Haaga-Helia
University of Applied Sciences

Behind the scenes of algorithmic trading

Raphaël Darbellay

Bachelor's Thesis
Degree Programme in
Business Information
Technology
2021

Haaga-Helia
University of Applied Sciences

Abstract

Date : April 10, 2021

Author
Raphaël Darbellay

Degree programme
Business Information Technology

Thesis title
Behind the scenes of algorithmic trading

Number of pages and appendix pages
89 + 18

Now more than ever, with the emergence of excessive speculation and social trading, it makes sense to talk about this topic, algorithmic trading.

The objective of this thesis will be the implementation of an autonomous trading software. This software should be able to operate without human intervention in the financial markets. It will follow a predefined strategy to find buying and selling opportunities. The criteria that will be used to evaluate the work will be the speed and performance of our software.

Throughout this report, a theoretical framework will be set up. This framework will be used for the implementation of the software. Every effort will be made to ensure that as many people as possible can develop the same software or duplicate it. No investment will be necessary. No real accounts will be used. The development of this software will be done at no cost.

This work is aimed at all readers who are passionate about algorithmic trading. Financial skills are not necessary, but programming skills are more than recommended to understand the importance of the work. The process of the development carried out will also be explained in detail. Every reader should be able to understand all the work that has been done. From the architecture to the algorithm, every element of this development will be shared in this document. As for the full code of the software written in Java, it will be available for download on the official GitHub repository for those readers who want it.

Finally, the algorithmic trading software that works and can be used on the stock markets will be tested. The results obtained will show that the performance in terms of analysis speed (about 6 milliseconds) will logically be better than that of a human for the same task. But also, that the strategy that will be implemented, although it works perfectly, could be improved afterwards to increase the performance.

Links
Repository: https://github.com/Raphael77777/BullittTWS.git
Release: https://github.com/Raphael77777/BullittTWS/releases/tag/v1.1.0

Keywords
algorithmic trading, automatic trading, computer program, high-frequency trading, trading strategies, stock market, software

Table of contents

1 Introduction

Trading has been going on for more than 2000 years. (Alti Trading, 2021) But from ancient Rome to today's markets, not many remains of the same thing. The development of the world of finance and more particularly the banking system has been so extraordinary that it sometimes says that it controls our society in the same way that politics could. Those who hold the money, often also hold the power.

In the last 50 years, the development of stock exchanges and all financial centres has been dazzling. The arrival of electronics and information technology has allowed a small revolution in many sectors, but more particularly in the trading business. Succulent mix of technology and financial engineering, trading algorithms are today much more present than most people would dare to hope. Ever faster and more powerful, these trading algorithms are said to account for up to 80% of the world's trading volume. (Café du Forex, 2018) If it has grown so fast, it is because this activity is profitable. Several tens of billions of dollars have been made by companies that carry out this activity at a level that could be described as obsessional.

Although developing an automatic trading system requires a more than colossal investment, we will try to prove that developing a functional trading algorithm is not impossible. In this way, we will dispel the myth that this activity is reserved only for privileged companies with astronomical capital. The goal will be to prove that a trading algorithm can easily beat a physical trader. The comparison will be mainly based on the speed of analysis and expected performance. The objective will be to perform better than a real investor but also faster. The speed will be measured against a simultaneous execution of the same task. The expected performance will be calculated using backtesting methods on the trading strategy that will be selected for this software.

This development of an algorithmic trading algorithm will be carried out after a global research on the trading industry. All the essential and useful elements for understanding this subject will be covered. From the arrival of electronics in the markets to the emergence of trading machines capable of placing orders automatically through algorithms, these topics will be examined in detail. The effects on the market, high frequency trading and its excesses as well as all the strategies used will be discovered in this theoretical part. Special attention will also be paid to the computer architecture used for this kind of software. This research work will allow to approach the development part with a certain ease but also to better apprehend and evaluate the obtained results.

This software is not intended to destroy companies that invest millions of dollars each year in research and development of powerful systems and algorithms. The aim is to use resources that are accessible to most people while maintaining an effective understanding of the system that will be put in place. In this sense, the content and vocabulary will be adapted to a novice population rather than a group of experts in automatic trading. The algorithmic trading system developed will be accessible online and its development should be kept simple to facilitate understanding.

There are some limitations regarding the algorithmic software we will try to develop. First, all transactions will be made through a fictitious account. This development will only be used for learning purposes and will not be used to make a profit. Secondly, some market data is not freely available. This is particularly the case for the live quotation of certain financial instruments. To overcome this problem, the system that will be developed will focus only on assets for which data is available free of charge. Lastly, the aim of the development is to set up an architecture that allows the use of a trading algorithm. The trading strategy used will not be the same as we might find on algorithmic trading systems that generate millions of euros in profits every year. As we will see later, the development of these algorithmic strategies requires special skills that are mostly found among quantitative analysts and that we do not have in this development process.

