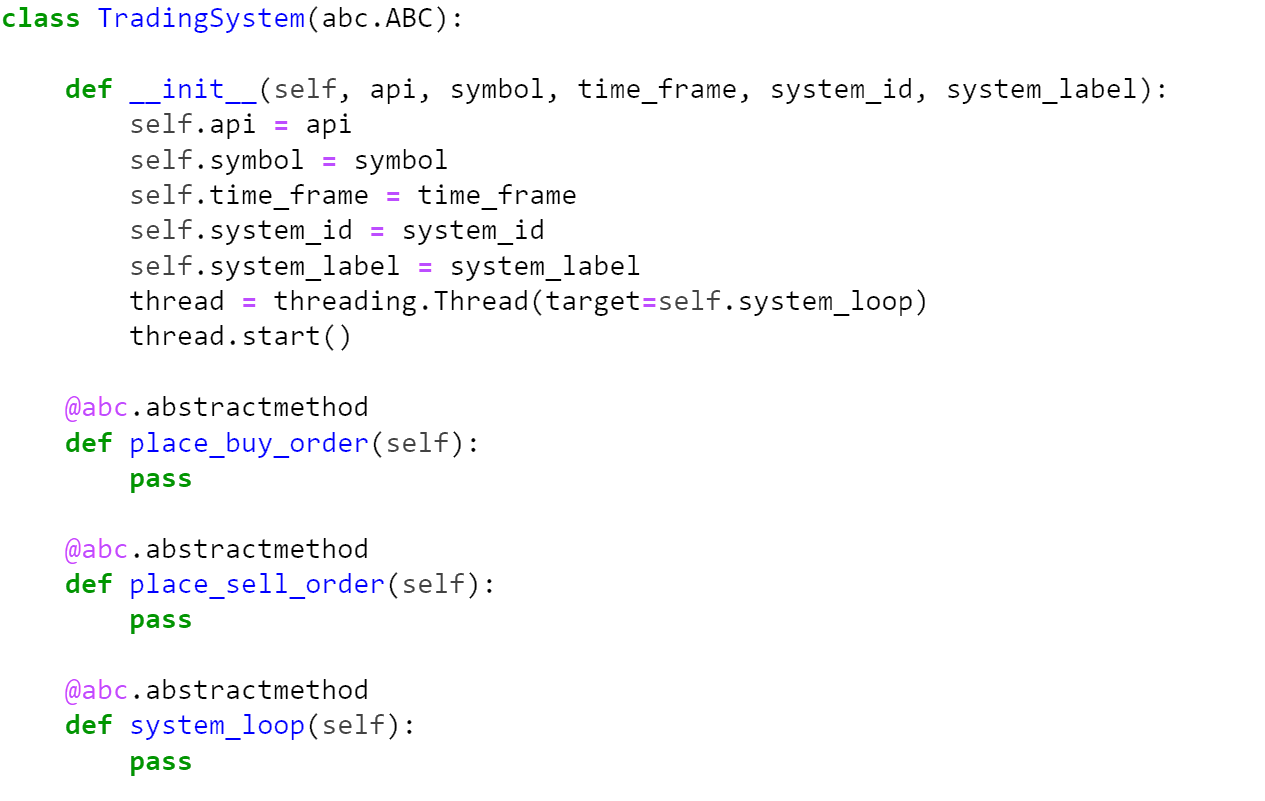
**Mini Project Report**

# Introduction and Overview

This assignment is about stock sell/buy prediction based on real stock data. In this assignment, we are asked to pick five stocks that had a good performance during the past year from NASDAQ, and almost all the tasks were based on these five stocks. The datasets record the data including date, opening price, high price, low price, closing price, volume. They could be downloaded from alpaca. markets used alpaca API. The content of the checkpoint report could be divided into the following parts: firstly, show the program and modules in stock prediction; additionally, explain the methods and metrics of back-test of each of 5 stocks and across them; besides, improved the model to the neural network and trained it; furthermore, applied the model on paper trading and ran it live; finally, mention contribution of each team member.

# Trading system development

The main idea is to construct an abstract TradingSystem class so that we can implement custom rulesets for each type of system we wish to trade. The code is fairly straightforward, allowing us to initialize the system and thread an infinite loop.



