predicting stock market movements". In: *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. 2016, pp. 1345–1350. DOI: 10.1109/SCOPES.2016.7955659.

[3] Ray Chen and Marius Lazer. "Sentiment analysis of twitter feeds for the prediction of stock market movement". In: *stanford edu Retrieved January* 25 (2013), p. 2013.

[4] Johan Bollen, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market". In: *Journal of Computational Science* 2.1 (2011), pp. 1–8. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2010.12.007. URL: https://www.sciencedirect.com/science/article/pii/S187775031100007X.

[5] Tobias Bauer, Fabian Beer, Daniel Holl, Ardian Imeraj, Konrad Schweiger, Philipp Stangl, Wolfgang Weigl, and Christoph P. Neumann. *Reddiment: Eine SvelteKit- und ElasticSearch-basierte Reddit Sentiment-Analyse*. Tech. rep. CL-2022-06. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2022. DOI: 10.13140/RG.2.2.32244.12161.

[6] Padmini Srinivasan Michael Rechenthin W. Nick Street. "Stock Chatter: Using Stock Sentiment to Predict Price Direction". In: *Algorithmic Finance* (2013). DOI: 10.3233/AF-13025. URL: https://content.iospress.com/articles/algorithmic-finance/af025.

[7] Vaanchitha Kalyanaraman, Sarah Kazi, Rohan Tondulkar, and Sangeeta Oswal. "Sentiment Analysis on News Articles for Stocks". In: *2014 8th Asia Modelling Symposium*. 2014, pp. 10–15. DOI: 10.1109/AMS.2014.14.

[8] Saloni Mohan, Sahitya Mullapudi, Sudheer Sammeta, Parag Vijayvergia, and David C. Anastasiu. "Stock Price Prediction Using News Sentiment Analysis". In: *2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*. 2019, pp. 205–208. DOI: 10.1109/BigDataService.2019.00035.

[9] Yang Yang Zhigang Jin and Yuhong Liu. “Stock closing price prediction based on sentiment analysis and LSTM”. In: *Neural Computing and Applications* 32 (2020), pp. 9713–9729. DOI: https://doi.org/10.1007/s00521-019-04504-2.

[10] Paul Brandl, Manuel Kalla, Dominik Panzer, Kevin Paulus, Manuel Pickl, Franziska Rubenbauer, Berkay Yurdaguel, and Christoph P. Neumann. *Neunerln: Eine MEVN-basierte Webanwendung zum kompetitiven Kartenspielen*. Tech. rep. CL-2023-11. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2023. DOI: 10.13140/RG.2.2.33933.31209.

[11] André Kestler, Antonio Vidos, Marcus Haberl, Tobias Dobmeier, Tobias Lettner, Tobias Weiß, and Christoph P. Neumann. *Computer Vision Pipeline: Eine React- und Flask-basierte Webanwendung zur No-Code-Bildverarbeitung mit Cloud-Deployment*. Tech. rep. CL-2023-08. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2023. DOI: 10.13140/RG.2.2.23866.98248.

[12] Jakob Götz, Uwe Kölbel, Maximilian Schlosser, Oliver Schmidts, Jan Schuster, Philipp Seufert, Fabian Wagner, and Christoph P. Neumann. *Nautical Nonsense: Eine Phaser3- und FastAPI-basierte Webanwendung für Schiffe-Versenken mit Cloud-Deployment*. Tech. rep. CL-2023-07. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2023. DOI: 10.13140/RG.2.2.17156.09601.

[13] Lukas Feil, Stefan Reger, Timon Spichtinger, Manuel Pickl, Gian Piero Cecchetti, Alexander Hammer, Berkay Yurdagül, and Christoph P. Neumann. *Torpedo Tactics: Eine MEVN-basierte Webanwendung für Schiffe-Versenken mit Cloud-Deployment*. Tech. rep. CL-2023-06. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2023. DOI: 10.13140/RG.2.2.22608.69120.