2 Theoretical framework

In this theoretical part, the trading industry will be studied. From the history of stock exchanges to the development of markets as we see them today. From standard trading algorithms to high frequency trading, with a small diversion through artificial intelligence. From the architecture of today's systems to the strategies used to find signals or those used to optimise order execution. All these elements will be popularised so that novice readers can grasp and understand the subject with a global view.

2.1 Early developments

To understand how algorithmic trading has managed to establish itself on all stock exchanges, we need to look more closely at how it has developed over the years. From the first financial markets to today's communication systems, via the first electronic market, these are all elements that have had a definite impact in establishing the presence of algorithmic trading.

2.1.1 Former financial markets

If we go back in history, we can understand that barter is already a form of negotiation. When a lumberjack provided wood for food, there was already an exchange taking place. In those days, it was enough for two people to agree to conclude a transaction. There were no regulations. Years later, we began to see the emergence of recognised exchange values, gold, silver, and precious metals. These means of payment or exchange values were used as currency long before the official appearance of modern state currencies.

A few centuries later, in Paris, London or New York, securities were exchanged at private fairs and exhibitions. These activities began in New York at the beginning of the War of Independence in 1776. (Attac-Québec, 2010) A few years later, on May 17th, 1792, 24 brokers gathered in Manhattan decided to create the New York Stock Exchange together. It was only in 1865, at the end of the Civil War, that the New York Stock Exchange moved to the place it still occupies today. (Attac-Québec, 2010)

The means of transmitting orders and information have always been the key element in the functioning of financial markets. Throughout history, stock exchanges have adopted all kinds of means to communicate. Starting with carrier pigeons, the telegraph, the telephone,

tyres and even roller skates, all these inventions made it possible to save time when transmitting information or executing a transaction. (Canal+, 2015)

It was only in 1867 that stock exchanges began to use the telegraph to transmit information, orders, and quotes for each asset. This invention would revolutionise the world of finance and more particularly the stock market, as would the Internet more than a century later. (Attac-Québec, 2010)

2.1.2 First electronic market

In 1971, the National Association of Securities Dealers Automated Quotations (NASDAQ) was founded. (Singh, 2009) From the outset, this exchange was electronic. It will mainly be used to trade technology stocks that do not have enough market capitalization to be listed on the largest financial markets in the United States.

The ease of access for issuers, the low costs, the importance of data, the quality and the simplicity of its organisation based on high-performance technology will quickly allow this stock exchange to become more than just a default stock exchange. (Singh, 2009) As soon as it appears, the NASDAQ is convincing, but it is during the explosion of technology stocks that this stock exchange will demonstrate its true potential.

As a result of its improvements, the NASDAQ became the reference for electronic markets. In the 80s, it began to welcome a lot of small technological start-ups such as Microsoft. Technological innovations will make the NASDAQ even more efficient, and the dot-com period will see the market capitalisation of several NASDAQ-listed companies soar. Having taken advantage of all the opportunities of the information age and thanks to technology companies with a lot of potential, it will become one of the most important stock market intermediaries in the United States and in the world.

In the wake of this technological change, other stock exchanges will adapt to new methods of communication and adopt electronics. After the telegraph, it is the computer that will succeed in making its mark following a series of scandals involving traders. It will then no longer be possible to turn back the clock. (Canal+, 2015)

2.1.3 Electronic communication networks

The birth of the NASDAQ in 1971 set the trend for the centuries to come. But the brokers at the time did not necessarily see this paradigm shift as a threat to them. (Singh, 2009) Although listings had become electronic, it was necessary to have a location within the exchange to have the privilege of trading. So, when an individual wished to buy a security, he had to first go to his bank. His bank would then contact the official trader present on the exchange to execute the transaction on behalf of the client. This method was restrictive. Everyone had access to the stock exchange but only a privileged few placed order on the markets. Transactions as described above were lengthy and expensive for the end customer. It was therefore logical that an alternative appeared.

At the beginning of the 1990s, discount brokers appeared on the stock exchanges. (Stock Trading Warrior, 2021) (Singh, 2009) They now allowed anyone to execute a transaction without necessarily going through a traditional banking institution. Anyone with a computer and an internet connection was now able to access the financial markets. The privilege that was previously reserved for institutional investors was now within everyone's reach. (Stock Trading Warrior, 2021) The systems created for the transmission provided data quickly, efficiently, and automatically. (Singh, 2009) It is these points that made it possible to drastically reduce traditional brokerage fees. (Lemettre, 2009) It is estimated that the average brokerage fee would have been reduced by a factor of ten over the first few years. (Singh, 2009)

These first few years also saw the appearance of new platforms. Yahoo and Microsoft began selling or offering investment information services. This information such as data, analysis or historical data was used by everyone to help in the decision-making process when buying or selling a financial asset. These platforms, partly free of charge, which offered a new advisory service, quickly became popular around the world by convincing people that stock market investment, trading and speculation was not just for professional brokers.