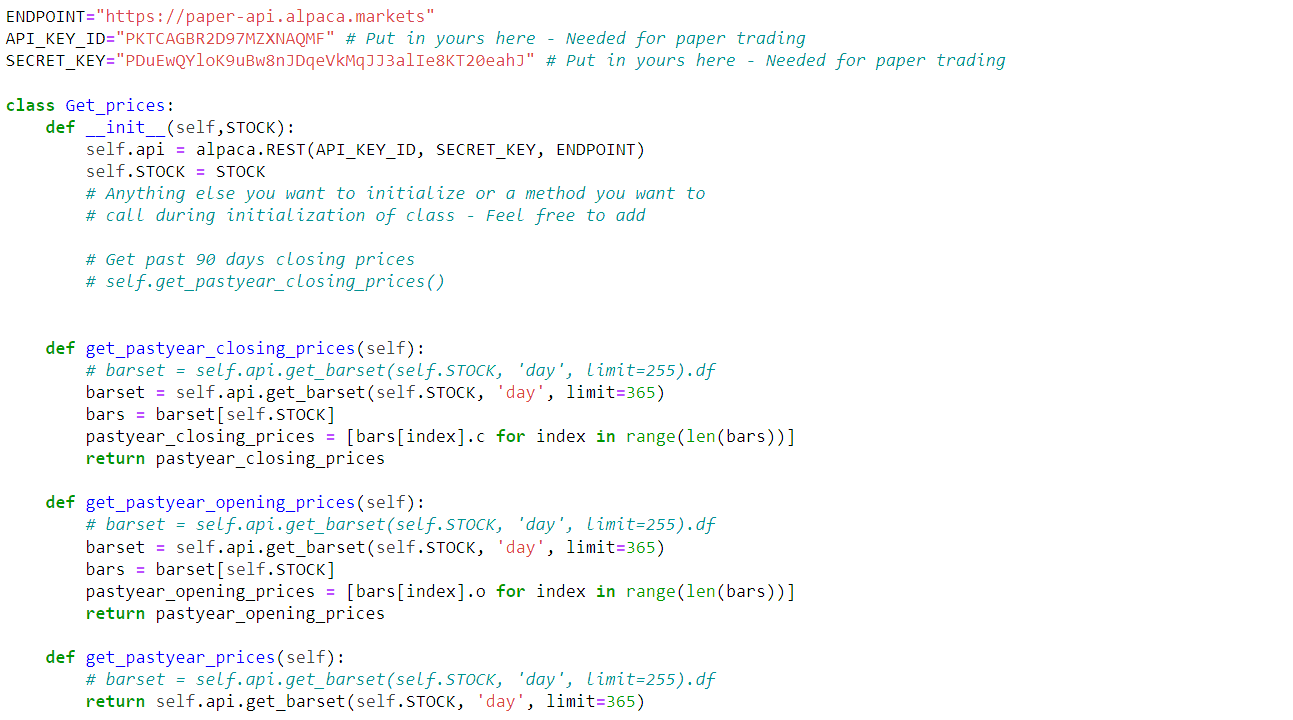
# AI trading model development

### Download the dataset

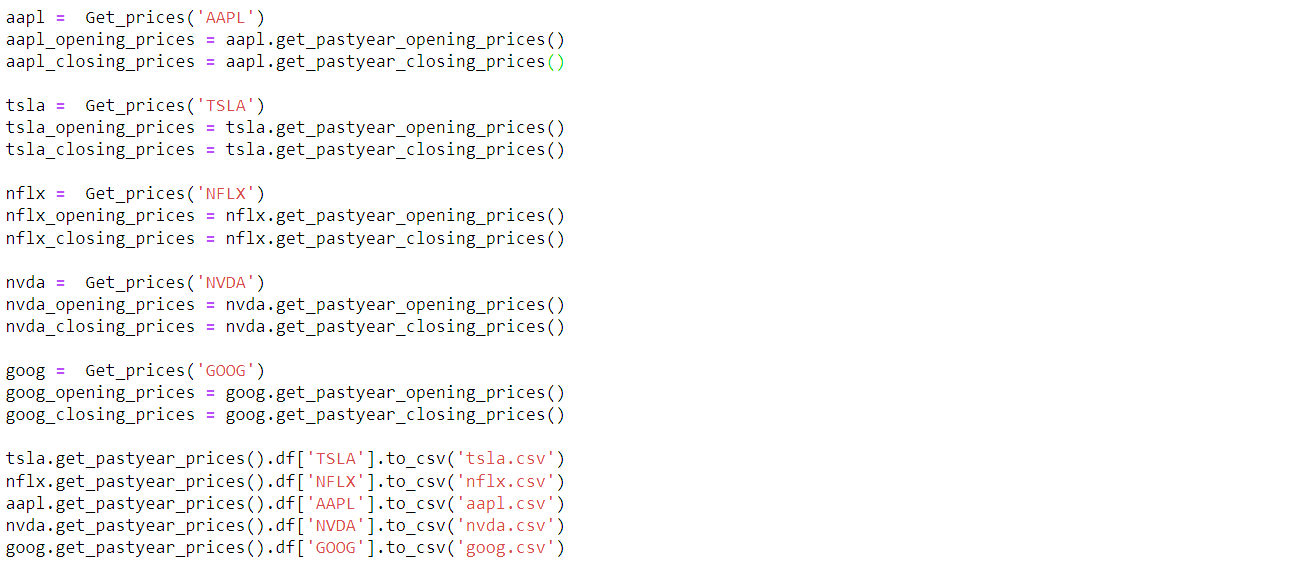
First of all, the five stocks that we have chosen from NASDAQ are AAPL, TSLA, NFLX, GOOG, and NVDA. These five all have good performance during the past year and we suppose that we can have much more benefit on them.

Connect to a brokerage firm, which will allow us to receive real-time data on transactions in securities of our interest. In this article, I'll be using Alpaca, the easiest free way to get started with algorithmic trading, and for our purposes, AI trading. Create an account and go to the dashboard to generate an API key.

To get the prices of these five stocks, we defined a class named Get\_prices, and in this class, we had some functions that obtaining the past year opening data, closing date, and complete data.



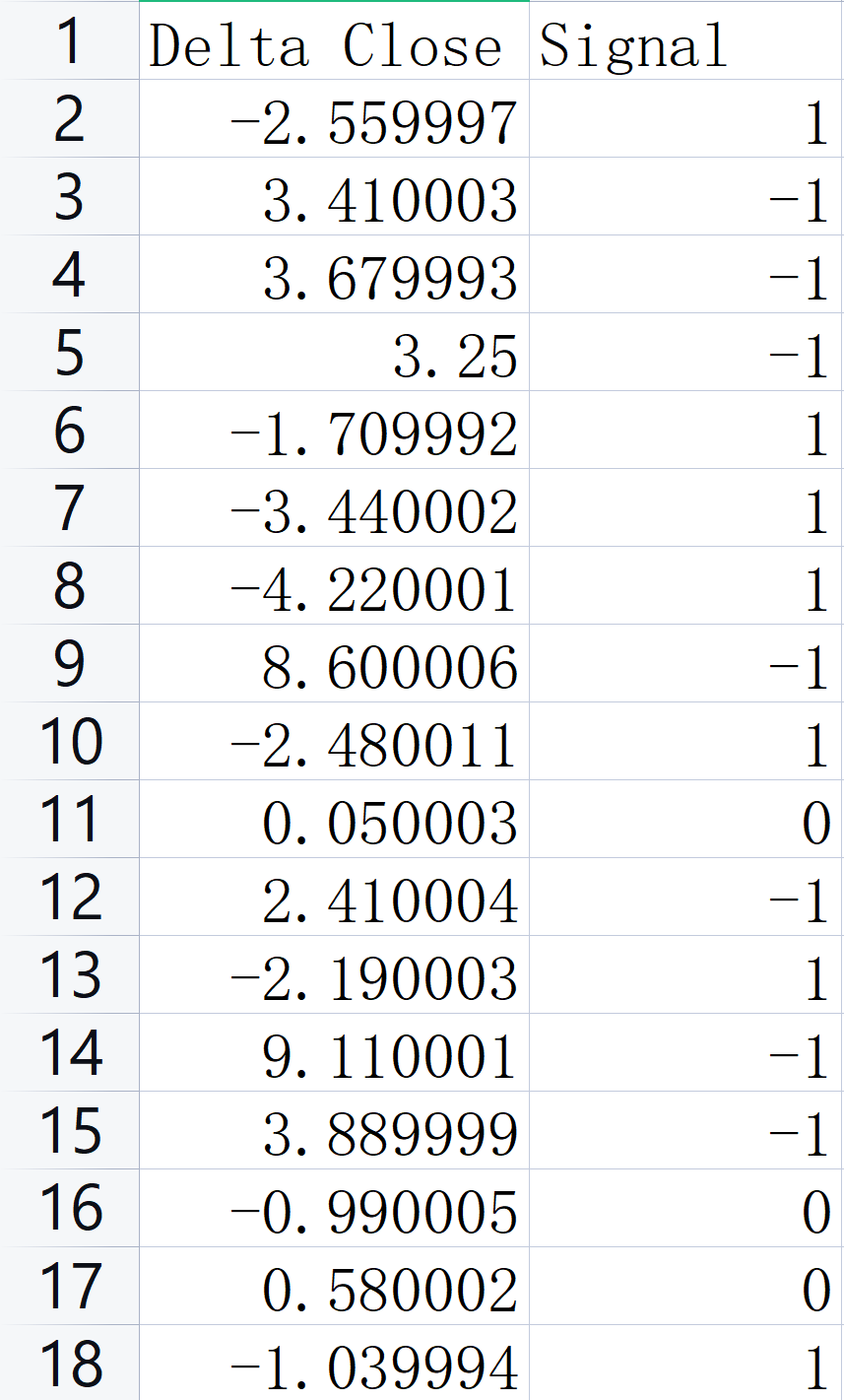
Then called this class to download the datasets and saved them as CSV files successfully.



### Preprocess the dataset

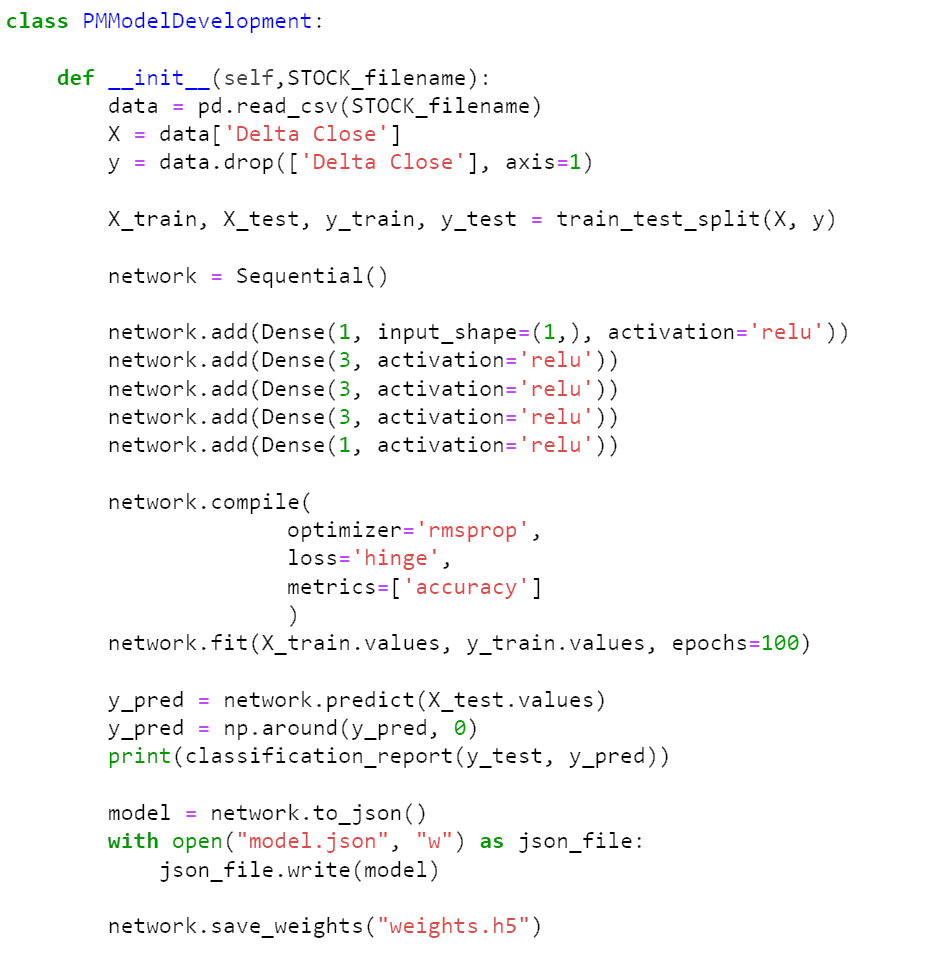
For this system, I will build and train an AI model as a portfolio manager for my system. The idea is to train a neural network to buy at a certain negative change threshold and sell stock prices at a certain positive change threshold. We are essentially teaching our AI to buy sediment and sell sediment. To train this neural network, I will build and annotate a dataset based on weekly historical market data for stocks and create a function called a signal that will produce values in the set {0,1, -1} based on changing thresholds.

