The Hong Kong Polytechnic University

Department of Computing

COMP4913 Capstone Project

Interim 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, 03 Jan 2023

Abstract

This is an interim report on an 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.

In addition to some background information and research on the existing system, this report also describes the progress of the first phase of my work, such as the parts that have been completed, the areas and problems identified during the process that can be improved, and future work. Some of the user interfaces that have been completed so far are also attached in Appendices for reference.

Table of Contents

List of tables and figures

1. Introduction

Learning about algorithm trading can be a daunting task 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 incredibly useful.

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 realistic market data and simulated trades to give users a hands-on experience in developing and executing trading strategies. By engaging in this simulation, users can gain a better understanding of 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 out 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 out 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 at hand. This can be particularly helpful for those new to algorithm trading, as 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 a real exchange with fake balance.

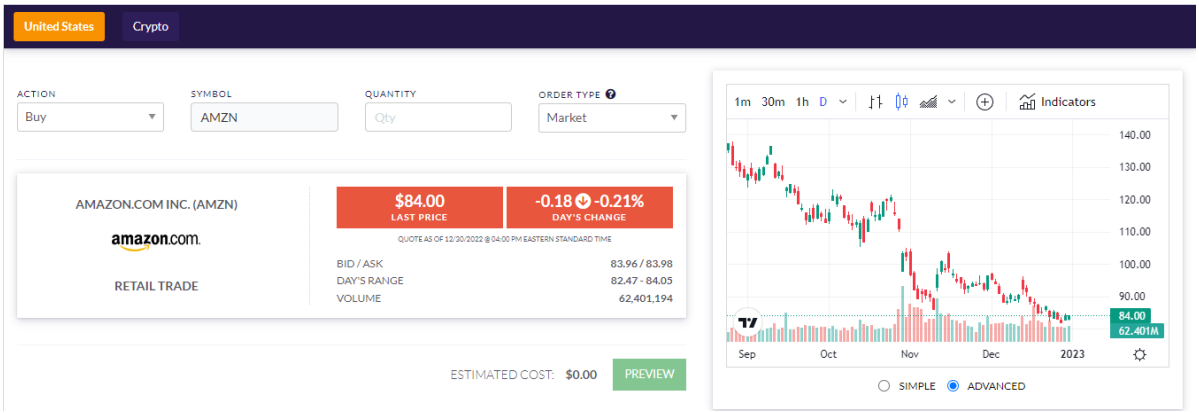


Figure 1 Real-time trading UI (Survivor, 2022)

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 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.

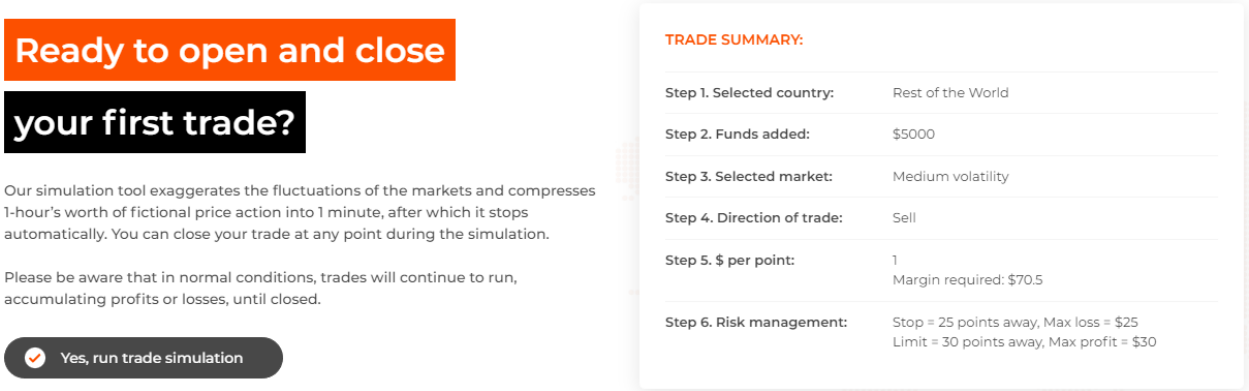


Figure 2 Step by Step setup (Nation, 2022)

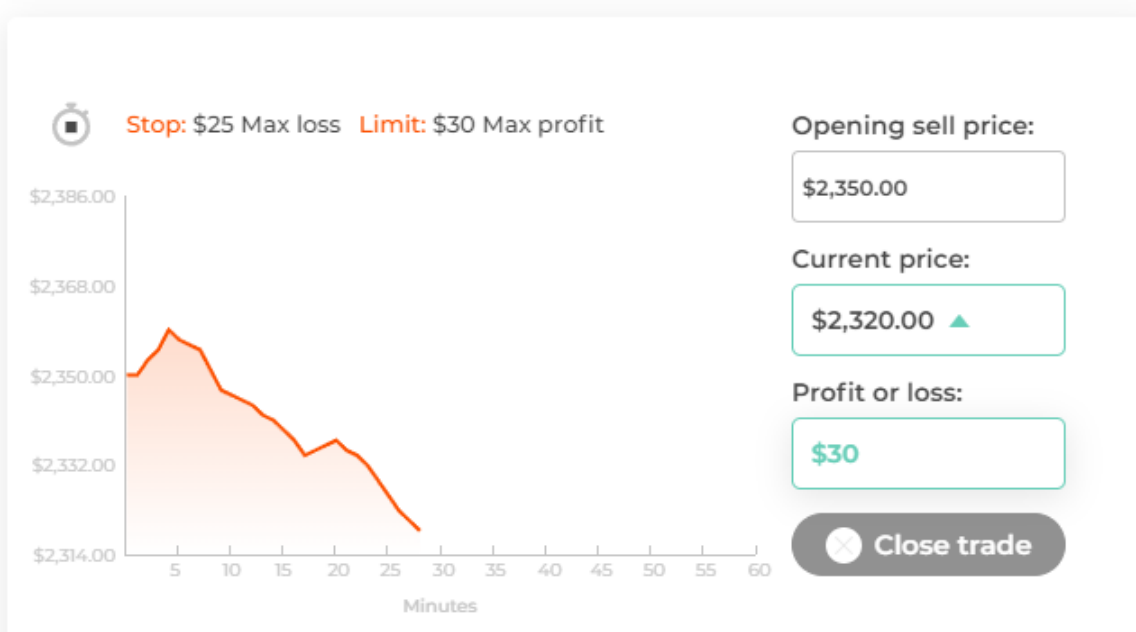


Figure 3 Simulation Result (Nation, 2022)

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 very simple. But I can learn some styles such as the profit movement chart, which can implement in my application.

