The Hong Kong Polytechnic University

Department of Computing

COMP4913 Capstone Project

Project Proposal

Simulation Game for Learning Algorithmic Trading

Student Name: Cheung Sui Wing

Student ID No.: 21027547D

Programme-Stream Code: 61431-SYC

Supervisor: Dr YIU Man Lung Ken

Submission Date: 23:59, 21 Oct 2022

Table of Contents

1. Background and Problem Statement

Nowadays, not a few people have the habit of investing and trading. For example, stocks, gold, and even cryptocurrencies. They all have a common goal, which is "Make more money." However, many of them fail and lose their money in the end.

There are many reasons for this, such as humans not reacting quickly enough to price changes, needing time to analyze, and even having investment emotions. These series of shortcomings increase the chance of losing money. Computers, on the other hand, are solving some of these problems that are unique to humans because they can execute various instructions and algorithms based on the original plan in a precise and unemotional manner. (Donadio, S. S. G., & Ghosh, S., 2019)

Algorithmic trading may not make you a winner in the trading market. But at least, it can increase the chances of becoming a winner.

There are also a number of trading tools or games on the market that allow people to access the market before using real money. And they can be organized into 2 categories.

1. **Use real market data, completely simulate the exchange**, except for the account amount is false. **Users can only submit buy/sell instructions.** This type of simulator allows users to feel the real market atmosphere, and because it is real time data, it has the unknown, so it is a good introductory tool. However, in the few examples I have tried, none of them support algorithmic trading. Figure 1 shows one of the captures of these type of games.

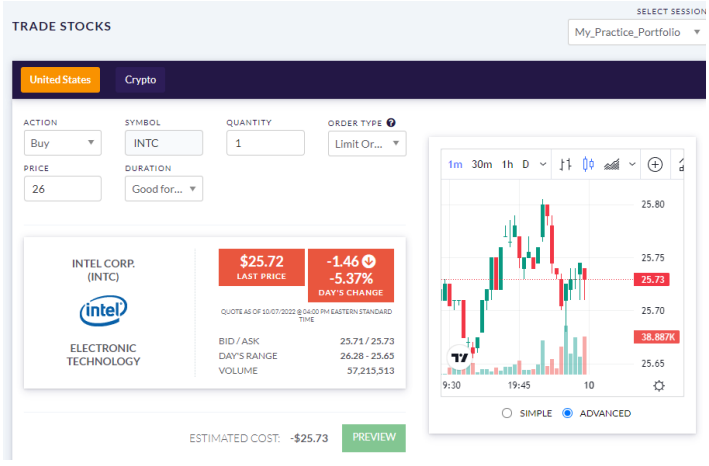

Figure 1 Real-time trading UI (Survivor, 2022)

2. **Use fake/historical data** to simulate trades. This type of site allows users to experience market movements more quickly. You can even know directly if the trade is profitable or losing. Figure 2 and Figure 3 shows the process of one of the examples on this type of websites.

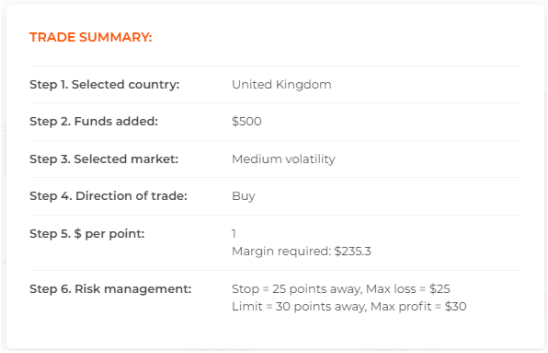

Figure 2 Step by Step rule setup (Nation, 2022)

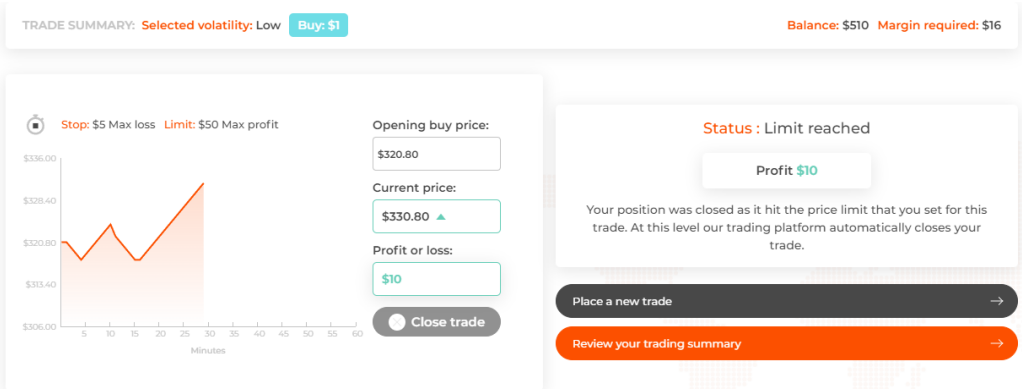

Figure 3 Simulation result (Nation, 2022)

In the process, I did not find a simulation game that allows users to set up complex algorithmic to do transactions automatically. So, I started to find some more professional software related to this topic.