To obtain an intact dataset of neural network training, including labels and predictors, we preprocessed the dataset: let the label be three different values ( -1, 0, 1 separately). If the stock’s closing price on the current day is lower than the stock’s closing price 7 days ago by $0.5, it means that the stock is moving lower, and the label is assigned as 1, the stock needs to be sold. Or if the price is higher than that of a week ago about $0.5, which means it is moving higher, the label is set as -1 and we need to buy it instantly.

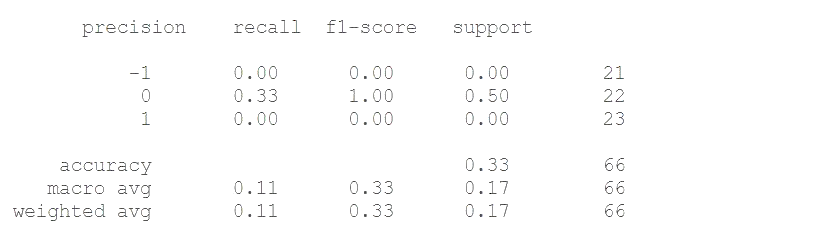


### Structure of neural network

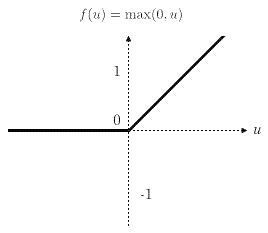
Now consider the architecture of this neural network. We will save the weights after backpropagation so that after successfully testing the model, we can deploy it.



We will use a sequence model, in this case, the choice of activation function is crucial, if we close our eyes and choose ReLu with a binary cross-entropy loss function, we will get a confusion matrix that looks similar to the following:



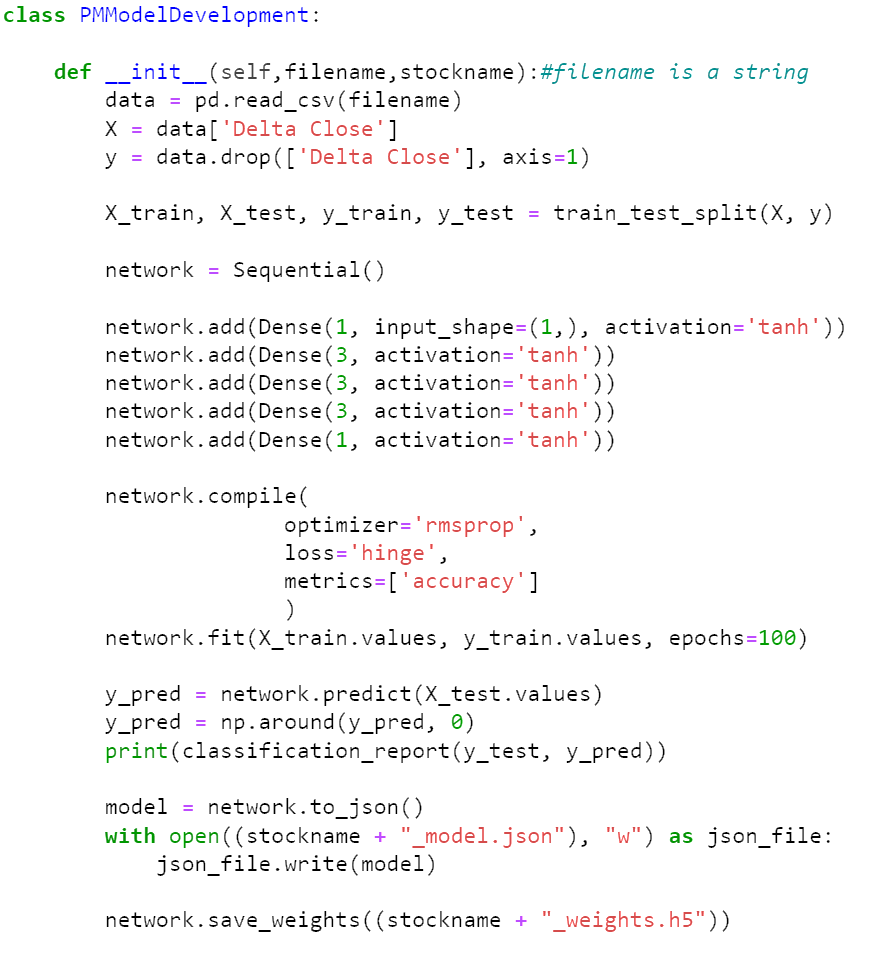
Actually, let's visualize the ReLu function, we can easily get this graph:



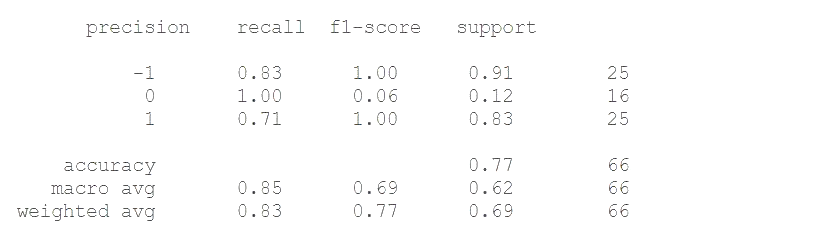
If we set negative input to zero, or expect binary output using loss function, hyperbolic tangent function, and hinge loss help here:



Updating our neural network, recognizing where we went wrong, we have the following model:

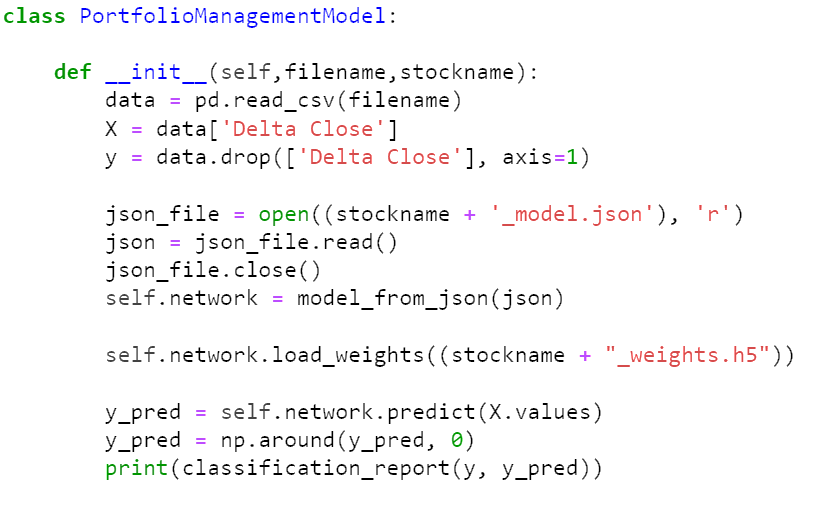


After training, we found a significant improvement in classification reports.