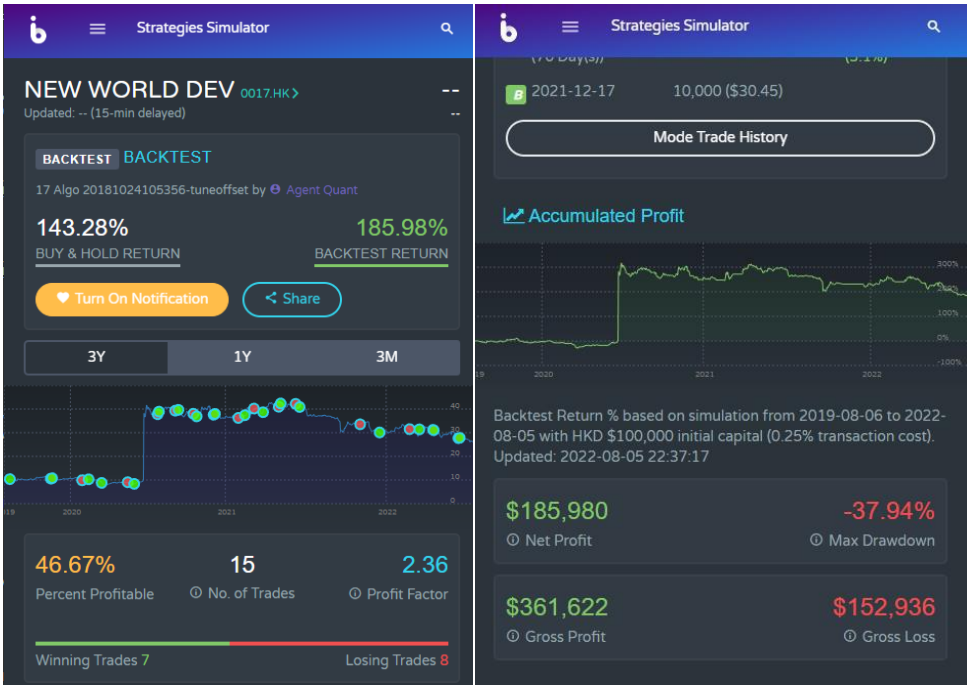


Figure 5 Screenshots of the report page of InvestBot. (InvestBot, 2022)

3. Proposed Solution

3.1. Platform

For easily accessible to a wide audience, I would develop the simulation trading game on a **web-based platform**. It allows users to access the game through a web browser. This means that users do not need to install any specialized software or tools in order to play the game, which can be convenient and reduce barriers to adoption.

Another advantage of using a web-based platform is that it allows developers to create interactive and dynamic applications that can run on a variety of devices and operating systems. This can be particularly useful for a simulation game, as it allows users to access the game from a range of devices, including desktop computers, laptops, tablets, and smartphones.

3.2. Data

Like many existing systems. I will use **historical data** to create a mirror real-world for the user to implement their algorithm. It contains some crypto or stock data for the user to select. And the data will be updated daily. Therefore, instead of downloading some set of historical data. I will use an API call to download the data. That means every time the user calls the function. It will download the up-to-date data.

3.3. Game Play

To make the learning experience more engaging and effective, the game could incorporate a number of features and elements. One such feature could be the ability for users to set up algorithm trades step by step, including both traditional techniques such as the Martingale algorithm and more freeform approaches that involve combining different technical indicators. This element of gameplay could provide a more immersive and hands-on learning experience, helping users to understand the complexities and nuances of algorithm trading.

To support this game play element, the game could include a variety of resources and tools, such as interactive tutorials, step-by-step guides, and visual aids. These resources could help users to understand the different steps involved in setting up an algorithm trade and provide them with the knowledge and skills they need to succeed. The game could also include tooltips or other forms of guidance to help users understand different keywords or concepts, as well as classical algorithms for users to try and more advanced, freeform approaches for more experienced users.

Also, the application will generate a report after each game so that users can review the data of the entire simulation. Perform analysis, reflection or learning from it.

3.4. Backend server

A RESTful backend server will also be developed.

- The purpose of this server is to
- Manage the connection between the application and the Database. Manage the update of historical data
- Doing simulation tasks according to different algorithms

4. System Design

4.1. System architecture

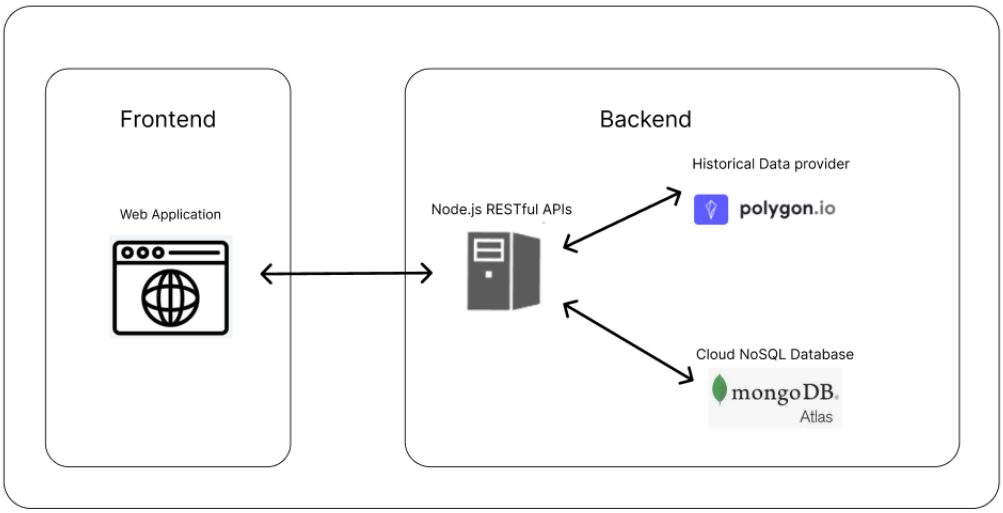


Figure 6 System architecture

Figure 6 shows the basic component involved in the system. The system comprises a front-end and back-end, which respectively to a web-based application and a server running in Windows.

- Frontend
  - A React web application for user to interact with the system
- Backend
  - A RESTful server to receive the user request from frontend. Execute simulation task and communicate with other third part component.
- Historical Data
  - Get historical data using HTTP request. Which can make the data up to date automatically.
- Database
  - MongoDB. A document database to store user account details, simulation setting etc.
- Security
  - JSON Web Token (JWT): JWTs contain the userID and username and the server can use the private key to verify the token is valid or not for route protection purpose.

