



Applied Industry Project : Stock Trading Bot

Group A3 – [Br@in]

Benny, Abin

Birru, Shivaganesh

Bhatt, Jaldhi Himanshu

Duarte Junior, Fabio

Patel, Manali Shaileshkumar

Applied Industry Project – 2024, Loyalist College in Toronto

AIP Project Advisor: Mr. Usman Ahmad

AIP Coordinator: Mr. Stanley Chor

April 08, 2024



Abstract

Stock trading involves the movement of trillions of dollars being invested daily with millions of decisions made by both humans and automation systems. The competitive landscape is extremely high, and traditional automation tools serve merely as triggers for user-predefined decisions. Thus, based on this scenario, we were challenged by the college to build a Stock Trading Bot based on Deep Reinforcement Learning techniques (referred to in this document as DRL), a solution aimed at modernizing existing technology. It would be capable of automating high-frequency trading using state-of-the-art technology, reducing risks, and increasing returns, with the goal of surpassing current technologies.

In conducting a study on market pain points and how AI has been applied in other sectors, parallel discussions also raised questions such as "Are AI solutions, like trading automations, increasing the gap between the rich and the poor?".

Therefore, we decided that our solution should not only modernize current solutions but also be accessible by reducing the complexity of the trading process to democratize access to our platform.

Our project was based, for the most part, on the pre-existing work of the AI4Finance-Foundation through its research, papers, and Python libraries made available in its open repository. This details our approach, focusing on technological innovation and the democratization of access to AI, based on the FinRL, FinRL-Meta, and FinGPT libraries to create not just a stock trading bot but a more comprehensive trading platform based on AI and cloud technologies.

**Table of Contents**

Abstract.....	2
Introduction.....	4
Literature Review	7
Stock Market domain and available technologies.....	7
Regarding insights and compliance.	10
Methods	11
FinRL and Deep Reinforcement Learning (DRL).....	11
Exploring Libraries	11
FinRL and FinRL-Meta	11
ElegantRL.....	12
Deep Q-learning and other approaches from scratch	13
FinGPT	13
Deep Reinforcement Learning Algorithms.....	14
Available algorithms and their characteristics.....	14
Available Datasets.....	15
Testing FinRL	16
Training and testing the agents:.....	18
A2C (Advantage Actor-Critic).....	19
DDPG (Deep Deterministic Policy Gradient).....	19
PPO (Proximal Policy Optimization)	19
TD3 (Twin Delayed Deep Deterministic policy gradient)	20
SAC (Soft Actor-Critic)	20
Trading (testing).....	21
Architecture	22
BrainLib.....	24
Batch Process	25
Findings.....	28
Hidden Overfitting	28
FinRL Installation and alternatives for CUDA.....	29
Creation of our own version of FinRL : FinRL-Mini.....	31
Ngrok https tunnel	33
Discussion.....	35
Our final product	35
Disclaimers and important point about the use of AI.	36
Simulation API running on AWS.	37
Motivation	37



BrainAPI Architecture.....	38
Swagger documentation	40
Walking though the Features.....	41
Main Page	41
UI Responsiveness	43
Subscription and Payment methods.....	44
Logged Session	45
Dashboard	46
Stock Marketplace.....	46
Manage Bots	47
View Assets.....	48
Run Simulation.....	48
AI Knowledge Hub	49
Profile	50
Conclusion	51
Wrapping up.....	51
Acknowledgement.....	51
Disclaimer	51
Recommendation	52
References	53

Introduction

The stock market is highly sought after as an option for asset generation, especially in the current economic situation. For example, research conducted by the website Finder shows that 26% of Canadians turn to the stock market for long-term savings purposes between 2022 and 2023. (King, R. (2023, May 10) Statistics and facts about the stock market. Finder Canada. <https://www.finder.com/ca/stock-trading/stock-trading-statistics>) [04].

On the other hand, the same research indicates that 37% of Canadians lack confidence in the trading market [04], which is understandable given the volatility and inherent risk of this niche, as well as the amount of financial education needed, an overload of information, and the necessity to closely follow every change happening through news related to target companies or



via information providers. In summary, there are many opportunities for expansion in this market if the trading process can be somehow simplified and/or made more accessible to the public, thus also easing life.

To achieve this goal, technology plays an essential role, especially when we talk about optimizing strategies in the complex and dynamic context of the Stock Market. Thus, guided by our Project Advisor, we explored the use of AI techniques based on the Deep Reinforcement Learning approach, inspired by the work of the AI4Finance-Foundation[\[01,02,03\]](#), to create an automation capable of performing stock trading, or as the Project Charter defines, “high-frequency trading.” Nevertheless, as already mentioned, our objective is not only to automate these operations but also to democratize the use of these technologies and the stock trading market. Therefore, we attempted to wrap around our solution in a platform with an easy-to-use interface and features that assist and instruct the user to optimize their experience and results when using our solution.

To achieve our goals, we utilized the following technologies:

- **Markov Decision Process** (MDP) through Deep Reinforcement Learning. (For the stock trading bot)
- **Natural Language Processing** (NLP), through the RAG (Retrieval Augmented Generation) technique, which combines simpler NLP techniques to identify user intent with APIs of Large Language Models (LLM) for user-friendly responses (For the Chatbot Aka AiKnowledgeHub)
- **FinRL & FinRL Meta**, a Python library made available by the AI4 finance team, used to train, test, and as a basis for creating simulated trading environments. [\[01,02,03\]](#),



We also used some free APIs and external data as follows:

- **Alpaca API** – For news and real-time stock prices and Exchanges.[[21](#)]
- **Finnhub API** – For obtaining news data and financial indicators of symbols for the batch process that populates the database.[[23](#)]
- **Cohere API** – API and library for accessing the LLM model “Command,” to be used by the chatbot. [[22](#)]

The following cloud technologies were also used to execute our solution:

- **Firebase** – Part of the Google Cloud Platform, responsible for database authentication (Firestore). [[24](#)]
- **AWS Cloud** – We used a free tier EC2 server to run our chatbot and our simulation feature based on historical data.[[20](#)]
- **Vercel** – We used Vercel, in the free tier, to execute our front-end. Vercel is a cloud platform with deployment integrated with GitHub, which greatly facilitated our work.
- **GitHub** – Used as a repository for our project with deployment integrated with the Vercel platform.
- **Ngrok** – This is an HTTPS “tunnel” used to convert our HTTP application to HTTPS for free and with few restrictions. This tool provides a fixed domain and an HTTPS tunnel application that can link any flow from a local server to the HTTPS port of your domain.[[05](#)]
- **Nginx** - Advanced Load Balancer, Web Server, & Reverse Proxy
- **Gunicorn** - Gunicorn is a HTTP server for WSGI applications.



- **Yahoo! Finance** – We used data extracted from this platform to train our DRL model and to execute simulation functionalities based on historical data and trading simulation. The data used are the symbols of the Dow30, a “portfolio” of the top 30 symbols selected as a market reference by Dow Jones.

Literature Review

Stock Market domain and available technologies.

As previously mentioned, our project was inspired by the work of the AI4Finance team, primarily through the FinRL, FinRL-meta, and FinGPT libraries[[01](#),[02](#),[03](#)]. However, before initiating any development or solution design, we needed to deepen our understanding of the market. For this purpose, we utilized the Investopedia platform. Investopedia serves as a database of articles about the financial market at large and is respected by numerous entities such as the Gramercy Institute, SABEW, and the Financial Communications Society, having received awards from these entities in 2020, 2021, and 2022[[14](#)]. Its articles are also cited in the financial sections of major news portals like NBC News, CNBC, and Yahoo Finance.

Given its undeniable credibility, editorial standards, fact-checking, and many other mechanisms to keep information as reliable as possible, we used it for research on understanding the business, market volatility, and specific terms of this domain. We observed that the information obtained from the mentioned sources and the way technology is being approached are in resonance with what is currently being used in the market. For example, the use of the volatility



indicator VIX as a tool to measure market sentiment and future volatility [15,16,17]. We note that the implementation of AI4Finance papers uses this type of approach for paper-trading model training. According to our interpretation, these data demonstrate that one of the market's major concerns is volatility, and various strategies and indices have been created, such as the VIX, on how to address this issue, which is reinforced by the technical papers researched. It's worth mentioning that we also spent extensive time conducting tests and following the various tutorials available on GitHub and the official page of the FinRL library.

For a broader view, we also consulted research sources that specifically demonstrated how people who are not considered professional traders view the market. Our research team discovered some data on this in the research conducted by the award-winning personal finance writer and real estate expert, Romana King(2023) [04], where she summarizes data from other research conducted between 2022 and 2023, stating relevant points like:

- "41% — Of [Canadian] investors do not invest in the stock market".
- "1 in 4 — Canadians were not confident in the stock market in 2022".

According to King, R. (2023, May 10). Statistics and facts about the stock market. Finder Canada.

<https://www.finder.com/ca/stock-trading/stock-trading-statistics>. [04]

To shape our technical design approach, we studied the AI4Finance team's papers from 2020, 2022, and 2023 (they mention 2024, but the publication date is in 2023) [01,02,03]. They explain the need for more advanced techniques to optimize trading strategies even in a dynamic environment. Besides discussions on the subject, they present a solution, a set of solutions encapsulated in the form of two libraries with complete tools for historical data manipulation,



DRL agent training, and various simulated trading environments, some with integrations to external brokers' APIs like Alpaca. Please refer to the reference section

To date, they maintain and update these libraries through a community with the principle of bringing state-of-the-art AI solutions to the financial market.

In summary, the researched data seem to indicate that trading market professionals are always looking for ways to mitigate the problems caused by market volatility. Investors, both professional and Canadian respondents in the surveys, are fearful or somewhat reluctant to invest in stock trading. On the other hand, there are studies, research, and a community leveraging modern technology for financial purposes addressing the same problems mentioned and maintaining an open-source community. Unfortunately, within the time constraints we had for research, we did not find a viewpoint in the literature that could contrast with what we found during this phase to have a more sober view on the topic. However, given that the use of AI4 Finance solutions was a project requirement and the research aligns with what they publish in their papers, we understood this to be sufficient to validate and clarify the directions and forms the project should take.

Thus, we combined the knowledge from these market research with the available technical knowledge on state-of-the-art (SOTA) AI technology to design a system that was simple to use, yet powerful, but could be executed within the constraints of our course, such as not having financial or technological support, having to use our own resources and freely available resources.



Regarding insights and compliance.

Research was conducted, primarily based on the IIROC (Investment Industry Regulatory Organization of Canada), focused on specific regulations for the use of AI and automation in Canada.

Here are the most important points produced by the research [10,11,12].

Automated Order System Definition: System generating or transmitting orders automatically on a pre-determined basis. Encompasses hardware and software, including smart order routers and trading algorithms.

Market Impact Risks: High-speed order transmission can quickly impact markets if issues arise. Marketplace participants are accountable for automated order system use, requiring prevention of interference with fair and orderly markets.

Regulatory Mandates: Marketplace participants must take reasonable steps to prevent interference. General understanding and annual testing of automated order systems are required.

Confidentiality Considerations: Acknowledgment of confidential client information but emphasizes the need for a sufficient level of knowledge to manage risks.