1. **Code base to creating the trading strategy, which is not beginner friendly.**
   TradingView. This is an advanced tool that allows people to test their algorithms. It provides both historical data and real-time data. People can program scripts to set up their algorithms. A similar tool is MetaTrade 4/5. These tools are very comprehensive, but one drawback is that you have to know how to program and have some knowledge, which is harder to get started and not a learning-oriented tool.
   Figure 4 shows a screenshot of TradingView. The bottom part is the area for the user to create their algorithm by programming. The upper chart shows the "Buy" and "Sell" signals given by the algorithm.

Figure 4 TradingView screenshot (TradingView, 2022)

2. **Report only, user will be learning nothing about the trading algorithm.**
   InvestBot. The application provides some good setup algorithms that allow the user to "follow" to get trade signals. The downside is that users do not know any details of the algorithm. Instead, they will know the detailed backtest report of the algorithm. Figure 5 shows some screenshots of the report.

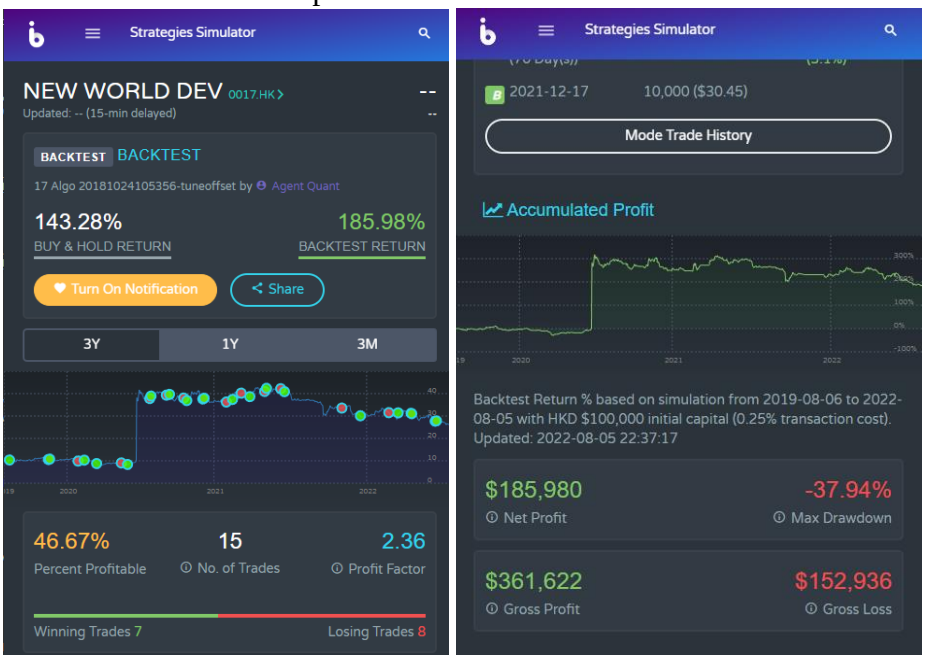

Figure 5 Screenshots of the report page of InvestBot. (InvestBot, 2022)

2. Objectives and Outcome

People generally have the misconception that you have to know how to program for algorithmic trading. Secondly, you need to know a lot of financial indicators, technical analysis, and even artificial intelligence. So, the learning threshold is considered to be very high.

In this project, I will try to break this stereotype, but of course, knowing what I said before may make your trading more accurate. But getting started doesn't need to be complicated. You can start with an easier way and then go deeply later if you are interested.

By using this tool, I want users to learn the 3 steps they need to know before implementing their trading plan programmatically.

- Create your trading logic
  - First, I will provide some classic trading strategies for users to try. For example, the Martingale and Average Cost method. Users need to set various parameters to create their own unique trading strategies.
  - Secondly, I would allow users to create a trading strategy based on technical indicators in an If-Then-Else model. There will be some preset indicators provided and allow the user to make their own algorithm with indicator/rule combination.
  - This step will be presented using the GUI as much as possible, hoping to achieve the low-code level, so that users can easily get started.

- Backtesting using historical data
  - After having a trading strategy, the user will be asked to select a trade for backtesting
  - Then, the server will apply the algorithm and the historical data to generate a detail report to the user.
  - After reviewing the report, users can go back to the previous step to change the parameters/algorithm, and then retest again.