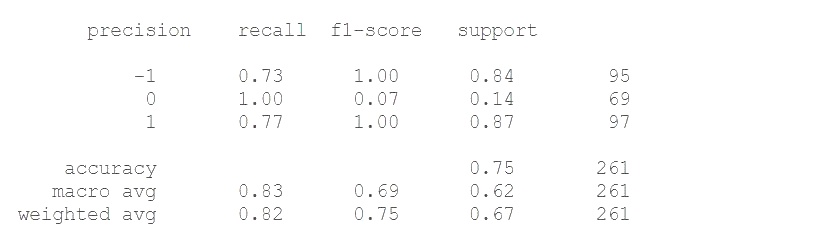


### AI Portfolio Manager

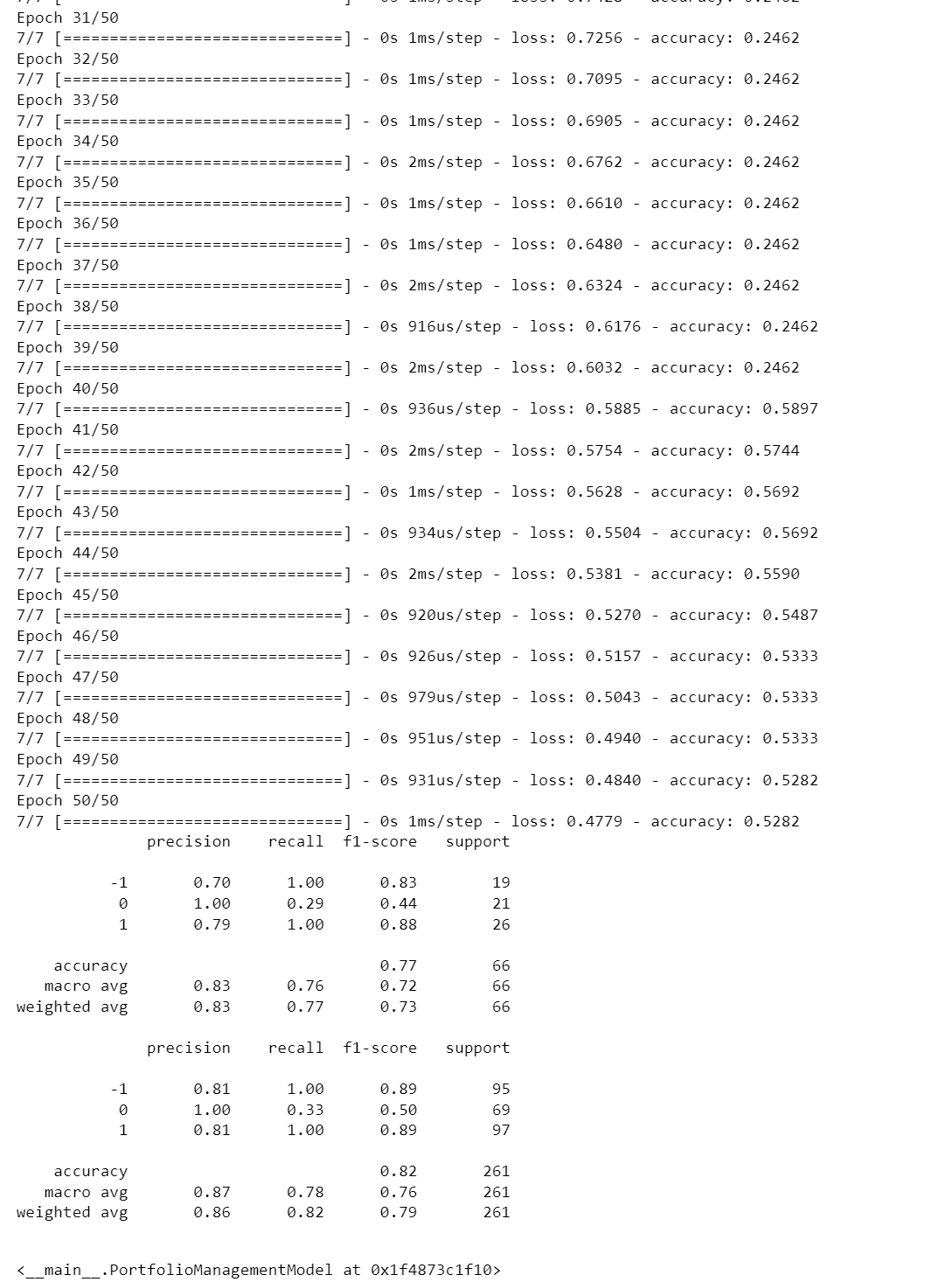
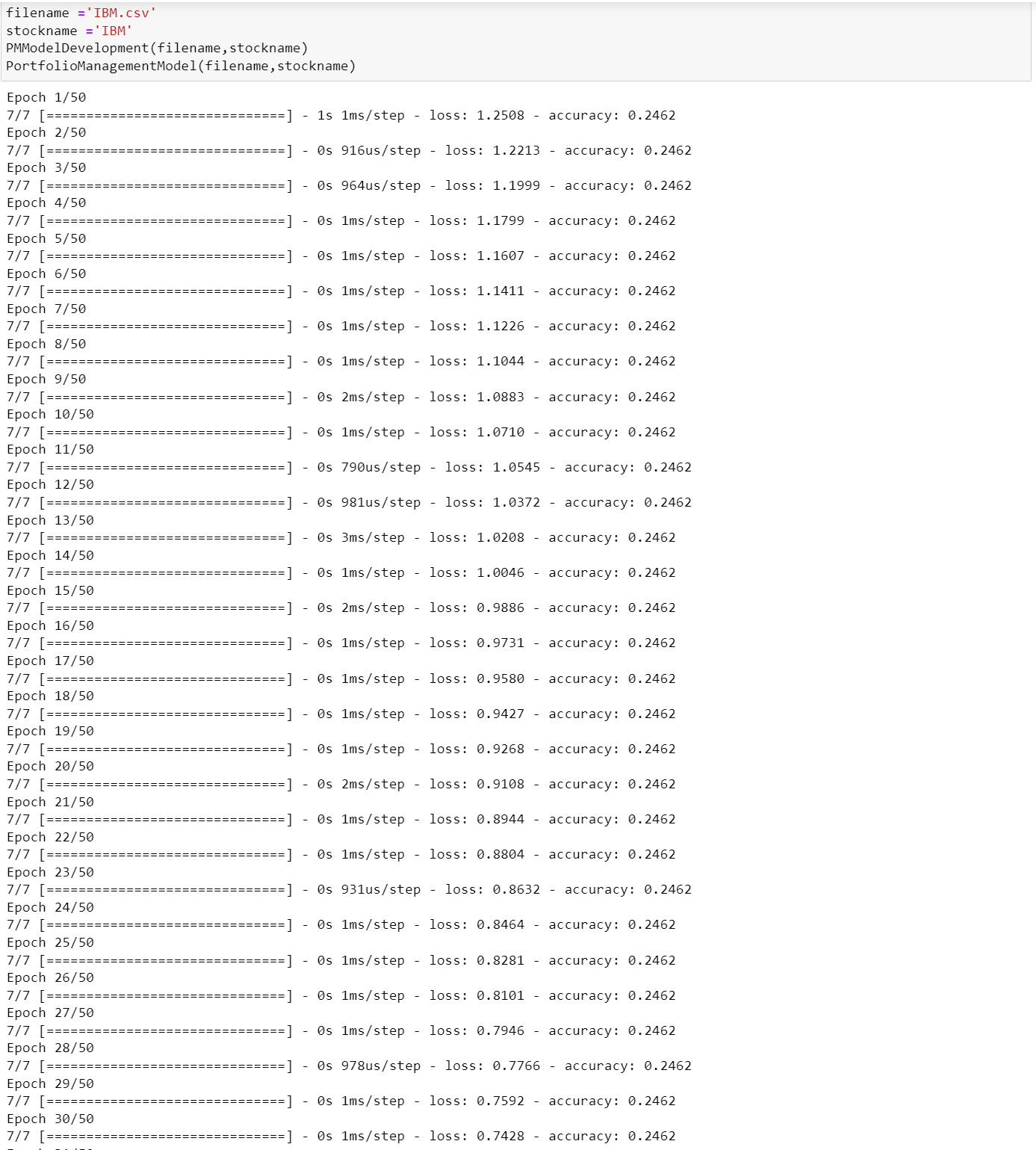
We've successfully developed our model by saving the model and loading it into the class that specifically hosts it



We verify the structure of the neural network and correctly loaded weights by looking at the classification reports for the entire dataset.