Fair and Orderly Markets Support: Provisions aim to support fair and orderly market functioning upon deploying smart order routers, trading algorithms, or other automated order system aspects.

Risk Mitigation Controls: Requirement for controls, including a "kill switch," to disable malfunctioning automated order systems promptly.



Essential Market Functioning: Recognition of the necessity of controls in mitigating risks posed by automated order systems to market functioning.

Methods

FinRL and Deep Reinforcement Learning (DRL)

The cornerstone of our project is the use of FinRL and its other components for the training, testing, and trading process of the DRL agents that are responsible for the trading transactions.

For this purpose, we based our research primarily on the paper by Liu, X.-Y., et al. (2022), "FinRL-Meta: Market environments and benchmarks for data-driven financial reinforcement learning," presented at NeurIPS, and on the official web documentation of the respective software/libraries.

Through this, we evaluated potential Agents, environments, and libraries.

Exploring Libraries

FinRL and FinRL-Meta

These paired open-source libraries provide both a pipeline for creating Reinforcement Learning models and allow the creation of a diverse training environment with a variety of data sources and integration with other libraries. Although the documentation and the library seem a bit outdated, with broken links in the 2021 documentation (see the references section) and version conflicts during installation, we were able to install and test their capabilities. Currently, they seem to be unified, which reduces implementation complexity. After testing and reviewing the papers and documentation, we can conclude that it is a very powerful and flexible library, suitable for our purposes. We will refer to them as one entity throughout the rest of the document. The



documentation for the library includes guidelines on how to conduct the installation on AWS Cloud environment and docker image.

It is divided into three layers, transparent to the implementer, abstracting much of the complex training process.

- **Data Layer:** It essentially provides functions to load the data it is integrated with, as well as for data cleaning and feature engineering.
- **Environment Layer:** It creates market environments in the standard used by OpenAI Gym, incorporating specific techniques to simulate real-world markets.
- **Agent Layer:** This is where the DRL algorithms are available, provided through an internal implementation of the ElegantRL library. The algorithms cater to various types of applications, such as stock trading and portfolio management, each with its characteristics, advantages, and disadvantages. However, we tested some using the tutorials provided in the documentation and saw promising results.

ElegantRL

Offers open-source DRL algorithms suitable for both continuous and discrete actions, as well as for multi-agent reinforcement learning scenarios. We indirectly tested some agents from this library via FinRL; however, ElegantRL stands out as a promising option should we choose an environment other than FinRL, such as OpenAI Gym. One notable advantage is that, unlike FinRL which is TensorFlow-based, ElegantRL operates on PyTorch, compatible with our available GPU hardware. Moreover, ElegantRL has minimal dependencies, requiring only Python version higher



than 3.6 and PyTorch above 1.0.2, making it more manageable. Therefore, if compatibility issues arise with FinRL, we can utilize ElegantRL independently.

Deep Q-learning and other approaches from scratch

We came across projects like the trading-bot by Prabhsimran Singh (refer to the reference section), which implements the Q-learning algorithm from the ground up. Although it's not a library by itself, it acts as a benchmark for building from scratch. Nevertheless, we think that replicating such an algorithm would be redundant since ElegantRL already offers an optimized DQN agent with support from the open-source community. Opting for an already established algorithm appears to be more beneficial than developing our own within this constrained timeframe. We also found another intriguing example, built from scratch, which employs a web scraper to fetch data from Yahoo and integrates a news API connection to perform sentiment analysis for making buy and sell decisions. While this approach is interesting, it does not use DRL, which may not align well with our project's expectations.

FinGPT

FinGPT is an open-source framework designed for Financial Forecasting and Advice, leveraging Natural Language Processing (NLP), specifically Large Language Models (LLMs). Within the scope of our project, it could serve as a Robo-Advisor, aid in Portfolio Optimization, and perform Financial Sentiment Analysis. The tool can be trained out-of-the-box using LLaMA or ChatGLM algorithms, but it also offers the flexibility to integrate with other LLMs, like ChatGPT, via APIs. It utilizes data from news, trends, and social media to create a repository for sentiment analysis. We tested the Robo-Advisor implementation and, although the results were impressive (it



searches for news about a company or ticker, then makes a forecast and provides investment advice based on sentiment analysis), the process is time-consuming. This leads us to believe that we might lack the computational resources required to run these more advanced technologies efficiently. We also reviewed the related paper, Yang, H. (Bruce), et al. (2023). FinGPT: Open-Source Financial Large Language Models. <https://doi.org/10.48550/arXiv.2306.06031>, but have not yet completed our research and understanding of its features.

Deep Reinforcement Learning Algorithms

Available algorithms and their characteristics

Algorithms	Input	Output	Type	State-action spaces support	Finance use cases support	Features and Improvements	Advantages
DQN	States	Q-value	Value based	Discrete only	Single stock trading	Target network, experience replay	Simple and easy to use
Double DQN	States	Q-value	Value based	Discrete only	Single stock trading	Use two identical neural network models to learn	Reduce overestimations
Dueling DQN	States	Q-value	Value based	Discrete only	Single stock trading	Add a specialized dueling Q head	Better differentiate actions, improves the learning
DDPG	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Being deep Q-learning for continuous action spaces	Better at handling high-dimensional continuous action spaces
A2C	State action pair	Q-value	Actor-critic based	Discrete and continuous	All use cases	Advantage function, parallel gradients updating	Stable, cost-effective, faster and works better with large batch sizes
PPO	State action pair	Q-value	Actor-critic based	Discrete and continuous	All use cases	Clipped surrogate objective function	Improve stability, less variance, simply to implement
SAC	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Entropy regularization, exploration-exploitation trade-off	Improve stability
TD3	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Clipped double Q-Learning, delayed policy update, target policy smoothing.	Improve DDPG performance
MADDPG	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Handle multi-agent RL problem	Improve stability and performance

FinRL. (2021). DRL Agents. Retrieved January 28, 2024, from https://finrl.readthedocs.io/en/latest/start/three_layer/agents.html#elegantrl-drl-library

Within the FinRL documentation itself, we found definitions and advice on top state-of-the-art DRL Agents, and to our surprise, the library mentioned already has an integrated implementation of the ElegantRL library and its algorithms, which can be easily implemented. We chose three models to test, using the default configurations from the documentation (refer to the "references" section) with an initial budget of \$1000000.00.



The training process took six minutes (on average) for 10 “episodes”. Using this basic configuration, we achieve a very positive result in the training PPO, DDPG, and SAC algorithms. With no tuning and a few “episodes” of training over a ten-year time frame on Yahoo Financial data, we get a risk ratio, observed here as Sharpe ratio, within what would be considered “good” in terms of performance.

Available Datasets

Guided by the FinRL paper, we reviewed the available datasets and chose to use Yahoo Finance for its ease of use and availability. We also encountered the possibility of using an API service called Alpaca, which promises to provide financial data for free for educational purposes and solution development, though we haven't thoroughly explored this option yet. Here is the list of dataset options available within the FinRL environment for reference, but we will primarily be working with Yahoo Finance.

Data Source	Type	Max Frequency	Raw Data	Preprocessed Data
Alpaca	US Stocks, ETFs	1 min	OHLCV	Prices, indicators
Baostock	CN Securities	5 min	OHLCV	Prices, indicators
Binance	Cryptocurrency	1 s	OHLCV	Prices, indicators
CCXT	Cryptocurrency	1 min	OHLCV	Prices, indicators
IEXCloud	NMS US securities	1 day	OHLCV	Prices, indicators
JoinQuant	CN Securities	1 min	OHLCV	Prices, indicators
QuantConnect	US Securities	1 s	OHLCV	Prices, indicators
RiceQuant	CN Securities	1 ms	OHLCV	Prices, indicators
Tushare	CN Securities	1 min	OHLCV	Prices, indicators
WRDS	US Securities	1 ms	Intraday Trades	Prices, indicators
YahooFinance	US Securities	1 min	OHLCV	Prices, indicators
AkShare	CN Securities	1 day	OHLCV	Prices, indicators
findatapy	CN Securities	1 day	OHLCV	Prices, indicators
pandas_datareader	US Securities	1 day	OHLCV	Prices, indicators
pandas-finance	US Securities	1 day	OHLCV	Prices, indicators
ystockquote	US Securities	1 day	OHLCV	Prices, indicators
Marketstack	50+ countries	1 day	OHLCV	Prices, indicators
finnhub	US Stocks, currencies, crypto	1 day	OHLCV	Prices, indicators
Financial Modeling prep	US stocks, currencies, crypto	1 min	OHLCV	Prices, indicators
EOD Historical Data	US stocks, and ETFs	1 day	OHLCV	Prices, indicators
Alpha Vantage	Stock, ETF, forex, crypto, technical indicators	1 min	OHLCV	Prices, indicators
Tiingo	Stocks, crypto	1 day	OHLCV	Prices, indicators
Quandl	250+ sources	1 day	OHLCV	Prices, indicators
Polygon	US Securities	1 day	OHLCV	Prices, indicators
fixer	Exchange rate	1 day	Exchange rate	Exchange rate, indicators
Exchangerates	Exchange rate	1 day	Exchange rate	Exchange rate, indicators
Fixer	Exchange rate	1 day	Exchange rate	Exchange rate, indicators
currencylayer	Exchange rate	1 day	Exchange rate	Exchange rate, indicators
currencyapi	Exchange rate	1 day	Exchange rate	Exchange rate, indicators
Open Exchange Rates	Exchange rate	1 day	Exchange rate	Exchange rate, indicators
XE	Exchange rate	1 day	Exchange rate	Exchange rate, indicators
Xignite	Exchange rate	1 day	Exchange rate	Exchange rate, indicators

Liu, X.-Y., et al. (2022), "FinRL-Meta: Market environments and benchmarks for data-driven financial reinforcement learning,"

Testing FinRL

As mentioned, we conducted tests on three different types of agents using a training window of ten years of historical data from Yahoo Finance (Dow Jones top 30 tickers) and the standard parameters from the tutorials provided in the official FinRL documentation. To measure the results, FinRL provides some metrics for comparing the models, as follows:

- Cumulative return: $R = V - V_0 / V_0$

V = final portfolio value

V_0 = original capital

- Annualized return: $r = ((1+R)^{365/t}) - 1$

t = trading days

- Annualized volatility $\sigma_a = \sqrt{\frac{\sum_{i=1}^n (r_i - \bar{r})^2}{n-1}}$,

r_i = annualized return in year i

\bar{r} = avg annualized return

n = number of years

- Sharpe ratio: $S = \frac{r - r_f}{\sigma_a}$ (The Sharpe ratio estimates investment returns versus risk.)

r_f = risk-free rate

- Max. drawdown: The maximal percentage loss in portfolio value.

	Annual return	Cumulative returns	Annual volatility	Sharpe ratio	Calmar ratio	Stability	Max drawdown	Daily value at risk
SAC	0.041	0.125	0.221	0.295	0.111	0.019	-0.372	-0.028
PPO	0.092	0.291	0.233	0.495	0.263	0.243	-0.350	-0.029
DDPG	0.139	0.457	0.217	0.708	0.449	0.739	-0.309	-0.027

With an initial budget of \$1,000,000.00 and a runtime between 5 and 6 minutes, we achieved good results both in terms of increasing the value of the assets and with a risk vs. investment ratio very close to the margin between "good" and "high."