4.2. Sequence diagram

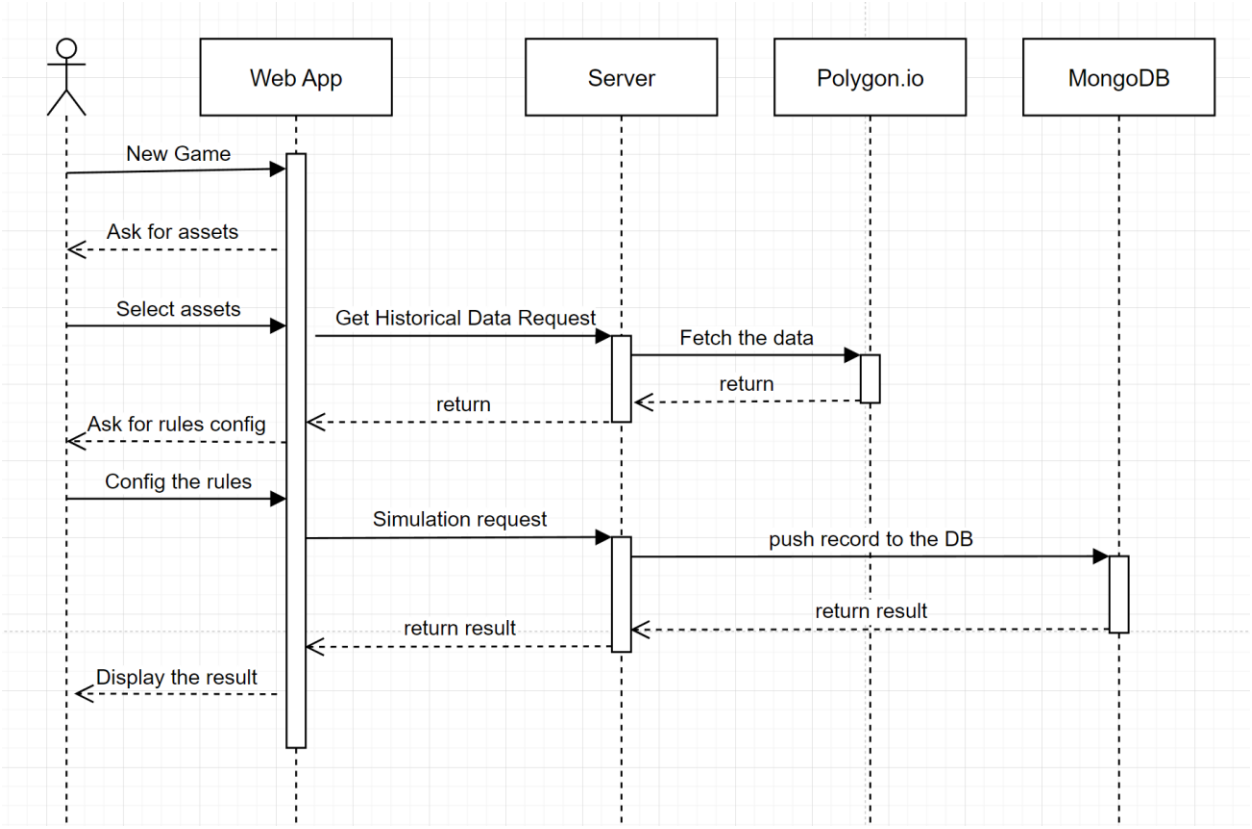


Figure 7 Main flow of the system

Figure 7 show the main flow of the system.

1. User ask for new game
2. Web app will ask for what game assets the user willing to pick
3. User select the assets
4. Web apps send request to the server
5. The server download/update the historical from Polygon.io
6. Get the result and save as JSON in server side.
7. Server return the updated historical data to Web app
8. Web app ask user to input the rules setting
9. User config the rules
10. Web app send the rules to the server and request the simulation
11. Server perform the simulation task and
12. Save the record to the DB
13. Return the result to the Web application
14. Display the result to user

4.3. Frontend

Framework: React.js

Table 1 show the route that the system contain

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

Table 1 Current route of the frontend application

*Means the route are protected route. Only login user can access the page.

There are three algorithms will be provided.

- Martingale (Finish)
    - 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.

    - The reason why I chose this strategy is that every time I play some gambling game, someone will suggest this so-called "Sure win if you have enough money" strategy. However, I know that this strategy is very risky, so I want users to experience the actual risk by trying this strategy. Figure 8 will show the flow of the strategy.

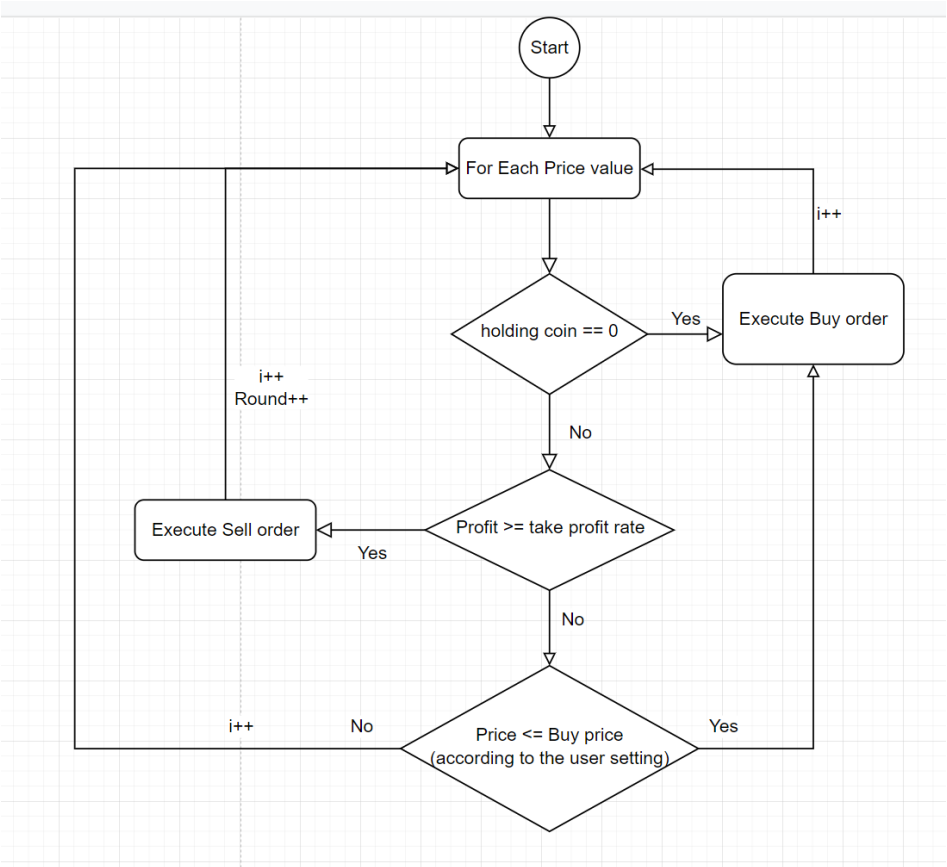


Figure 8 Simple Flow of Martingale Strategy

When implementing this algorithm in the trading market. We need to determine what is WIN and LOSE

For WIN, I will provide an input field called “Take Profit Ratio” for the user to set what WIN is. For example, 1% means when the profit of the holding coin/stock is 1%, then you will sell all the holding to take profit.

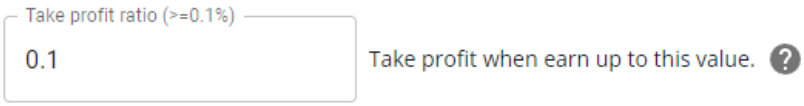


Figure 9 Take Profit Field

For LOSE, the user can set up an array object for the system as a reference to execute the buy order.

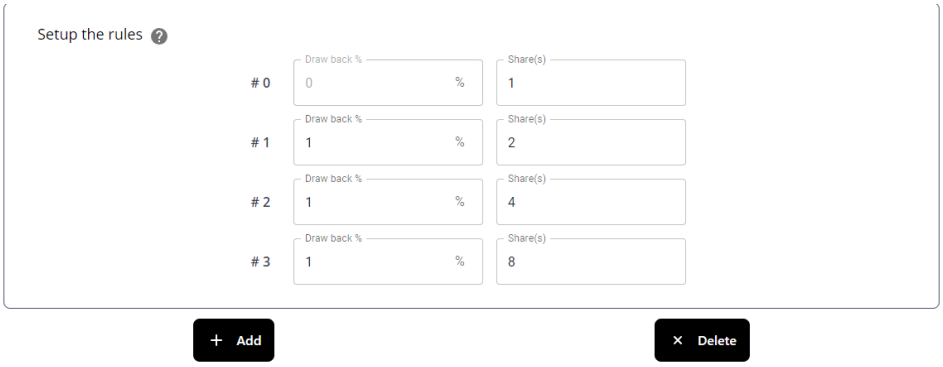


Figure 10 Martingale Buy Setting

Figure10 shows the UI of how the user can set up the rules.

"**Drawback %**" means how much % of the price goes down the system will execute the Buy order. For example, 1% means that when the price is lower than the last Buy Price with 1%. When the day close price lower than this price. It will execute the Buy order.

