ECE FINAL PROJECT REPORT

For my final project, I decided to make a Paper Trading Simulator, I am very pleased that I was able to implement real-time market data into this program. Doing this was difficult as I first had to find a provider that allows access to their API and find a tool to scrape the data from the website. I chose to use AlphaVantage for the API access, primarily as it was one, I found to allow free attempts. I was limited to 25 requests a day which also limited my testing but was enough to ensure that it was functioning. In terms of the data scraping, I found that libcurl seemed to be a common library and only needed the homebrew dependency, which I already had installed. This process was the largest learning curve throughout the project as I had never attempted to scrape web data. After various tutorials and research, I was able to get this implementation to function.

A screenshot of a computer program

Description automatically generated

Stocks.CPP

Starting with the "stocks" C++ file there are 3 main functions in this script.

Write Call Back:

This function is part of the web scraping process and is what processes the HTTP request.

ExtractPrice:

This function is how the price is found from the AlphaVantage Api response. Since the Alpha Vantage request returns Various information about the stock other than the price, we need to find the information that Is important to the purchase of an asset, the price. This will sort through using what we know about the returned information.

GetFromAV

This function is the main functionality of this part of the script, it first requests the data, passes the API key, and then uses the functionality from the ExtractPrice function. It then returns the price of the passed in Ticker, along with error checking.

Account.cpp

The C++ script titled "account.cpp" deals with "everything else", this includes account creation, password validation, and assets. The Assets section has its class as it is the most complex as it needs to be able to handle viewing, saving, loading, and purchasing assets.

Here are the functions within this Class and their functionality.

ViewAssets:

This function's purpose is to loop over the user's assets and print out various information doing calculations such as their total account value, and profit/ loss, as well as printing information about their assets such as current market price.

purchaceAssets

The purpose of This function is the second action that a user can perform, adding a set amount of shares to their portfolio, This function takes the user's ticker information through a command input as well as asking how many shares they would like. It also checks to see if the portfolio already contains assets of this name so It can add them, but it assumes they will correctly capitalize as shown in this photo. This could be fixed by casting the lowercase letters to capitals using the ASCII table, but I did not see this as a necessity as it functions if the user knows to enter the ticker with consistency. The final purpose of this and the most crucial is pushing this into the vector accordingly.

A screenshot of a computer

Description automatically generated

LoadAssets

The purpose of This function is to load the assets already existing in the file into a vector that can be read, past, and modified in later actions. This is how the system "remembers" previous actions but is just reading what was written in the last session. This function is called every time someone logs in

SaveAssets

This function is how purchased assets are written to a file with the username of the provided account. This method of saving allows for easy storage and purchase history for each account and avoids cluttering a single file with all user's purchases. Although this function is short, it is crucial to this program's functionality.

CreateAccount

This function is used to keep track of who is using the program so that their assets can be viewed. This most definitely is not a secure way of storing user information especially as all user's information would be stored in a single text file without hashing, but for this program, I disregarded this. All this program does is store a username and password and write it to the text file where account information is stored.

Main

The main function is where most of the user interaction comes from, it will direct the user according to their decisions to the proper function as well as run what is needed for them to do what they choose. This is also where authentication is done and again is not the most secure.

Known Flaws

* Case sensitive for Ticker Prices
* Exits the Menu after any action is performed.
* If an unknown ticker is added, it will be placed in assets with a value of $0, and alpha vantage will return an error.
* Can only process 25 requests a day ( due to free API limitation)

A screen shot of a computer

Description automatically generated

EX-"Tesla " is the name of a company, not the Ticker name the alpha vantage Error states it cannot find a value as the name is not a ticker.

Final thoughts

I am very happy with the results of this project as I was able to use tools I wasn't familiar with to learn how resources may be used in real applications. I also created a basic account system where user information is stored which in this case is simple but plays a large role in the overall software industry. Not only this, but this system actually could be useful if expanded upon as it could be used as a paper trading system to further someone's understanding of financial markets without the risks. That being said, other tools out there do this functionality at a higher level.