- Demo test of real time data
  - When you are satisfied with your current trading logic, you can use real-time data to make a demo trade. Test whether the trading strategy is also suitable for the future.

To add some game elements, I will set up a leaderboard. Which rank the user according to the profit margin of their algorithm setting.

Expected outcome

Frontend:

- **A website**. Allowing people to register and login to use above function.

Backend:

- **A Servers**. Handle communication of database and the website. Get data from third party API. Processing the backtesting and demo trade function.
- **Database.** Storing users/algorithm setting data

3. Project Methodology

The approach used in this project will be an incremental development. Figure 6 shows the details of the model. At the beginning of the project (this proposal), I will conceptualize the main functions needed for this system (Section 2). Divide into individual functions, and then complete them one by one. Section 4 will show the details.

Each feature will have a development cycle of about 2-3 weeks. Each cycle consists of three phases: analysis, development, and testing. Then after each feature is completed, I will try to invite some of my friends around for trial use. The purpose is to collect user feedback.

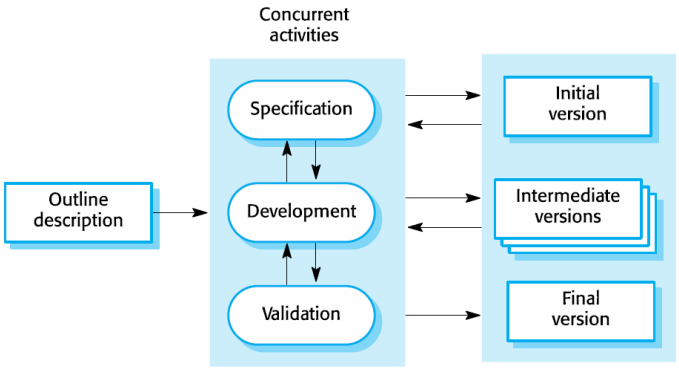

Figure 6 Incremental Development Model (Pei, 2022)

4. Project Schedule

2022-10-23 to 2022-12-31 (W is for Week)

W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
Milestones 1			Milestones 2			Milestones 3			

2022:

Milestones 1 (10-24 to 11-13):

- Setup the react/server-side/DB of the projects
- Landing page
- Account related function (e.g., create account/login/profile)

Milestones 2 (11-14 to 12-04) (Maybe busy due to the end of the semester):

- Access the polygon.io API to get the market price data
- Build the GUI to teach the user to create their algorithm step by step (Martingale)

Milestones 3 (12-05 to 12-31): (Include Exam period)

- Build the logic of the Martingale
- Build report page of Martingale
- Interim Report + presentation video

2023-01-02 to 2023-04-09

W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14
Milestones 4			Milestones 5			Milestones 6			Milestones 7				

2023:

Milestones 4 (01-02 to 01-22):

- The GUI/Logic/Report of Average Cost method

Milestones 5 (01-23 to 02-12):

- The GUI/Logic/Report of the customize indicator combined algorithm

Milestones 6 (02-13 to 03-05):

- The demo run of the finished three type of algorithm using current market data
- The user can logout and login to check the run result after a period.

Milestones 7 (03-06 to 04-09):

- The continuous report/record of the demo run
- Ranking page
- Buffer time for unfinished things/Bug fixing/ final Testing
- Final report / one page summary

5. Resources Estimation

Hardware:

- Desktop/Laptop with networking access

Software:

- Frontend:
  - React (JS library for building websites)
    - Lightweight Charts (build candlestick charts) (free version)
- Backend:
  - Node.js (server-side language)
    - Express.js (use to create RESTful API Server)
    - Mongoose (manage the connection to MongoDB)
  - Python (server-side language)
    - Pillow (Image library for generate report graph)
  - Database
    - MongoDB atlas (free Cloud database)

Historical/Real Time Data

  - Polygon.io API (stock/crypto) (free version)

**The above lists are some of the most important libraries.*

**More libraries may be added as needed during the process of program development.*

References

Donadio, S. S. G., & Ghosh, S. (2019). *Learn algorithmic trading : build and deploy algorithmic trading systems and strategies using Python and advanced data analysis (1st edition).* Birmingham ; Mumbai : Packt Publishing.

InvestBot, L. (2022, 10 09). *InvestBot*. Retrieved from https://www.investbotapp.com/

Nation, T. (2022, 10 09). *Trade Nation*. Retrieved from https://tradenation.com/trading-simulator

Pei, D. M. (2022). *COMP3211 02.Software Processes.* The Hong Kong Polytechnic University.

Survivor, W. S. (2022, 10 09). *Wall Street Survivor*. Retrieved from https://www.wallstreetsurvivor.com/

TradingView, I. (2022, 10 09). *TradingView*. Retrieved from https://www.tradingview.com/