Stock Overflow

University of California, Riverside

Github Repo Link: https://github.com/CS-UCR/final-project-stockoverflow

Authors

Name	Github Profile
Aditya Acharya	https://github.com/adialachar
Eric Ong	https://github.com/ericong18
John Shin	https://github.com/jshin029
Ji Hwan Kim	https://github.com/kimjihwan0208
Neal Goyal	https://github.com/nealgoyal

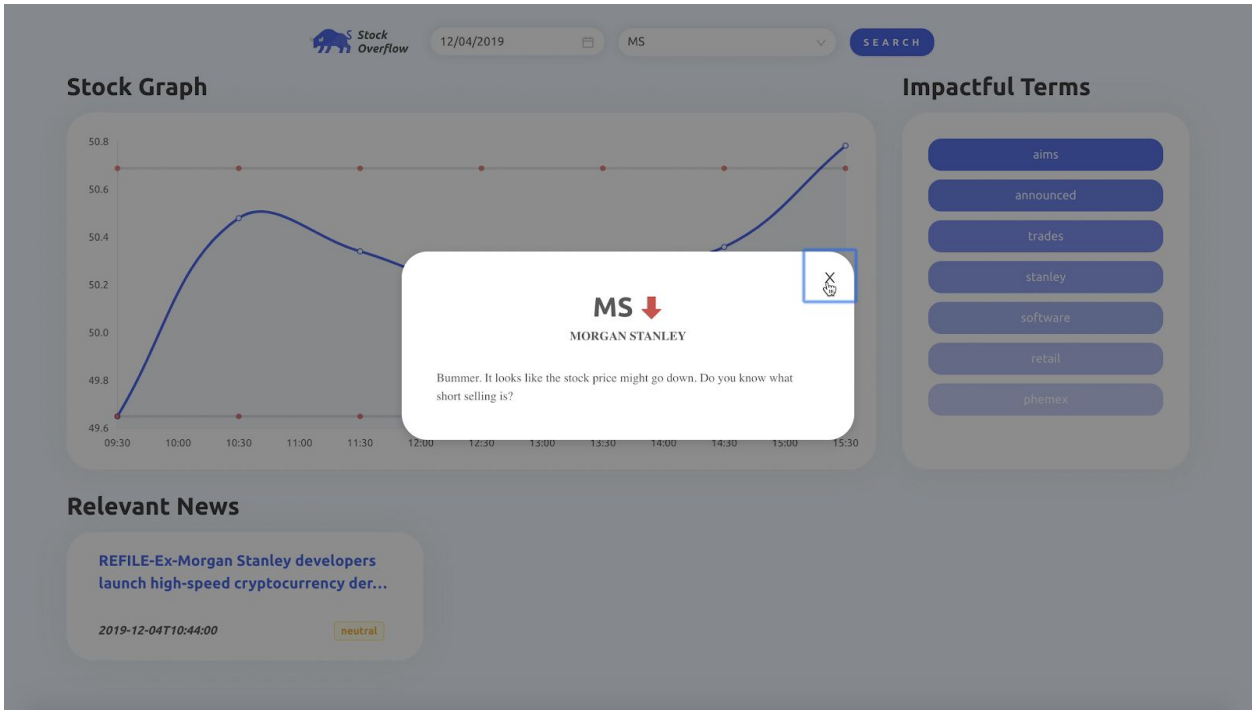
Table of Contents

Overview

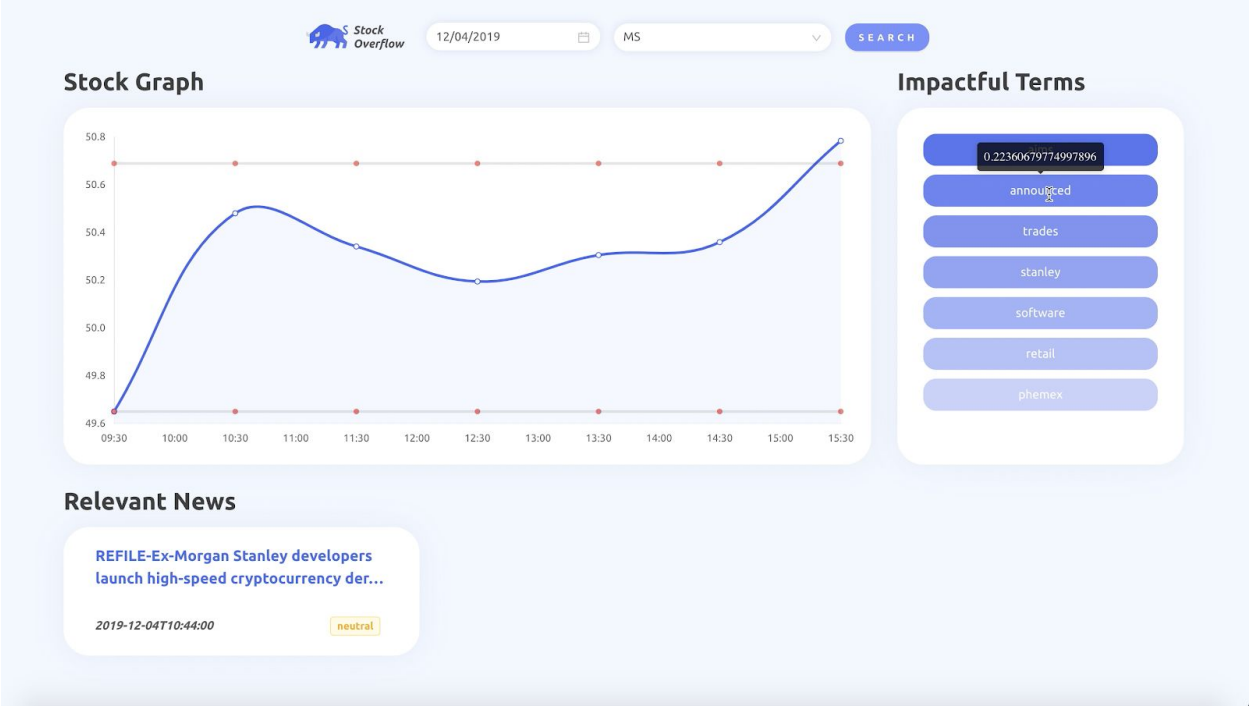
Stock Overflow is a full-stack web application that uses news data to predict changes in stock prices. Not only does it show the prediction, but it also retrieves the news articles that are most relevant to that stock. It also shows the most impactful keywords associated to the price of the stock of the day. Collectively, these features in the application will help provide insights for users through historical and current stock trends.

