2021 Senior Project Reflective Paper

Oliver Rochester

12/1/2021

Advisor: Dr. Kowalczyk

Committee: John Sarkela, Dr. Appleton

Introduction

As a requirement for the completion of a Bachelors degree in computer science from Northern Michigan University, a final project must be attempted. This project has to have at least 1,000 of functional code and be presented at the end of the semester of which you started it. For my final project, I chose to create a single-page paper trading web application. I chose to pursue this project topic because of my interest in the stock market and finance, and chose to create a web-based application from the knowledge I gained from the client-side programming class I completed.

Description of project.

My project is defined as a full-stack single page web application. A single page application (SPA) is a powerful alternative for multiple page applications (MPA). An SPA interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of a web browser loading entire new pages like an MPA would. An application that is 'full-stack' means that the application has client, server and database that communicate and work together. To fully connect all the pieces, lastly, we have the term ‘paper trading’, this means to buy and sell stocks with simulated money using real-time stock prices.

What does it do?

When you come to the landing page of the application, you first see that this application is a user-based app, and if you want to use the tools of the application, you must create an account and sign in. Once signed in, you are presented with various different pieces of information on the page. At the top left, you see that you have an amount of $100,000. This is the starting amount that all users are given to start trading. Below that, there is an area where it gives you the option to buy a stock and however many shares of that stock. When you purchase a stock, it sends out a request to get the latest price on that stock and calculates the purchase total, then completes your order if the all the requirements are satisfied. Below that, is a list of the top 25 performing stocks just to give the user some potentially helpful stock market information. These are displayed by web scraping Yahoo Finance’s Top Gainers web page.

In the middle column, you are able to search for individual stock information, a search returns the current stock price, today’s open price, yesterday's close price, the 50-day moving average, the day high, the day low, the 52-week low, and the 52-week high. It also displays a graph. When the initial graph is loaded, the 7-day trend is presented. There are also other buttons to display the yearly trend, six-month trend, and 3-month trend. The trend line is green if the overall trend is positive and red if it is overall negative. At the very bottom of the middle column is a list of three current news articles in relation to the ticker symbol that was searched. This also uses an API to retrieve this data. The API does the best it can to find news article related to the ticker because not every stock has news on it every day.

The third column on the far right displays all of the positions that you currently hold. The metadata of a position shows a refresh button, the ticker, how many shares you have, the gains or losses of the position, and an option to sell. If it is a loss, it will be red, and if it is a gain, it will be green. Above that shows the total profit amount, so you don't need to calculate in your head.

What problem does it solve?

This type of application mostly solves one of the biggest problems that modern-day stock market traders face. It's the ability to buy and sell stocks and test out trading strategies while not having to risk your own, real money. This is a very helpful tool and can help people steer clear from spiraling down into a financial depression. The reason that this 'mostly' solves one of the biggest issues is that when you buy or sell stocks using this application, your actions have zero effect on the current stock price, like it would if you used real money on a real trading platform.

Describe how you designed the project.

Most of my inspiration for the feature design of this application came from a different paper trading web app, called Alpaca Trading. The biggest feature design that I used was the 'Buy Stock' feature and the positions list feature. The design layout came from personal preference. I implemented the dark theme because I am a big fan of the dark theme on almost all apps and when I program using VSCode.

Describe how you completed the project.

I would consider this final project my first long term project. In order to successfully complete it, I had to use my time management skills more than I ever had before. I have heard stories from previous years where students either try to tackle a project bigger than they can handle, or finish a project where they expected to take longer. For my project, this was not the case. I believe I chose a project that required a good amount of work, that kept me busy all semester and allowed me to have a good balance between this, my other classes, and work. There were some stretches where I lost motivation to keep working on the project, but were made up for with hard work during other times.

What languages did you use?