"**Shares**" mean the division of the investment. Users can edit this value to determine the buying value of each order. Such as not double but triple the shares

- Dollar Cost Average (DCA) (TODO in next stage)
    - The dollar cost average dollar strategy is a way to determine the cost of goods sold and ending inventory value by averaging the cost of all units of the same type of inventory purchased during a given period.
    - DCA can be a simple and effective way to build a diverse investment portfolio, as it allows an investor to gradually accumulate different stocks or other assets over time. I think the user can learn something by trying this strategy

- Custom Indicator (TODO in next stage)
    - This is a more advance options for the user to set up their rules by combine different technical indicator. Such as Moving average, Relative strength index (RSI) and On-balance volume (OBV) etc.
    - The goal is to allow users to learn some different analysis indicators

4.4. Backend

Farmwork: Node.js + express.js + mongoose.js

Route	Description

/user/signUp	/* create user
	URL:localhost:3000/user/signUp
	Method: POST
	body:
	{
	"username": "test",
	"password": "12345"
	}
	*/

/user/singIn	/* login
	URL:localhost:3000/user/signIn
	Method: POST
	body:
	{
	"username": "test",
	"password": "12345"
	}
	*/

/user/edit	/* edit
	URL:localhost:3000/user/edit
	Method: POST
	body:
	{
	"user": "{
	_id : xxxxxxxxxxxx
	username: xxx
	xxx: xxx
	...
	},
	"token": xxxxxxxxx
	}
	*/

/user/view	/* edit
	URL:localhost:3000/user/view
	Method: POST
	body:
	{
	"token": xxxxxxxxx
	}
	*/

/user/viewRecord	/* view user record with record ID
	URL:localhost:3000/user/view
	Method: POST
	body:
	{
	"token": xxxxxxxxx
	"record_id": xxxxxxx
	}
	*/

/simulation/	/* do simulation
	URL:localhost:3000//simulation/
	Method: POST
	body:
	{
	token: xxx,
	type: 1,
	algoType: 1,
	... (rule object with token)
	}
	*/

/his/getCryptoData	/* get historical data (crypto)
	URL:localhost:3000//his/getCryptoData
	Method: POST
	body:
	{
	type: 1,
	ticker: 'X:DOGEUSD',
	from: '2020-12-19',
	to: '2022-12-19',
	token: xxx
	}
	*/

Table 2 The Route that the RESTful Server contain

Apart from /signIn and /signUp, all routes are protected with the AuthToken function. Which is used to verify the token is valid or not.

Example:

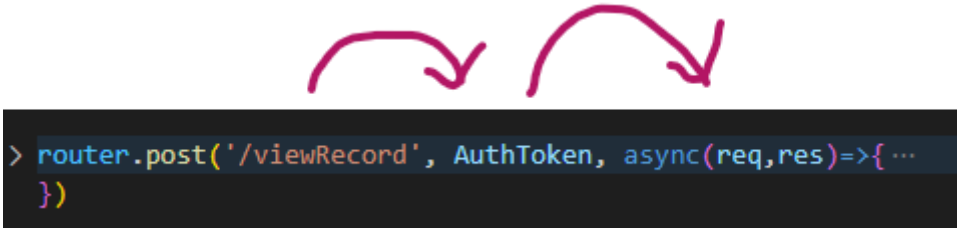


Figure 11 Protected route example

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

5. Progress to date

Table 3 shows the progress of the Milestones/Function proposed in the proposal

Milestone 1		
Job	Description	Progress
Setup React Project	Learn the basic and setup a React project. Such as ○ Build the layout ○ Manage the Route navigation ○ How to protect private route ○ React Hook ○ Learn different frontend library like tailwind CSS, Lightweight chart etc	✓
Setup Server Project	Learn the basic and setup a RESTful Server. Such as ○ (Access Control Allow Origin) CROS setup ○ Mongoose (model/DB connection) ○ Express.js (Route management) ○ JSON Web Token. (Route protection) ○ Hashing password	✓
Setup Database	Create the MongoDB environment for this project	✓
Authorization	○ Develop the UI in frontend for user to sign in or signup ○ Web form validation ○ Storing user token using cookies ○ Using React-Auth-Kit to valid the user	✓
Milestone 2		
Get historical Data	○ Server route for get historical data ○ Logic of updating the historical data ○ Use candlestick chart to display the data	✓
Build the GUI of Martingale	○ Build the Step by Step UI to lead the user to input the setting/rules ○ State object control ○ Form validation ○ Tooltip for explaining the step ○ Send the setting object to the server	✓
Milestone 3		
Build the login of the Martingale	○ Build the simulation Router ○ Build the simulation logic ○ Push the rules object as record to the user record field ○ Return a list of buy/sell execution to user	✓
Summary Page	○ History page to display all the simulation records ○ Mini profit movement chart	✓

	○ Record page, display the summary of a single record ○ Simulation animation using candlestick chart and marker(buy/sell) ○ Details profit movement chart ○ Buy sell record summary table	

Table 3 Completed task

6. Problem and Enhancement

As mentioned in section 5, all milestones 1-3 have been completed. However, many areas could still be enhanced. Such as:

- Result accuracy
  - As all timestamps in the historical data are "day", I use the "daily closing price" as a signal to determine whether the algorithm will execute a buy or sell option. However, sometimes the ups and downs of a day can be very large, for example in Figure 12. The full day highs and lows are 1938.98 and 700 respectively. Using the day as a time span may cause the user to miss a stop loss/profit taking/trade opportunity.

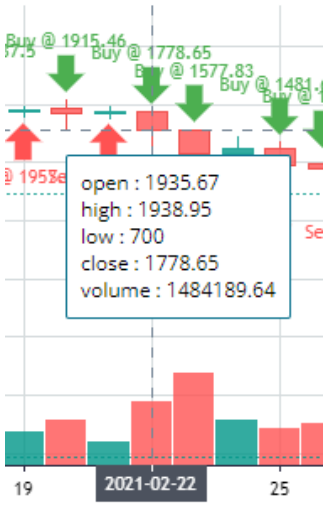


Figure 12 ETH price on 2021-02-22

  - Therefore, I think it is important to let the user know that the time stamp is also an important decision-making position. I will also try to find out the timestamp in smaller units in the future. At the moment there are only a limited number of requests due to the free account of Polygon.io. So, for the time being, I am using the original data in "days".

- Cheating
    - It is easy for people to cheat because they are aware of the historical data. The leader board function has become meaningless. Therefore, in the future, a random data generation function or random historical data selection function may be added to allow users to choose.

- Tooltip text
    - At the moment only some of the features have a tooltip to explain what that is. I will get some users to try them out and add various tooltips based on their feedback when the system is finished

7. Next Stage

Over the next semester, I will focus on completing the remaining development work for the simulation game. This will include finishing Milestones 4-7, which include the functions related to the Average Cost method, custom indicator, and the leaderboard gameplay. Here is the updated Milestones 4-7:

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):

- Leader board game play. (Not using fix historical data, maybe real time data or random data)
- Fix the problem that mention at section 6
- Finish all the tutorial/tooltips

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

- Buffer time for unfinished things/Bug fixing/ final Testing
- Final report / one page summary

8. Conclusions

Overall, the past month has been a productive one for this project. I have made good progress on the development of the new app and have learned a lot about programming skills and knowledge in the trading aspect. I have also had the opportunity to work on some new and challenging problems, which has helped me to expand my knowledge and capabilities in this area.

One of the key challenges that I have faced during this period has been the start of the project. which requires learning how to set up the web application and server. Reading a lot of documentation and video consume so much time, but I have been able to overcome this by finishing bit by bit every day.

Looking ahead, I plan to continue working on the app and to focus on other algorithms, such as the Average Cost method and a more flexible custom indicator method. I am confident that with continued hard work and dedication, I will be able to successfully complete this project and achieve my personal goals.