Final Demo Video: https://www.youtube.com/watch?v=YMJh-efLch4

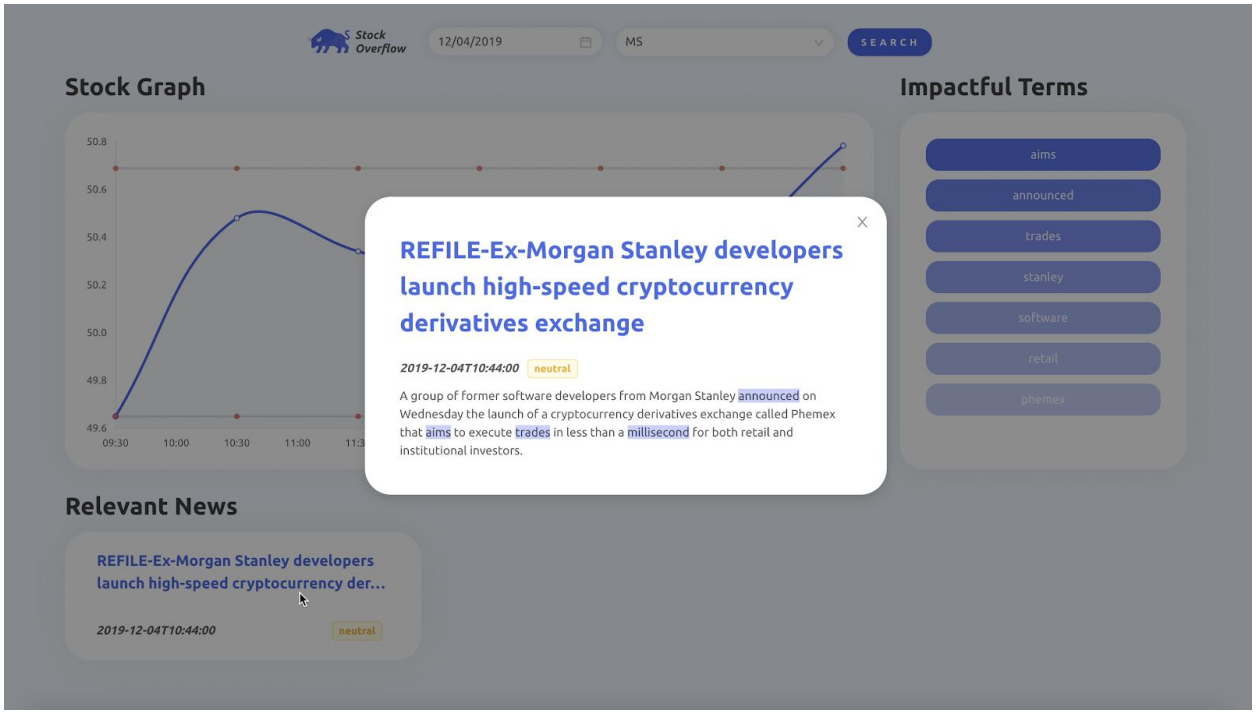
Stock Prediction



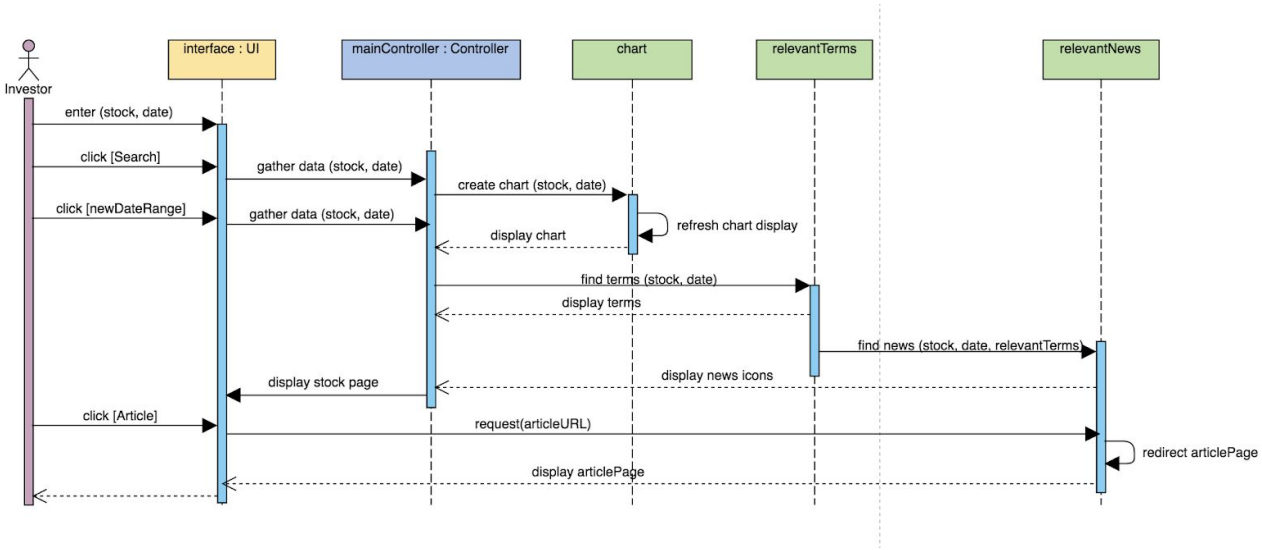
Overall View



Articles



1. Sequence Diagrams



User Actions:

