

The Hong Kong Polytechnic University
Department of Computing

COMP4913 Capstone Project

Final Report

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, 11 Apr 2023

Abstract

This final report is about the development details of the algorithmic simulation game. The application is a web-based software that allows the user to play the game through a browser while learning various aspects of algorithmic trading through the game process. The game uses past market data to simulate trading, allowing the user to experience the process of developing and executing trading strategies. By participating in this simulation, users can better understand the intricacies and nuances of algorithmic trading and improve their skills in this area.

This report includes a review of background information and related systems, highlighting how they can engage users and enhance their comprehension of trading strategies. Additionally, it discusses the system overview and design, outlines the challenges encountered during game development, and proposes recommendations for improving the game's usability and functionality.

Table of Contents

1. Introduction.....	1
2. Related Systems Review.....	2
2.1. Wall Street Survivor	2
2.2. Trade Nation	2
2.3. TradingView	3
2.4. InvestBot.....	4
3. Game Overview	5
3.1. Frontend Platform.....	5
3.2. Data.....	6
3.3. Backend	7
3.4. Game Play.....	8
3.5. Game Record and Leader Board	9
4. System Design	10
4.1. System architecture.....	10
4.2. Sequence diagram.....	12
4.3. Frontend.....	13
4.4. Backend	14
4.5. Algorithm provided (Game Level)	16
5. Testing.....	21
6. Integration with Course in PolyU	22
7. User Interface.....	23
7.1. SignIn/SignUp	23
7.2. Dashboard.....	23
7.3. New Game – Pick Assets.....	24
7.4. New Game – Select Algorithm.....	24
7.5. New Game – DCA rules setting	25
7.6. New Game – Martingale rules setting	26
7.7. New Game – Custom Rules setting	27
7.8. History	30
7.9. History:id.....	30
7.10. Leader Board.....	32
8. Limitation.....	33
9. Conclusions.....	34
References	35
Appendices	36

List of Figures and Tables

<i>Figure 1 Real-time trading UI (Survivor, 2022)</i>	2
<i>Figure 2 Step by Step setup (Nation, 2022)</i>	2
<i>Figure 3 Simulation Result (Nation, 2022)</i>	3
<i>Figure 4 TradingView screenshot (TradingView, 2022)</i>	3
<i>Figure 5 Screenshots of the report page of InvestBot. (InvestBot, 2022)</i>	4
<i>Figure 6 Frontend development platform</i>	5
<i>Figure 7 Services provider of the historical data</i>	6
<i>Figure 8 Database service provider</i>	7
<i>Figure 9 Node.js</i>	7
<i>Figure 10 Express.js</i>	7
<i>Figure 11 Mongoose.js</i>	7
<i>Figure 12 System architecture</i>	10
<i>Figure 13 Protected route example</i>	11
<i>Figure 14 Main flow of the system</i>	12
<i>Figure 15 Route navigation map</i>	13
<i>Figure 16 DCA flow chart</i>	16
<i>Figure 17 Simple Flow of Martingale Strategy</i>	17
<i>Figure 18 Take Profit Field</i>	18
<i>Figure 19 Martingale Buy Setting</i>	18
<i>Figure 20 Flow chart of Custom Rules</i>	19
<i>Figure 21 Structure of the condition groups</i>	20
<i>Figure 22 Rule format</i>	20
<i>Figure 23 Test Result for each Indicator</i>	21
<i>Figure 24 Code Example of SMA Test</i>	21
<i>Figure 25 Syllabus of COMP4141 (PolyU, COMP4141 Syllabus, 2021)</i>	22
<i>Figure 26 Syllabus of COMP4531 (PolyU, COMP4531 Syllabus, 2020)</i>	22
 Table 1 Current route of the frontend application.....	 13
Table 2 The Route that the RESTful server contains	15

1. Introduction

Learning about algorithm trading can be daunting for those new to the field, as it involves a complex set of concepts and techniques that require a strong foundation of knowledge and skills. This is where a simulation game for learning algorithm trading can be instrumental.

A simulation game is a digital educational tool that allows users to learn about a topic through interactive gameplay. In the case of algorithm trading, the game utilizes actual market data and simulated trades to give users hands-on experience in developing and executing trading strategies. By engaging in this simulation, users can better understand the complexities and nuances of algorithm trading and improve their skills in a safe and controlled environment.

One of the key benefits of a simulation game is that it allows users to learn about a topic in a low-risk setting. In the case of algorithm trading, users can test different strategies and see how they perform in real-time market conditions without the risk of real-world financial losses. This can be incredibly valuable for those just starting in algorithm trading, as it allows them to get a feel for the process and develop their skills without the pressure of actual financial stakes.

In addition to the interactive gameplay, many simulation games also include educational resources and tutorials to help users understand the concepts and techniques involved in the topic. This can be particularly helpful for those new to algorithm trading. It can provide a strong foundation of knowledge and skills to build upon as they engage in the simulated trading environment.

Overall, a simulation game for learning algorithm trading is a valuable resource for anyone interested in learning about this topic. It provides an immersive and interactive learning experience that allows users to develop their skills and knowledge in a safe and controlled environment. Whether you're a beginner looking to get your feet wet in algorithm trading or an experienced trader looking to hone your skills, a simulation game can be a valuable tool in your learning journey.

2. Related Systems Review

2.1. Wall Street Survivor

Wall Street Survivor is a website application that allows users to execute Buy/Sell operations on a selected stock/crypto. Users can create different profiles, each with an initial balance of \$100,000, since it uses Real-Time Data. It can avoid cheating like people already know the historical trend and base on that movement to do the Buy/Sell operation. But the weak point is that the option of Action is less. Only simple Buy/Sell/Short/Cover orders can be used. Overall is like you are trading in an actual exchange with a fake balance.

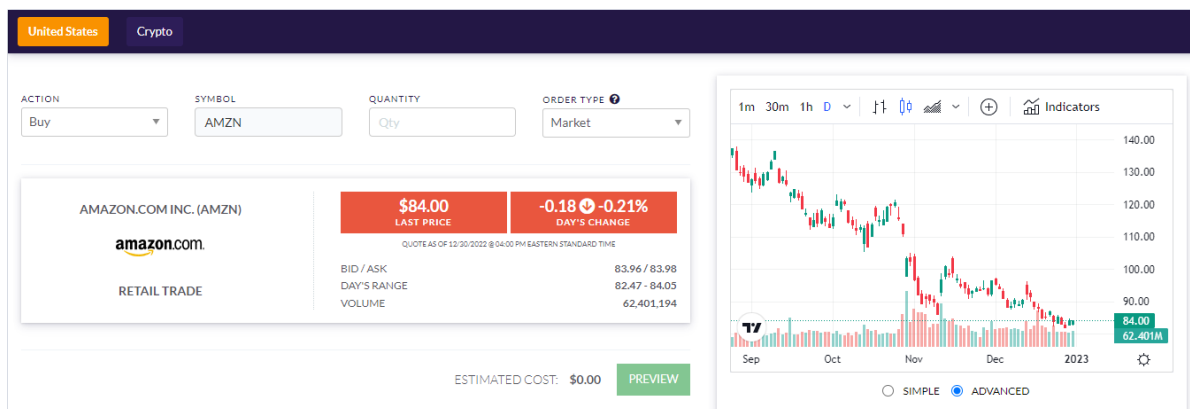


Figure 1 Real-time trading UI (Survivor, 2022)

2.2. Trade Nation

Trade Nation is a web application that allows users to execute Buy/Sell operations with randomly generated data. It has a step-by-step UI to lead the user to input the setting, like how much you want to invest or when you will stop loss. An explanation will be provided between steps to let the user know what is happening. When playing the simulation game, it randomly generates price data according to the user setting. Then execute the Buy/Sell/Stop Loss automatically.

The screenshot shows the Trade Nation step-by-step setup interface. On the left, a prompt asks 'Ready to open and close your first trade?' with a button to 'Yes, run trade simulation'. On the right, a 'TRADE SUMMARY' table lists the following steps:

TRADE SUMMARY:	
Step 1. Selected country:	Rest of the World
Step 2. Funds added:	\$5000
Step 3. Selected market:	Medium volatility
Step 4. Direction of trade:	Sell
Step 5. \$ per point:	1 Margin required: \$70.5
Step 6. Risk management:	Stop = 25 points away, Max loss = \$25 Limit = 30 points away, Max profit = \$30

Figure 2 Step by Step setup (Nation, 2022)

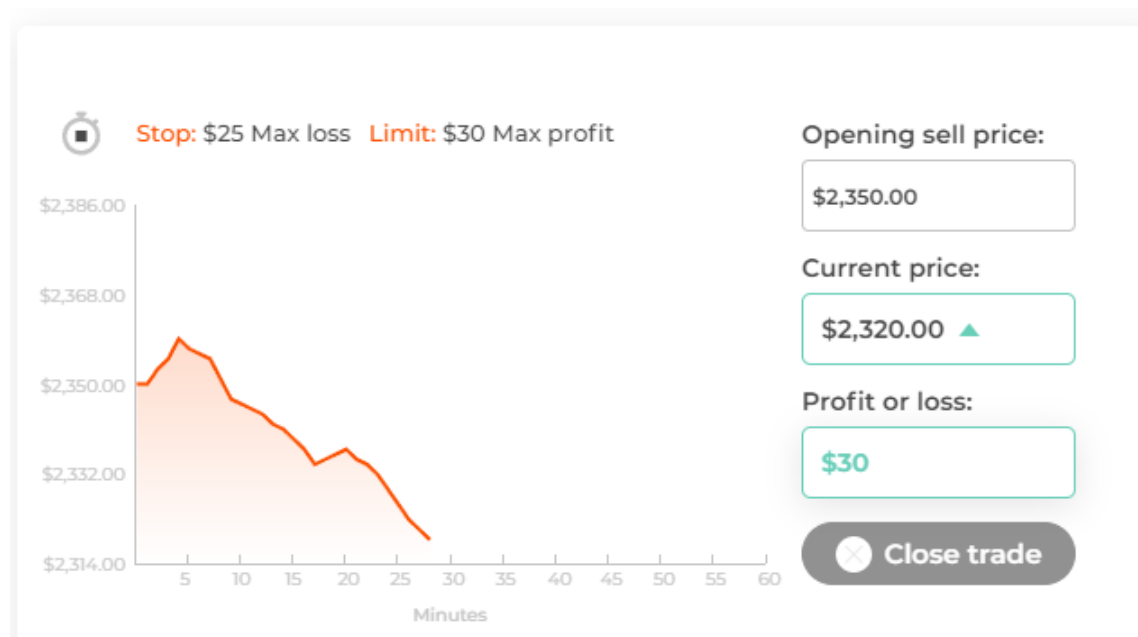


Figure 3 Simulation Result (Nation, 2022)

2.3. TradingView

TradingView is a professional website that provides historical and real-time Stock/Crypto data with an advanced chart. Many people use it to do technical analysis. One of the functions is to allow users to write a piece of code and apply it to the data. Then the system will simulate the Buy/Sell accordingly. It is a comprehensive tool but very hard to use since people need to learn to code before they want to apply their algorithm.