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

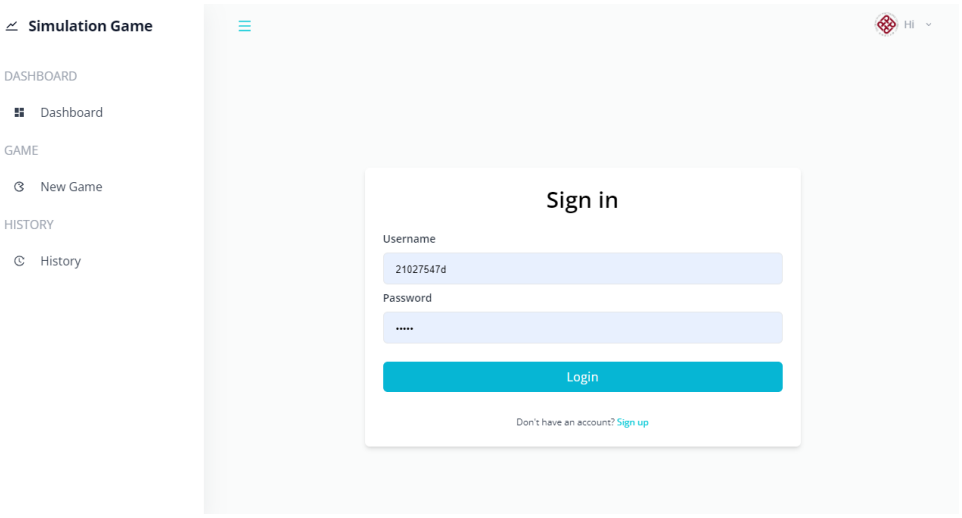
Survivor, W. S. (2022, 10 09). *Wall Street Survivor*. Retrieved from https://www.wallstreetsurvivor.com/

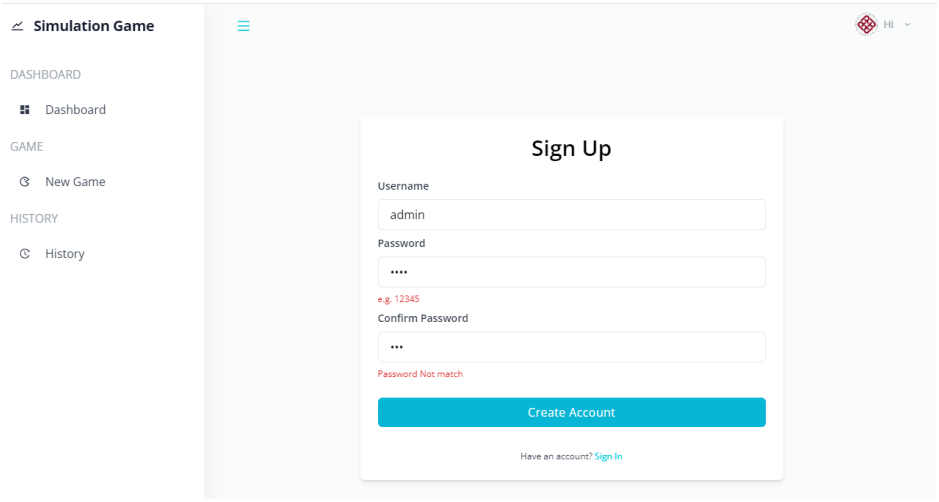
TradingView, I. (2022, 10 09). *TradingView*. Retrieved from https://www.tradingview.com/