The result of training is as follows:



# Deploy AI models to trading systems

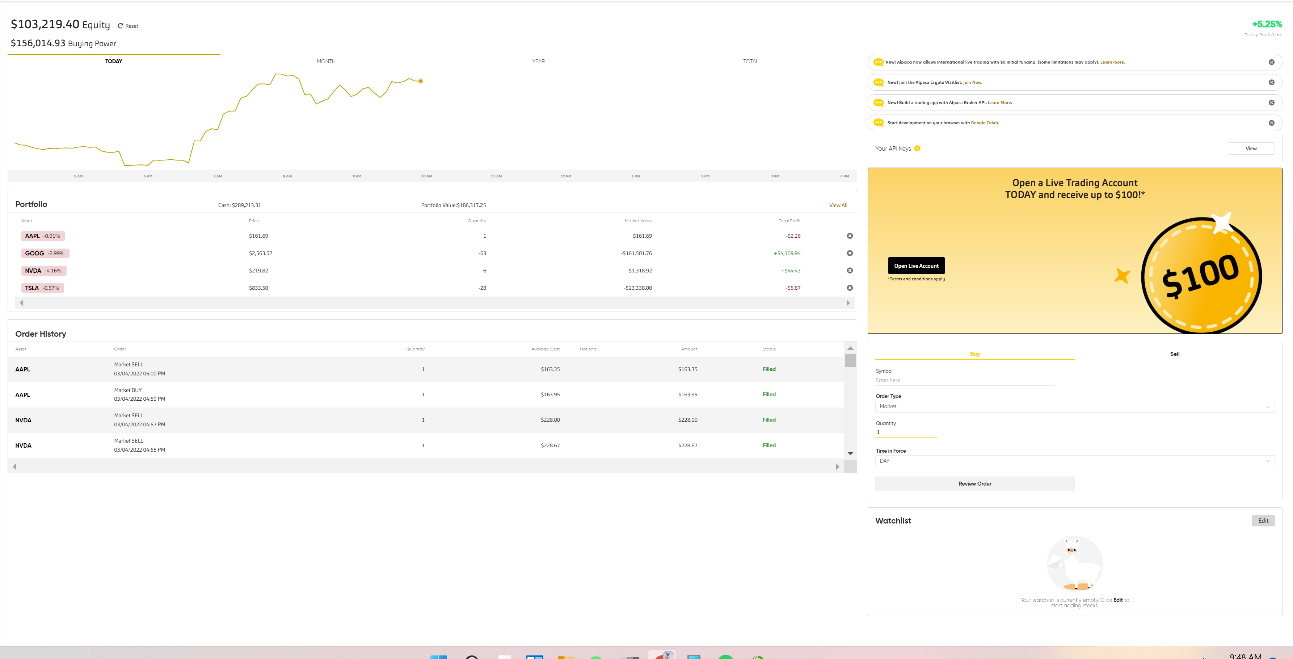
When rethinking the implementation of the abstract TradingSystem class, we have PortfolioManagementSystem. We updated the abstract functions to fulfill their respective purposes.

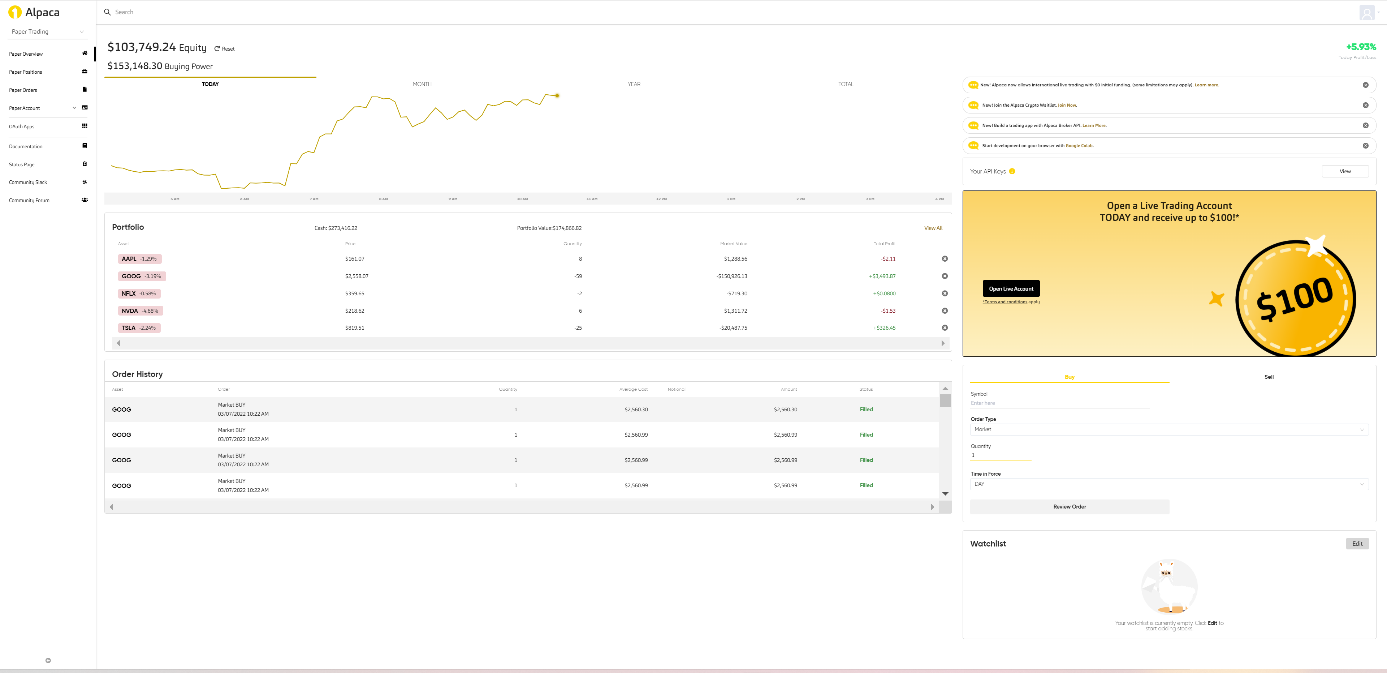


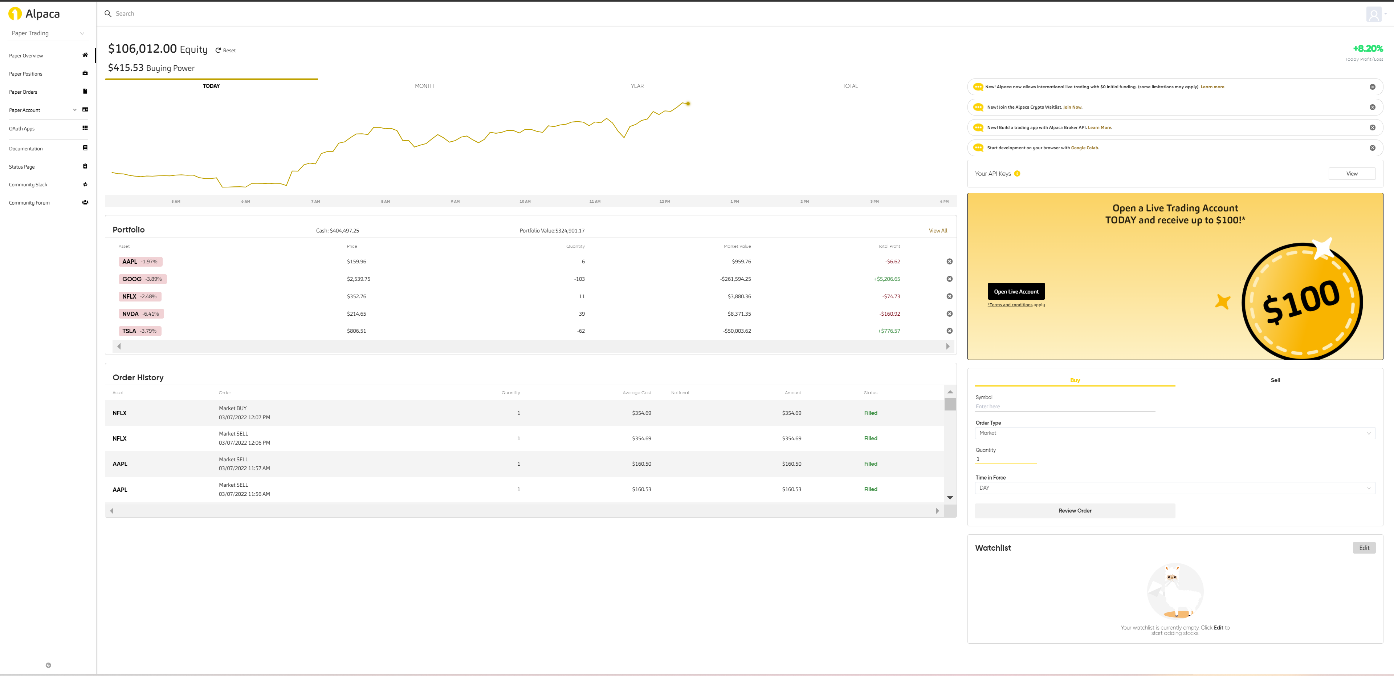
Let's talk about system\_loop. system\_loop initializes variables for close this week, close last week, current delta, and days. The infinite loop (the thread of the concurrent system) is responsible for collecting data once a day and determining if we have reached the weekly split. After reaching the weekly split, the variable is updated and we consult our AI whether to buy or sell.