Figure 4 TradingView screenshot (TradingView, 2022)

2.4. InvestBot

InvestBot is an application that provides some algorithms already set up. And allow users to "Follow" and get the trading signal. Users can review how the algorithm is applied to historical data. The report is straightforward. But I can learn some styles, such as the profit movement chart, which I can implement in my application.

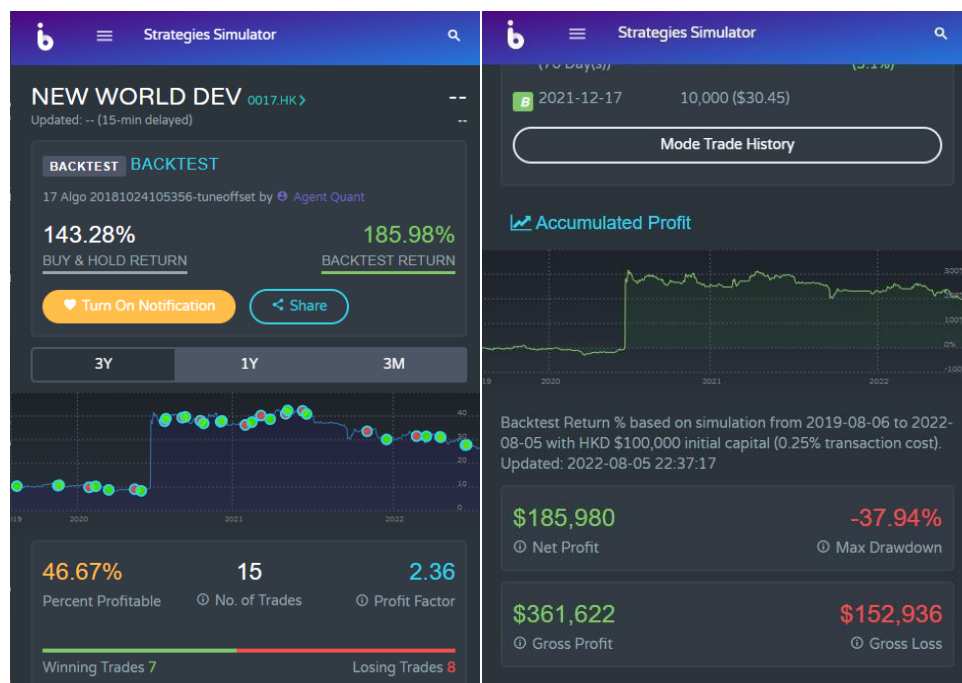


Figure 5 Screenshots of the report page of InvestBot. (InvestBot, 2022)

3. Game Overview

3.1. Frontend Platform

The simulation trading game has been developed as a web-based platform to provide easy accessibility to a broad audience. By creating the game as a web application, users can access the game through a web browser without the need to install any specialized software or tools. This approach has several advantages, making it an ideal choice for a simulation game. For instance, a web-based platform allows developers to create interactive and dynamic applications on various devices and operating systems. This can be particularly useful for a simulation game, enabling users to access the game from multiple devices, including desktop computers, laptops, and tablets.

Furthermore, a web-based platform can provide easy access to updates and new features, as changes can be implemented on the server side without requiring users to download or install anything. This can help ensure that users always have access to the latest version of the game and enjoy a seamless experience without worrying about compatibility issues. Additionally, as the platform is hosted on the web, it is easier to share and promote, increasing its potential audience and making it accessible to a broader range of users.

As part of the development process, the web-based simulation trading game has been developed using React, a popular JavaScript library for building user interfaces. React was chosen based on its ability to create modular and reusable components, making it easy to manage and maintain complex web applications. By using React, the platform provides users with a highly responsive and interactive experience, enabling seamless and engaging gameplay. Moreover, React's virtual DOM is optimized for performance, ensuring the platform runs smoothly, even on lower-end devices.

React's modular and reusable components also allow developers to add new features and functionalities to the platform quickly and efficiently. This is important for a simulation game, as it enables the platform to evolve and improve over time based on user feedback and new market conditions. Additionally, React's component-based architecture makes it easier to maintain and test the platform's codebase, helping to ensure its reliability and stability.



Figure 6 Frontend development platform

3.2. Data

3.2.1. Historical Data

The simulation trading game utilizes historical data to create a realistic and relevant environment for users to test their trading strategies. This approach is a common practice in algorithmic trading simulations, and it allows users to experiment with different assets and market conditions. The game features crypto and stock data, giving users access to a diverse range of assets.

To ensure that the simulation remains accurate and relevant, the game uses polygon.io, an API service provider, to download up-to-date data instead of a set of historical data. This allows the game to provide users with market conditions as close to real-world situations. The data is updated daily, providing users with the latest market trends and movements to work with.

Using polygon.io as the API service provider is a practical solution that simplifies the development process. It eliminates the need to manually update or download large historical data sets, saving time and resources. The simulation can accurately reflect current market conditions using up-to-date data, providing users with a valuable experience.

Overall, the simulation trading game's use of historical data and polygon.io as the API service provider ensures that users can experiment with a realistic and diverse range of market conditions. Using up-to-date data, the game can provide users with a current and accurate reflection of the market, simplifying the development process and improving the overall user experience.



Figure 7 Services provider of the historical data

3.2.2. User Data

The simulation trading game stores user data to track different simulations and provides users with a record of their past simulations. The user data stored includes the user information, the selected assets, the selected trading algorithms, and the simulation setting.

To store this data securely and efficiently, the simulation trading game uses MongoDB Atlas, a cloud-based database service that provides a robust and scalable solution for data storage. MongoDB Atlas offers a range of features that make it an ideal choice for storing user data, such as automatic scaling, automatic backups, and robust security protocols.

Using MongoDB Atlas as the database provider ensures that user data is stored securely and efficiently, providing a reliable and scalable solution for managing user data. MongoDB Atlas's cloud-based solution enables easy access to user data from anywhere worldwide, making it an ideal choice for a web-based simulation trading game.



Figure 8 Database service provider

3.3. Backend

The simulation trading game requires a backend server to manage user data and provide simulation tasks based on different trading algorithms. The simulation trading game uses Node.js, Express.js, and Mongoose to develop the backend server.

Node.js is a popular runtime environment that enables developers to run JavaScript code on the server side. Node.js's event-driven architecture and non-blocking I/O make it a lightweight and efficient choice for building scalable server-side applications. Express.js is a web application framework that runs on top of Node.js, providing robust features for building web applications, such as routing, middleware, and templates. Mongoose is an Object Data Modeling (ODM) library that provides a high-level abstraction layer over MongoDB, enabling developers to define schemas and models for their data.



Figure 9 Node.js

Using Node.js and Express.js as the backend technologies provide a lightweight and efficient solution for managing user data and giving simulation tasks.



Figure 10 Express.js

Using Mongoose enables developers to define schemas and models for user data, making it easier to manage and manipulate data. Additionally, the combination of Node.js, Express.js, and Mongoose provides a flexible and scalable solution for developing a backend server that can handle large amounts of user data and simulation tasks.



Figure 11 Mongoose.js

The backend server provides a RESTful API that enables the simulation trading game to communicate with the database and perform simulation tasks based on different trading algorithms. The RESTful API defines a set of routes and methods that enable the simulation trading game to perform read, write, and update operations on user data and simulation tasks.

In addition, the server uses JSON Web Tokens (JWT) for authentication and authorization to ensure the security of user data and simulation tasks. JWTs provide a secure and efficient means of transmitting information between parties, enabling the game to restrict access to sensitive data and resources to authorized users. The backend server generates a JWT upon user login, which is then stored on the client-side application for subsequent requests to the server. JWTs provide a secure way of transmitting information, reducing the risk of data breaches and unauthorized access. Combining Node.js, Express.js, Mongoose, and JWTs delivers a robust and secure solution for managing user data and simulation tasks.

3.4. Game Play

The simulation trading game provides users with an engaging and interactive learning experience, enabling them to experiment with different trading strategies and better understand algorithmic trading. The game is developed with three levels, each providing a different challenge and level of complexity.

- **Level 1:**

The game's first level is DCA (Dollar-Cost Averaging), a simple trading strategy that involves investing a fixed amount of money at regular intervals. DCA is a popular and effective strategy for long-term investors and provides users with a basic understanding of algorithmic trading.

- **Level 2:**

The game's second level is Martingale, a higher-risk trading strategy that involves doubling down on losing trades. While Martingale can be profitable in the short term, it requires careful risk management and can lead to significant losses if not executed correctly. The Martingale level gives users a deeper understanding of risk management and the importance of carefully managing their trading strategies.

- **Level 3:**

The game's third level is Custom rulemaking, where users can create their own trading rules by adding different conditions such as price, volume, and indicators. Only this mode will record in the leaderboard ranking system. This level provides users with the most advanced and customizable trading experience, enabling them to experiment with different trading strategies and develop their unique approach to algorithmic trading.

The game also provides various resources and tools to help users understand trading strategies and develop their skills through step-by-step guides. Additionally, the game offers tooltips and other forms of guidance to help users understand different keywords and concepts.

3.5. Game Record and Leader Board

- **History:**

The simulation trading game also provides a history feature for users to check their past simulation records. The history feature enables users to review their previous simulations and gain insights into their trading strategies and performance.

