**Project Description**

Name of project: TradingView Lite: Stock game extraordinaire

The tradingview lite application allows users to pull up a stock's chart with the click of a button. In addition, users can take trades on their favorite stocks and track their portfolio with access to portfolio metrics.

**Competitive Analysis**

Tradingview: Obviously, tradingview is what this project is based off of. Tradingview is built on a much more robust react platform that allows graphics to be very smooth. In addition, Tradingview allows users to create their own scripts through an implementation of javascript called pinescript. My project will be different from this because it will be built on tkinter graphics with charts completely self-generated, and it will offer a paper trading platform that has portfolio analytics.

TowardsDataScience StockProjects: These projects often result in very useable, interactive charts that can be embedded anywhere with very nice graphics and let the user gain the piece of information that they are looking for. My project will be different because it will use tkinter to generate charts as opposed to a module like matplotlib or plotly. In addition, it will offer portfolio analytics tools that these projects don't contain. In addition, I will be utilizing actual market data from an API as opposed to being from the finance module.

**Structural Plan**

Structurally, the plan will be to have two files. One called iex.py and another called tp.py. Iex.py will be the one doing all of the calculations and drawing the graphs and handling the portfolio analytics. Tp.py will serve as a client file for iex.py and use its functions and the stock class to represent a stock and run all of the calculations and display them.

**Algorithmic Plan**

One of the most algorithmically difficult parts of my project will be creating the charts all by myself. This will be accomplished using the data from the iex API as well as the cmu_112 graphics module. I will calculate the position of the candlesticks by calculating the amount of pixels that I must represent per dollar move in the stock. I will then multiply this value with the difference in the open, high, low, and close prices of the actual stock and add them to the pixel height of the bottom of the graph. This will result in a unique function for finding a y-value in pixels from a "y-value" of the stock. Then, I will graph the body and the wicks separately based on the open, high, low, close pixel values.

Another one of the most algorithmically difficult parts of my project will be figuring out how to save charts that have been drawn on by the user. I intend to do this by saving all of the y-values

of the candles and the coordinates of the lines in a list and just redrawing them whenever the user wanted to redraw them.

Aside from these things, I will also need to figure out how to do the portfolio analytics (a lot of coding math into the values) and I need to figure out how I will be able to display all of this information.

**Timeline Plan**

TP1: Be able to draw the candlesticks for one stock and have a rough UI for the trading and analytics portfolio

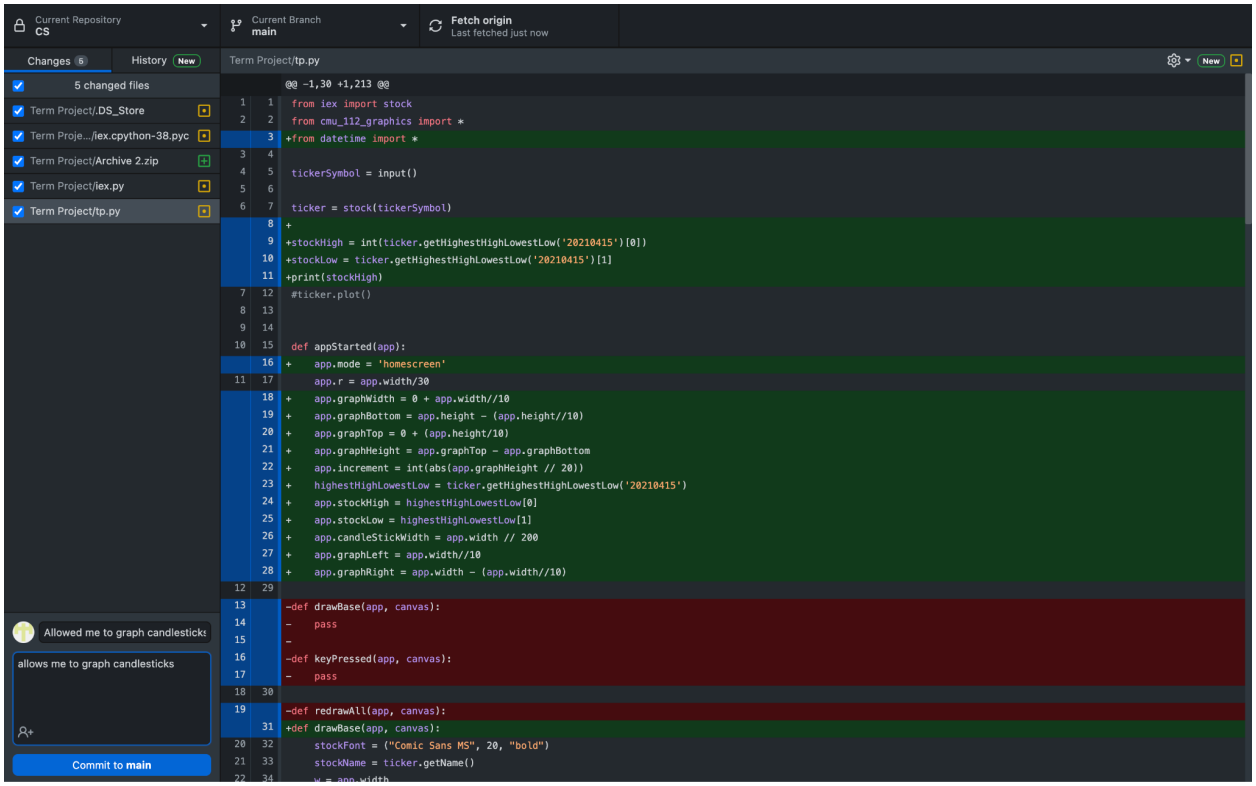
TP2: Be able to zoom in and out of the graph using buttons and be able to make a trade and have portfolio analytics for the portfolio. Be able to "remember" the user's previous settings for a given chart. Be able to draw lines on the graph. → MVP