I used a variety of tools and languages in this project. The main language I used was JavaScript, but I did not use vanilla JavaScript. For the client-side code, I used a front-end JavaScript framework called Vue.js. Vue.js allows you to store data in a state-like object, and call this data from the html file to be displayed. Whenever you change the state data, it automatically re-renders on the page. I also used a fair amount of Python in this project as well. I found that the Yahoo Finance API that I use to retrieve stock information was much cleaner and easier to use using python rather than making an HTTP request. For client-server communication, I used AJAX. I chose to use AJAX over other options like Socket.IO because there are no shared data structures between all the users of the application. To store data, I used a database called NEDB. NEDB is basically a light-weight version of MongoDB. You could compare this to SQL and SQLite. One of the reasons that I chose NEDB is because it stores data in JSON notation. This makes the data very easy to work with since JSON and JavaScript go nicely together. To implement the stock charts in my applications I used a JavaScript library called chart.js. I had never used this tool before, but it was very easy to use and was well documented online. To retrieve stock market information, I used three different API's. The first was the Yahoo Finance API that I talked about previously. I also used the Polygon.IO API and the MarketAux API. Polygon was used for getting historical data in order to create the graphs, MarketAux was used to retrieve news articles about certain companies, and Yahoo Finance was used to get basic stock information. To manage my code base, I used GitHub. After any change had been made to the code, I would always push the changes to GitHub immediately. The last thing anyone wants is losing a large amount of progress made on a project. Lastly, to get the list of the 25 top performing stocks, I used a python library called BeautifulSoup, which is very powerful tool to scrape data from the web. The only difficult part about using this library, is using regular expressions to parse the scraped data.

What are the new technologies you had to master?

The new technology that I had to master was using NEDB. There are a lot of nice features with this data storage tool. First, all you need to do to get it loaded into your project was an NPM install and then a require function at the top of the server file. Also, like I talked about previously, it uses JSON which is extremely easy to read to the human eye. There also were a few issues I had with NEDB. Firstly, I ran into a few syntax issues when trying to query the database. The biggest issue I had was with scoping. When making a query from the database, after the data is returned, the data is required to be used within an arrow function. If you try and store the returned data in a variable or data structure outside the scope of the arrow function, it will not work. This resulted in a little bit of messy nested code. This was one of the biggest bugs I had to face and took me the longest to figure out.

What went well?

I would consider my biggest successes for the duration of this project was the ability to create a project that is a real-world tool with good functionality. I am also proud of my ability to manage my time efficiently in order to properly deliver a final product. Another feature I am proud of that I almost did not implement were the stock graphs. I feel that the graphs bring a good sense of visualization to the application and it really brings it all together and looks complete.

What didn't go well?

If I had to pick one, I would say my biggest issues with this project was running into the fact JavaScript is a single threaded language. One of my goals was to have an automatic and constantly updating positions list. I wanted to create separate threads where they would be continuously calling the Yahoo Finance API to be checking for a price change. I was able to find potential techniques online, but was not able to learn and implement the strategies due to a lack of time. Another issue that I ran into was that making a request to the Yahoo Finance API is so incredibly slow. One of the reasons behind this is because it is one of the only free stock market API's that actually give out decent data. With this fact, Yahoo creates a delay for their server responses due to the number of requests they get from around the world to prevent server overloads. Due to this issue, I was forced to except the fact that when buying, selling, and updating positions, it was going to be a little slow. The last moderately big issue I had was that there were some extremely good stock market API's out there, but they cost a lot of money. I ended up having to use all of the free versions of each API which restricts you in some ways. One of the restrictions is that you only get a certain amount of API calls per minute or per day.

What might you have done differently if you could do it over?

If I had the opportunity to restart this project, I would have tried harder to learn about the potential work arounds when it comes to "multi-threading" in JavaScript. I would have looked more closely into using promises and async/await functions. I also would have tried harder to reduce the number of breaks I took from the project. Without the breaks, I believe I could have made the UX friendlier, and possibly added another feature or two. Also, would have maybe asked for funding from computer science department, so that way I could be obtain a higher-level API Subscription to receive better data.

Conclusion

To conclude on the reflection of my 2021 Senior project, I would say that it was an overall success. I was able to plan, execute, and produce a final product within the given time constraint, while managing other aspects of my life. I am proud of myself for the continuous effort I put in through out the semester, even though there were times where I thought I was not going to be able to finish it. I would like to thank my advisors and project committee for approving this project because it is something I am passionate about. Lastly, I would also like to thank the entire Computer Science and Math Faculty for an extremely good experience during my 4 years at Northern Michigan University.