[14] Rebecca Kietzer, Baran Baygin, Carl Küschall, Jonathan Okorafor, Luca Käsmann, Michael Zimmet, Michael Ippisch, and Christoph P. Neumann. *Stockbird: Eine React-basierte Webanwendung mit serverless Cloud-Deployment zur Analyse des Einfluss von Tweets auf Aktienkurs-Schwankungen*. Tech. rep. CL-2023-04. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2023. DOI: 10.13140/RG.2.2.32675.02083.

[15] Christian Rute, Alex Müller, Alexander Rudolf Wittmann, Arthur Zimmermann, David Nestmeyer, Julian Tischlak, Matthias Wolfinger, and Christoph P. Neumann. *FancyChess: Eine Next.js-basierte Cloud-Anwendung zum Schachspielen*. Tech. rep. CL-2023-03. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2023. DOI: 10.13140/RG.2.2.19253.24802.

[16] Anastasia Chernysheva, Jakob Götz, Ardian Imeraj, Patrice Korinth, Philipp Stangl, and Christoph P. Neumann. *SGDb Semantic Video Game Database: Svelte- und Ontotext-basierte Webanwendung mit einer Graphen-Suche für Videospiele*. Tech. rep. CL-2023-02. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Mar. 2023. DOI: 10.13140/RG.2.2.11272.60160.

[17] Johannes Horst, Manuel Zimmermann, Patrick Sabau, Saniye Ogul, Stefan Ries, Tobias Schotter, and Christoph P. Neumann. *OPCUA-Netzwerk: Angular- und FastAPI-basierte Entwicklung eines OPC-UA Sensor-Netzwerks für den Heimbereich*. Tech. rep. CL-2023-01. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Mar. 2023. DOI: 10.13140/RG.2.2.22177.79209.

[18] Alexander Ziebell, Anja Stricker, Annika Stadelmann, Leo Schurrer, Philip Bartmann, Ronja Bäumel, Ulrich Stark, and Christoph P. Neumann. *Wo ist mein Geld: Eine MERN-basierte Webanwendung für gemeinsame Ausgaben mit Freunden oder Kollegen*. Tech. rep. CL-2022-11. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2022. DOI: 10.13140/RG.2.2.28888.67847.

[19] Bastian Hahn, Martin Kleber, Andreas Klier, Lukas Kreussel, Felix Paris, Andreas Ziegler, and Christoph P. Neumann. *Twitter-Dash: React- und .NET-basierte Trend- und Sentiment-Analysen*. Tech. rep. CL-2022-07. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2022. DOI: 10.13140/RG.2.2.15466.90564.

[20] Florian Bösl, Helge Kohl, Anastasia Chernysheva, Patrice Korinth, Philipp Porsch, and Christoph P. Neumann. *Explosion Guy: Cloud-basiertes Matchmaking für einen graphischen Bombenspaß*. Tech. rep. CL-2022-05. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2022. DOI: 10.13140/RG.2.2.18822.34882.