Each record in the history feature includes the simulation record, buy and sell animations on the candlestick chart, profit movement chart, and buy/sell/stop-loss order execution table. These visualizations provide users with a clear and detailed view of their trading performance and enable them to analyze their trading strategies in more detail.

Additionally, each record includes statistics on the simulation's overall performance, such as final profit and maximum drawdown. These statistics provide users with a clear understanding of their overall performance and enable them to identify areas for improvement.

- **Leader Board**

The simulation trading game includes a leaderboard feature that adds a competitive element to the system and enables users to check other users' best performance. The leaderboard ranks users according to their ROI (Return on Investment) and provides a fun and engaging way for users to compete with each other.

The leaderboard feature also considers the updated historical data on the server side, meaning that ranking positions are adjusted based on price changes when the data is updated. This ensures that the leaderboard accurately reflects users' performance and provides a fair and transparent ranking system.

The leaderboard feature adds engagement to the simulation trading game, encouraging users to improve their trading strategies and compete with other users. It provides a fun and interactive way for users to learn more about algorithmic trading and gain insights into other users' strategies and performance.

4. System Design

4.1. System architecture

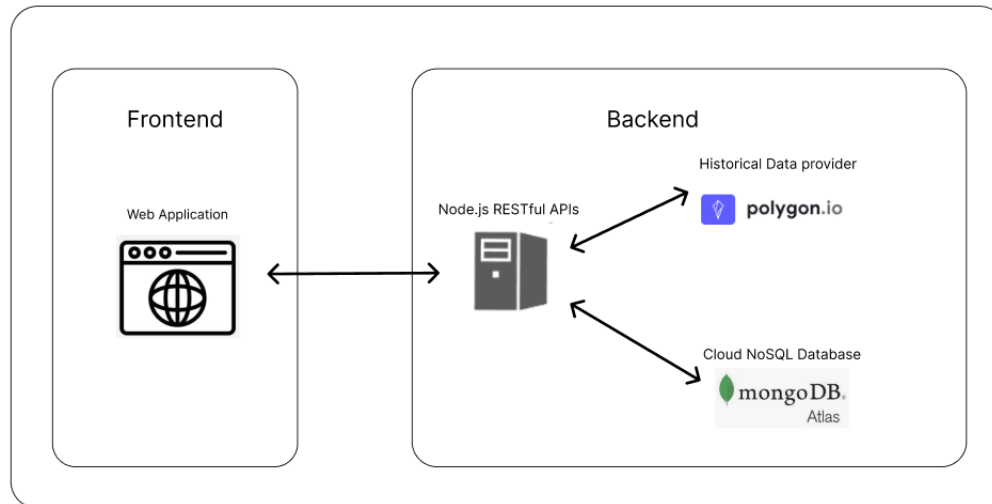


Figure 12 System architecture

Figure 12 shows the system architecture. The system comprises a front and back end, working together to provide a seamless user experience. The front end is built using React, providing an intuitive and user-friendly interface for users to interact with the system. The back end is a RESTful server, receiving user requests from the front end and executing simulation tasks. It also communicates with other third-party components to complete various tasks.

Historical data is retrieved using HTTP requests to ensure the system uses the most up-to-date data. All user account details, simulation settings, and other relevant data are stored in a MongoDB document database, which the server can easily access.

JSON Web Tokens (JWTs) protect routes and ensure system security. These tokens contain user identification information, such as the user ID and username, and can be verified by the server using a private key. This provides a secure way to authenticate users and protect sensitive data.

The various components of the system interact with each other through APIs and databases. Here is how each component interacts with the others:

- **Frontend and Backend:**

The frontend and backend communicate with each other via RESTful APIs. The front end sends user requests to the back end using HTTP requests, and the back end processes the requests and sends a response back to the front end. This communication is vital for the system's core functionality, as the front end relies on the back end to execute simulation tasks and retrieve data.

- **The back end and Historical Data:**

The back end retrieves historical data using HTTP requests, which it sends to an external source. The data source provides the historical data in a format that the rear end can use, and the back end stores the data in a JSON format locally for future use.


- **Backend and Database:**

The back end communicates with the database to retrieve and store data as needed. It sends queries to the database to retrieve user account details, simulation settings, and other relevant data. The back end also writes new data to the database as needed, such as when a user creates a new account or adds their simulation settings.

- **Security and Backend:**

The security component of the system, which uses JWTs, interacts with the back end by verifying the user's identity when accessing protected routes. When a user attempts to access a protected route, the back end checks the JWT to ensure the user is authorized to access the route. If the JWT is valid, the user is granted access. If the JWT is invalid, the user is denied access.

Example:



```
> router.post('/viewRecord', AuthToken, async(req,res)=>{ ...  
  })
```

Figure 13 Protected route example.

If AuthToken function return fail, the server will not execute the request and response function on the right side.

4.2. Sequence diagram

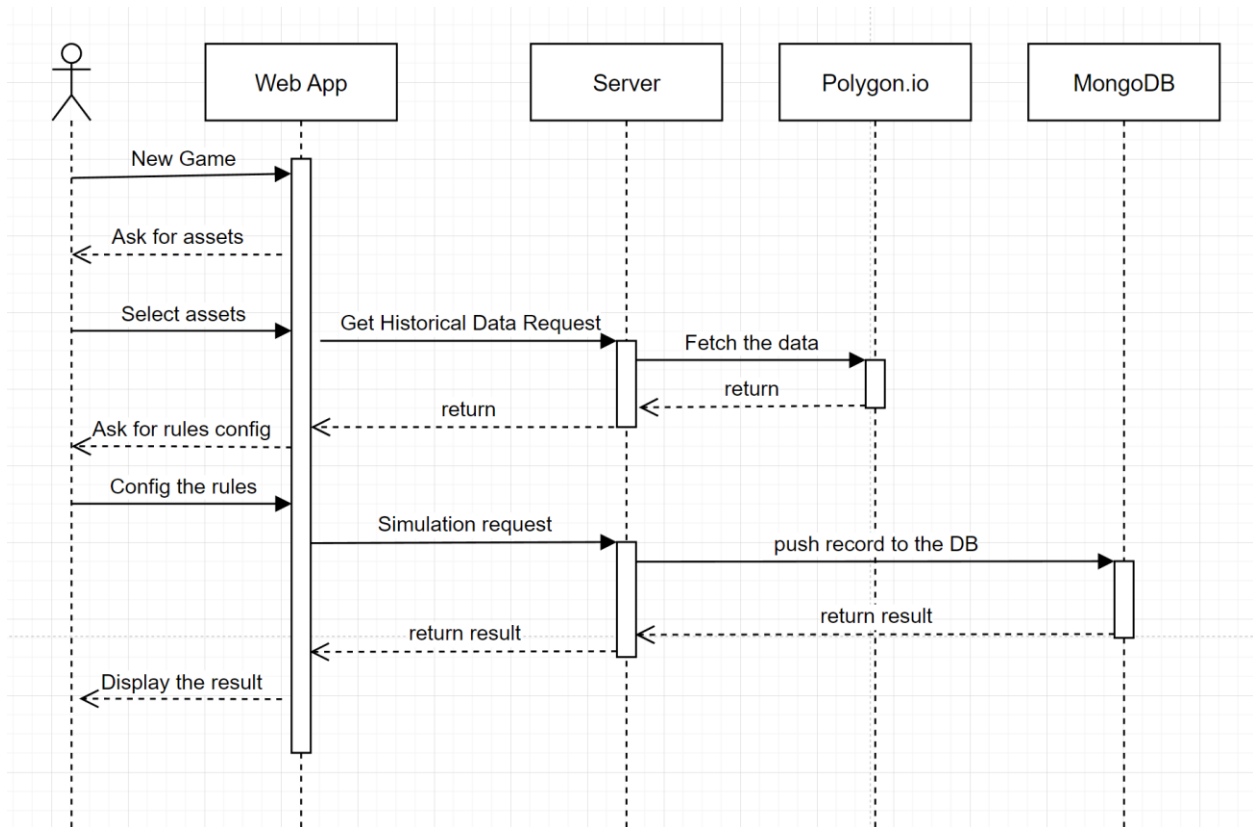


Figure 14 Main flow of the system

Figure 14 shows the main flow of the system.

1. The user asks for a new game.
2. The web app will ask for what game assets the user is willing to pick.
3. The user selects the assets.
4. Web apps send requests to the server.
5. The server download/update the historical from Polygon.io.
6. Get the result and save it as JSON on the server side.
7. The server returns the updated historical data to the Web app.
8. The web app asks the user to input the rules set.
9. User config the rules
10. The web app sends the rules to the server and requests the simulation.
11. The server performs the simulation task and
12. Save the record to the DB.
13. Return the result to the Web application.
14. Display the result to the user

4.3. Frontend

The frontend is developed by React.js Table 1 shows the web's route.

Route	Description
/login	For the user to login to the system
/ SignUp	For the user to sign up for their account
*/Dashboard	Introduce some information about the system
*/GameInti	For the user to create a simulation game [1] Pick up an asset [2] Pick up an Algorithm [3] Setup rules by different input parameters [4] Confirm page [5] Send to the server and redirect to /History/:id
*/History	List all the records of the login user
*/History/:id	List the summary of that record id
*/ LeaderBoard	List the ranking of all the users

Table 1 Current route of the frontend application

*Means the route is a protected route. Only login users can access the page.