For instance, by running the SAC-type Agent after training in an environment simulating a real exchange with a 2-year timeframe (2019-2020), we achieved an estimated return of approximately \$3,258,353.00.



df_account_value_sac

	date	account_value
0	2019-01-01	1000000.00
1	2019-01-02	999878.01
2	2019-01-03	994063.54
3	2019-01-04	1003457.76
4	2019-01-05	1006023.54
...
726	2020-12-27	1472454.26
727	2020-12-28	1472454.26
728	2020-12-29	1474867.57
729	2020-12-30	1473139.29
730	2020-12-31	1484592.56

```
day: 3650, episode: 10
begin_total_asset: 1000000.00
end_total_asset: 4258353.13
total_reward: 3258353.13
total_cost: 1907.99
total_trades: 57084
```

Training and testing the agents:

Tickers.: Top 30 Dow Jones.

Start.: 2010-01-01.

End.: 2023-12-01 (testing data from 2022 to 2023)

We tested the following agents by varying some hyperparameters and increasing the amount of data compared to the previous version. We used the following agents, utilizing the stable baseline 3 library (initially, we used the elegantRL library but encountered compatibility issues and decided to switch): a2c, ddpg, td3, ppo, sac. The a2c and ddpg agents (which we used in the first prototype) do not have many hyperparameters. In contrast, the td3, ppo, and sac models allow adjustments to the batch size, buffer size, and learning rate.

During training, all of them performed well, with varying execution times for similar configurations. One notable aspect was the SAC model, which initially showed extremely high profits with a Sharpe ratio above 1.0, considered very good. However, as training progressed, these metrics decreased, with the Sharpe ratio seeming to stabilize around 0.8. This could



indicate that the initial results were due to overfitting. We plan to explore this characteristic further and measure it more accurately to understand what is happening.

Here are the training results:

A2C (Advantage Actor-Critic)

Combines value-based and policy-based approaches, using the advantage function to reduce variance.

Execution time : 14.94 minutes

```
begin_total_asset: 1000000.00
end_total_asset: 4069528.36
total_reward: 3069528.36
total_cost: 1346.43
total_trades: 47821
Sharpe: 0.813
```

DDPG (Deep Deterministic Policy Gradient)

Learns policies in high-dimensional, continuous action spaces using off-policy data and the Bellman equation.

Execution time :25.58 minutes

```
begin_total_asset: 1000000.00
end_total_asset: 4430345.86
total_reward: 3430345.86
total_cost: 1297.55
total_trades: 38508
Sharpe: 0.776
```

PPO (Proximal Policy Optimization)

Optimizes policy gradients by maintaining a balance between exploration and exploitation, with simplified updates.

Execution time : 12.35 minutes

Hyper parameters:

```
"n_steps": 2048,
"ent_coef": 0.01,
"learning_rate": 0.00025,
```



```
"batch_size": 128,  
"device":"mps"
```

```
{"n_steps": 2048, 'ent_coef': 0.01, 'learning_rate': 0.00025, 'batch_size': 128, 'device': 'mps'}  
Using mps device  
day: 2956, episode: 90  
begin_total_asset: 1000000.00  
end_total_asset: 4109897.89  
total_reward: 3109897.89  
total_cost: 385210.66  
total_trades: 80245  
Sharpe: 0.801
```

TD3 (Twin Delayed Deep Deterministic policy gradient)

Improves DDPG by using two value functions to reduce overestimation and delayed policy updates.

Execution time : 25.1 minutes

Hyper parameters:

```
"batch_size": 100,  
"buffer_size": 1000000,  
"learning_rate": 0.001,  
"device":"mps"
```

```
{"batch_size": 100, 'buffer_size': 1000000, 'learning_rate': 0.001, 'device': 'mps'}  
Using mps device  
begin_total_asset: 1000000.00  
end_total_asset: 4951350.69  
total_reward: 3951350.69  
total_cost: 1190.37  
total_trades: 32596  
Sharpe: 0.854
```

SAC (Soft Actor-Critic)

Focuses on maximizing a trade-off between expected return and entropy, encouraging exploration.

Execution time : 47.47 minutes

Hyper parameters:

```
"batch_size": 128,  
"buffer_size": 100000,  
"learning_rate": 0.0001,  
"learning_starts": 100,
```

```
"ent_coef": "auto_0.1",
"device": "mps"
```

```
{"batch_size": 128, "buffer_size": 100000, "learning_rate": 0.0001, "learning_starts": 100, "ent_coef": "auto_0.1", "device": "mps"}
begin_total_asset: 1000000.00      begin_total_asset: 1000000.00
end_total_asset: 7251436.69       end_total_asset: 4608151.59
total_reward: 6251436.69         total_reward: 3608151.59
total_cost: 112983.38           total_cost: 3901.65
total_trades: 70517             total_trades: 46641
Sharpe: 1.034                   Sharpe: 0.841
```

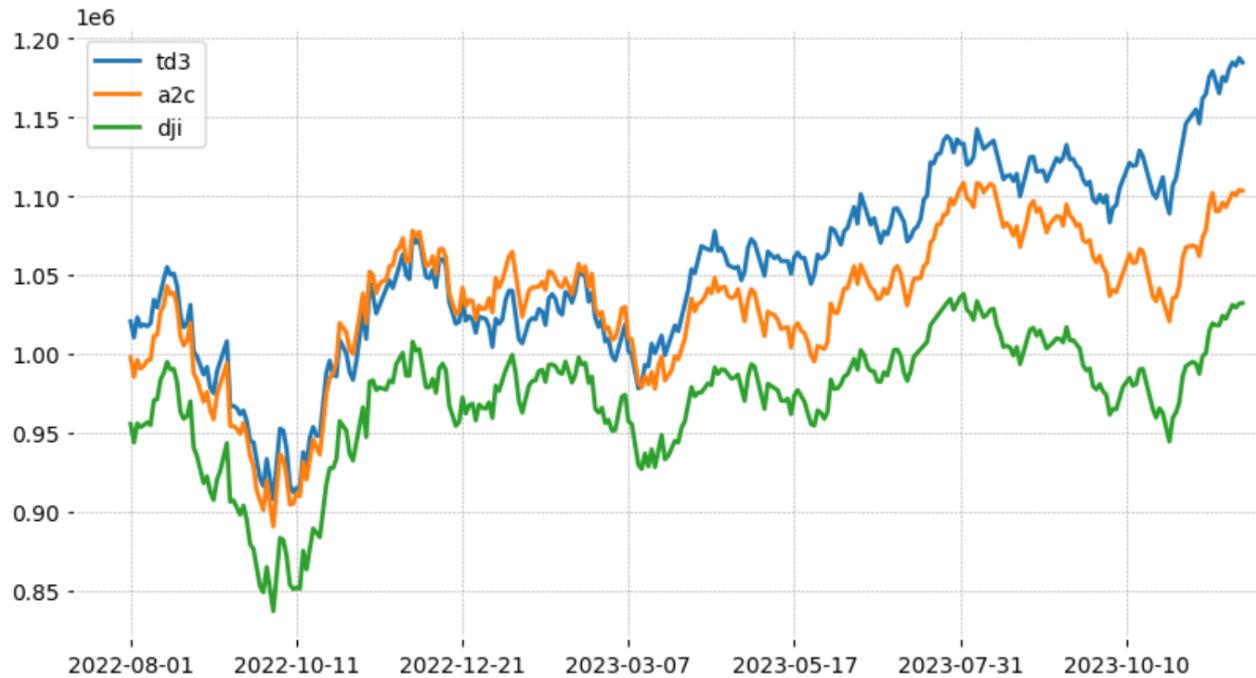
Trading (testing)

During testing, we also tested with data from 2021-10-01 to 2023-12-01. An interesting observation is that all models, including the average and the Dow Jones Industrial Average (DJI), had a negative peak on 2021-11-10, likely due to some atypical event on that date. In the long term, the PPO model surpassed the SAC agent, but it exhibited much instability; during training, PPO had the highest risk index of all.



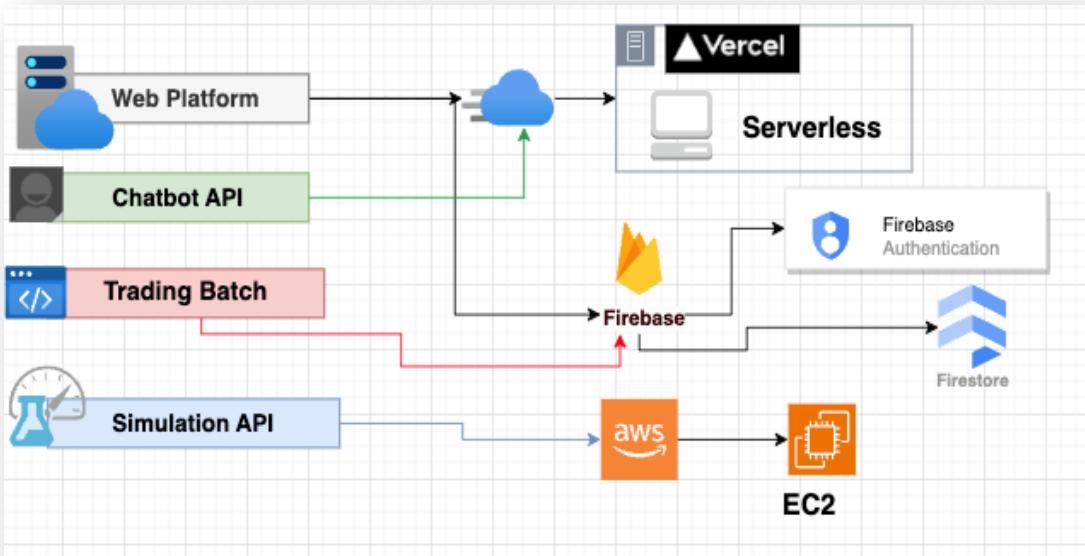
The SAC agent showed the least variation, but its financial gains were not as good as those of A2C and TD3.

The A2C model performed very well, but in the long term, the td3 model showed the best result, with moderate variation and achieving profit most of the time.