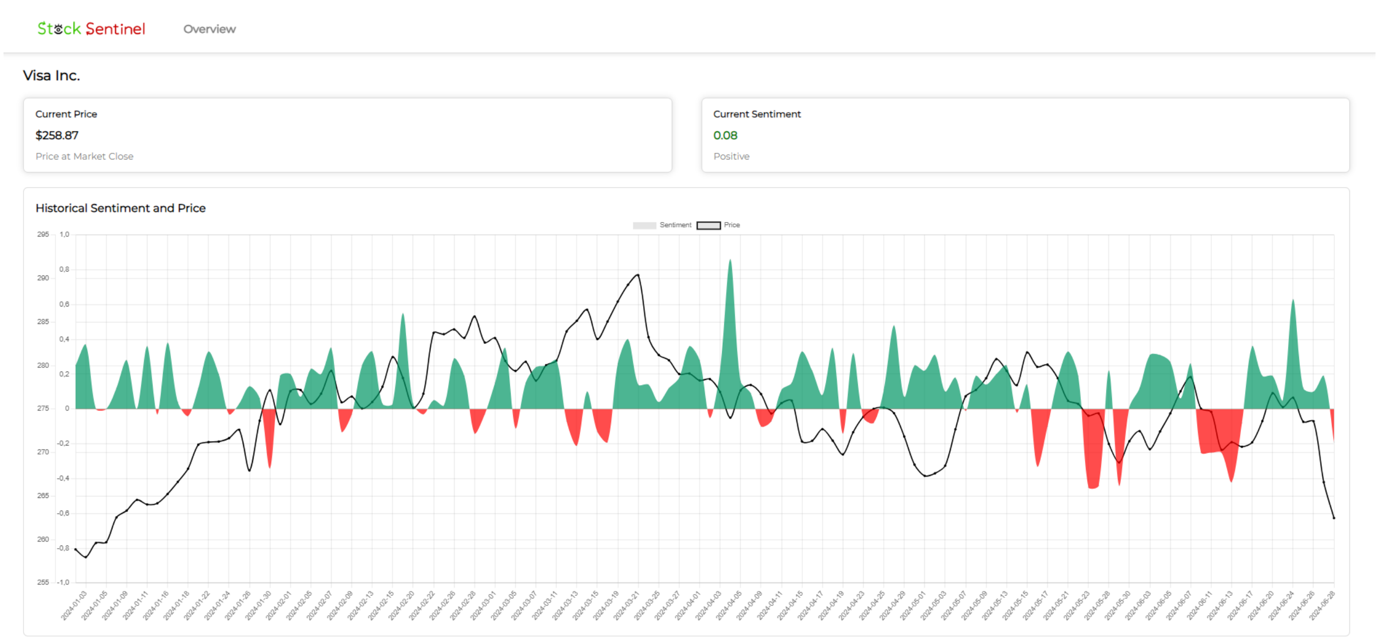
[21] Dominik Smrekar, Johannes Horst, Patrick Sabau, Saniye Ogul, Tobias Schotter, and Christoph P. Neumann. *OTH-Wiki: Ein Angular- und FastAPI-basiertes Wiki für Studierende*. Tech. rep. CL-2022-04. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2022. DOI: 10.13140/RG.2.2.25533.23526.

[22] Johannes Halbritter, Helge Kohl, Lukas Kreussel, Stephan Prettner, Andreas Ziegler, and Christoph P. Neumann. *Graphvio: Eine Graphdatenbank-Webanwendung für integrierte Datensätze von Streaminganbietern*. Tech. rep. CL-2022-01. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, Mar. 2022. DOI: 10.13140/RG.2.2.12111.46244.

[23] Tobias Bauer, Albert Hahn, Lukas Kleinlein, Nicolas Proske, Leonard Wöllmer, Andrei Trukhin, and Christoph P. Neumann. *Covidash: Eine MEAN-Variation-basierte Webanwendung für Inzidenz-Zahlen und Impffortschritt in Deutschland*. Tech. rep. CL-2021-06. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2021. DOI: 10.13140/RG.2.2.33921.84321.

[24] Cameron Barbee, Tim Hoffmann, Christian Piffel, Tobias Schotter, Sebastian Schuscha, Philipp Stangl, Thomas Stangl, and Christoph P. Neumann. *FireForceDefense: Graphisches Tower-Defense-Spiel mit Kubernetes-Deployment*. Tech. rep. CL-2021-05. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2021. DOI: 10.13140/RG.2.2.20500.07048.

[25] Egidia Cenko, Madina Kamalova, Matthias Schön, Christoph Schuster, Andrei Trukhin, and Christoph P. Neumann. *Med-Planner: Eine Angular- und Django-basierte Webanwendung um ärztliche Termine übersichtlich zu verwalten*. Tech. rep. CL-2021-04. Ostbayerische Technische Hochschule Amberg-Weiden, CyberLytics-Lab an der Fakultät Elektrotechnik, Medien und Informatik, July 2021. DOI: 10.13140/RG.2.2.19409.71528.