1. Entering Stock Info
2. Search
3. Resizing stock chart
4. Clicking article

2. Detailed Design

a. High-level Design Description

The front end will display the stock data for the last 24 hours of a particular stock on the dashboard. There will also be several cards below the graph displaying blurbs about news articles. To the right of the graph, there will be key terms pertaining to the stock from the day's news articles. All of this data will be received through different API calls to the backend. The backend will then retrieve the pre-processed data and return it to the front end.

b. Front-End Design

As mentioned previously, the frontend will be built using React along with React Hooks. The use of Hooks/functional components will allow the application to be more maintainable, as each component doesn't necessarily need to be tied to the UI directly. We will be using standard AJAX requests (via the *fetch* API in JavaScript) to fetch data from the backend.

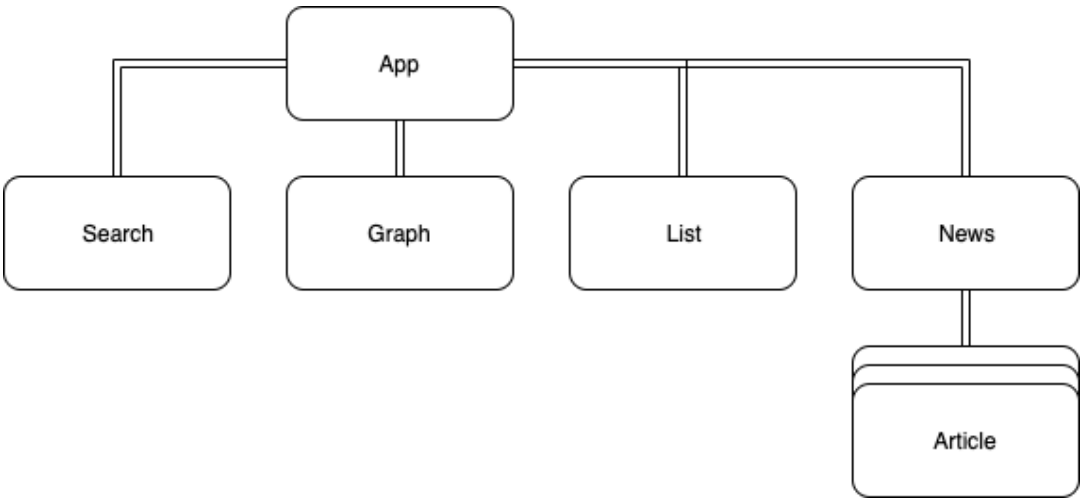
There will be a *Search* component that will allow the user to query the database. This component will render search results of stock symbols while providing some sort of sentiment score correlating to the symbol.

Below the Search component, a *Graph* component will be rendered to display the stock history of the searched company symbol. The graph will also show the opening and closing prices of the stock on the current day.

To the right of the graph, there will be a *List* component that displays the most impactful terms associated with the stock symbol, ranked from most impactful to least impactful.

Under the *Graph* and *List* components, there will be a *News* component, which will render a grid of relevant news articles associated with the stock symbol's company. These articles will be displayed as a "card", which will display information such as the news sentiment. When the user clicks on a card, a modal will render, which will show key, highlighted terms in the news article associated with the sentiment.

The diagram below represents the structure of the frontend.



c. Back-End Design (including Database)

There are several parts to the backend, as described below.

The first is the pre-processed news articles. We will take the data at this link https://github.com/Danbo3004/financial-news-dataset?fbclid=IwAR0yLhOoXeWuIxShnoluaRCdU0mmrYEQ3d7Af15rkWJxKV15oO0DsNwoggA

and the stock symbol, the date, and the keyword from the article body and store them in a document database. We will then have Elastic Search index these documents so that queries are less expensive and faster, as the amount of data we will be dealing with is enormous.