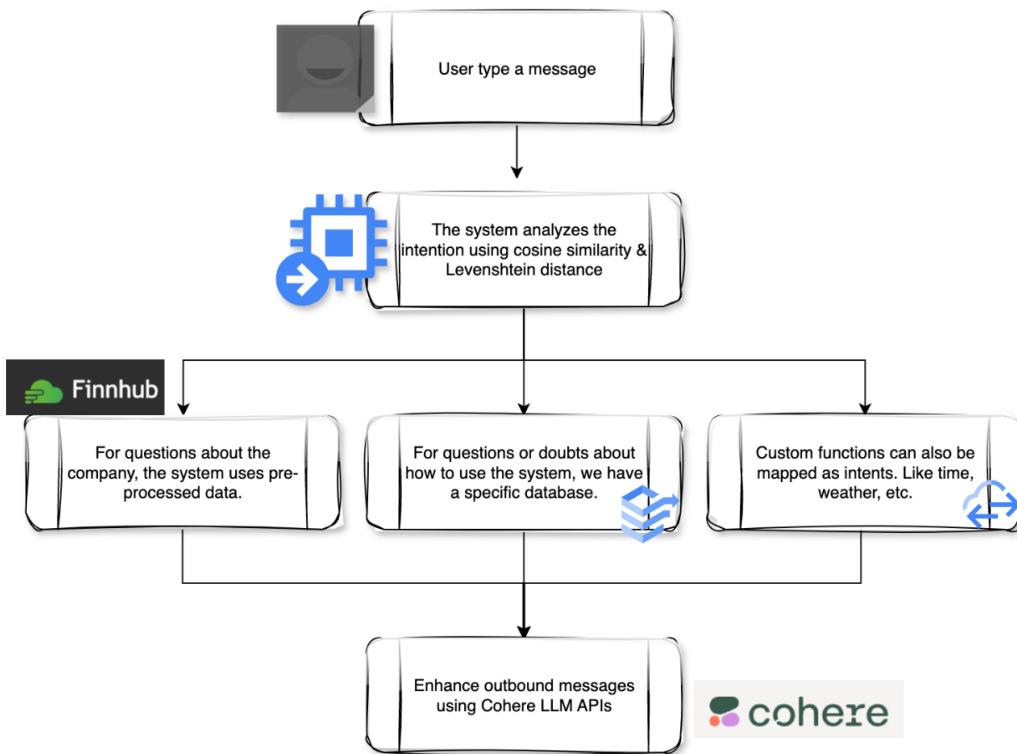


Architecture

To make all our solutions working together without costs, our application was divided into smaller services, each running in a different environment, but all are interconnected at some point. We are using different cloud solutions in integration with local batch process.

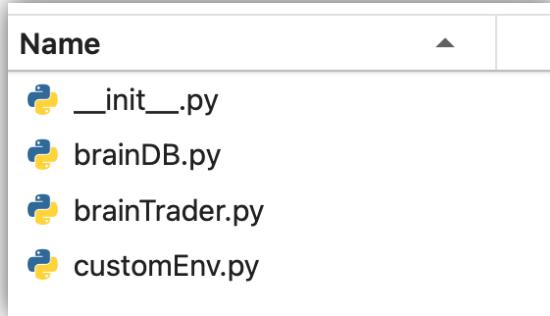


As for the AI Knowledge Hub, part of the Chatbot API, is using RAG architectures by using Coherer's API for the LLM and Finnhub to generate financial information in the batch process.



BrainLib

We created this library to facilitate development, the complexity of the code, and the number of libraries we use because of FinRL (for example). We encapsulated the main functions by organizing our library as follows:



BrainDB.py -> Contains simplified functions for accessing data in local and cloud databases already with connection settings. As in the example above, to check the data of the bots in the cloud, it is only necessary to create an instance of the brainDB.FireStoreDB class and then call the list_bots() function. It's that simple.

CustomEnv.py -> This is the class containing the most recent version of our customized FinRL-meta environment.

BrainTrader.py -> This is the class that controls the integration between data, agent, and environment. It encapsulates complex tasks and libraries necessary for the bot's execution. In the code snippet below, for example, we are using the brainTrader.GenericTrader class. The GenericTrader class was made to work with customized agents and data. This generalization makes it easier to use different agents and different data files.

```
1 def run_trading(a_path,a_type,d_path,t_limit,i_amnt,s_date,e_date,en
2     from brainLib.brainTrader import GenericTrader
3     import matplotlib.pyplot as plt
4     import pandas as pd
5
6     trader = GenericTrader()
7
8     simulation_args = {"agent_path": a_path,
9                         "agent_type": a_type,
10                        "data_path": d_path,
11                        "trade_limit": int(t_limit),
12                        "initial_amount": str(i_amnt),
13                        "start_date": s_date,
14                        "env": "",
15                        "end_date": e_date,
16                        "symbol": symbol,
17                        "user": user,
18                        "resume_session": resume_session
19
20
21
22     _, _,_ = trader.start_simulation(**simulation_args)
23 }
```

With the GenericTrader class, we just pass the data and call the “start_simulation()” function.

Batch Process

We are using the Apscheduler library to execute the “run_session” function at a specific frequency. So, every x minutes, the batch process will run the trading based on the information registered by the interface and the data generated in the previous session.

```
1 from apscheduler.schedulers.blocking import BlockingScheduler
2
3 scheduler = BlockingScheduler()
4
5 scheduler.add_job(run_session, 'interval', minutes=1)
6
7
```

```

1 def run_session():
2     symbol = ""
3     a_path = ""
4
5     ## Get list of bots
6     bots = data_base.list_active_bots(today)
7
8     ## For each bot
9     for bot in bots.index:
10
11         ### Format some parameters
12         if bots["symbol"][bot] == "DOW30":
13
14             symbol = ""
15             a_path = "agents/dow30_high.mdl"
16             a_type = "ddpg"
17
18         else:
19
20             symbol = (bots["symbol"][bot]).lower()
21             a_path = f"agents/{symbol}_td3_high.mdl"
22             a_type = "td3"
23             print("a_path : ", a_path)
24
25
26     ## Run session
27     run_trading(a_path,
28                 a_type,
29                 "data/trading_set.csv",
30                 100,
31                 str(bots["investment"][bot]),
32                 last_session,
33                 today,
34                 "",
35                 symbol,
36                 bots["userId"][bot],
37                 False)
38
39     ## Update portifolio
40     memory = pd.read_csv("results/state_memory.csv")
41     memory = memory.query("date=='"+ today +"'")
42
43     ## Get New balance
44     balance = memory["money"].iloc[-1]
45     print("balance")
46
47     ## Update bot balance
48     data_base.update_bot(bots["botId"][bot], investment=balance)
49
50     ## Update bot Balance in the cloud
51     from brainLib.brainDB import FirestoreDB
52     cloudBase = FirestoreDB()
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
755
756
757
757
758
759
759
760
761
762
763
764
765
765
766
767
767
768
769
769
770
771
772
773
774
775
775
776
777
777
778
779
779
780
781
782
783
784
784
785
786
786
787
787
788
788
789
789
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
16
```

ticker	money	price	shares	total	fees
CRM	11.120357681276076	164.75	0	0.0	9.98
CSCO	11.120357681276076	46.79556655883789	2.0	93.59113311767578	9.98
CVX	11.120357681276076	165.80503845214844	0.0	0.0	9.98
DIS	11.120357681276076	107.24048614501952	0.0	0.0	9.98
GS	11.120357681276076	346.4787902832031	0.0	0.0	9.98
HD	11.120357681276076	305.732421875	0.0	0.0	9.98
HON	11.120357681276076	201.33436584472656	0	0.0	9.98
IBM	11.120357681276076	127.73370361328124	0	0.0	9.98
INTC	11.120357681276076	27.23270034790039	0.0	0.0	9.98
JNJ	11.120357681276076	156.04408264160156	61.0	9518.689041137695	9.98
IDM	11.120357681276076	125.26661720288672	0.0	0.0	9.98

Outcome of the batch execution for the example we registered in the user interface

(demonstrated at the beginning of this document):

The screenshot shows a Jupyter Notebook environment. On the left, there's a file explorer window titled '01 - final / results /' containing several CSV files: 'last_state.csv', 'state_memory.csv' (which is selected), 'save_action_memory.csv', 'asset_memory.csv', 'account_rewards.csv', 'account_value.csv', and 'actions.csv'. The 'state_memory.csv' file has a timestamp of 'seconds ago'. On the right, there are two data preview windows. The top one is for 'state_memory.csv' with columns: date, ticker, money, price, shares, total, fees. It shows two rows: Row 1 (date 2023-01-27, ticker AAPL, money 10000.0, price 144.93391..., shares 0, total 0.0, fees 0.00) and Row 2 (date 2023-01-30, ticker AAPL, money 134.63832..., price 142.02392..., shares 68.0, total 9657.62..., fees 9.86). The bottom window is for 'trading_set.csv'.

	date	ticker	money	price	shares	total	fees
1	2023-01-27	AAPL	10000.0	144.93391...	0	0.0	0.00
2	2023-01-30	AAPL	134.63832...	142.02392...	68.0	9657.62...	9.86

The batch process outcome displays the state of the previous session:

AAPL, \$10,000 available for the bot to trade, the ticker's price on that day, and the number of shares the user held on that day, followed by the next row representing the trading day. In the second row, one can observe that the money now is the remainder from the day's transactions. We also have the share amount, the total asset value (number of shares * ticker price), and the total fees paid in the transactions for that session.



Findings

Hidden Overfitting

It's very easy to fall into overfitting, although it's hard to identify clearly.

As noted in previous sections, when we tested the models, the results were quite impressive. We were able to increase our portfolio value by more than fourfold using models that were trained for just a few minutes over a 10-year period of historical data.

```
day: 2956, episode: 90
begin_total_asset: 1000000.00
end_total_asset: 4109897.89
total_reward: 3109897.89
total_cost: 385210.66
total_trades: 80245
Sharpe: 0.801
```

When we apply these types of models to more recent test data, they end up performing well, though not as promisingly as in the training phase. However, it's noticeable that they become "biased" towards certain symbols, keeping them in the portfolio even when their market value is declining. It's as if they learn something along the lines of "holding Apple stocks is always good; they will always perform well."

In summary, great care must be taken when tuning these models, as it's possible to fall into the "illusion" that they can perform very well when, in fact, they are being "lazy" or "biased" towards a point of just buying more and more of a stock, even when it's no longer performing well in the present. The solution we found to balance this was to execute several rounds of tuning, carefully observing the behavior and distribution of the portfolio chosen by the model and adjusting the hyperparameters and training iterations.



FinRL Installation and alternatives for CUDA

We encountered numerous issues with the installation and operation of FinRL-Meta. Apparently, in the last three months, there has been a consolidation of all repositories by the AI4Finance team, resulting in broken or moved links to examples and sources, parts of the source codes being outdated or having conflicting versions of the same classes, bugs, among other issues. Consequently, many of the code examples were also broken, outdated, or incomplete, and it wasn't even possible to run them on Google Collab without making adaptations.

Additionally, the official version of the library is completely outdated. If you try to install it via pip, you will need to downgrade your Python version and many of the main libraries.

We found an installation alternative in the documentation, which was to install directly from the GitHub repository using the pip package manager, which was a pleasant surprise. Thus, by using the command `pip install git+<repository path>`, it's possible to install a library directly from its most recent version on GitHub, without needing any authentication or additional steps. With this information, we managed to install it in the Google Collab environment for testing, and even though we used GPU and TPU, we understood that the training process took a long time to complete.

Among the equipment available to our team were two gaming laptops and two Apple laptops with Apple Silicon (ARM64) architecture. However, it was an impossible task to install the library on Windows due to dependencies on development tools like C++ compilers among other requirements that seemed "strange" to us. Despite much effort, it was not possible, making



running the system using CUDA unfeasible since the machines with Nvidia GPUs ran on Windows and even virtual machines couldn't install the entire environment.

Parallelly, based on past experiences in college, we identified that the libraries we were trying to use ran PyTorch behind the scenes, which allows the use of MPS or Metal Performance Shaders, which is Apple's equivalent to CUDA.

Thus, we cloned the FinRL repository and added a command so that the library could now be compatible with the use of MPS, allowing us to run the model training using the GPU of Apple laptops. Then, we used the direct installation feature from GitHub to install the modified library.