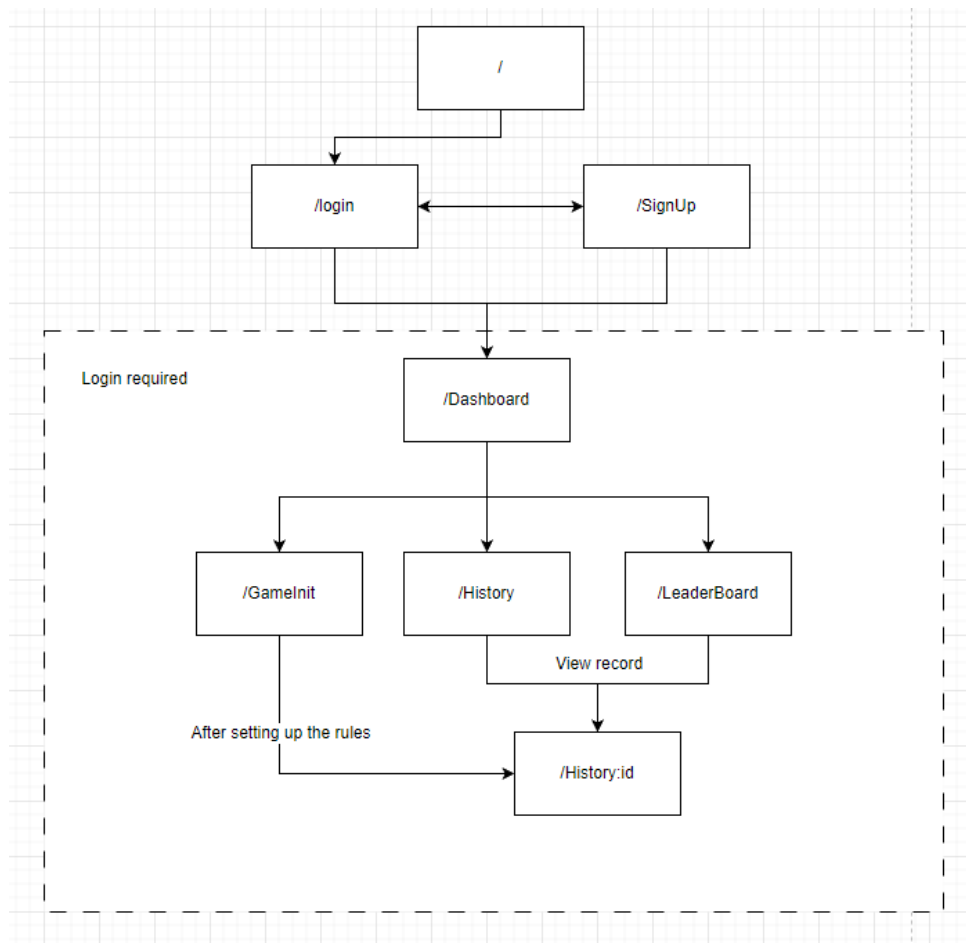


Figure 15 Route navigation map

4.4. Backend

Framework: Node.js + express.js + mongoose.js

Route	Description
/user/signUp	<pre>/* create user URL:localhost:3000/user/signUp Method: POST body: { "username": "test", "password": "12345" } */</pre>
/user/singIn	<pre>/* login URL:localhost:3000/user/signIn Method: POST body: { "username": "test", "password": "12345" } */</pre>
/user/edit	<pre>/* edit URL:localhost:3000/user/edit Method: POST body: { "user": "{ _id : xxxxxxxxxxxxxx username: xxx xxx: xxx ... }, "token": xxxxxxxxxx } */</pre>
/user/view	<pre>/* edit URL:localhost:3000/user/view Method: POST body: { "token": xxxxxxxxxx } */</pre>

/user/viewRecord	<pre> /* view user record with record ID URL:localhost:3000/user/view Method: POST body: { "token": xxxxxxxxx "record_id": xxxxxxxx } */ </pre>
/simulation/	<pre> /* do simulation URL:localhost:3000//simulation/ Method: POST body: { token: xxx, type: 1, algoType: 1, ... (rule object with token) } */ </pre>
/simulation/getRank	<pre> /* get ranking data URL:localhost:3000/simulation/getRank/ Method: POST body: { token: xxx } */ </pre>
/his/ getHistoricalData	<pre> /* get historical data URL:localhost:3000/his/ getHistoricalData Method: POST body: { type: 1, ticker: 'X:DOGEUSD', from: '2020-12-19', to: '2022-12-19', token: xxx } */ </pre>

Table 2 The Route that the RESTful server contains

Table 2 shows all the API routes of the RESTful server. Except for/signIn and /signUp routes, all other routes within the system are secured using the AuthToken function. This function is employed to authenticate and validate the token to ensure it is valid before allowing access to the requested route.

4.5. Algorithm provided (Game Level)

There are three algorithms provided in the game. Here is the design of each level. Each level of the game has its particular purpose for the user to learn or try.

- Dollar Cost Averaging (DCA)
 - Dollar Cost Averaging (DCA) can be a good strategy for beginners who are just starting to invest because it is simple to understand and can help reduce the impact of market volatility. By investing a fixed amount of money at regular intervals, regardless of market conditions, beginners can avoid the temptation to make emotional decisions based on short-term market fluctuations. Additionally, DCA can help beginners build a diversified portfolio over time, which can help reduce risk and increase the potential for long-term returns.

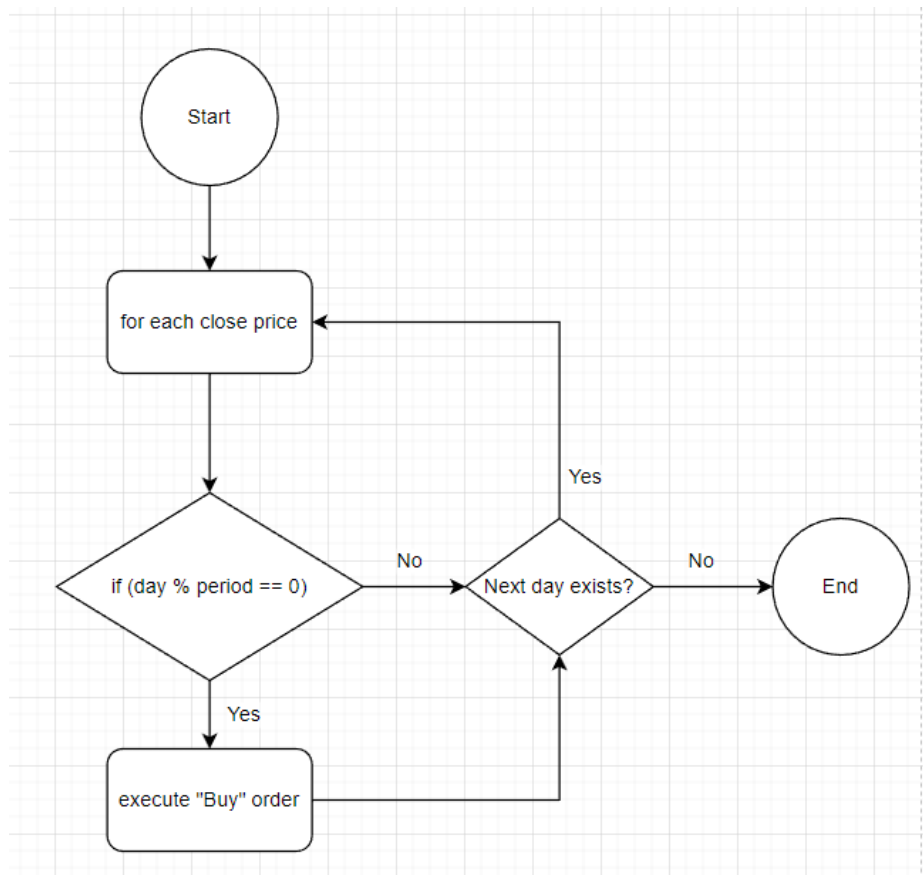


Figure 16 DCA flow chart

The DCA algorithm is implemented with a logic structure outlined in Figure 16. Each day is checked to determine whether it falls within the specified user-defined period via a modulo operation to execute the algorithm. If the current day satisfies this condition, a buy order is executed; otherwise, the algorithm continues to the next day in the data sequence.

- Martingale
 - The Martingale method is a betting strategy that involves increasing the amount of the next bet after a loss so that the first win would recover all previous losses plus a profit equal to the original stake.
 - However, it is a high-risk algorithm. Therefore, I also add the stop loss, stop earn, and price range setting on top of the algorithm. The purpose is to let the user understand the importance of risk management.

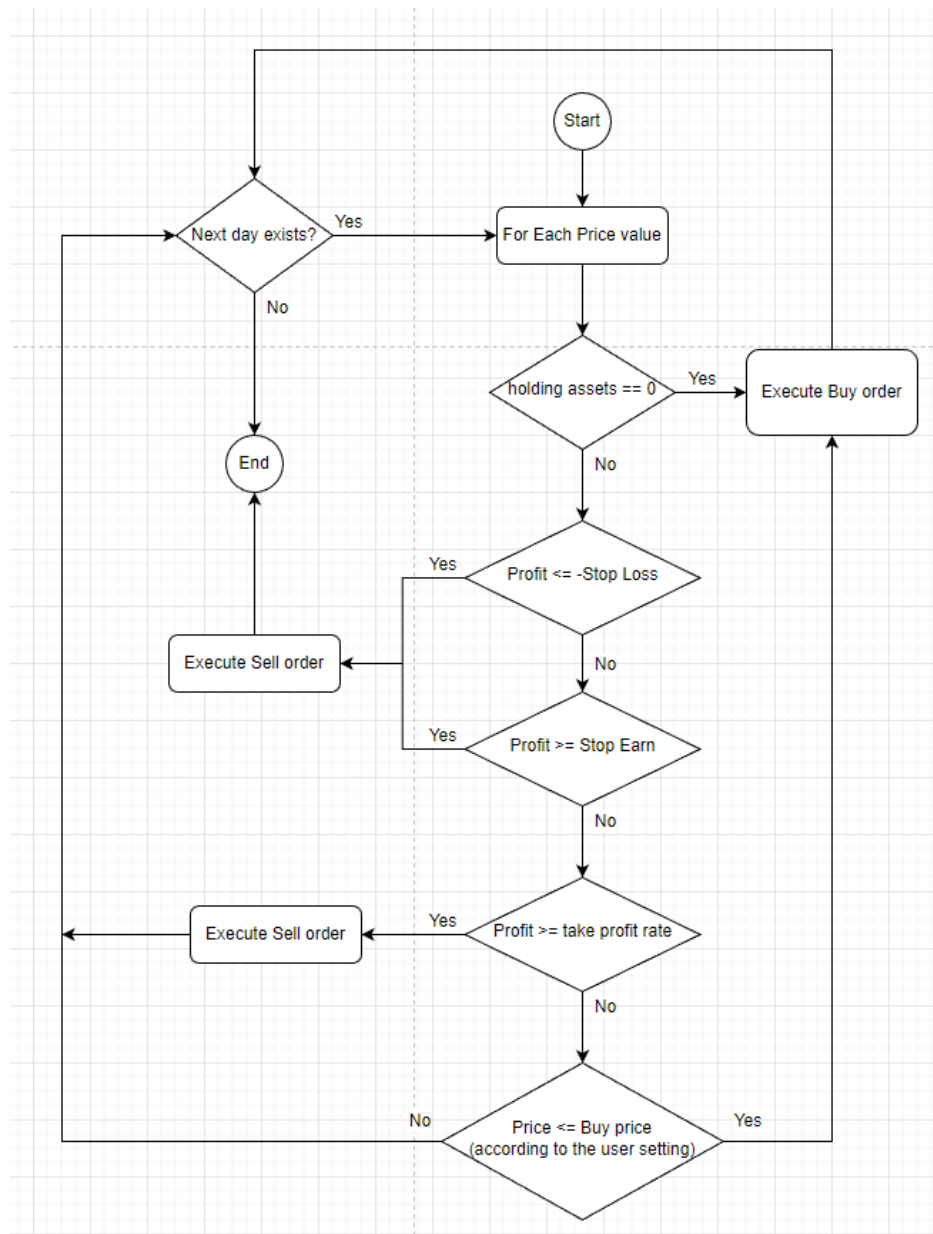


Figure 17 Simple Flow of Martingale Strategy

Win:

In implementing the Martingale method in the trading market, defining the parameters that indicate a win or a lose is crucial. An input field titled "Take Profit Ratio" will be made available to users to signify a win.

