

Software Requirements and Design Document

For

Group <2>

Money Mills

Version 3.0

Authors:

Jeffrey A.

Brian H.

Roy W.

Marcelo Z.

1. Overview (5 points)

The project revolves around querying the API from a service called TD Ameritrade to gather, sort, and organize the data and input them into an algorithm called Binomial Distribution. 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)

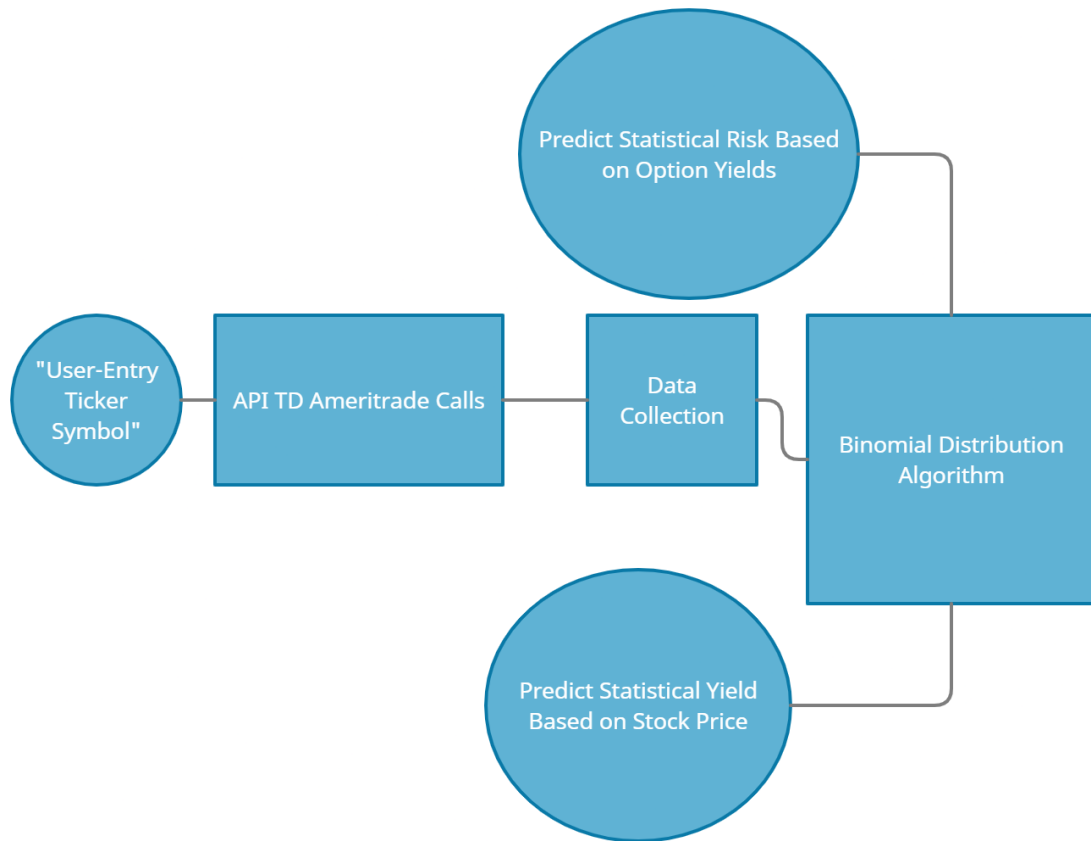
The high priority requirements would most likely involve being able to access the options chain and stock data from the API that is chosen and store them temporarily in an easily readable format. This is an integral part of the program and without any data, no algorithms can be run to determine an appropriate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

A medium priority requirement would be to utilize the data and input them into a Binomial Distribution 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.