The MPS-compatible version of FinRL is available at <https://github.com/FabioD-Junior/FinRL> and involves a simple line of instruction that was replaced in the paper trading module.

```
207
208         self.states = None # assert self.states == (1, state_dim)
209         self.device = torch.device(
210     +         f"cuda:{gpu_id}" if (torch.cuda.is_available() and (gpu_id >= 0)) else "cpu"
211     +         ##@TO DO : Test MPS version
212     +         f"cuda:{gpu_id}" if (torch.cuda.is_available() and (gpu_id >= 0)) else ("mps" if
           torch.backends.mps.is_available() else "cpu")
```

In summary, running a simple basic example of the library for the first time was very challenging, but once we had a complete understanding of what we were doing, we were able to overcome the problem. This made us look at an improvement point that is ignored nowadays and is also not widely disclosed, which is the use of MPS for AI purposes. Most people developing AI solutions with Python are not even aware of this option to replace NVIDIA's CUDA, and due to its ARM64 architecture nature, it consumes much less energy than an approach using regular GPUs. However, we not only applied the use of MPS in our project but also enabled the FinRL library so that other people from the open community can also be aware of and enjoy this alternative.



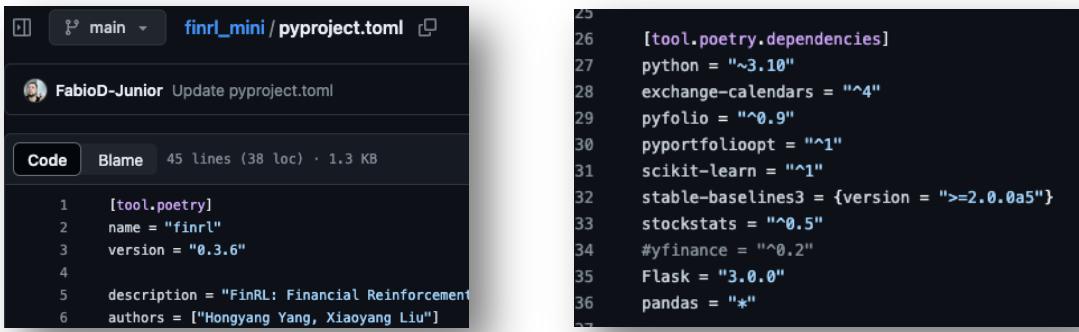
Creation of our own version of FinRL : FinRL-Mini

The processes of training and executing DRL agents, especially when there's a large amount of data, require a lot of resources, burdening any basic system. It can present bottlenecks in processing, demanding a long time to finalize processes that seem simple to the user, as well as being prone to memory overflow, disk, or in the case of using cloud technologies, a very high charge if the automatic provisioning of resources is not being well managed and monitored. One point that we are proud of in this project was the ability to use our creativity to circumvent some of these problems and still achieve an excellent result, for example, executing the simulation in a cloud environment (AWS) within the free tier.

After a great effort analyzing the source codes of FinRL, which is available as open source on GitHub, we had the idea to perform "reverse engineering" to understand how this library works and to create a new version containing only what is necessary to specifically execute our purpose. As explained in the previous section, the FinRL library tries to bring the most varied and useful tools for the development of DRL solutions for finance, making it generalist and more specific for training the model and exploration than for the process of executing the created solution. Thus, we cloned the entire library repository and downloaded it locally, removing the components and classes that were used only for training or that were not related to our solution, such as the environment module used to connect with Chinese exchanges, the environment and dependencies used to access the Alpaca environment, methods of downloading data that were not from Yahoo Finance among others. We discovered in this way that this library does not use the common dependency mapping process that we learned in college and are used to, such as

the use of requirements.txt, instead, it uses a pair of files to map what must be installed for the library to function. These files are:

- **Pyproject.toml** – Contains the description and metadata of the library, such as the author's name, license, etc., the version of Python compatible with the library, and the main libraries that must be installed first.
- **Poetry.lock** – Responsible for the dependencies necessary to install the main libraries related in the **pyproject.toml**, as well as any other extra dependency (such as examples, templates, etc.).



The screenshot shows a GitHub repository page for 'finrl_mini'. The 'Code' tab is selected, displaying the 'pyproject.toml' file. The code content is as follows:

```
25
26 [tool.poetry.dependencies]
27 python = "~3.10"
28 exchange-calendars = "^4"
29 pyfolio = "^0.9"
30 pyportfolioopt = "^1"
31 scikit-learn = "^1"
32 stable-baselines3 = {version = ">=2.0.0a5"}
33 stockstats = "^0.5"
34 #yfinance = "^0.2"
35 Flask = "3.0.0"
36 pandas = "*"
```

We also made adaptations in these files with the intent of installing only the relevant dependencies and excluding extra dependencies that are never used, for example, one of the libraries has as an extra dependency a gaming environment for Atari, which probably should be used as an example for studying and training agents capable of playing video games.

Our library, FinRL-Mini, derived from the FinRL library is available on our GitHub. We maintained all the original credits and information of FINRL, adding the information that this is a lighter version adapted to run using fewer resources and only for execution, not training models.

It can be installed using the command:



```
pip install git+https://github.com/FabioD-Junior/finrl_mini/
```

Once again, through our creativity and problem-solving ability, we managed to contribute to the open-source community while solving a complex problem.

The screenshot shows a GitHub repository page for 'finrl_mini' owned by 'FabioD-Junior'. The repository is public. At the top, there are navigation links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the header, there's a search bar and a pin/unwatch button. The main content area displays a list of 16 commits from 'FabioD-Junior' made 3 weeks ago. The commits are:

File / Action	Description	Time
finrl	Delete finrl/applications/stock_trading/fundamental_stoc...	last month
LICENSE	Add files via upload	last month
README.md	Add files via upload	last month
poetry.lock	Update poetry.lock	last month
pyproject.toml	Update pyproject.toml	3 weeks ago
requirements.txt	Add files via upload	last month
setup.cfg	Add files via upload	last month

Ngrok https tunnel

During the deployment of the application, we encountered issues during testing. Unlike running locally, when we run in production, the web interface defaults to HTTPS due to Vercel's cloud configuration. However, the AWS standard, where the backend runs, is the HTTP protocol. To change this setting, it is the developer's responsibility to manually configure and provide the

SSL certificate for use in authentication. This situation, where one solution runs on HTTPS and another on HTTP, creates a security error in the browser, preventing the platform from operating properly.

Since we do not have financial resources for this project and are only using free solutions, we did not have the means to purchase a domain and register a digital certificate, nor was our attempt to use a self-signed certificate productive.

After a lot of research, we discovered Ngrok, which is an HTTPS tunnel solution. It converts the entire server flow to a fixed free domain with HTTPS output in real time.

```
ngrok
      (CTRL+C to quit)

Full request capture now available in your browser: https://ngrok.com/r/ti

Session Status          online
Account                  junior.hehe@gmail.com (Plan: Free)
Version                 3.8.0
Region                  United States (us)
Latency                 46ms
Web Interface           http://127.0.0.1:4040
Forwarding              https://piglet-moved-yearly.ngrok-free.app -> http://127.0.0.1:4040

Connections             ttl     opn      rt1     rt5     p50     p90
                        43      0       0.00    0.00    0.05    6.26

HTTP Requests
-----
POST   /bot/simulate      200 OK
OPTIONS /bot/simulate    200 OK
POST   /chatbot/ask        200 OK
OPTIONS /chatbot/ask      200 OK
POST   /chatbot/ask        200 OK
OPTIONS /chatbot/ask      200 OK
```

This solution also allows us to observe all the calls made to the endpoint through an intuitive web interface.



All Requests

[Clear Requests](#)

about 23 hours ago Duration 94.36ms

IP 204.101.131.2

POST /bot/simulate	200 OK	94.36ms
OPTIONS /bot/simulate	200 OK	3.77ms
POST /chatbot/ask	200 OK	5.87s
OPTIONS /chatbot/ask	200 OK	1.9ms
POST /chatbot/ask	200 OK	4.95s
OPTIONS /chatbot/ask	200 OK	6.99ms
POST /bot/simulate	200 OK	89.85ms
OPTIONS /bot/simulate	200 OK	4.08ms
POST /bot/simulate	200 OK	98.1ms
OPTIONS /bot/simulate	200 OK	4.29ms
POST /chatbot/ask	500 INTERNAL SERVER ERROR	1.48s
OPTIONS /chatbot/ask	200 OK	4.85ms
POST /chatbot/ask	200 OK	4.07s
OPTIONS /chatbot/ask	200 OK	4.42ms
POST /bot/simulate	500 INTERNAL SERVER ERROR	15.32ms
OPTIONS /bot/simulate	200 OK	3ms
POST /bot/simulate	500 INTERNAL SERVER ERROR	21.74ms
OPTIONS /bot/simulate	200 OK	6.92ms
POST /bot/simulate	500 INTERNAL SERVER ERROR	39.44ms
OPTIONS /bot/simulate	200 OK	5.45ms

POST /bot/simulate

[Summary](#) [Headers](#) [Raw](#) [Binary](#)[Replay](#)

257 bytes application/json

```
{  
    "data_path": "data/trading_set.csv",  
    "trade_limit": 100,  
    "buy_upper_limit": 200,  
    "sell_upper_limit": 200,  
    "initial_amount": 50000,  
    "symbol": "AAPL",  
    "start_date": "2023-12-28",  
    "end_date": "2024-02-28",  
    "user": "Generic User",  
    "resume_session": false,  
    "orientation": "records"  
}
```

200 OK

[Summary](#) [Headers](#) [Raw](#) [Binary](#)

3178 bytes application/json

```
{  
    "asset_history": "[{\\"date\\": \"2023-12-29\", \\"ticker\\": \"AAPL\", \\"money\\": 30647.3368972778, \\"price\\": 192.2846374512, \\"shares\\": 100, \\"total\\": 19228.4637451172, \\"fees\\": 19.33, \\"sharpe\\": null}, {\\"date\\": \"2024-01-02\", \\"ticker\\": \"AAPL\", \\"money\\": 11399.6446884155, \\"price\\": 185.4034118652, \\"shares\\": 200, \\"total\\": 137080.6823730469, \\"fees\\": 38.56, \\"sharpe\\": -13.4122798103}, {\\"date\\": \"2024-01-03\", \\"ticker\\": \"AAPL\", \\"money\\": 78.7269565125, \\"price\\": 184.0151977539, \\"shares\\": 261, \\"total\\": 48027.9666137695, \\"fees\\": 49.87, \\"sharpe\\": -14.9973672844}, {\\"date\\": \"2024-01-04\", \\"ticker\\": \"AAPL\", \\"money\\": 78.7269565125, \\"price\\": 181.6781768799, \\"shares\\": 261, \\"total\\": 47418.0041656494, \\"fees\\": 49.87, \\"sharpe\\": -18.3520602043}, {\\"date\\": \"2024-01-05\", \\"ticker\\": \"AAPL\", \\"money\\": 78.7269565125, \\"price\\": 180.9490966}
```

Therefore, we were able to proceed with our deployment using this solution while keeping our entire process running for free.

Discussion

Our final product

As already mentioned in the section on the architecture of our solution, guided by our mission to democratize access to trading using modern AI technology while being able to run the system on



a structure that had no cost due to the restrictions of this AIP course, we divided the project into modules and created a user-friendly visual interface, well-organized, and easy to use, consolidating the features into a single platform.