Take profit ratio ($\geq 0.1\%$)

0.1

Take profit when earn up to this value.
?

Figure 18 Take Profit Field

Figure 18 shows the field. This field will enable users to set their desired win ratio, such as 1%, which means that when the profit on the holding coin/stock reaches 1%, the algorithm will sell all holdings to realize the profit.

Lose:

To signify a lose, users can create an array object as a reference for the system to execute the buy order. This array object will enable the system to identify the price of the coin/stock that should purchase. If the coin/stock price falls below this reference value, the algorithm will execute a buy order to minimize losses. This approach will give users greater control over their trading strategy and ensure that losses are managed effectively.

Setup the rules ?

# 0	<div>Draw back %</div> <div>0%</div>	<div>Share(s)</div> <div>1</div>
# 1	<div>Draw back %</div> <div>1%</div>	<div>Share(s)</div> <div>2</div>
# 2	<div>Draw back %</div> <div>1%</div>	<div>Share(s)</div> <div>4</div>
# 3	<div>Draw back %</div> <div>1%</div>	<div>Share(s)</div> <div>8</div>

+ Add

× Delete

Figure 19 Martingale Buy Setting

Figure 19 shows the UI of how the user can set up the rules.

The "**Drawback %**" field denotes the percentage by which the price must decline for the system to execute a buy order. For instance, if the user sets the drawback percentage to 1%, the algorithm will initiate a buy order if the price falls below the last buy price by 1%. The algorithm will execute a buy order if the day's closing price is lower than this threshold.

The "**Shares**" field refers to the division of the investment and enables users to customize the buying value of each order. Users can adjust this value to suit their investment preferences, such as tripling the number of shares instead of doubling them. This approach allows for greater flexibility in determining each trade's size and can help optimize trading strategies based on the user's risk tolerance and investment goals.

- Custom Rules

- The "Custom Rules" approach provides users various rule-setting options, including current and previous open, high, low, close prices and volume.
- Additionally, this approach offers more advanced options that enable users to set up rules by combining various technical indicators, such as
 - Moving Average, Relative Strength Index (RSI),
 - Moving Average Convergence Divergence (MACD).
 - SMA: Simple Moving Average
 - EMA: Exponential Moving Average
 - ADX: Average Directional Index
 - Stochastic Oscillator
- These tools can assist users in analyzing market trends and identifying profitable trades, thereby helping to optimize their trading strategy. This approach empowers users with greater control over their trades and enables them to customize their rules based on their trading style and risk tolerance.

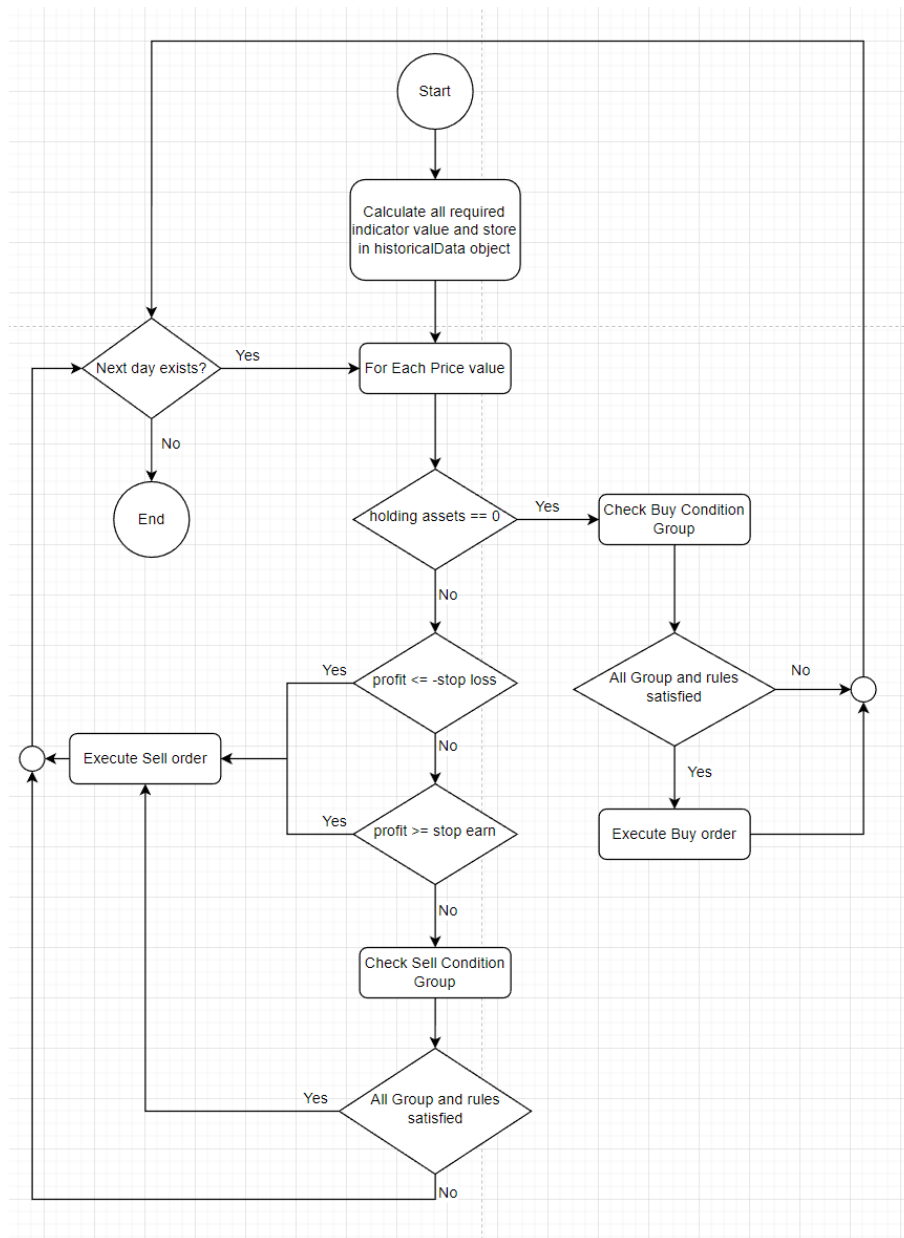


Figure 20 Flow chart of Custom Rules

Buy/Sell condition groups:

The system allows users to create multiple groups of buy/sell conditions, each containing multiple rules. The groups have their operation type, which can be selected as "And", "Count", or "Not". Here is a breakdown of the group operation types and meanings:

- "And": All rules within the group must pass for the group to be considered true.
- "Not": All rules within the group must fail for the group to be considered true.
- "Count": Only a certain number of rules within the group need to pass for the group to be considered true.

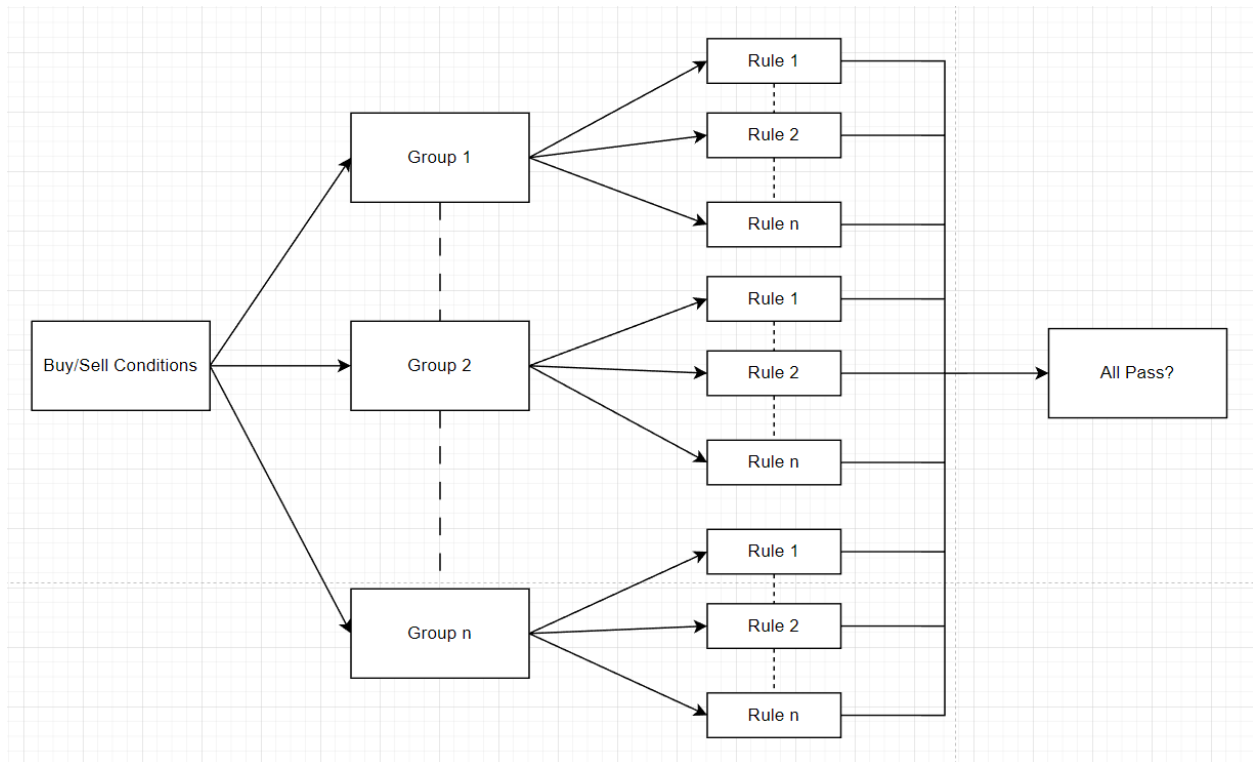


Figure 21 Structure of the condition groups

Figure 21 shows the condition group structure. The system processes all the rules within the group based on the selected group operation type. Once all the rules are passed, the system generates a final result. If all the rules within the group pass, the system executes the buy/sell order.