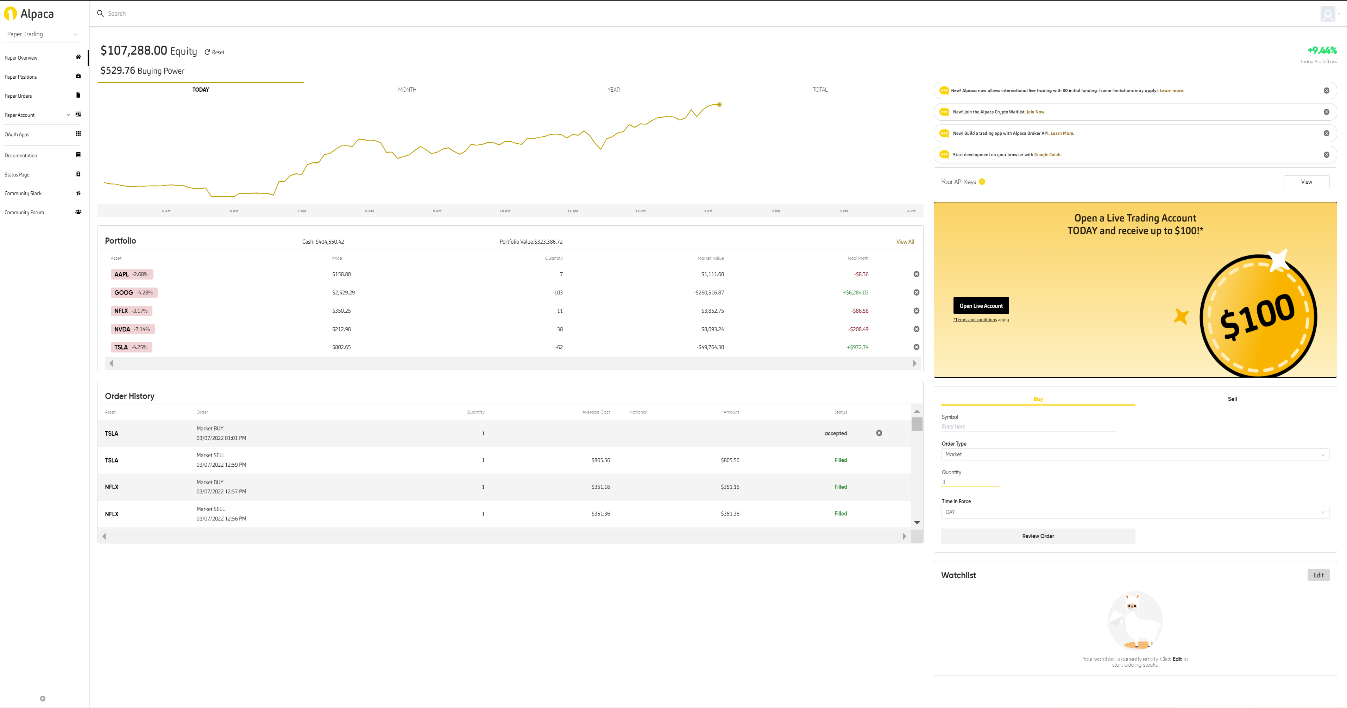
# Paper trading

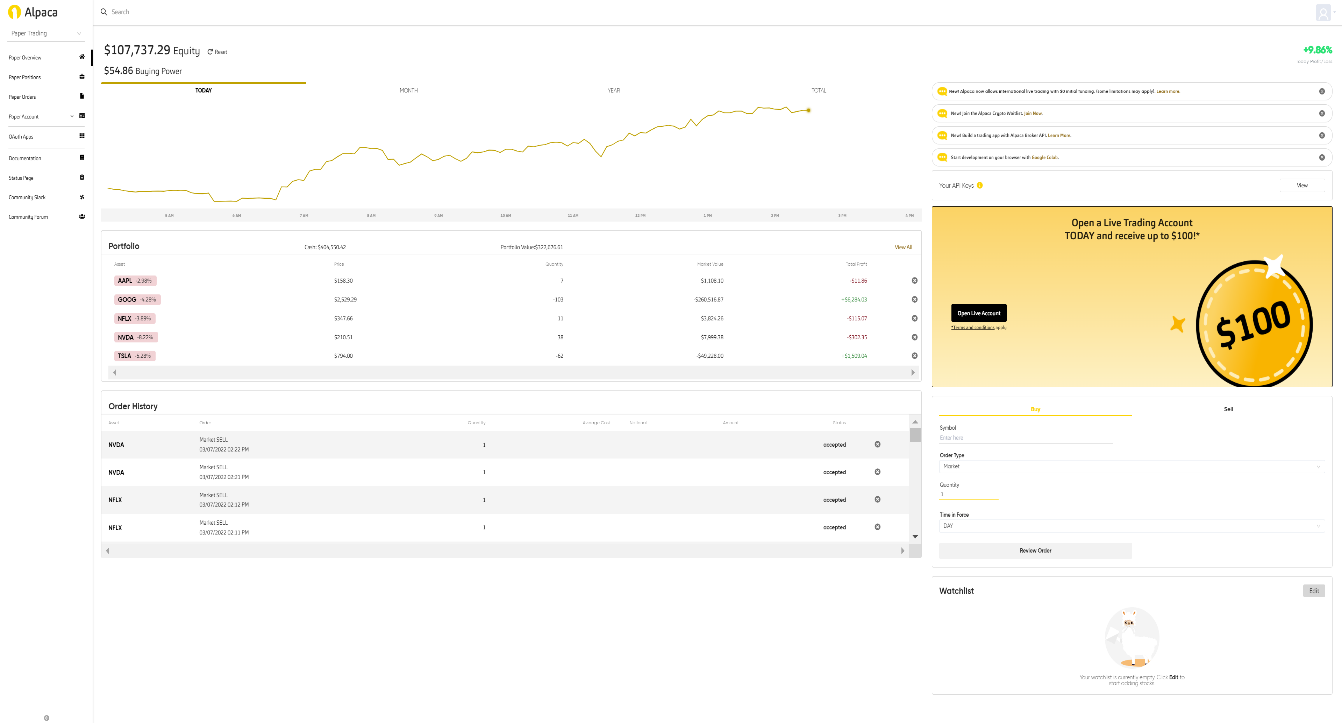
The result of paper trading is as follows:

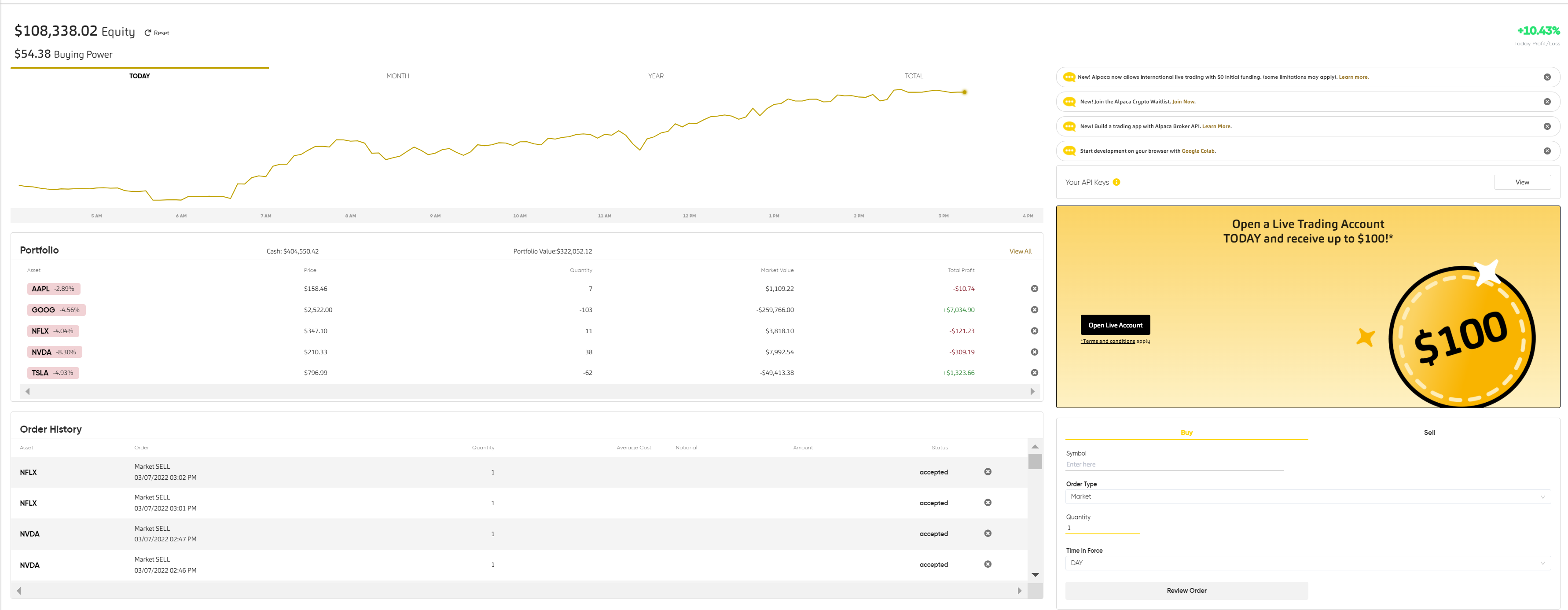


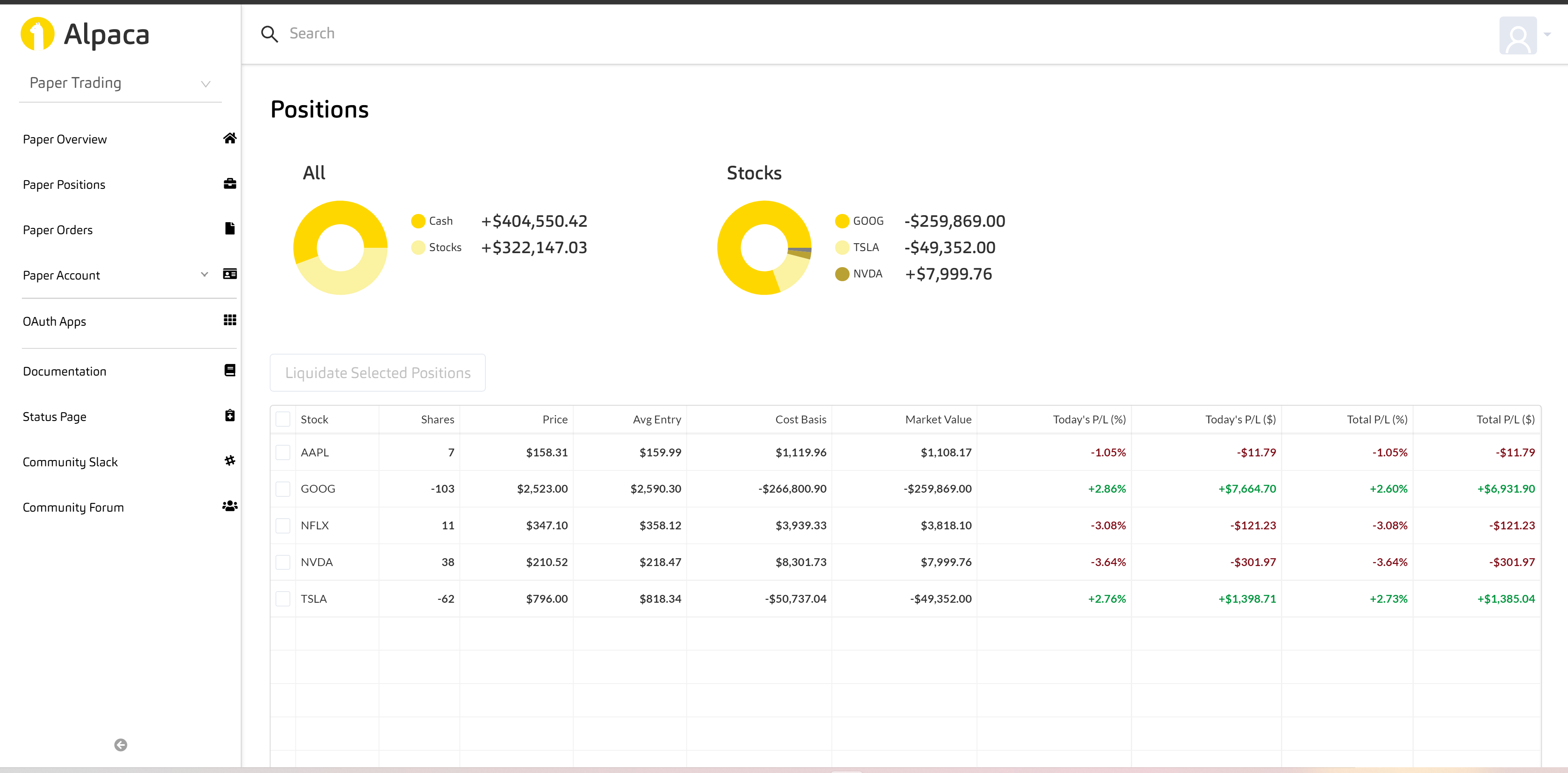
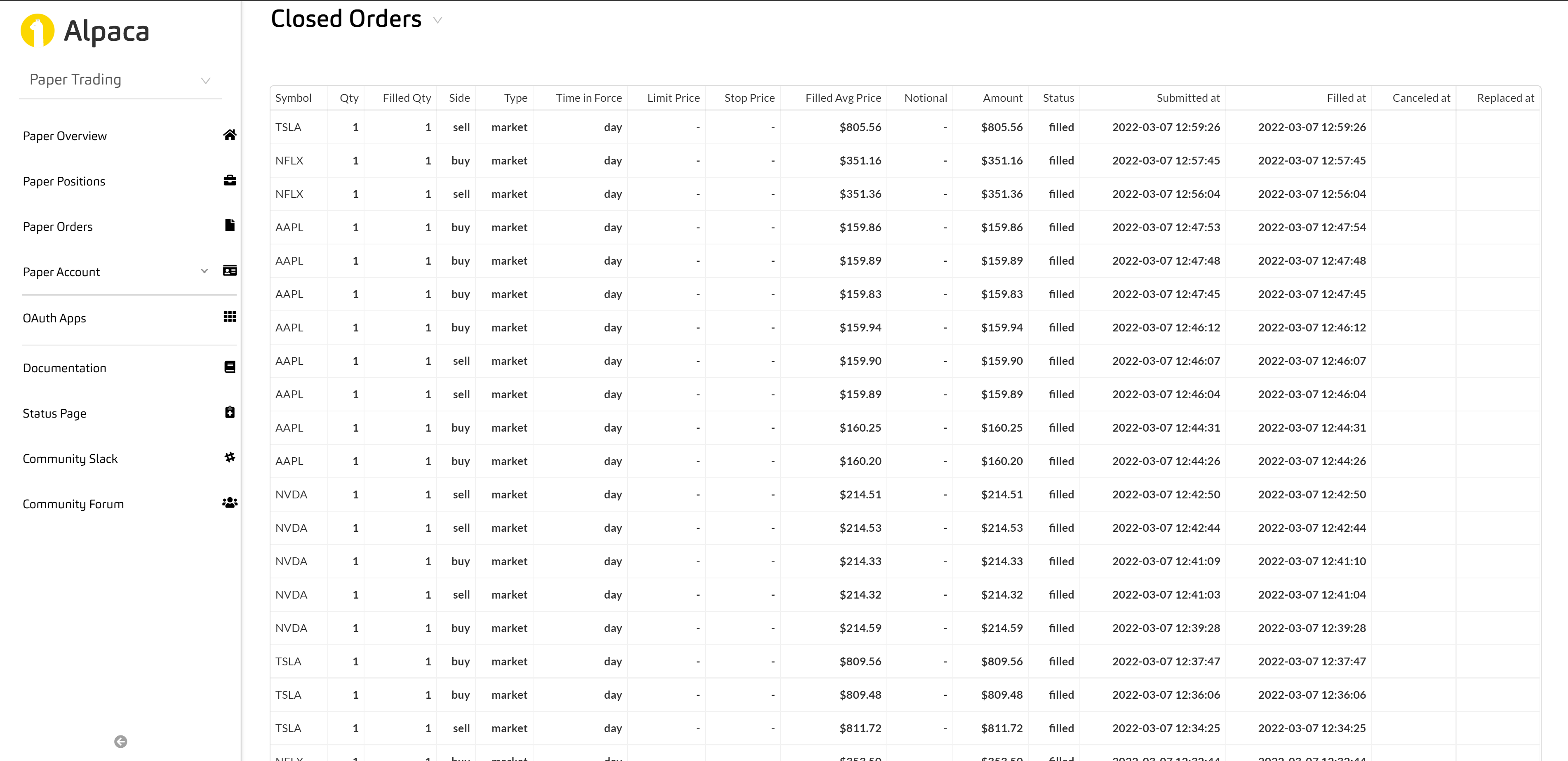