Stock Sentinel Overview

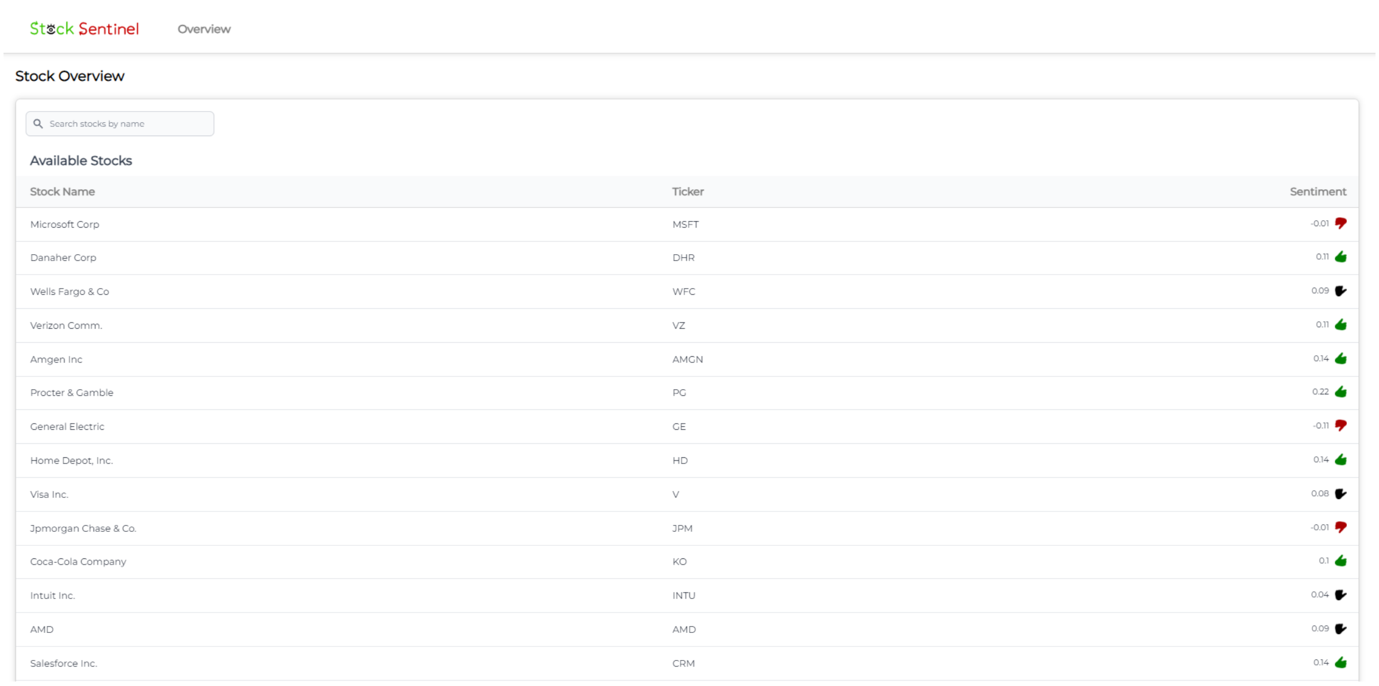
Visa Inc.

Current Price
$258.87
Price at Market Close

Current Sentiment
0.08
Positive

Historical Sentiment and Price

Figure 4: Dashboard view of a selected Stock


Stock Sentinel Overview

Stock Overview

Search stocks by name

Available Stocks

Stock Name	Ticker	Sentiment
Microsoft Corp	MSFT	-0.01 👎
Danaher Corp	DHR	0.11 👍
Wells Fargo & Co	WFC	0.09 👉
Verizon Comm.	VZ	0.11 👍
Amgen Inc	AMGN	0.14 👍
Procter & Gamble	PG	0.22 👍
General Electric	GE	-0.11 👎
Home Depot, Inc.	HD	0.14 👍
Visa Inc.	V	0.08 👉
Jpmorgan Chase & Co.	JPM	-0.01 👎
Coca-Cola Company	KO	0.1 👍
Intuit Inc.	INTU	0.04 👉
AMD	AMD	0.09 👉
Salesforce Inc.	CRM	0.14 👍

Figure 5: Overview Page / Homepage