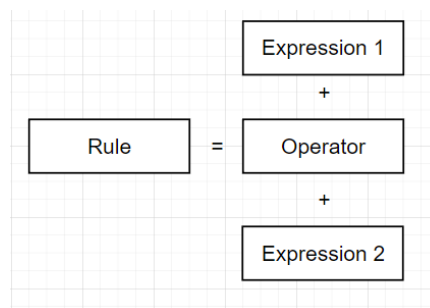


Figure 22 Rule format

As shown in Figure 22, each rule within the system consists of three components: "Expression1", "Operator", and "Expression2". To execute the compare operation, the system utilizes a function that maps these expressions and operators based on their respective types with the processed historical data.

5. Testing

In the Custom Rules section, I implemented a unit test using Jest to ensure the accuracy of the calculations performed by the indicators. This ensures that the system produces reliable and accurate results. Figure 23 shows the result of the testing.

```
PASS UnitTest/algo.test.js
  SO
    ✓ calculates the Stochastic Oscillator correctly (2 ms)
  ADX
    ✓ calculates the ADX correctly (2 ms)
  RSI_function
    ✓ calculates RSI correctly (1 ms)
  SMA function
    ✓ should calculate the SMA values correctly (1 ms)
  calculateEMA function
    ✓ calculate the EMA values correctly (37 ms)
  MACD function
    ✓ calculate the MACD values correctly (2 ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:  0 total
Time:        0.637 s
Ran all test suites.
```

Figure 23 Test Result for each Indicator

For example, when testing the SMA indicator, we created test cases to verify that the indicator accurately calculates the moving average for different time intervals and correctly identifies trends in the data. Microsoft Excel is an excellent tool for calculating the result, which can make me easier to get to correct answers for testing.

```
describe('SMA function', () => {
  it('should calculate the SMA values correctly', () => {
    const data = [
      {c: 1},
      {c: 2},
      {c: 3},
      {c: 4},
      {c: 5},
      {c: 6},
      {c: 7},
      {c: 8},
      {c: 9},
      {c: 10},
    ];

    const periods = 5;

    const expectedOutput = [
      {c: 1, sma5: null},
      {c: 2, sma5: null},
      {c: 3, sma5: null},
      {c: 4, sma5: null},
      {c: 5, sma5: 3},
      {c: 6, sma5: 4},
      {c: 7, sma5: 5},
      {c: 8, sma5: 6},
      {c: 9, sma5: 7},
      {c: 10, sma5: 8},
    ];

    expect(SMA(data, periods)).toEqual(expectedOutput);
  });
});
```

Figure 24 Code Example of SMA Test

6. Integration with Course in PolyU

The software developed in this paper can be utilized in various courses at The Hong Kong Polytechnic University, specifically in finance, technology, and algorithmic trading courses.

For example, in the course COMP4141 Crowdfunding and E-Finance, students can use the simulation game to understand e-trading technology, systems, and algorithmic trading. They can try out different trading strategies and analyze the results, gaining valuable practical experience that can supplement the theoretical concepts taught in the course.

6. Beyond Crowdfunding: Crowdsourcing and Monetisation of Crowdsourcing
7. E-Trading: Technology, Systems and Algorithmic Trading
8. Digital Banking: Online and mobile banking services integrated with innovative digital technologies, e.g. strategic analytics tools, social media interactions, and a focus on user experience
7. E-Trading: Technology, Systems and Algorithmic Trading

Figure 25 Syllabus of COMP4141 (PolyU, COMP4141 Syllabus, 2021)

Similarly, in the course COMP4531 Emerging Topics in FinTech, the simulation game can be utilized to explore the current state of the major thematic areas in the FinTech ecosystem, including e-trading and algorithmic trading. Students can test various trading strategies, assess the impact of different parameters, and gain hands-on experience in a simulated environment.

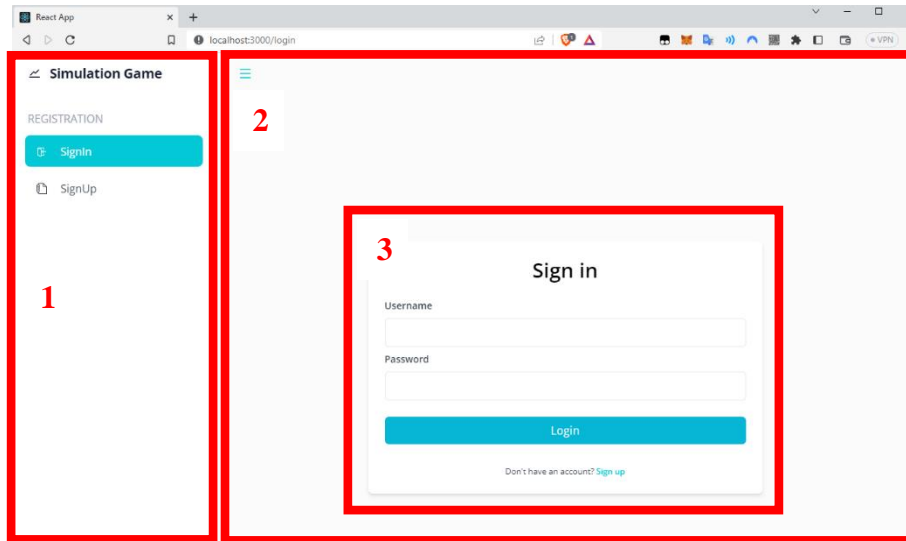
1. Current state of the major thematic areas in the FinTech ecosystem such as infrastructure (e.g., identity, privacy, security), crowdfunding (e.g., types, platforms, applications) e-trading (e.g., algorithmic trading) and peer-to-peer (P2P) lending.

Figure 26 Syllabus of COMP4531 (PolyU, COMP4531 Syllabus, 2020)

In conclusion, the software developed in this paper has the potential to be used in various courses at The Hong Kong Polytechnic University, allowing students to gain practical experience in trading technology and algorithmic trading, thereby supplementing their theoretical knowledge.

7. User Interface

7.1. SignIn/SignUp



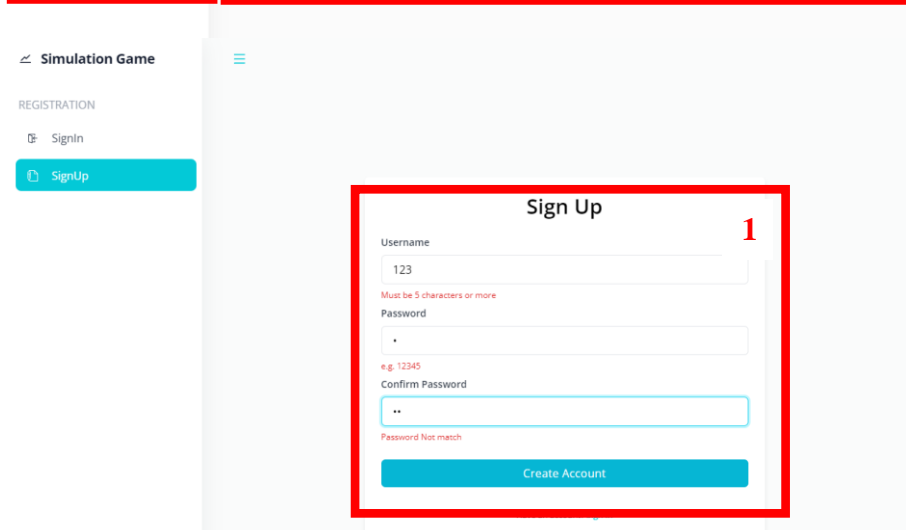
The website has a

1: left slide bar with navigation buttons

2: a main content section,

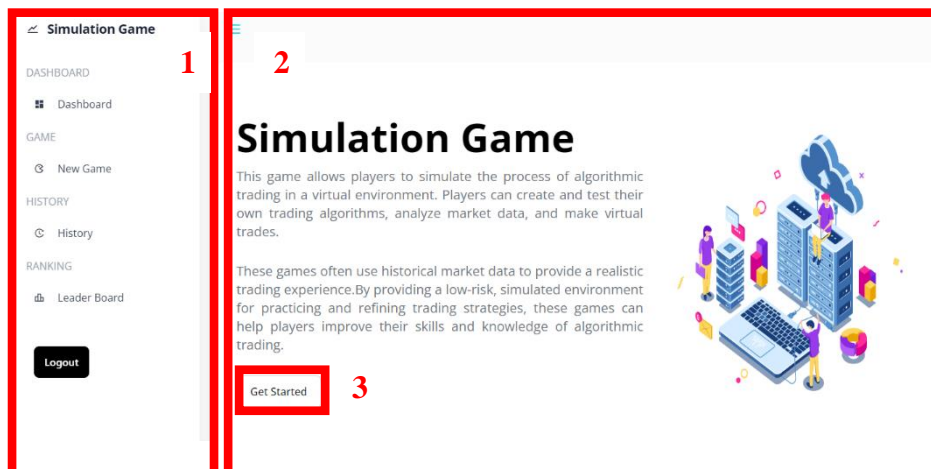
3: login forms.