After these documents have been processed and inserted into our document database, we have to set up a back-end server to be able to query the database and return the data to the front end. For this, we will be using Flask, a Python microframework. Flask will receive API calls to its endpoints from the front-end React app, and based on the API call and the information in the request, it will query the document database and retrieve the desired data.

Based on the database we use, there are several client libraries for Python that we can use in our Flask app. For example, if our document database ends up being MongoDB, we can simply use the PyMongo client and store the documents with the built-in functions.

After the data is aggregated and organized, we will put it through a neural network to try to predict whether there will be a stock increase or decrease based on the content of the news article. We are currently investigating different neural network architectures.

d. Design Constraints

A mock-up of the user interface is shown below. From a general perspective, the application should respect browser windows maintaining a 16:9 aspect ratio. This means that any sort of resizing that affects the 16:9 aspect ratio will not be optimal for the user experience. The application, however, is not optimized for mobile, but may be in the future.

e. Engineering Standards

Description	Standard	Governing body
A standard way of naming elements and class names in HTML/CSS files to enforce uniqueness among class names.	BEM	Get BEM

Contributions

Eric

Summary

I mainly worked on the frontend user interface using React, along with Jest to write unit and integration tests for several frontend components. Throughout the project, there was a lot of back-and-forth communication between the frontend and backend. In regards to the communication, often times, the data received from the backend would change (ie. the JSON object attributes and the underlying data would change), so a lot of the frontend work mainly involved data "massaging" and processing as well. In addition to the frontend, when it came to developing the machine learning model, I worked with the rest of the team to implement logistic regression with ***sklearn***. Once that was completed, it was important to make sure that the updated results (from the machine learning model) were received in the frontend so the user would know the appropriate prediction.

Status Reports

Week 5

So far, John and I have worked on completing the skeleton of the frontend, which has been more or less complete. More specifically, I worked on fetching stock symbols pertaining to a date and allowing the user to search with these parameters. In addition to that, using the data sent back from the backend, I was able to display a list of impactful terms based on the news articles for the stock symbol for that date.

Week 6

I worked with Aditya on receiving data from the backend and connecting it with the News component on the frontend. Using the data I received, I was able to display the summaries and sentiments for each news article, while highlighting the important terms in the article summaries. A lot of the work mainly focused on styling and parsing the data from the backend.

Week 8

Now that the frontend is able to receive current/recent news articles, I've started to work with the others on the machine learning model. I began looking into logistic regression, and how we would use it to predict whether or not the stock price will go up or down the next day.

Week 9

The frontend is now completely done. After the frontend fetches the prediction based on the machine learning model, I rendered the output to the user, so he/she has an idea if the stock will go up or down the following day of the date selected. In addition to any UI/UX changes, I also took some time to write unit tests using Jest. This helped achieve 100% code coverage for most of the components.

Neal

Summary

I worked on two parts of the project: the web scraper and documentation. The purpose of the web scraper is to get real-time news data. I built the web scraper using Beautiful Soup library and Python. I researched various sources to gather financial news data. I ended up scraping Reuters to stay in line with the current dataset's news source.

I worked heavily on documentation. I managed the Kanban board and filled out the README. I supported the team on the essays by providing a variety of potential topics to write about the ethical issues of stock predictions. I created Milestone I, Milestone II, and the Final presentation and helped assign the team parts to say during the demo.

Status Reports

Week 5

I have been working on implementing the web scraper. So far, I have learned how to use the Beautiful Soup API and have tested it out on random online datasets. I am working on running it with our intended data sources (Bloomberg and Reuters) to extract the web pages and store the data in documents. I have been working closely with the rest of the team to keep track of our progress with the project. Outside of developing, I updated the README and Kanban board to keep KirbyUpB organized and following agile practices.