We believe that with this we have achieved our objective of bringing modernization and ease to trading, in addition to providing relevant information about the stock market to the person operating the system.

Disclaimers and important point about the use of AI.

It's always important to clarify that we cannot guarantee profit under any circumstances. What our system does is try to optimize your chances of maintaining as high a portfolio value as possible using AI decision-making techniques. Again, even simulation is not a guarantee of profit or that the system will behave exactly as in the simulation of historical data.

Regarding chatbot functionalities, despite our efforts to keep information up to date and as accurate as possible, the nature of AI chatbots' operation means they have a margin for erroneous responses technically called "hallucination." If you ask a question outside the scope of the symbols and companies we operate with on the platform, you may get results that do not represent the truth. Once again, we emphasize that the chatbot is subject to "hallucinations," and may provide inaccurate or fictional information as if it were real.



Simulation API running on AWS.

Motivation

Our first idea was running all features on AWS, but it is not possible due to the constraints, in this case DRL takes a lot of memory (mainly) in order to run because of the “state space”, transformed into a numeric tensor(vector/matrix, depending on which naming stand you prefer) grows as the number of symbols and tickers, and the more complex the bigger the size of the trained model. The CPU would not be a problem since running the already trained model is not heavy in computational terms, different from training which we had to use GPUs for the execution, the main problem is the memory. We could do that, but it will be costly, to keep the solution running in the free tier of amazon we had to be creative. In the previous sections is cited that our solution is divided into different features, this approach of using multiple modules running in different clouds and environments emerged from this challenge to come up with a solution given the situation of scarcity of computational resources.

This section will present and describe how we used our creativity to create the simulation API, branded as BrainAPI, which is also a product feature offered by our solution where the users within top subscription can have access to implement our solution in their own code or automation.

Since we could not install and deploy FinRL lib on Vercel, due computational resource restrictions, we had to use an alternative where we could have more control over the environment to make the necessary adaptations for everything to work normally. Thus, we used a virtual machine on EC2's free tier running Ubuntu server.



We had to adapt the solution to run with 1GB of memory, so we restricted the number of tickers per simulation to just one and created a light version of the FinRL library.

The result was more than satisfactory, our simulation API can run with all security components, middleware, and API gateway using only 1GB of memory, 1 vCPU, and still achieve a response time of less than 1 second (as you can see in the screenshot below).

BrainAPI Architecture

We designed the API architecture to be lightweight yet robust. One of our primary concerns was handling concurrent calls. Typically, in AI course activities, the solutions were not designed to be accessed by more than one user simultaneously, nor was there a focus on the security and performance of the application.

We decided to create the API using the RESTful pattern. However, despite applying the main concepts, we had to be flexible in some aspects of it to adapt to our use case. Therefore, we cannot strictly call our API RESTful but rather REST-like, meaning it applies the principles of REST, but some details were not implemented. For example, we had to use only the POST method for some applications that should have used the GET method according to the RESTful standard, due to server limitations or to improve performance in some cases.

That said, our API operates in a stateless manner, meaning there is no specific session, and we do not maintain user data or caches. This is particularly advantageous for scenarios where microservices are implemented.

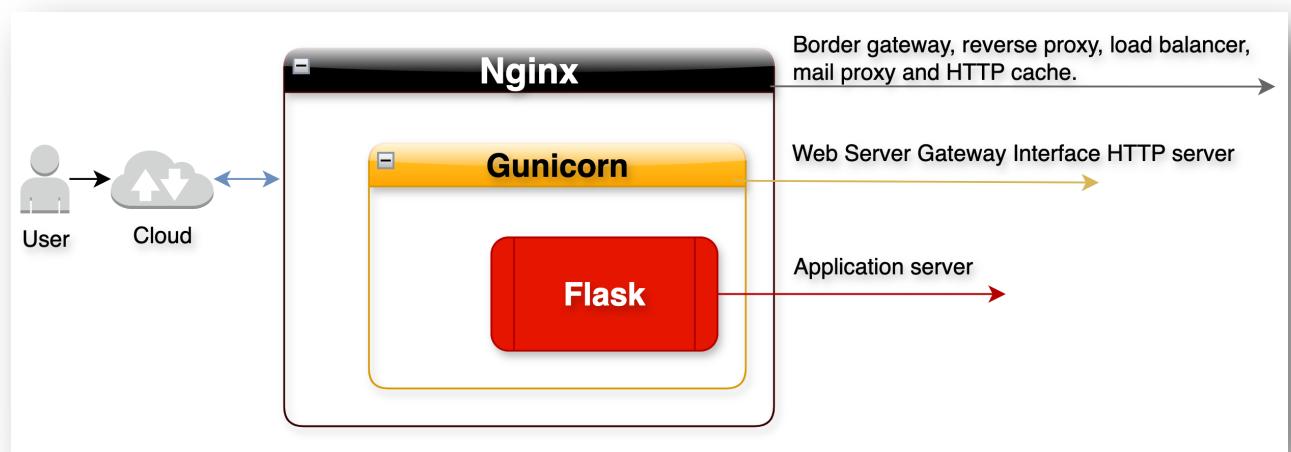
We implemented a bearer authentication, meaning a user with a valid token can access the application. The token is generated through the login endpoint, where the user enters their credentials and once validated, receives a token with temporary validity.

As mentioned, we needed a robust, multi-user, and secure solution, which could not be achieved using only Flask as the server, as it is intended for testing and development purposes, not production. Thus, we encapsulated Flask within two additional layers: a WSGI to manage multi-user calls and a reverse proxy gateway for access management, call handling, load balancing, and traffic management.

Therefore, we chose the following technologies to complement the API's architecture:

- **Nginx** - For the edge gateway. It serves various functions like load balancing and reverse proxy.
- **Gunicorn** - A **WSGI** HTTP Server capable of running multiple Python processes.

This is the diagram of how these technologies interact within our design.



Swagger documentation

As we would offer this API as a service, our team developed a swagger documentation to help our developer customers.

The screenshot shows the Swagger Editor interface. On the left, the API definition is displayed in JSON format:

```

1 openapi: 3.0.0
2 info:
3   title: Trading Simulation & Chatbot API
4   description: API for authentication, trading simulation, and chatbot interactions.
5   version: "1.0.0"
6 servers:
7   - url: http://localhost:5000/
8 paths:
9   /auth/login:
10    post:
11      summary: Login Endpoint
12      description: Authenticates a user and returns a JWT token.
13      requestBody:
14        content:
15          application/json:
16            schema:
17              type: object
18              properties:
19                username:
20                  type: string
21                secret:
22                  type: string
23                  required:
24                    - username
25                    - secret
26      responses:
27        '200':
28          description: Successful login.
29          content:
30            application/json:

```

On the right, the API documentation is shown under the heading "Trading Simulation & Chatbot API 1.0.0 OAS 3.0". It includes a "Servers" dropdown set to "http://localhost:5000/". Below it, the "default" section lists three endpoints:

- POST /auth/login** Login Endpoint
- GET /bot/simulate** Trading Simulation
- GET /chatbot/ask** Chatbot Interaction

This screenshot provides a detailed view of the "POST /auth/login" endpoint under the "default" section. It includes:

- Parameters:** No parameters.
- Request body:** Set to "application/json".
- Example Value | Schema:**

```
[
  {
    "username": "string",
    "secret": "string"
  }
]
```
- Responses:**

Code	Description	Links
200	Successful login.	No links
400	Invalid credentials.	No links



The screenshot shows the API documentation for the `/bot/simulate` endpoint. The endpoint is described as "Trading Simulation".

Parameters:

Name	Description
<code>agent_path</code> * required	agent_path
<code>agent_type</code> * required	agent_type
<code>data_path</code> * required	data_path
<code>trade_limit</code>	trade_limit
<code>buy_upper_limit</code>	buy_upper_limit
<code>sell_upper_limit</code>	sell_upper_limit
<code>initial_amount</code>	initial_amount
<code>start_date</code>	start_date
<code>env</code>	env
<code>end_date</code>	end_date

Responses:

Code	Description
200	Successful simulation execution. Media type: application/json Controls Accept header.
401	Invalid or missing token.

Example Value | Schema:

```
{  
  "asset_history": "string",  
  "current_balance": 0,  
  "asset_value": 0,  
  "sharpe_ratio": 0  
}
```

Walking through the Features

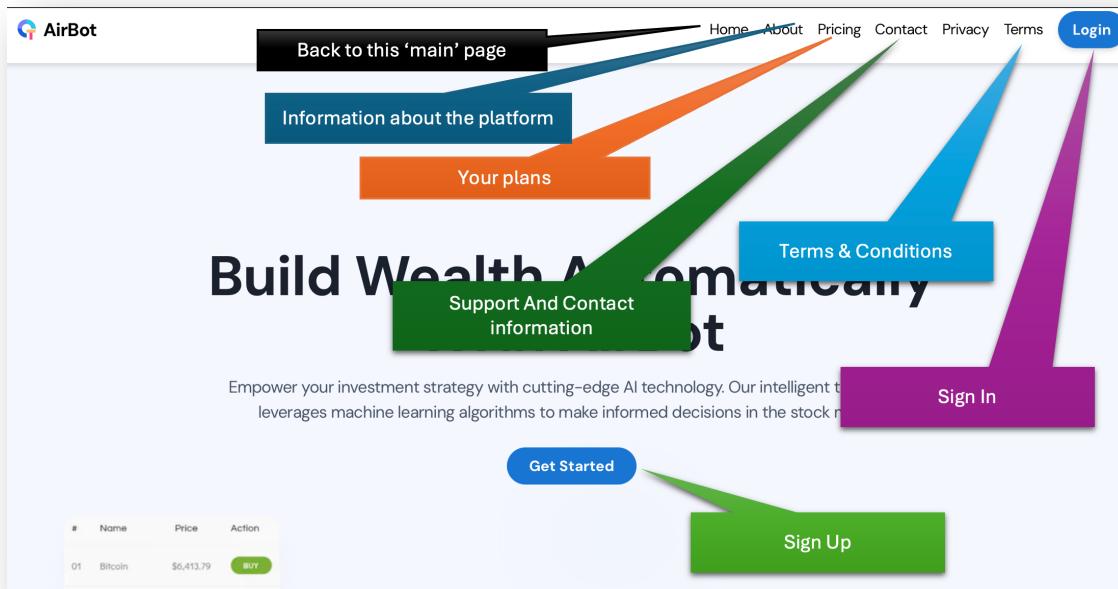
Main Page

Firstly, you can access the application using the following URL:

<https://brain-trading-bot.vercel.app/> this is the official link to your bot platform.

For more information about our team's features and products, you can visit the official Brain's team page at: <https://jaldhihimanshubha.wixsite.com/brain>.

Upon accessing the platform, you will find the following page:



Home: Returns to the platform's main page.

About: Directs to the About page.