Without authentication, the website will only display the SignIn and SignUp options for the user.



1: Sign Up form. The red alert text is the form validation message.

7.2. Dashboard



When users log in, this is the first UI they can see.

1: The left slide bar with the main navigation button.

2: some description of the game.

3. Button that can go to play the game.

7.3. New Game – Pick Assets

1: Step by Step form, tell the user where they are.

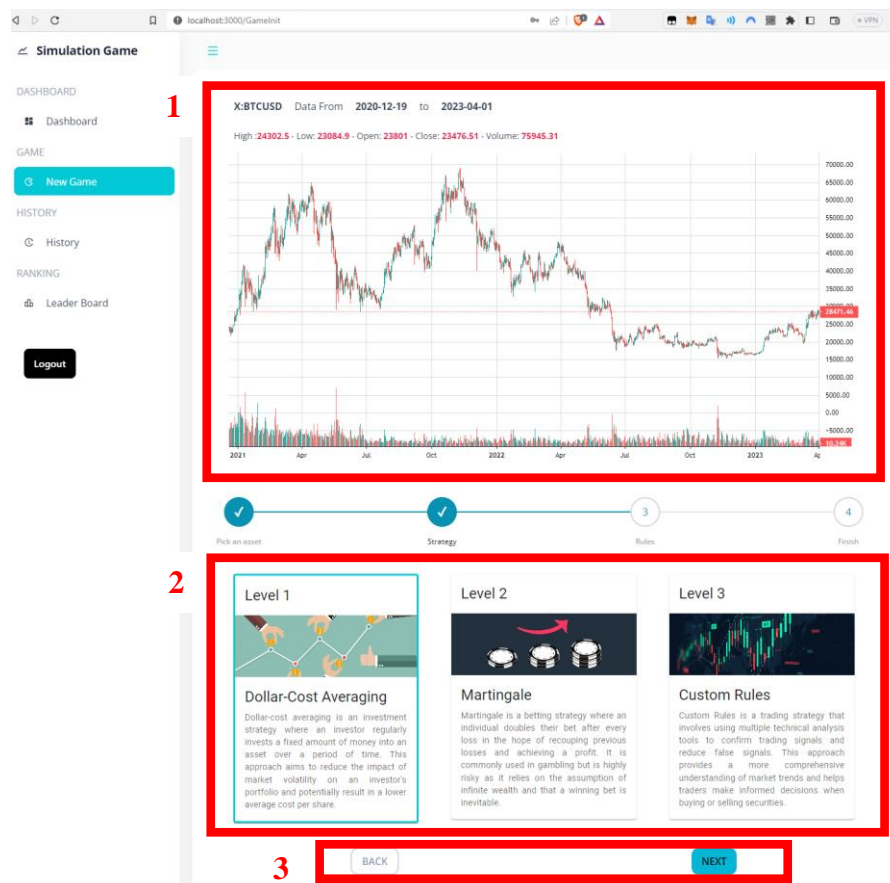
2: Select Crypto or Stock market

3: Select the trading asset

4: Go Next stage

5: User can also view the chart here if they click this button

7.4. New Game – Select Algorithm



7.5. New Game – DCA rules setting

Setup the DCA rules

Select the buy period.

Example:
Buy period = 1 day
Buy amount = 10\$ / 1 shares
Then the bot will buy the amount of coin / shares every 1 trading day.

Buy Amount of every trading.

Total Trading Day during the period : 834 day(s)

Approximate amount of Investment required: 834 USD

BACK NEXT

- 1: The tooltip tells the user what it is.
- 2: The period select option for the user.
- 3: Investment in each trade
- 4: The date range for the simulation
- 5: Updated calculation data for the user to know when the input change.

Parameters

Market Type:	Crypto
Assets:	X:BTCUSD
Algorithm:	Dollar-Cost Averaging
Invest Period	1 Day(s)
Investment every time	\$1 USD
Total Investment	\$834 USD
Date Range:	2020-12-19 to 2023-04-01

BACK CONFIRM

- 1: DCA Parameters table for the user to review their setting.

7.6. New Game – Martingale rules setting

Progress bar: Pick an asset (✓), Strategy (✓), Rules (✓), Finish (4)

1 Setup the rules ?

#	Draw back %	Share(s)
# 0	0 %	1
# 1	1 %	2

+ Add x Delete

Total drawback : 1.00% ? **2**

Total shares: 3 ? **3**

Investment **3** 100 Your Investment.

Take profit ratio (=0.1%) **4** 0.1 Take profit when earn up to this value. ?

Stop Loss : % **5** 100 Stop loss when hit this value. ?

Stop Earning % **6** 0 Stop algorithm when earning hit this value. ?

Date Range: ? **7** 2020/12/19 – 2023/04/01

Upper price **8** 0 Pause the algorithm when the price **Pass** this value. ?

Lower price **9** 0 Pause the algorithm when price **Below** this value. ?

BACK NEXT

1: The parameter list of the Martingale (refer to section 4.5 - Martingale). The Drawback % and Share(s) filed can be edited according to the user's need. Also, they can add or delete the rules by clicking the Add or Delete button.

2: Calculate text for the user to check the maximum drawback and shares needed to buy.

3: Investment – has the minimum requirement according to the rules set.

4: The take-profit ratio

5: Stop loss

6: Stop earning

7: Data range for the simulation

8&9: upper and lower prices. This means the bot will only buy or sell between the price.

1

Parameters

Market Type: Crypto

Assets: X:BTCUSD

Algorithm: Martingale

Main Rules:

	Index	Drawback	Shares
#0		0%	1
#1		1%	2

Total Drawback %: 1.00%

Total Shares : 3

Investment \$100

Take Profit Ratio: 0.1%

Stop Loss: 100%

Stop Earn: 0%

Date Range: 2020-12-19 to 2023-04-01

Price Range: 0 to 0

BACK CONFIRM

1: Martingale Parameters table for the user to review their setting.

7.7. New Game – Custom Rules setting

4

Finish

Rules

Strategy

Pick an asset

1 Buy Rules ▶

2 Sell Rules ▶

3 Risk Management ▶

BACK NEXT

1: Collapse button for Buy Rules

2: Collapse button for Sell Rules

3: Collapse button for Risk Management

The screenshot shows a multi-step configuration process for a trading strategy. At the top, a progress bar indicates four steps: 'Pick an asset', 'Strategy', 'Rules', and 'Finish'. The 'Rules' step is currently active.

Buy Rules Section:

- 1:** A red box highlights the 'Investment USD' input field, which contains the value '1000'.
- 2:** A red box highlights the 'Group #0' section, specifically the 'Group Operator' dropdown menu set to 'And'.
- 3:** A red box highlights the 'Rules #0' section, showing 'Expression 1' as 'Close Price', 'Operator' as '>', 'Expression 2' as 'Number', and 'value' as '1'.
- 4:** A red box highlights the 'Add Rules' button.
- 5:** A red box highlights the 'Add Buy Condition Group' button.

Sell Rules Section:

- 6:** A red box highlights the entire 'Group #0' configuration area for sell rules, including the 'Group Operator' (set to 'And') and the 'Rules #0' configuration (set to 'Close Price' > 'Number' > '1').

Risk Management Section:

- 7:** A red box highlights the 'Risk Management Parameters' section, showing 'Stop Loss -%' set to '100' and 'Stop Earning %' set to '0'.

At the bottom of the interface are 'BACK' and 'NEXT' buttons.

1: Investment value

2: Group Operator of that group

3: Expression 1 – Operator – Expression2, in some cases, It will request the user to input the parameters, E.g., Number

4: Button to add rules

5: Button to add groups

6: Sell Groups (Same control as Buy Group)

7: Stop loss and stop earning

Group #1 Remove Group x 1

Group Operator: And

Rules #0: Expression 1: Close Price, Operator: >, Expression 2: Number, value: 1

Rules #1: Expression 1: Close Price, Operator: >, Expression 2: Number, value: 1, Remove Rule #1 2

Add Rules

Add Buy Condition Group

1: If the condition groups contain more than one group. Then a Remove Group will appear for the user to remove the group.

2: If there is more than one rule inside the group. Then a Remove Rule will appear for the user to remove the rule.

1

Parameters

Market Type: Crypto

Assets: X:BTCUSD

Algorithm: Custom

Investment: \$1000

Buy Rules

Group #0 (And) Pass when all rules in this group should be satisfied.

- Rule 0: Close Price < Number - value: 30000

Sell Rules

Group #0 (And) Pass when all rules in this group should be satisfied.

- Rule 0: Close Price >= Number - value: 30000

Risk Management

Stop Loss: 100%

Stop Earn: 0%

BACK CONFIRM

1: Custom Rules Parameters table for the user to review their setting.

7.8. History

Simulation Game

DASHBOARD

Dashboard

GAME

New Game

HISTORY

History

RANKING

Leader Board

Logout

Records					Profit Movement	Details
Record Time	Algorithm Type	Trading Pair	Investment			
2023-04-02 14:11:34	Martingale	X:BTCUSD	\$100			View
2023-04-02 14:11:19	Martingale	X:BTCUSD	\$100			View
2023-04-02 14:08:57	Martingale	X:BTCUSD	\$1000000			View
2023-04-02 14:08:30	Martingale	X:BTCUSD	\$100000			View
2023-04-02 14:07:48	Martingale	X:BTCUSD	\$100			View

1: The history record table

2: In the simple profit movement chart, Red means the final result is losing while Green is earning.