Week 6

I primarily worked on setting up Beautiful Soup. I am working on scraping articles from Bloomberg and Reuters and store it into the document. Aside from development, I updated the Kanban board and requirements on GitHub.

Week 8

I web scraped ~70 world news articles from Reuters using BeautifulSoup and Python. These articles will be ingested by MongoDB and displayed on our news cards for real-time news information. Aside from development, I've been working on documentation and updating the Kanban board.

Week 9

I finished the presentation and updated any documentation. I worked with the team to coordinate a time to meet for practicing our final demo.

Aditya

Summary

I created the insertion script and created the backend. The backend connects the front end to Elasticsearch and the ML model.

Status Reports

Week 5

I have been working on configuring Elastic Search with the Hadoop cluster that the professor set up for us. Currently, our mongo atlas cluster takes about 2 seconds to search ~1500 documents. We project that we'll have about 100,000 documents, so it's clear that our current solution is not feasible.

Week 6

I worked on curating the article data to be used by the front end. I also worked with Ji hwan on how to configure our elastic search cluster to “skinny” down our data and make retrieval times faster. After all the data has been inserted into the mongo cluster, we will be implementing the ES cluster for use by the front end.

Week 8

I successfully skinnied down the data and inserted all of the data into our elasticsearch cluster. There were some hiccups at first but we finally got everything working. I have been working on just rescrapping and reinserting data into the mongo database to be inserted into elastic search.

Week 9

I re ran the scrapping script and attempted to make it a cronjob. I also connected the backend to the trained machine learning model using pickle, which is a python serialization method.

John

Summary

I helped make the frontend with eric which included making the initial mockup/figma, preprocessing the data from the backend to visualize the data, and implemented and tested different ml models

Week 5

Similar to Eric, the frontend is mostly completed and a majority of what is left is just connecting to the backend. I worked on fetching the graph data that is displayed on our homepage as well as setting up the fetching for the data that is passed into the search parameters. The graph uses chart.js and the graph retrieves its data points from the Alpha Vantage api.The remaining endpoints are still be completing such as displaying the article, but the basic fetch calls have already been implemented.

Week 6

I worked on plotting the graphs with current data and am now moving on to working on the web scraper. I have been using both beautifulsoup and scrapy on the bloomberg news data to see which would work better.

Week 8

I worked on the ml model and set up logistic regression for the data. The data had to be truncated and some features were left out because it made the predictions worse. The results were plotted on a confusion matrix and the model obtained around a 70-78% accuracy for our current dataset of 10,000 documents.

Week 9

I worked on the ml model again and did more feature selection to see which features actually affected the accuracy. Tested out other models such as linear regression but found it hard to determine the accuracy as in if a prediction was valid or not. Helped visualized the predictions through confusion matrix.

Ji Hwan

Summary

I designed the database schema with Aditya and helped Aditya with writing the insertion script that processes and pushes the news and stock data to MongoDB. I also helped John with writing our logistic regression script using scikit learn.

Status Reports

Week 5

I have been working with Aditya on ElasticSearch. We have been researching into how to configure it to store our data and how to connect it to our current front and back-end for query purposes.

Week 6

I worked on selecting the data that will be used as index on elastic search so that the queries made can retrieve the necessary information that user needs much faster. As Aditya mentioned, we will be implementing the ES cluster for use by the front end.

Week 8

I worked on selecting the data that will be used as index on elastic search so that the queries made can retrieve the necessary information that user needs much faster. As Aditya mentioned, we will be implementing the ES cluster for use by the front end.

Week 9

I looked up pickle in order to save the model we trained. I asked John to test the model without the sentiment feature to check the impact of that feature which lowered our accuracy by a minute 0.2%. We also tested the model with a larger dataset.

Team Management/Collaboration

Throughout the project, our team focused on sticking with an Agile style of development. This allowed us to develop new features when needed and to iterate quickly once those features were complete.