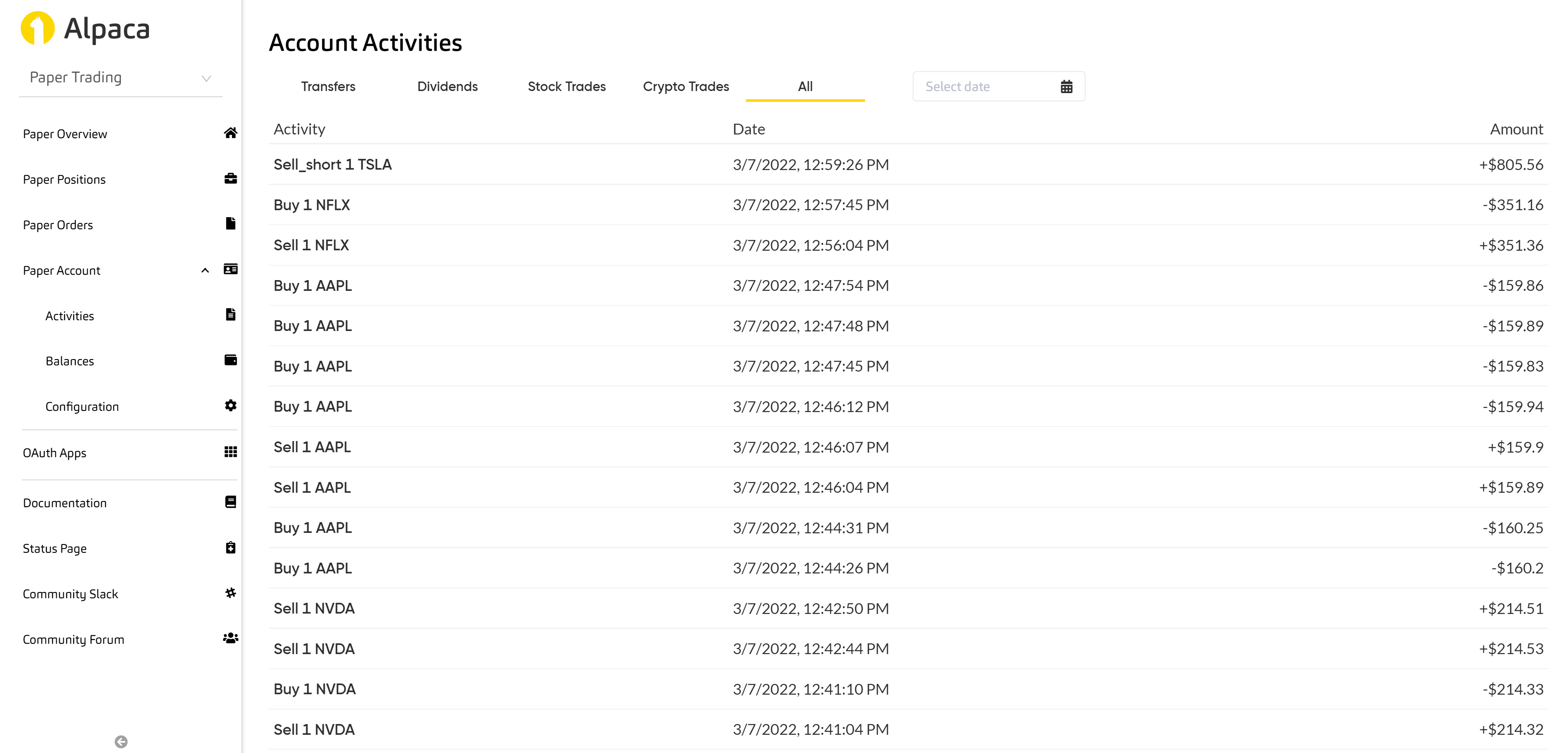


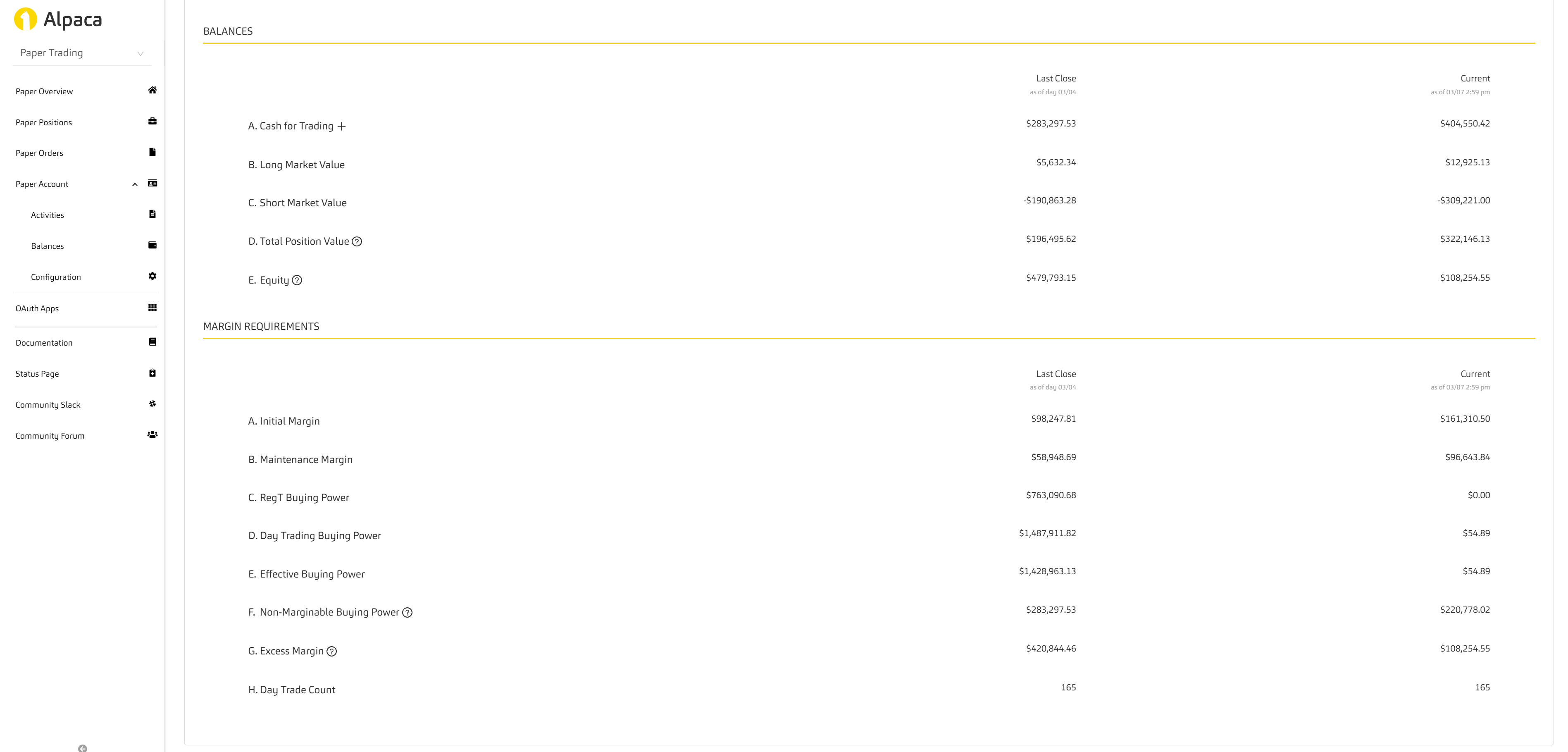












As of March 7, the final total asset is $108338 and the ratio of Profit/Loss is 10.43%.It can be seen that the model works well.

# Conclusion and contribution of each team member

We discussed how to connect to a brokerage, specifically this example of Alpaca. We then created the TradingSystem class itself and its inherited fields and implemented the class in a system dedicated to portfolio management. Afterward, we built an AI model to make trading decisions and discussed the lack of mathematical understanding behind the scenes. Finally, we deploy the model into the implemented system, giving our AI the ability to buy, sell, and hold.

In the mini-project, there are two people in our team: Xiangyu Gao and Rui Li. Xiangyu is mainly responsible for the model establishment, writing main codes, and parameter debugging, while Rui is mainly responsible for finding reference materials, correcting codes, and selecting model methods. Throughout the project, we found each other online and decided to team up together. During this time, we were coding face-to-face three times a week, because face-to-face communication allowed us to program more efficiently. Although we have our division of labor, the overall procedure is the result of both of us. We all feel very happy to work with each other.