TP3: Have a (AI) helper bot to help you make trades based on some criteria (not completely clear yet). Have (live?) data updating on the chart and be able to show a live P/L running. Have another page for scanners (imported from other websites?)

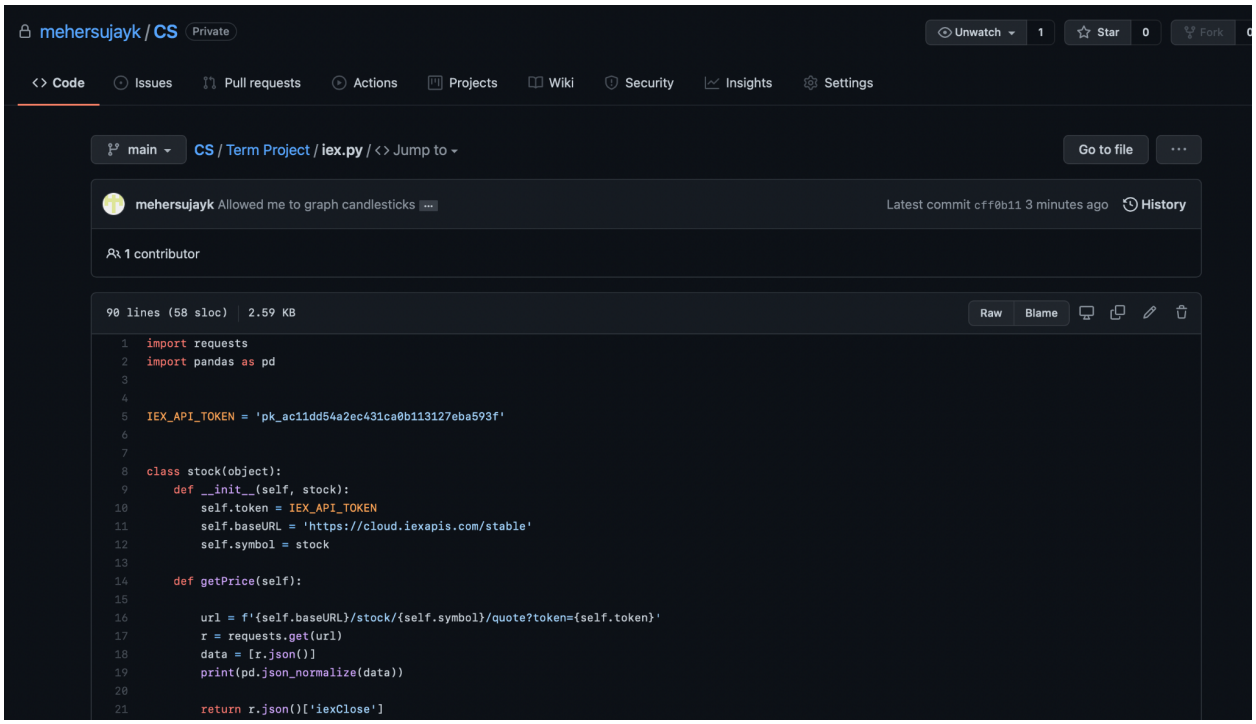
**Version Control Plan**

For version control, I will be using github desktop that automagically detects when I have made a change in the repository and allows me to assign notes and commit and push it to a private git repository in the cloud.

Github desktop



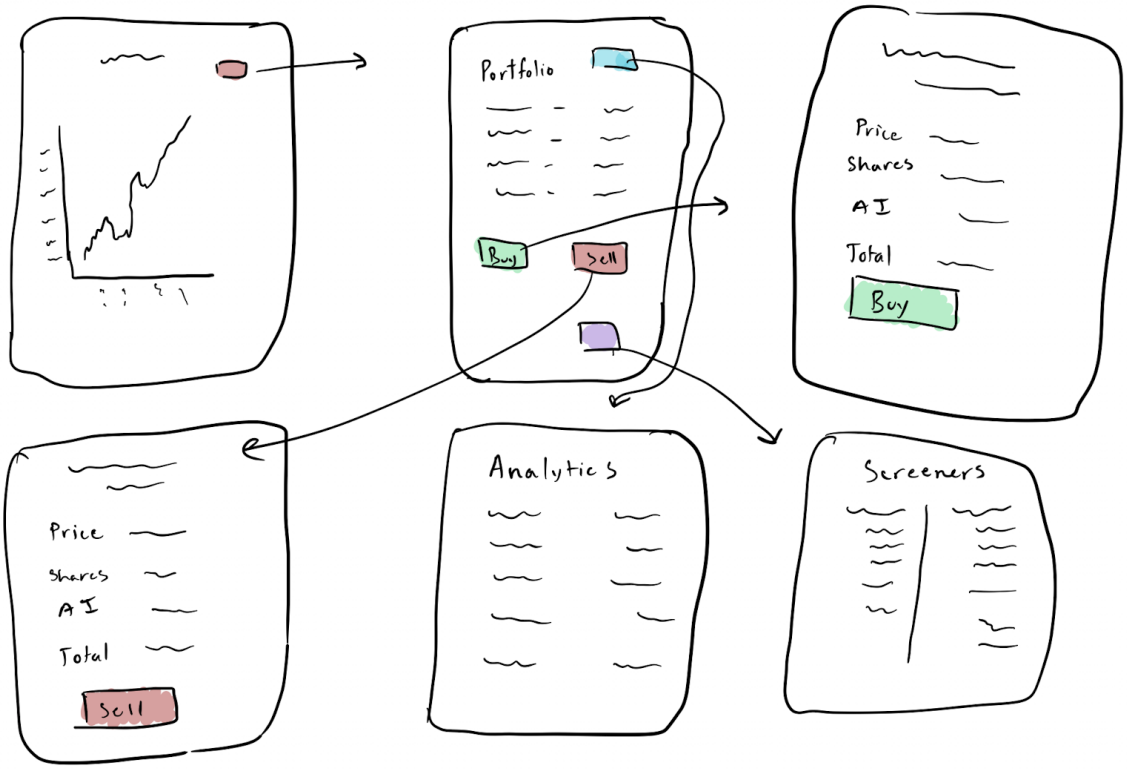

My repository



**Module List**

Requests → Passed Tech demo
Pandas → Passed Tech Demo
Iex cloud API → Passed tech demo with requests
Cmu_112_graphics → no tech demo needed

**Storyboard**




**TP 2 Update:**

The storyboard did not change very much from TP1 to TP2. My mentor and I decided that the AI and screeners can be done for after MVP, so that will not be present for the TP2 code. In addition, I created a lot of UI elements for this TP2 submission that weren't present on the TP1 submission along with creating functions that occur when these UI elements are interacted with. All in all, the project as a whole did not change much from TP 1 to TP 2 as I worked on completing more of it.