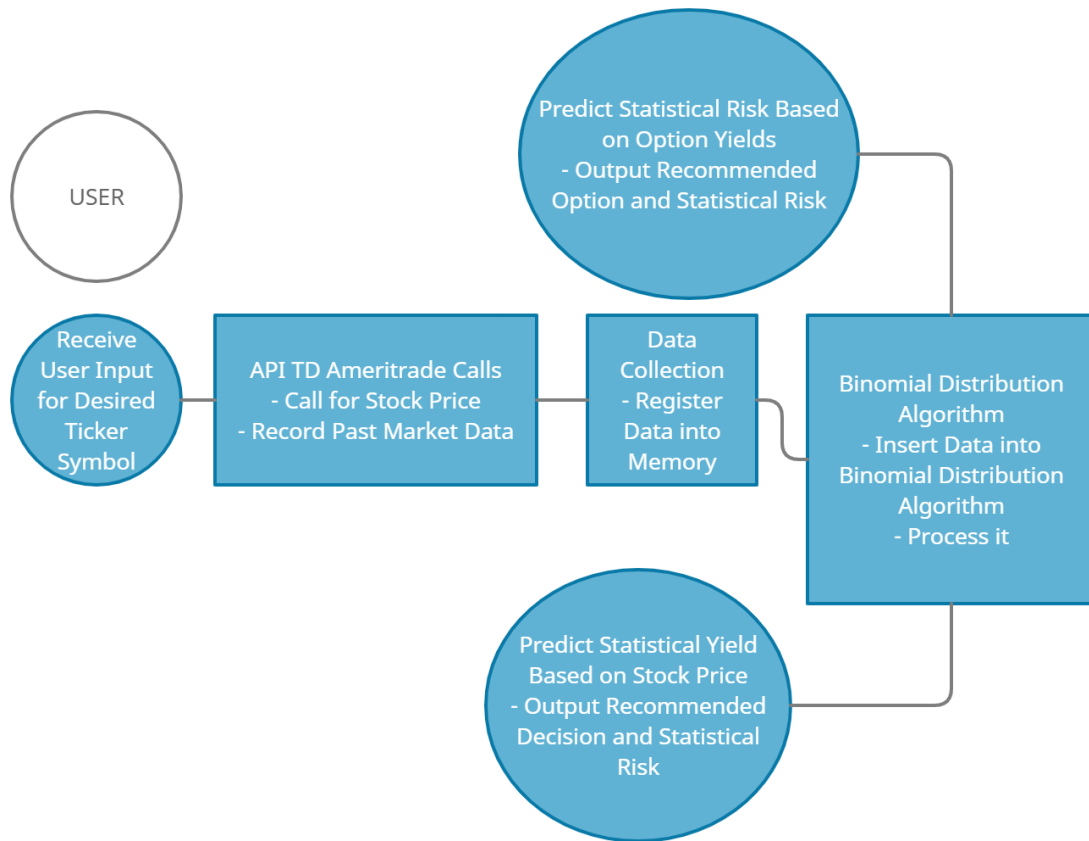
3. Non-functional Requirements (10 points)

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 since the keys are given to one person who has private information tied to it. As for performance and mainly reliability, this will not be a concern because the API key we will be using currently allows 120 calls per minute, which is basically one call every half a second. This makes it reliable for fast, day trading users since our limit is very high. 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 desired ticker symbol.

7. Assumptions and Dependencies (5 points)

Compared to our previously selected API Polygon.io, TD Ameritrade's API is more efficient since it allows for 120 calls per minute. As we previously mentioned in our first RD document, we wanted to search for a better API in order to obtain better and faster information for our stock program. Now that we utilize TD Ameritrade, hopefully we can get more reliable and up to date stock information.

Compared to our previously selected algorithm Black Sholes Model, we had to switch to using Binomial Distribution due to how American-exercise options work. American-exercise options can be exercised before the expiration date. The Black Sholes Model does not account for that because it was designed for European-exercise options. European-exercise options are only exercised at expiration. There are some American stocks that are traded using European-exercise options, such as the SPX index. However, that is not a commonly traded instrument on American stock markets. As a result, we had to use a different algorithm for theoretical prices.