Appendices

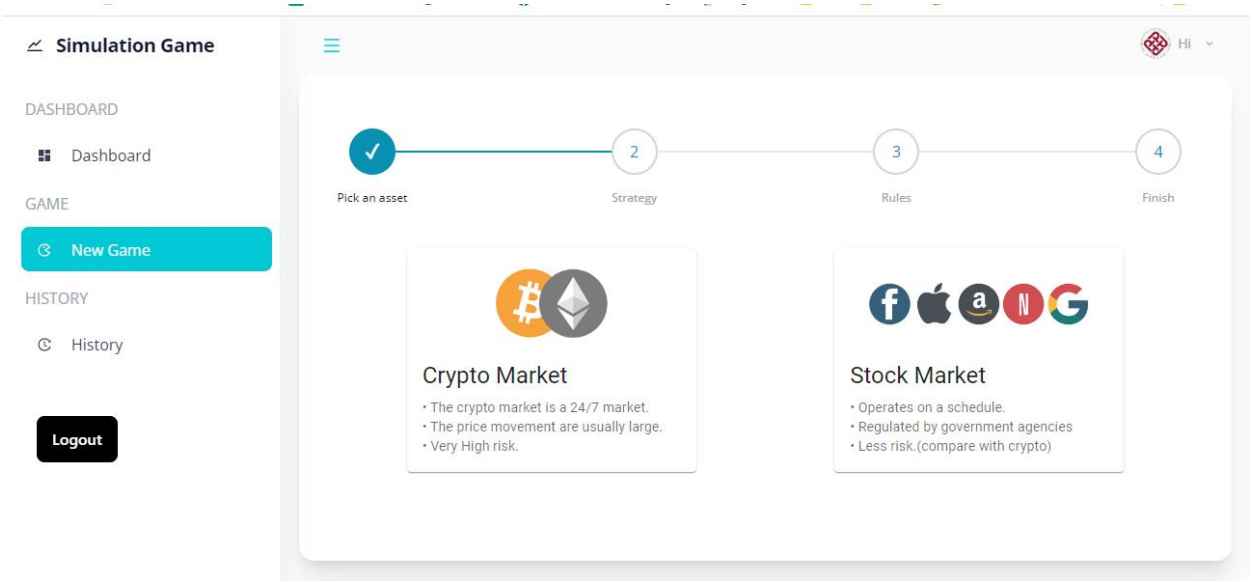
Current UI

Sign In/ Sign Up

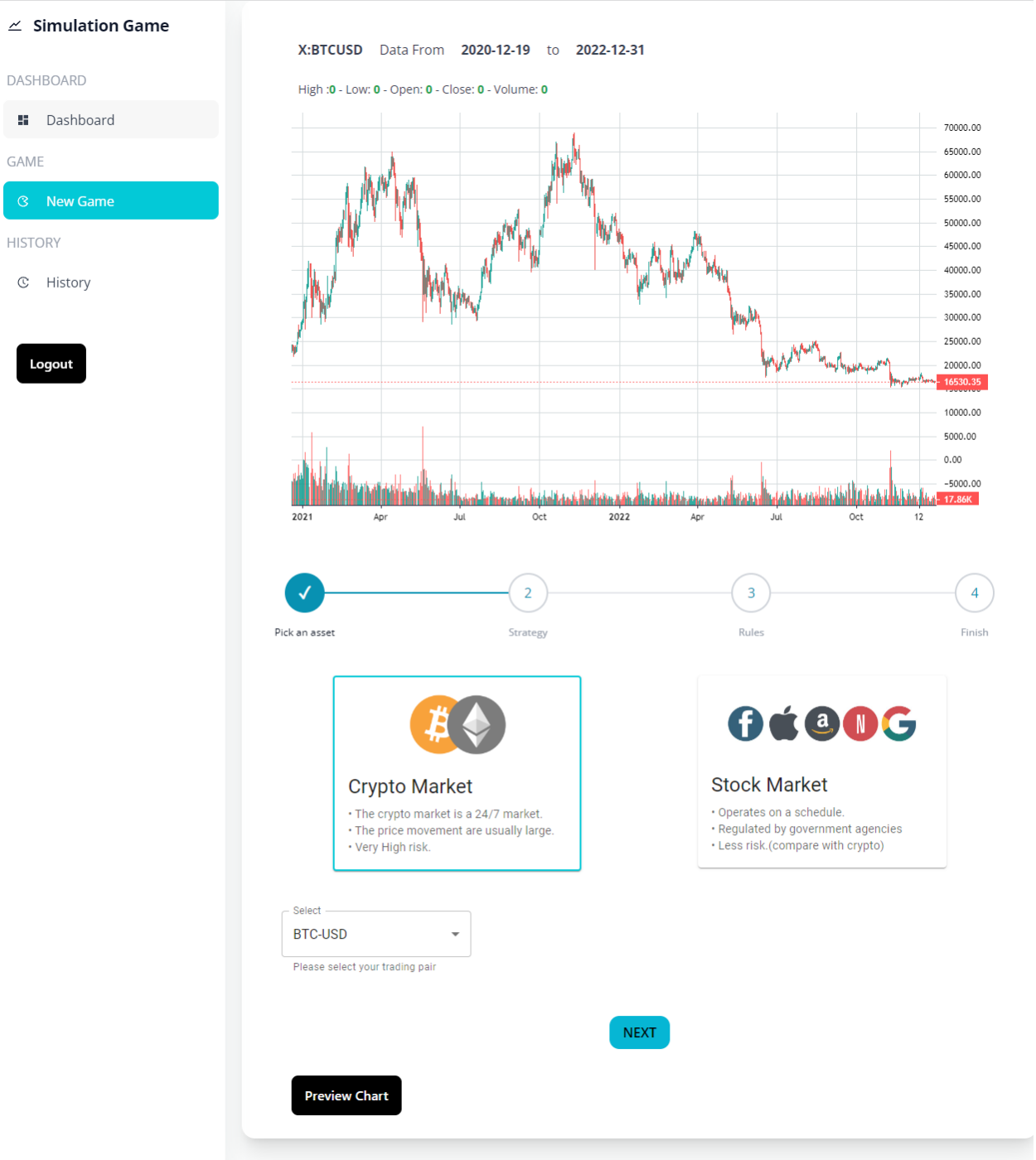




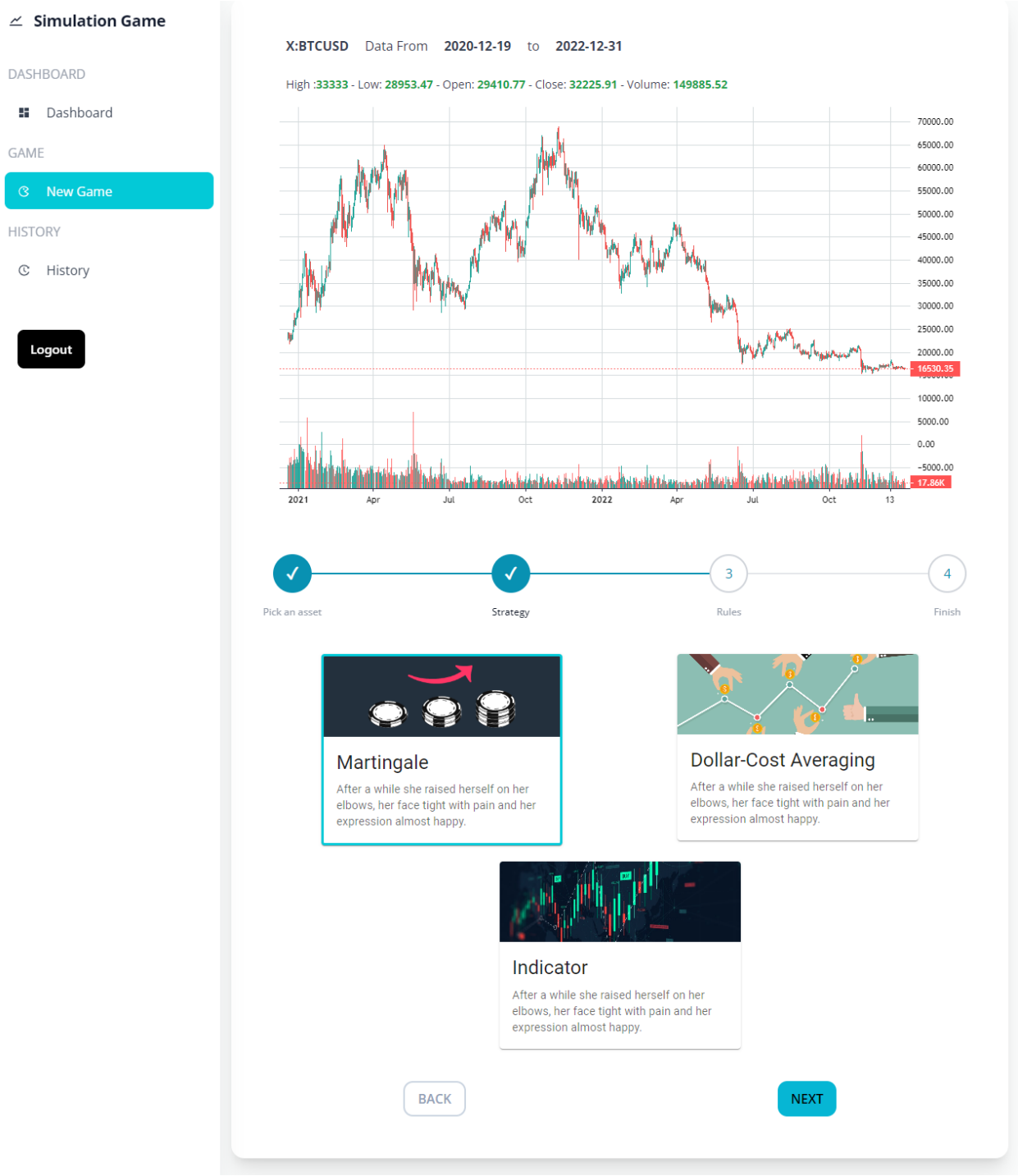
New Game



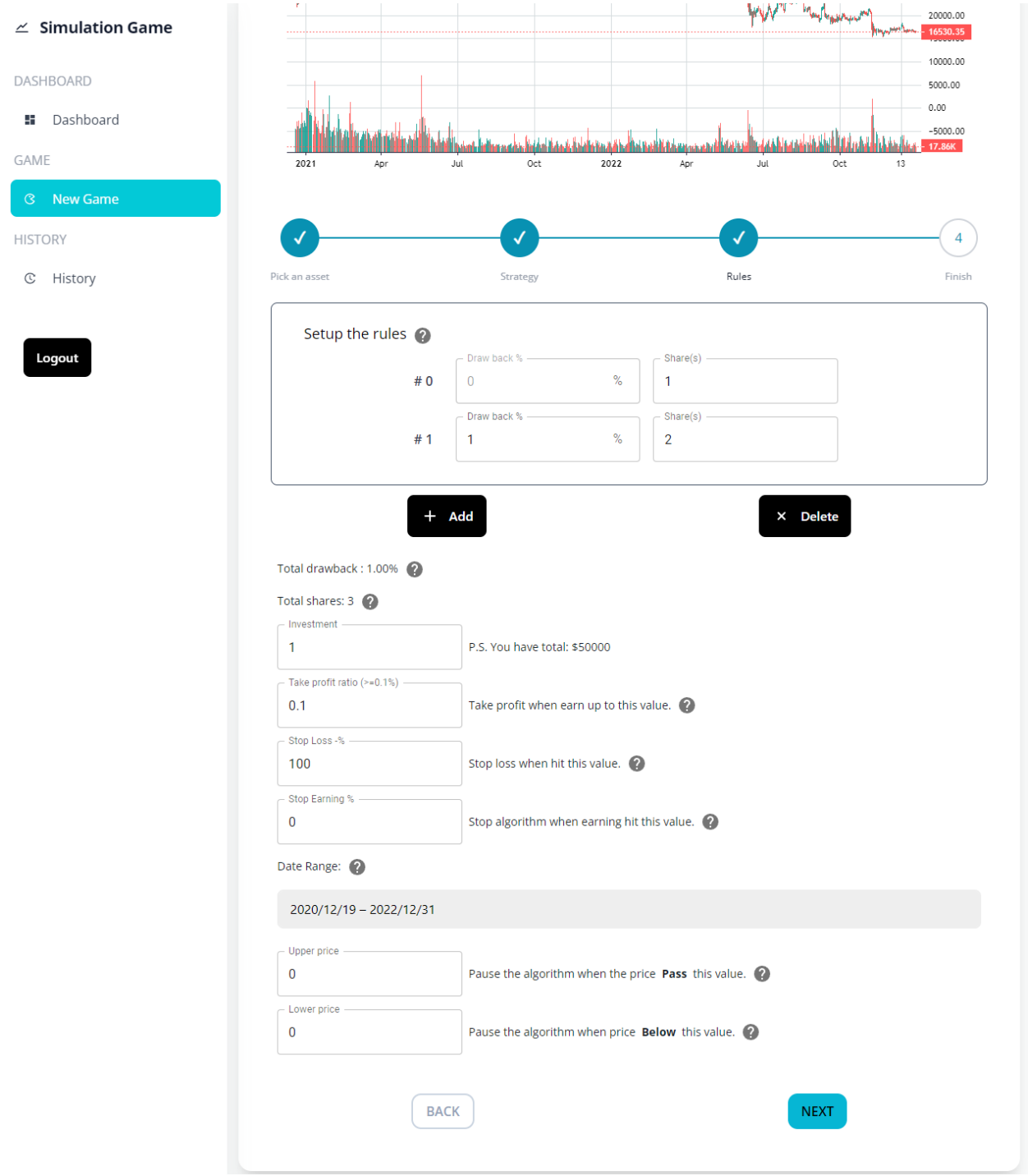
Pick assets


Simulation Game
DASHBOARD
Dashboard
GAME
New Game
HISTORY
History
Logout
X:BTCUSD Data From 2020-12-19 to 2022-12-31
High :0 - Low: 0 - Open: 0 - Close: 0 - Volume: 0
70000.00
65000.00
60000.00
55000.00
50000.00
45000.00
40000.00
35000.00
30000.00
25000.00
20000.00
16530.35
10000.00
5000.00
0.00
-5000.00
17.86K
2021
Apr
Jul
Oct
2022
Apr
Jul
Oct
12
Pick an asset
2
Strategy
3
Rules
4
Finish
Crypto Market
• The crypto market is a 24/7 market.
• The price movement are usually large.
• Very High risk.
Stock Market
• Operates on a schedule.
• Regulated by government agencies
• Less risk.(compare with crypto)
Select
BTC-USD
Please select your trading pair
NEXT
Preview Chart