3: View button for the user to click and navigate to the target record

7.9. History:id



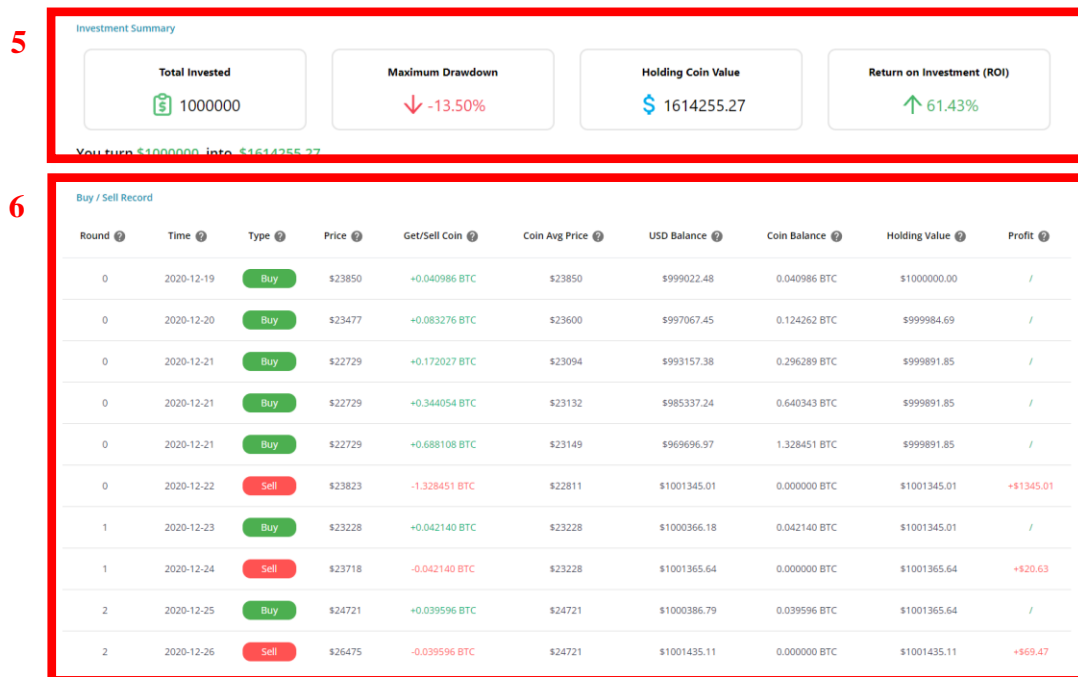
1: Button for the user to review the rules

2: The animate candle stick chart for the user to check when the bot executes buy/sell order

3: The chart displays the profit movement over time and is green if it is winning.

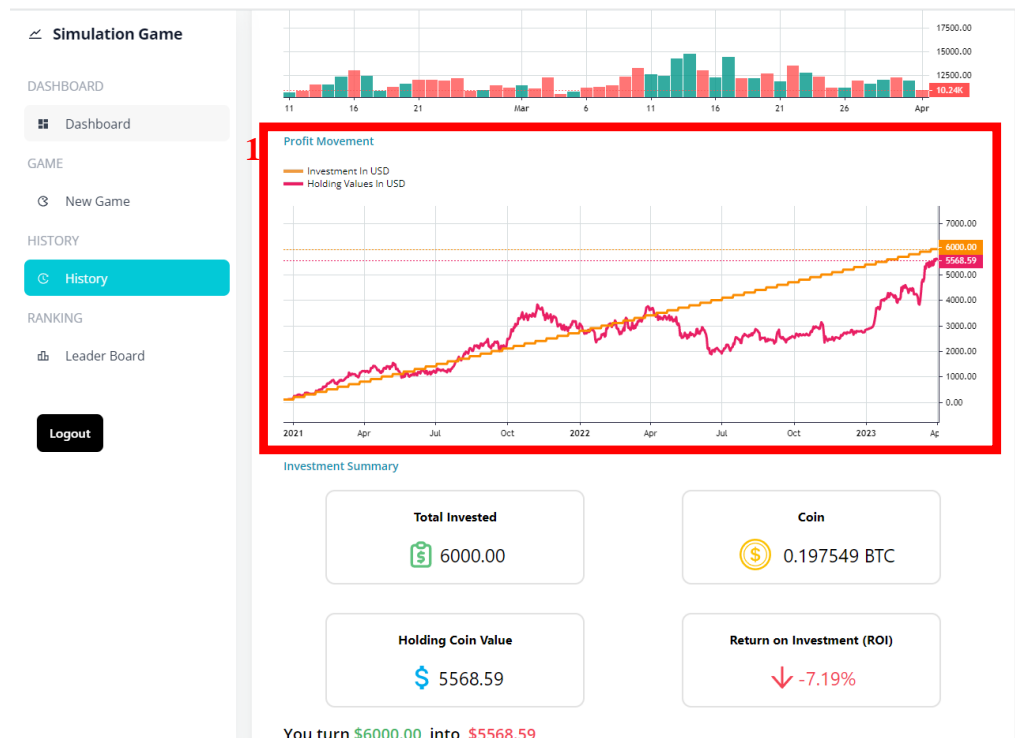


4: The chart turns red if the final value in USD is less than the investment.



5: The investment summary, such as the maximum drawdown and ROI

6: The details execution table



1: The DCA record's profit chart features a line graph with the Investment USD value in orange and the Holding Value in Red, enabling users to compare the two.

7.10. Leader Board

The screenshot shows the 'Simulation Game' dashboard with the 'Leader Board' tab selected (labeled 1). The table displays the top 10 records ranked by ROI. The first three records have rank icons (1st, 2nd, 3rd). Each row includes a 'View' button for details. The table is paginated, showing 10 rows of 2 out of a total of 2 pages.

Rank	Username	Investment	Current Value	Last Update	Current ROI %	Details
1	test1	\$1000	1755.04\$	2023-04-01	+755.04\$ 75.50%	View
2	admin	\$1000	1573.83\$	2023-03-31	+573.83\$ 57.38%	View
3	admin	\$1000	1517.53\$	2023-03-31	+517.53\$ 51.75%	View
4	test1	\$1000	1517.23\$	2023-04-01	+517.23\$ 51.72%	View
5	test1	\$1000	1441.76\$	2023-04-01	+441.76\$ 44.18%	View
6	21027547d	\$1000	1390.12\$	2023-04-01	+390.12\$ 39.01%	View
7	test1	\$1000	1390.12\$	2023-04-01	+390.12\$ 39.01%	View
8	test1	\$1000	1282.68\$	2023-03-31	+282.68\$ 28.27%	View
9	21027547d	\$1000	1214.06\$	2023-03-31	+214.06\$ 21.41%	View
10	test1	\$1000	1135.48\$	2023-04-01	+135.48\$ 13.55%	View

1: The leader board table. It only ranks the record of Custom Rules according to their ROI. The first three will have different Rank Icon. Users can also click the View to check how others set their rules.

8. Limitation

There are a few limitations to the current system. Here are some of the notable ones:

- **Time Interval of Historical Data:**

Using daily timestamps in the historical data may have specific limitations. These include potential limitations in capturing intraday market movements, delayed information, limited timeframes, and increased potential for noise in the data. These limitations may impact the effectiveness of the system's signals and may not be suitable for all trading strategies. It is essential to consider these limitations carefully when using daily timestamps and ensure they align with the employed trading strategy.

- **Trading Fees and Slippage:**

The trading system's lack of consideration for trading fees or slippage can significantly affect live trading. Trading fees, which include commissions, exchange fees, and other costs associated with executing trades, can quickly add up and significantly impact the profitability of the trading strategy.

Slippage, which refers to the difference between the expected price of a trade and the actual price at which it is executed, can also impact the profitability of the system's signals. To account for these costs, it's important to backtest the strategy with trading fees and slippage included and adjust the trading strategy or risk management approach accordingly.

- **Free plan for API services**

Using a free plan for API services may result in limitations and errors. The limit of 5 requests per minute may result in error if many users fetching the historical data. The maximum limit of 2 years of historical data may be insufficient for some trading strategies that rely on longer-term trends and patterns.

- **Flexibility**

The trading system may have limitations in terms of flexibility, such as limited customization options, indicator options, inflexible rules, and limited timeframes. These limitations may impact the system's effectiveness for specific trading strategies and result in missed opportunities or ineffective signals.

9. Conclusions

In conclusion, the simulation trading game system provides users a fun, interactive, and educational tool to learn about algorithmic trading and test their trading strategies in a risk-free environment.

The system's architecture, frontend and backend, and algorithm design have been outlined in detail, along with the API routes and testing procedures.

The three algorithm levels provided in the system, Dollar Cost Averaging, Martingale, and Custom Rules enable users to explore different trading strategies and learn about risk management.

However, there are limitations to the system, such as the time interval of historical data, trading fees, slippage, limitations of using a free API plan, and limited flexibility.

Overall, the system provides a valuable tool for individuals interested in algorithmic trading to learn and develop their trading skills.

References

- 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>
- PolyU, H. (2020, Dec). *COMP4531 Syllabus*. Retrieved from <https://www.polyu.edu.hk/comp/docdrive/ug/subject/COMP4531.pdf>
- PolyU, H. (2021, Jun). *COMP4141 Syllabus*. Retrieved from <https://www.polyu.edu.hk/comp/docdrive/ug/subject/COMP4141.pdf>
- 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/>

Appendices

You can check the README.md in the source code folder for the installation guide.

☰ README.md



Simulation Game for Algorithm Trading

Simulation Game for Algorithm Trading is a trading simulation game designed for users to test their trading strategies using real-world financial data. This README file provides instructions for installation and usage of the application.

Installation

1. Ensure that you have installed the latest version of Node.js and NPM on your local machine.
2. Clone this repository to your local machine using Git:

```
git clone [repository URL]
```

Server Installation

1. Navigate to the server directory and install the server dependencies using NPM:

```
cd server
npm install
```

2. Create a `.env` file or change the `.env.example` file in the server directory and fill it with the following information:

```
PORT=[port number] (e.g. 3000)
MONGO_PW=[MongoDB password]
MONGO_DB_NAME=[MongoDB database name]
HOST=[host name] (e.g. localhost)
JWT_KEY=[JWT secret key] (e.g. myjwt)
POLYGON_KEY=[Polygon API key]
```

You can obtain a MongoDB password and database name by signing up for a [Cloud MongoDB account](#). To obtain a Polygon API key, you can sign up on the [Polygon.io website](#).

3. Run the server using NPM:

```
npm start
```

4. To run unit tests:

```
npm test
```

Web Application Installation

To use the web application, follow these steps:

1. Navigate to the web_app directory and install the dependencies using NPM:

```
cd web_app
npm install
```

2. Open the `.env` file in `web_app/.env` and set the `REACT_APP_SERVER_HOST` variable to the target server host, for example:

```
REACT_APP_SERVER_HOST=http://localhost:3000
```

3. Run the React app using NPM:

```
npm start
```

This will launch the web application in your browser.