Software Requirements and Design Document

For

Group <2>

Money Mills

Version 1.0

Authors:
Marcelo Z.
Jeffrey A.
Brian H.
Roy W.

1. Overview (5 points)

The project revolves around querying the API from a service called *Polygon.io* to gather, sort, and organize the data and input them into an algorithm called the *Black-Scholes Model*. The UI will be terminal based and ideally, the user would enter a desired stock ticker and data will be formatted to determine the optimal entry/exit points if the pre-existing trends show a possibility of success.

2. Functional Requirements (10 points)

The high priority requirements would most likely involve being able to access the options chain and stock data from the API that is chosen and store them temporarily in an easily readable format. This is an integral part of the program and without any data, no algorithms can be run to determine an appropriate output.

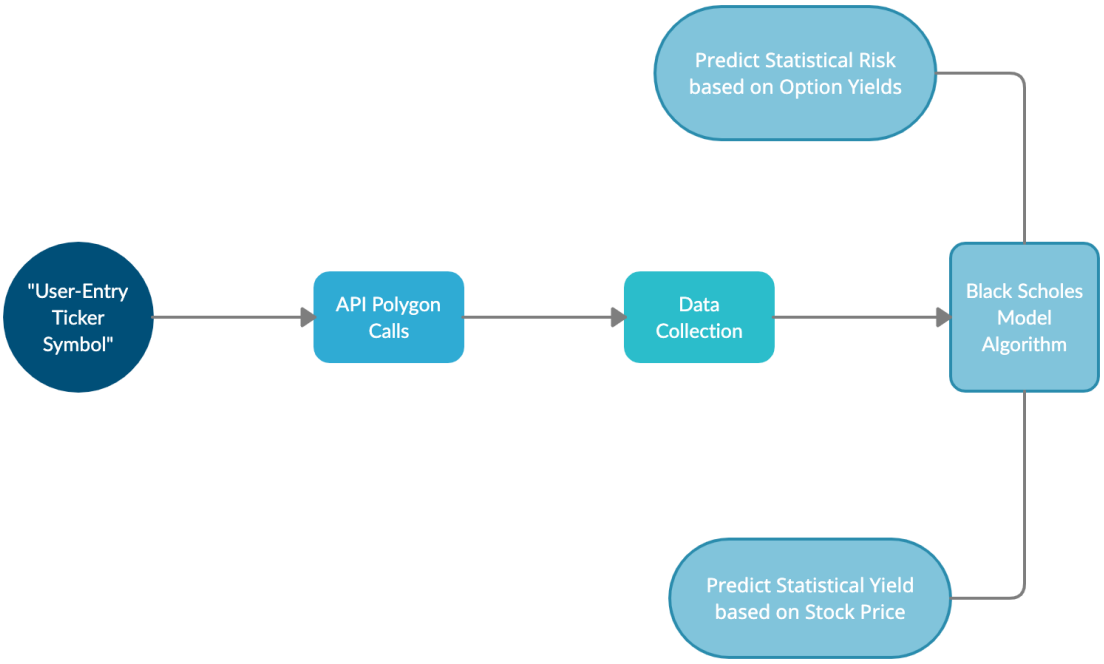
A medium priority requirement would be to utilize the data and input them into the Black Scholes Model as well as reformat the results into a comprehensive description for which option has the highest probability of success. Establishing a matrix or graph of the options chain would also be considered to enhance readability and comparisons.

The lowest priority requirement mostly involves the interface and its aesthetics. Because most of the back-end framework is still underdeveloped, there is almost no reason to begin “polishing” the front-end since it provides little to no substance to the actual program.

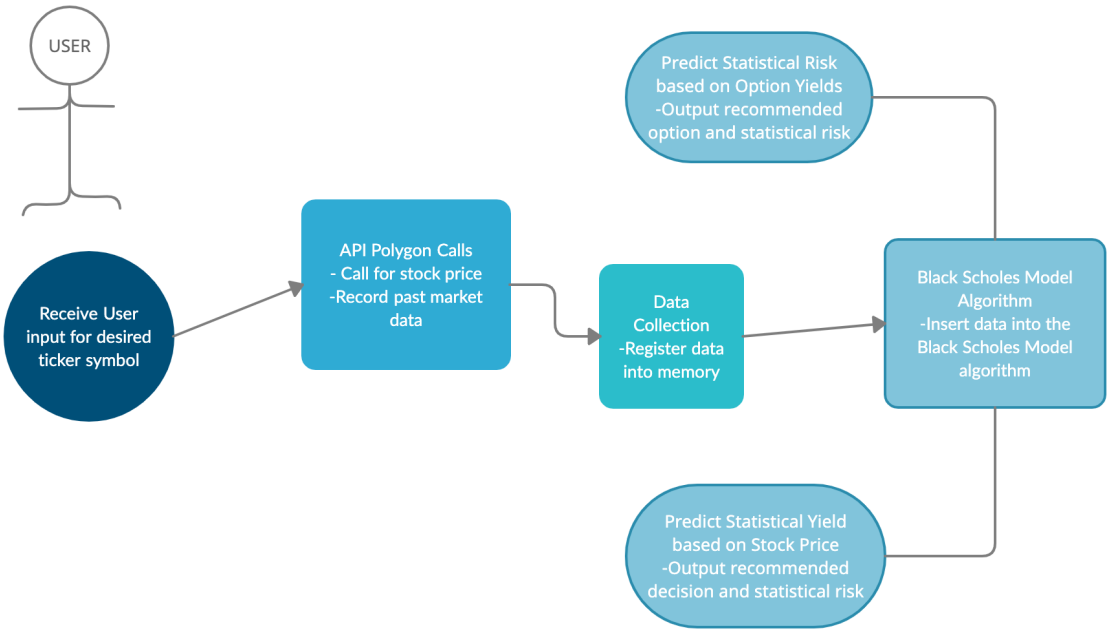
3. Non-functional Requirements (10 points)

Because the data is all accessible relatively easily (albeit unorganized and scattered), the security of the program mostly resides in the ability to keep API keys hidden or inaccessible to the user due to the fact that the keys are given to one person who has private information tied to it. As for performance and mainly reliability, this will mostly be out of our control due to the fact that the API key we will most likely be deciding on is currently time locked unless our budget increases to afford higher tiers of access. This makes it somewhat unreliable for fast, day trading users since our limit is currently locked at 5 calls per minute. Aside from that, the requirement for reusability will be tied directly to the cooldown for how many times the key can be accessed. Regarding software quality, the requirement will simply involve efficient and simple code that follows with the original program proposal/plan.

4. Use Case Diagram (10 points)



5. Class Diagram and/or Sequence Diagrams (15 points)



6. Operating Environment (5 points)

The program will be written in python and the user will be able to access the program via the website. The website will have a basic user interface where the user will be able to enter their desires ticker symbol.

7. Assumptions and Dependencies (5 points)

One of the main dependencies in our project is the fact that we must rely on an API to receive the necessary stock data. The API we selected has a cost of $100 per month, as a result we decided to use the free trial version which has a limit of 5 API calls per minute.

If 5 calls per minute are not enough for us to get the information to input into the black scholes model then we may have to search for alternative API's or purchase the full version of the current API.