J.P. Morgan Project Proposal: Electronic Tool to Trade ACME ETFs based on Time Weighted Average Price (TWAP)

Team Name: Exception Handlers

Team Members: Leon Song (ls3233), Aaron Ong (ao2591), Jackie Lin (jl4162), Gabrielle Taylor (gat2118)

Team Number: 9

Trello Link: https://trello.com/exceptionhandlers

Github Link: https://github.com/ls3233/ase/tree/master/price_is_right

Current User Workflow: Right now our customers do trades manually. They receive a buy/sell order from a manager (e.g. sell 1000 shares), then they split the order up into smaller chunks and call the exchange house periodically to attempt to sell. We want to automate this process so it is less tedious.

Intended Users:

v1.0 will only have one single trader as the intended user. The trader will log on to our web app, place an order and observe its progress throughout the day.

How does our Architecture work:

Our architecture has 3 main elements. Users, Parent Orders and Child Orders.

When a user wishes to sell something, he logs in and places a Parent Order. Our algorithm will then cut the Parent Order up into smaller Child Order chunks, then attempt to sell each chunk using the API that J.P. Morgan provided.

Proposed Solution: v1.0 of our tool will be simple. It will allow **one** user to log in and input an order. We'll use the TWAP algorithm to split the order into even chunks, and attempt to trade those chunks periodically until the market closes, logging both successful and failed trades. We will only support **sell** orders for v1.0

Nuances of the Algorithm:

1. We split the order into even chunks.
2. Attempt to trade each chunk every X minutes (depending on how much time we have left before market close)
3. If a trade fails, try again. If it fails again, split it further and sell even smaller chunks (split in half)
4. Repeat this loop until the order is completed.

Assumptions:

1. Orders all come in at the start of the day, when the market opens. No order will come in afterwards.
2. Whenever a trader tries to sell, we maintain no inventory i.e. we assume he has enough quantity to sell.

Error cases:

1. User cannot trade ETFs that he does not own (i.e. we should keep an inventory of the ETFs that a user has, and verify his orders against his inventory)

2. User cannot trade same ETF more than once in such a way that total quantity sold exceeds inventory (i.e. the system should evaluate sell orders based on total quantity of previous orders rather than completed partials)

Features for subsequent versions:

1. Be able to cancel or pause a trade midway
2. Keep some notion of an inventory, so we can ensure a user sells only shares that he owns.
3. Support multiple traders logging in
4. Support multiple simultaneous trades
5. Improved Algorithm: T.W.A.P can be a naive algorithm that does not react to market conditions, once we get the tool up and running we can aim to develop more sophisticated techniques.

Technology Stack:

Language: Python

Framework: Django

Database: SQL

How does our Framework talk to our database, and how this relates to Python:

Django has a built-in ORM (Object Relational Mapping) which allows us to easily manipulate SQL (and many other databases like Postgres, FYI). Essentially we can type things in Python like:

Parent_order = User.objects.get(ParentOrder_id == 3)

Django then converts this into the appropriate SQL query and returns the correct parent order to us.