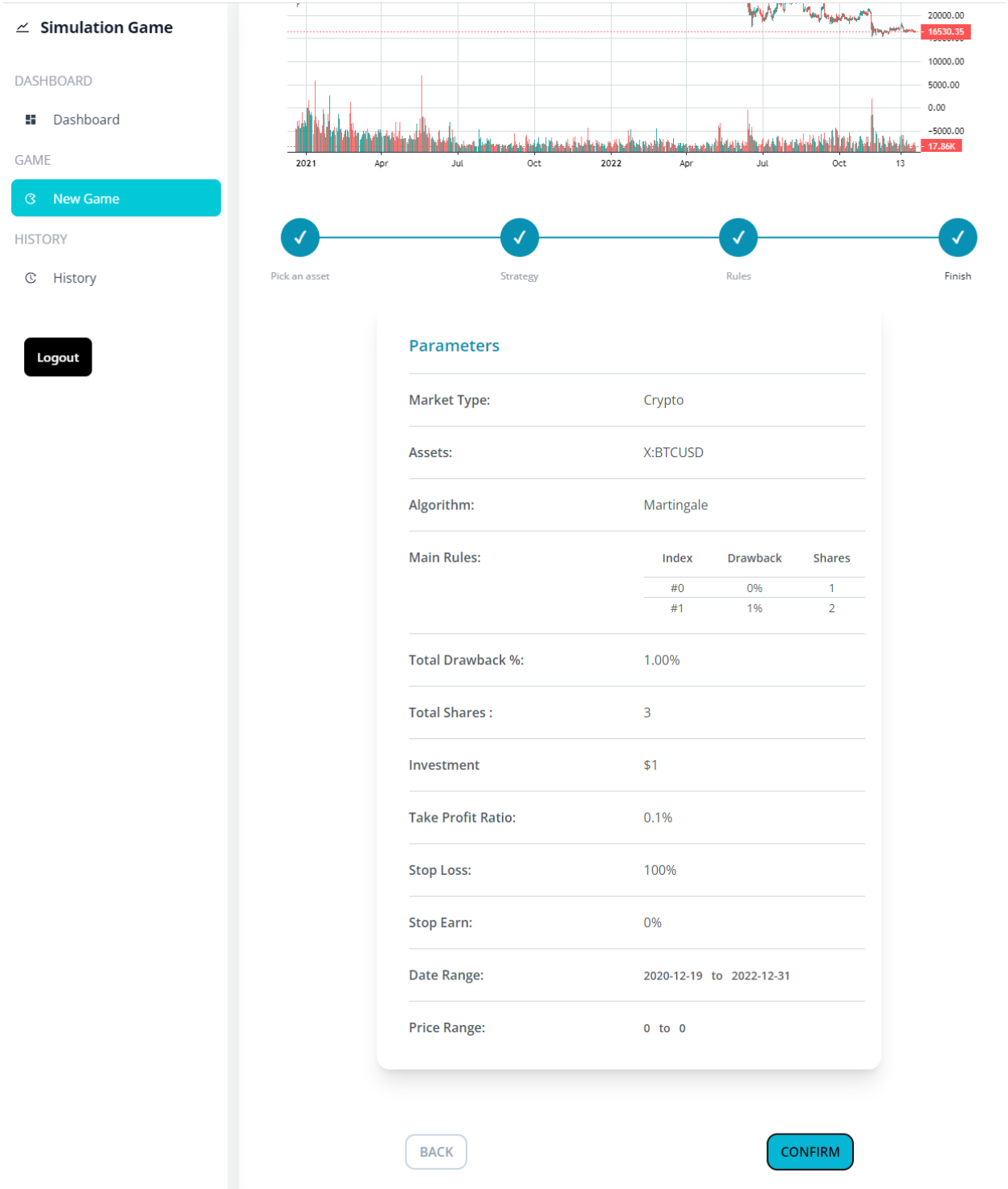
Select Algorithm



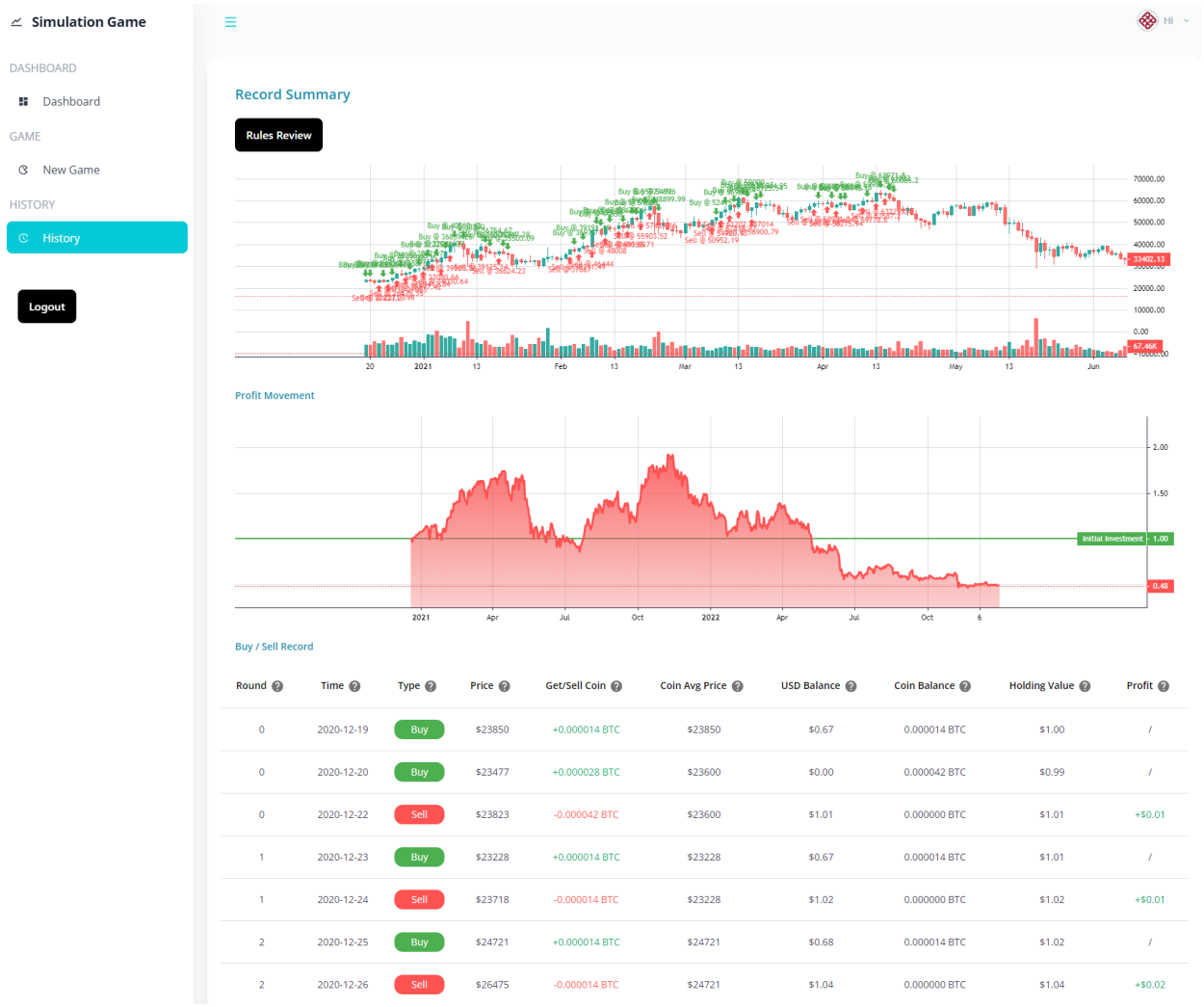
Rules setting of Martingale



Confirmation page

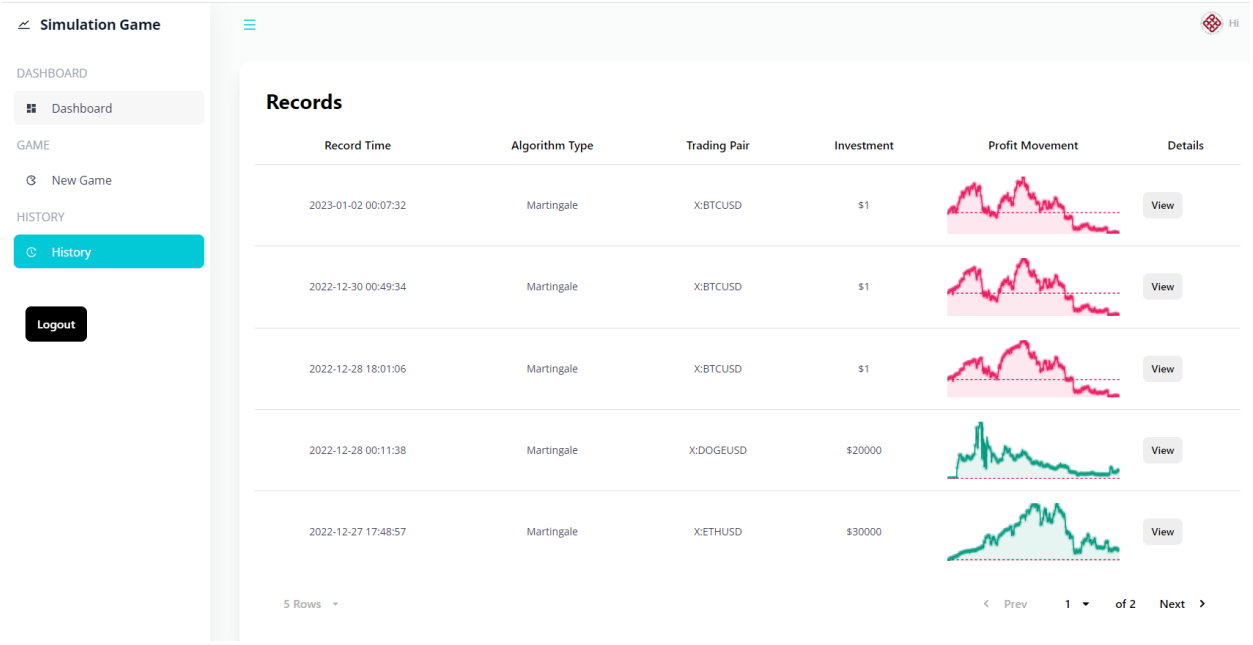


Simulation Result page



Round	Time	Type	Price	Get/Sell Coin	Coin Avg Price	USD Balance	Coin Balance	Holding Value	Profit
0	2020-12-19	Buy	$23850	+0.000014 BTC	$23850	$0.67	0.000014 BTC	$1.00	/
0	2020-12-20	Buy	$23477	+0.000028 BTC	$23600	$0.00	0.000042 BTC	$0.99	/
0	2020-12-22	Sell	$23823	-0.000042 BTC	$23600	$1.01	0.000000 BTC	$1.01	+$0.01
1	2020-12-23	Buy	$23228	+0.000014 BTC	$23228	$0.67	0.000014 BTC	$1.01	/
1	2020-12-24	Sell	$23718	-0.000014 BTC	$23228	$1.02	0.000000 BTC	$1.02	+$0.01
2	2020-12-25	Buy	$24721	+0.000014 BTC	$24721	$0.68	0.000014 BTC	$1.02	/
2	2020-12-26	Sell	$26475	-0.000014 BTC	$24721	$1.04	0.000000 BTC	$1.04	+$0.02

Record Page


Simulation Game
DASHBOARD
Dashboard
GAME
New Game
HISTORY
History
Logout
Hi
Records
Record Time
Algorithm Type
Trading Pair
Investment
Profit Movement
Details
2023-01-02 00:07:32
Martingale
X:BTCUSD
$1
View
2022-12-30 00:49:34
Martingale
X:BTCUSD
$1
View
2022-12-28 18:01:06
Martingale
X:BTCUSD
$1
View
2022-12-28 00:11:38
Martingale
X:DOGEUSD
$20000
View
2022-12-27 17:48:57
Martingale
X:ETHUSD
$30000
View
5 Rows
Prev
1
of 2
Next