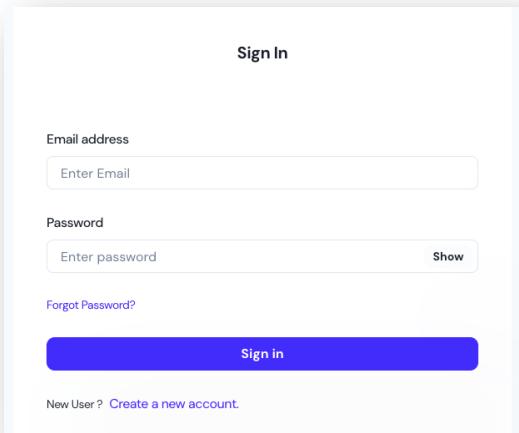
Pricing: Displays the page with our subscription plans.

Contact: A form to get in touch and ask questions.

Privacy & Terms: Terms and conditions of use.

You will also find two buttons: "Login" at the top right corner and "Get Started" at the center of the page.

As a user who already has an account, simply click the first button, and enter your credentials.

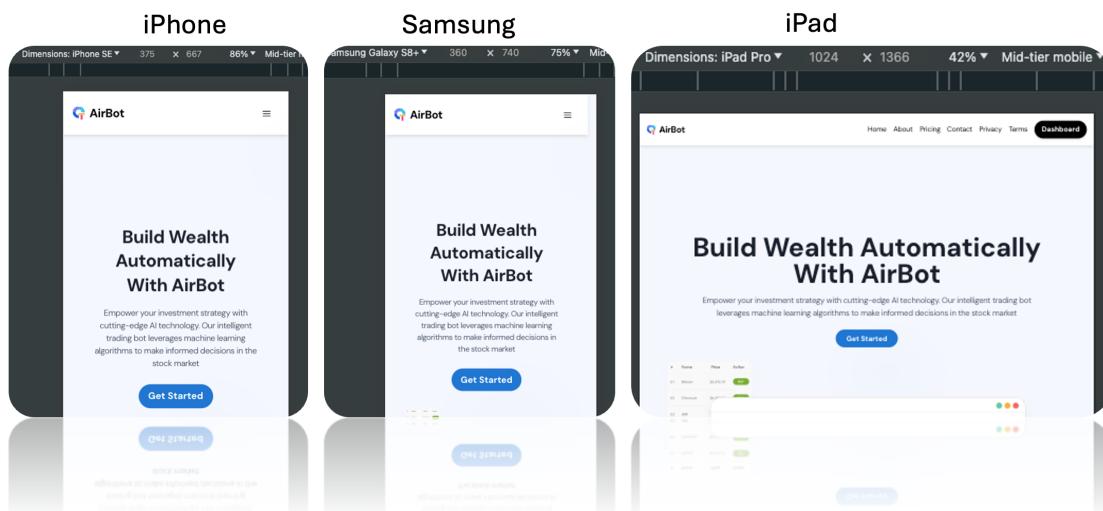


If you are a new user, you can click on "Get Started" and complete your registration in a few simple steps.

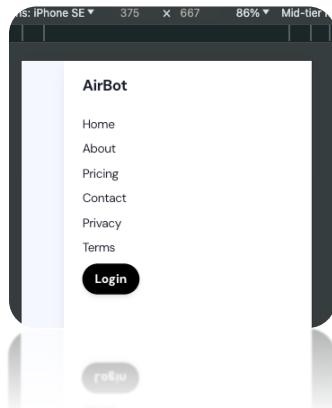
UI Responsiveness

Our platform is designed with user convenience and simplicity in mind. With these ideals, we've created pages that adapt to the size of your device's screen, whether it's a smartphone, tablet, PC, or any other device with a compatible browser.

Below, you can see a simulation of how the page resizes behaves across different devices.



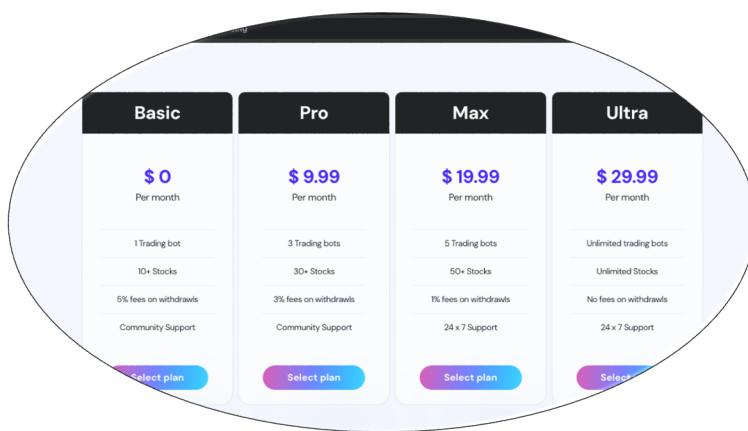
The navigation bar, depending on the screen size and your browser settings (to display desktop or mobile versions), is replaced by a "hamburger menu." When clicked, this menu displays the same options as the navigation bar, maintaining a layout that is more visible for your device's screen size.



Subscription and Payment methods

After entering your data, click on "Sign Up" to choose your plan and add funds to your account.

There are four aspects that differentiate the plans:



Basic	Pro	Max	Ultra
\$ 0 Per month	\$ 9.99 Per month	\$ 19.99 Per month	\$ 29.99 Per month
1 Trading bot	3 Trading bots	5 Trading bots	Unlimited trading bots
10+ Stocks	30+ Stocks	50+ Stocks	Unlimited Stocks
5% fees on withdrawals	3% fees on withdrawals	1% fees on withdrawals	No fees on withdrawals
Community Support	Community Support	24 x 7 Support	24 x 7 Support
Select plan	Select plan	Select plan	Select



1. The **number of bots** you can register (single trading or multi-trading).
2. The **number of stocks available for trading**; in other words, the list of stocks depends on the plan you select.
3. **Fees**. A transaction fee is charged for each withdrawal, the value of which depends on the selected plan.
4. **Support**. Starting from the MAX plan, you can have 24x7 support from our team for any questions about the platform, issues, etc.

The value of the chosen plan will be charged monthly.

Logged Session

Once you complete your registration or log in, you will be redirected to the logged-in area, specifically to the Dashboard page. Here are the sessions available for registered users:

The screenshot shows the AirBot Dashboard interface. At the top, there's a navigation bar with links for Dashboard, Stock Marketplace, Manage Bots, AI Knowledge Hub, and Profile. Below the navigation, there are several key statistics: Earnings (\$350.4), Total Invested (\$642.39), Bots Running (154), and Total Projects (2935). A 'News' section displays two recent articles. One article is titled 'YouTube Is Slow And Laggy In The Browser For Everyone: Here's Why And How You Can Fix It' and the other is 'Passion To Profit: The Journey Of Tezza's Rise In The Photo-Editing World'. At the bottom of the dashboard, there's a 'Feedback' section with a link to 'https://www.braintrading.com/feedback'.

Account info + Recent news about the stock market.

Live information about exchanges and stock prices

Area do create and customize our bots

Chatbot assistant to help with your questions about trading

Your plans & system preferences

Feedback

Dashboard

As mentioned earlier, here you can find both a summary of your account information, such as the active number of bots, earnings, investments, etc., as well as navigate through the latest financial market news.

The screenshot shows the AirBot dashboard interface. At the top, there are four summary cards: Earnings (\$350.4), Total Invested (\$642.39), Bots Running (154), and Total Projects (2935). Below these, a section titled "News" displays two recent articles:

- YouTube Is Slow And Laggy In The Browser For Everyone: Here's Why And How You Can Fix It**
After several user reports that YouTube is being throttled, the Google subsidiary has confirmed this. Here's how you can fix slow YouTube.
<https://www.benzinga.com/news/23/11/35891315/youtube-is-slow-and-laggy-in-the-browser-for-everyone-heres-why-and-how-you-can-fix-it>
Author: Rounak Jain
03/31/2024 16:20 PM
- Passion To Profit: The Journey Of Tezza's Rise In The Photo-Editing World**
In 2017, amidst the challenges of affording their \$2,800-per-month rent in New York, Tessa Barton and Cole Herrmann turned their creative and technical talents towards developing photo filters for Adobe Lightroom. This endeavor laid the groundwork for Tezza, a photo-editing app named after Barton's college nickname.
<https://www.benzinga.com/news/24/03/37987272/passion-to-profit-the-journey-of-tezzas-rise-in-the-photo-editing-world>
Author: Bibhu Pattnak
03/31/2024 15:43 PM

Stock Marketplace

Here you can browse through a list of stocks from various exchanges and check their price status.

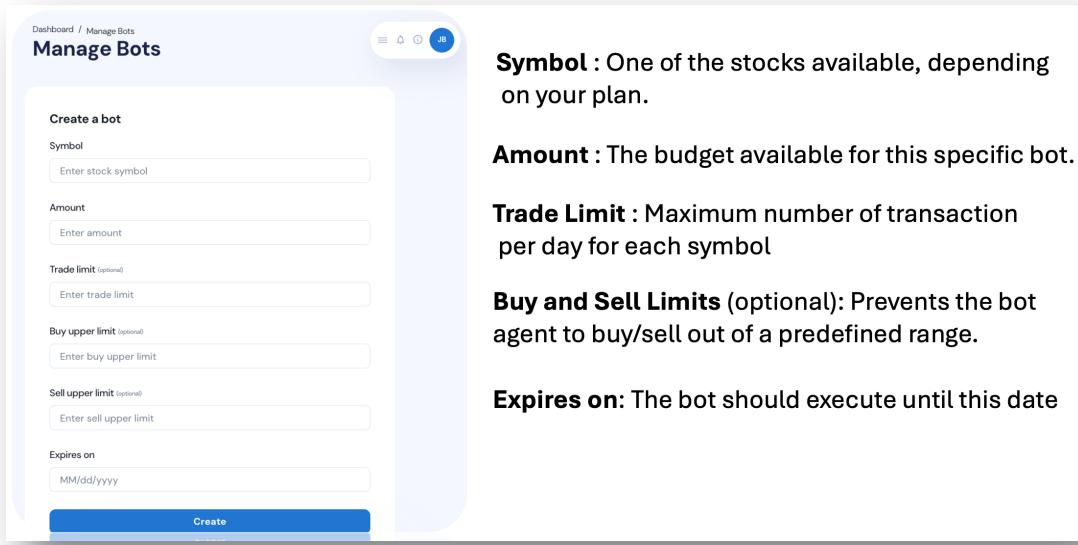
The screenshot shows the Stock Marketplace section of the AirBot dashboard. On the left, a sidebar menu includes Dashboard, Stock Marketplace (which is currently selected and highlighted in blue), Manage Bots, AI Knowledge Hub, and Profile. The main area is titled "Stock Marketplace" and displays a table of stocks with columns for SYMBOL, EXCHANGE, and ACTION (with a "View" button). The stocks listed are:

SYMBOL	EXCHANGE	ACTION
A	NYSE	View
AA	NYSE	View
AAA	ARCA	View
AAU	BATS	View
AACG	NASDAQ	View
AACI	NASDAQ	View
AACIU	NASDAQ	View
AACIW	NASDAQ	View

To the right, a detailed chart for the stock AAPL is shown, plotting price in USD (173.2) against time from May 2023 to April 2024. The chart features a purple line representing the price trend over time, with a light purple shaded area indicating the price range or volatility.

Manage Bots

Here you can create, delete, and apply simulations on trading bots. You can start by clicking on the "Create Bot" button and customize your bot by filling out the options below.