Typically, we all worked in 1 or 2 week sprints, designating a set of features that we wanted to be complete for a particular sprint. Since we all agreed upon who would work on which part of the application (ie. frontend vs. backend), it was intuitive for us to split up our tasks so that we would be comfortable working on a particular feature, with or without another person.

Once we completed a sprint, we would each talk about what we were able to accomplish, as well as tasks we struggled with and weren't able to complete. In the case of the latter, it was important for us to finish our tasks on time (or as much as possible), since we had less than 10 weeks to complete it. Therefore, we would help pick up any pieces left behind by either pair programming to help solve the issue or have another team member who was comfortable with the work and was available to take the task.

Speaking of pair programming, during each sprint, it was quite common for us to pair program. Since, most of the time, the features we each individually worked on were closely related, pair programming and helping a team member who was behind was not uncommon at all. For this collaboration process, not only did we learn to help out when appropriate, but it also gave us the opportunity to understand the dynamics of working with people who have different skill sets, mentalities, etc.

Design Process

We worked in an Agile environment from the start of the project following best practices for Scrum. Following the diagram above, here is a brief timeline of our design process:

Week 1-2:

Identify the Need

Brainstorm project ideas. We were looking at multiple project ideas that provide a real solution to the average Americans problems. We also were assessing constraints with looking at our 10-week timeline to ensure that we can feasibly finish this project.

Research the Problem

After narrowing down our problem, we did extensive research on whether this problem exists and what is the current market for our problem. We found alternates accomplishing similar goals like CNBC Mad Money. However, no one was delivering a simplified application that provides data-driven trends.

Develop Possible Solutions

We also looked into datasets on financial news. We researched technologies that we would need to implement this project like REACT and Watson. We created a list of potential approaches with technologies that can be used to complete this project. We looked into ML models as well.

Plan: Select a Solution

We ended up consolidating our ideas into a final plan. We mapped how the entire quarter was going to look like and divided up the work accordingly. We worked modularly with dividing up components of the project and used branching, issues, and PRs on GitHub.

Week 3-8

Build Prototype

Week 3: We started building out the skeleton web application. We also gathered historical data and ingested it within MongoDB.
Week 4-5: We connected the frontend and backend to display our news data for our graph and impactful terms.
Week 6: We added the functionality for the news cards.
Week 7: We created a web scraper to gather real-time news data.
Week 8-9: We built our logistic regression ML model.

Week 10

Test and Evaluate Prototype

We tested our frontend to see if all the components are within BEM standards.

Redesign as Needed

This was an iterative process throughout the quarter, but we would adjust the app as needed when we would add a new feature.

Lessons Learned

We learned three key takeaways from this project:

- Research more before jumping into a project
- Planned times to meet for paired programming
- Overestimated our technologies

Researching More

We jumped into the project with a lot of passion and ambition. We overlooked multiple edge cases that caused us to run into issues like figuring out that news articles need to be removed if they are on a federal holiday.

If we were to redo the project, we would spend an additional week doing researching and scoping out potential edge cases and putting ourselves in the position of the user to see what their needs and concerns are for investing in stocks.

Planning Times

We worked very efficiently throughout the project and met up during the weekly labs. Our communication chain was primarily through Messenger. However, we found that we could plan times to meet up every 1-2 weeks to do some paired programming for more collaboration.

Overestimation

When planning out our project, we were confident in our scripts and ML model. However, we found that the queries were taking too long and the ML model was much more difficult to implement. We should have allocated more time toward these portions of our project since they are a huge portion of our system architecture.

Developed vs. Proposed

We were mostly on-par with our initial plan. Here are a few things we didn't have enough time to complete:

1. **Implementing Linear Regression**: We implemented logistic regression for our ML model. If given more time, we would explore linear regression to find a more accurate model to analyze our data.

2. **Dockerize Application:** We would eventually like to deploy our application instead of running it on localhost.

3. **Researching More Sources:** We worked with Reuters. If given more time, we would expand our database to pull articles from a variety of websites (i.e. Fortune, CNBC)