Task - 1

Purpose

We want to process the data feed of stock A and stock B’s price to enable us to analyse when trading for the stock should occur.

Acceptance Criteria

● *getDataPoint* function should return the correct tuple of stock name, bid\_price, ask\_price and price. Note: price of a stock = average of bid and ask

● *getRatio* function should return the ratio of the two stock prices

● main function should output correct stock info, prices and ratio

● Upload a git patch file as the submission to this task

● Bonus: All unit tests inside client\_test.py, added/existing have to pass

**My achievements** :

● I made the necessary changes in the functions named *getDataPoint* and *getRatio* to get the data pipeline ready.

● I wrote / added several test cases for testing the functions. This really helped.

Task - 2

Purpose

The objective of this task will be for you to fix the client-side web application so that it displays a

graph that automatically updates as it gets data from the server application (*see Before and After*

*images below*) Currently, the web application only gets data every time you click on the 'Start

Streaming Data' button and does not aggregate duplicated data.

Acceptance Criteria

● This ticket is done when the graph displayed in the client-side web application is a continuously updating line graph whose y axis is the stock’s top\_ask\_price and the x-axisis the timestamp of the stock. The continuous updates to the graph should be the result of

continuous requests and responses to and from the server for the stock data.

● This ticket is done when the graph is also able to aggregate duplicated data retrieved from

the server.

**My achievements** :

● Added properly comments to the code integrated into the react app.

Task - 3

Purpose

You will use perspective to generate a live graph that displays the data feed in a clear and visually

appealing way for traders to monitor this trading strategy.

Recall that the purpose of this graph is to monitor and determine when a trading opportunity may

arise as a result of the temporary weakening of a correlation between two stock prices. Given this

graph, the trader should be able to quickly and easily notice when the ratio moves too far from the

average historical correlation. In the first instance, we'll assume that threshold is +/-10% of the 12

month historical average ratio.

● This ticket is done when the numbers from the python script render properly in the live

perspective graph. This means the ratio between the two stock prices is tracked and

displayed. The upper and lower bounds must be shown on the graph too. And finally,

alerts are shown whenever these bounds are crossed by the ratio (the guide below will

also give more detail and visuals to help you understand these requirements better)

● This ticket is done when a patch file is uploaded along with a video or audio explanation of the final chart you have produced .

**My achievements** :

● Added properly comments to the code integrated into the react app.