The screenshot shows a 'Manage Bots' interface with a 'Create a bot' form on the left. The form fields include:

- Symbol:** Enter stock symbol
- Amount:** Enter amount
- Trade limit (optional):** Enter trade limit
- Buy upper limit (optional):** Enter buy upper limit
- Sell upper limit (optional):** Enter sell upper limit
- Expires on:** MM/dd/yyyy

On the right, descriptive text explains the fields:

- Symbol :** One of the stocks available, depending on your plan.
- Amount :** The budget available for this specific bot.
- Trade Limit :** Maximum number of transaction per day for each symbol
- Buy and Sell Limits (optional):** Prevents the bot agent to buy/sell out of a predefined range.
- Expires on:** The bot should execute until this date

The trading agent bot will execute once a day through an automatic batch process that will perform trading actions, such as buying, selling, or "holding" stocks.

Depending on your plan, you may also choose the multi-paper option, where a stock portfolio is available to diversify your portfolio. If you choose a multi-paper option, the bot will manage your portfolio to optimize performance and value.

After creating your bot, you can navigate to the ACTION column and choose one of three options:

EXPIRES ON	ACTION
30 Apr, 2024	
30 Apr, 2024	 View Assets Run Simulation Delete

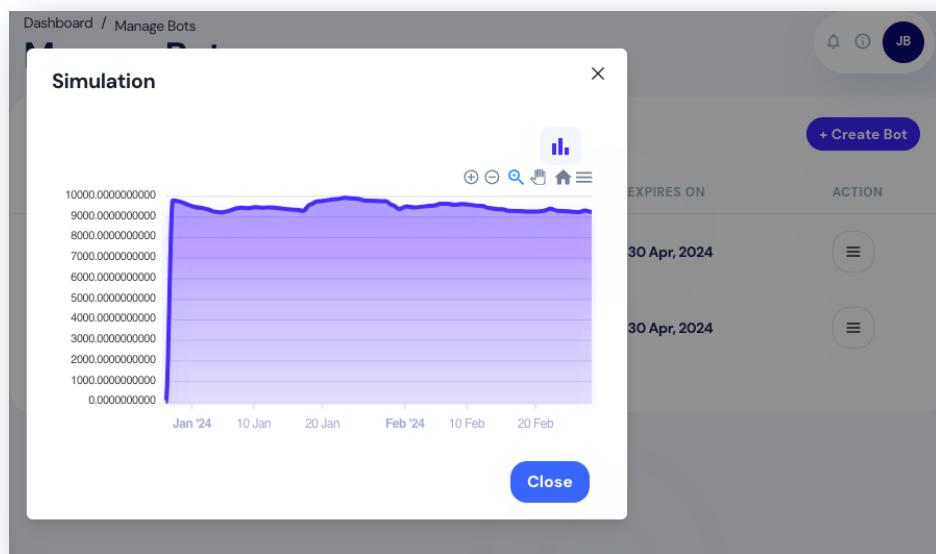
View Assets

History of buy and sell actions performed by the bot. You can track the number of shares for each stock symbol as well as its closing value on that day and how much fee was paid to the Exchange for that transaction.

Transactions		
SYMBOL	PRICE	DATE
AAPL	124.1666488647461	2023-01-05
AAPL	125.49749755859376	2023-01-04

Run Simulation

You can simulate a run of your bot, for comparison and strategy validation purposes, applying the customized settings of the bot you registered on historical data, generating a simulated result of what your bot's performance would have been in that situation.





Delete: Removes your bot from the database, performing a logical deletion, so that historical data can be retained.

AI Knowledge Hub

A chatbot for questions about stock marketing and system use. Simply say hi and start a chat with our Airbot. As the chatbot is only informational, it does not maintain the conversation session and does not "remember" what is being discussed in the AI Hub window.

The screenshot shows a web browser window for the 'AI Knowledge Hub' on localhost:3000. The left sidebar lists navigation options: Dashboard, Stock Marketplace, Manage Bots, AI Knowledge Hub (which is selected and highlighted in blue), and Profile. The main area is titled 'AI Knowledge Hub' and contains a conversation log. The user's messages are in grey boxes, and the bot's responses are in white boxes with a purple border. The conversation includes: 'Hello, how can I help you?', 'It is about : 2024-03-31 22:14:35', 'I'm the Air Bot \o/', and 'I exist in the digital realm to help you with your questions and tasks.' The user also sends messages asking for the time, name, and origin, which the bot ignores.



Dashboard / AI Knowledge Hub

AI Knowledge Hub

Knowledge Hub

where are you from?

I exist in the digital realm to help you with your questions and tasks.

Explain me about Apple

:::: Report for the Symbol AAPL::::
:: About the company ::
Company Introduction:Apple Inc is a leading entity in the Technology sector. Incorporated and publicly traded since 1980-12-12, the company has established its reputation as one of the key players in the market. As of today, Apple Inc has a market capitalization of 2705108.71 in USD, with 15441.88 shares outstanding. Apple Inc operates primarily in the US, trading under the ticker AAPL on the NASDAQ NMS – GLOBAL MARKET. As a dominant force in the Technology space, the company continues to innovate and drive progress within the industry.

Profile

In the profile section, you can review your registration data and change your plan.

Dashboard / Profile

Profile

Jaldhi Bhatt
bhattjaldhi27@gmail.com

Current plan
Pro
CAD 9.99 / Month
[Change plan](#)



Conclusion

Wrapping up

We were able to grasp how powerful the approach of deep reinforcement learning is and its various applications. This was very instructive and clarifying, especially since we did not have this subject in our AI and Data Science course, opening doors to this potent AI tool.

Despite having researched the investment market, we understand that there are still many aspects that we possibly did not cover in our study and our solutions.

Nonetheless, we believe our work was fruitful; we managed to consolidate all the content of our research through our creativity in utilizing available technologies in a very promising automated trading solution.

In the end, we understand that our solution to try to democratize the use was successful, with just a few clicks, the user can create a long-term investment.

Acknowledgement

We thank Mr. Stanley Chor, coordinator of the AIP, for his coaching and sharing his vast knowledge and experience, which helped and propelled us to give our best in this project.

We also thank our Project Advisor, Mr. Usman Ahmad, for guiding and directing us through this journey.

Disclaimer

Once again, it is important to reinforce that this is a purely academic project. We do not intend to guarantee that this project, as it is, can be used in a real environment. Nor can we affirm or



guarantee any kind of profit or that the project's applications are free from flaws, even though we have conducted several tests.

Recommendation

We understand that there is much room for improvement. If we had more resources, we could work even more on training and optimizing our models, use an infrastructure more compatible with our applications, and, most importantly, explore more the legal aspects associated with trading done by AI automations. Although we understand that we are in compliance with legal standards, as per our research, we recognize that we are not qualified to have the final say on these aspects. Ideally, we would have a legal team to ensure the legality of our platform and operations.

We also understand that our AI knowledge hub feature, which was basically an extra feature we created alongside the stock trading bot, did not receive the attention it deserved. Despite achieving great results, it could be improved if we had access to computational resources to fine-tune our own LLM model instead of using third-party APIs to format our responses in the RAG implementation.

References

- [01] Liu, X., Xia, Z., Yang, H., Gao, J., Zha, D., Zhu, M., Wang, C. D., Wang, Z., & Guo, J. (2023, April 25). Dynamic datasets and market environments for financial reinforcement learning. arXiv.org. <https://arxiv.org/abs/2304.13174>
- [02] Liu, X., Xia, Z., Rui, J., Gao, J., Yang, H., Zhu, M., Wang, C. D., Wang, Z., & Guo, J. (2022, November 6). FINRL-META: Market Environments and Benchmarks for Data-Driven Financial Reinforcement Learning. arXiv.org. <https://arxiv.org/abs/2211.03107>
- [03] Liu, X., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., & Wang, C. D. (2020, November 19). FINRL: a deep reinforcement learning library for automated stock trading in quantitative finance. arXiv.org. <https://arxiv.org/abs/2011.09607>
- [04] King, R. (2023, May 10). Statistics and facts about the stock market. Finder Canada. <https://www.finder.com/ca/stock-trading/stock-trading-statistics>
- [05] Quickstart | Ngrok documentation. (2024, April 10). <https://ngrok.com/docs/getting-started>
- [06] Beginner's Guide. (n.d.). https://nginx.org/en/docs/beginners_guide.html
- [07] Gunicorn - WSGI server — Gunicorn 21.2.0 documentation. (n.d.). <https://docs.gunicorn.org/en/stable/>
- [08] IIROC Rules. (2024, March 18). IIROC. <https://www.iroc.ca/rules-and-enforcement/iroc-rules>
- [09] Quickstart — ElegantRL 0.3.1 documentation. (n.d.). <https://eleganrl.readthedocs.io/en/latest/helloworld/quickstart.html>
- [10] TMX Group. (n.d.-b). Market regulation. TSX.com. <https://www.tsx.com/trading/toronto-stock-exchange/trading-rules-and-regulations/market-regulation>
- [11] Investment dealer and partially consolidated rules. (2024, January 14). Canadian Investment Regulatory Organization. <https://www.ciro.ca/rules-and-enforcement/investment-dealer-and-partially-consolidated-rules>
- [12] Beers, B. (2024b, March 18). How Long Does it Take for Brokers to Confirm a Trade? Investopedia. <https://www.investopedia.com/ask/answers/112814/how-long-does-it-take-broker-confirm-trade-after-it-placed.asp>
- [13] Hayes, A. (2023b, December 22). How does the stock market work? Investopedia. <https://www.investopedia.com/articles/investing/082614/how-stock-market-works.asp>



- [14] About us. (2024, April 2). Investopedia. <https://www.investopedia.com/about-us-5093223>
- [15] Plenty of individual investors are 'worried' about the stock market: Survey. (October 17, 2023). <https://finance.yahoo.com/video/plenty-individual-investors-worried-stock-165022329.html>
- [16] Beers, B. (2022, September 30). Why put money into a volatile stock market? Investopedia. <https://www.investopedia.com/ask/answers/why-put-money-volatile-stock-market/>
- [17] Team, I. (2023, November 27). How to trade the VIX: 4 Ways explained. Investopedia. <https://www.investopedia.com/stock-analysis/2012/4-ways-to-trade-the-vix-vxx-vxz-tvix-xxv0504.aspx>
- [18] Introduction — FinRL 0.3.1 documentation. (2021). <https://finrl.readthedocs.io/en/latest/start/introduction.html>
- [19] Deploying to Vercel. (2024). <https://vercel.com/docs/deployments/overview>
- [20] Tutorial: Get started with Amazon EC2 Linux instances - Amazon Elastic Compute Cloud. (2024.). https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html
- [21] About Broker API. (n.d.). Alpaca API Docs. <https://docs.alpaca.markets/docs/about-broker-api>
- [22] Generate API Reference | Legacy - Cohere Docs. (n.d.). Cohere AI. <https://docs.cohere.com/reference/generate>
- [23] Finnhub.Io. (n.d.). Finnhub - Free realtime APIs for stock, forex and cryptocurrency. <https://finnhub.io/docs/api>
- [24] Documentation. (n.d.). Firebase. <https://firebase.google.com/doc>
- [25] Yang, H. (Bruce), et al. (2023). FinGPT: Open-Source Financial Large Language Models. <https://doi.org/10.48550/arXiv.2306.06031>