Very quickly, the practice spread. From a very closed circle of traders, the stock exchange has developed into a simple service accessible on the internet. More than 40% of transactions were then executed by investors who were not directly present on the stock exchange but who used information systems and Internet to conclude transactions. (Singh, 2009)

The opening of stock exchanges spread from one corner of the globe to the other in the years following these few changes. This gave the possibility of a new democratisation of the stock exchange and a massive influx of liquidity from non-institutional investors who saw this as a real opportunity for returns.

2.2 Current market effects

After studying the development and arrival of algorithmic trading in our modern society, we will see what changes this revolution has brought in its wake. From a simple decision-making aid to automated intelligent management, trading algorithms have completely changed the industry and dramatically altered the fundamental values of the way we invest.

2.2.1 Data driven culture

From an oil-based economy, we are rapidly starting to change paradigm. The oil of the 21st century is no longer black and viscous, but digital and sensitive. The data that is generated by all digital exchanges today is as many new opportunities that can be exploited. Traders have understood this and have adapted. From a flesh-and-blood broker to a high-powered computer capable of replacing the trader, a key element seems to be the appearance of new technologies. Above all, these new technologies rely mostly on data and use it to generate sometimes very large profits.

Our society is changing. The appearance of all these new masses of data and the appropriate tools for processing them allows us to innovate in many areas. In the field of finance in particular, data allows us to speed up exchanges. From just a few quotes per minute previously, quotes are now valued several times per second on the world's most important stock exchanges. This increase in the number of quotes over time suggests that short-term opportunities have also increased. With a few quotations per minute, it was logical to assume that a large change could not be achieved in minutes or few minutes. Today, prices are continually updated, which also offers more variation and more data to analyse for new opportunities.

Trading algorithms use all this new data that is generated to try to predict the future price, set a trend or even guess the intention of other investors. They can be used to transmit buy signals to other traders, but the most successful traders are directly capable of making the investment decision and executing the trades. With the right tools, meaning efficient

algorithms, these super-powerful computers are capable of analysing thousands of data and closing a trade before any information is displayed on the broker's computer screen.

This culture of algorithmic trading emerges because of big data, and this is no mere coincidence. The multiplication of data on financial assets but also on the stock exchanges itself has made it possible to develop algorithmic trading models that are today more efficient than many physical traders. (MJV Team, 2019) Trading, like many other sectors, has been able to take advantage of big data to adapt to this new data processing culture.

2.2.2 Error reduction

Error is human, but it is not always involuntary. Scandals in the financial industry are not new and many include physical traders. Rogue traders, i.e., rebellious brokers, may be directly involved in several embezzlements or manipulations.

Jérôme Kerviel is a salaried French trader who has caused Société Générale to lose a total of 4.91 billion euros. (Rousseau, 2013) By hiding the real size of the open positions, he has managed to expose the bank to around 60 billion euros. (Rousseau, 2013) At this stage, a simple change in prices could have caused an immediate collapse of the bank, dragging many other investors in the wake of one of the largest French banks. Fabrice Tourre and Bruno Michel Iksil are two other traders who will have managed to cost more than 2.5 billion euros to their employer. (Rousseau, 2013) Still using a manipulative approach and hiding key information that could have been used to avoid such a huge carnage.

More recently but still under investigation, the Liborgate. This is a manipulation of the LIBOR rate. An agreement between several traders from the largest banks would have made it possible to rig this rate according to the interests of all the players in the most total illegality. (Vigaud, 2012) Let us recall in passing that the LIBOR rate defines the value of more than 300'000 billion financial assets throughout the world and that it also has an impact on real estate loans. (Rousseau, 2013)

In all these situations, the employers of these traders had to compensate financially for the market manipulation with substantial fines or simply cash a loss on their investment. The appearance of trading algorithms, at a time when scandals were making headlines, was a way for banks to say "STOP". (Albanese, 2015) (Vigaud, 2012)

More reliable, faster, emotionless, efficient and with no social charges, many arguments have, in just a few years, convinced the major financial players to invest in this field. We

can see countless reasons behind the adoption of trading algorithms, but from a technical point of view, the level of risk is more stable. An algorithm is programmed not to exceed a certain level of risk no matter what the situation. Unlike a trader who might be driven by these emotions instead of his rational common sense.

Voluntary manipulation by agreement between traders and concealment of portfolio exposure are situations that simply cannot occur with an algorithm. The algorithm is pre-programmed to follow a road map. Therefore, and considering that the algorithm has been written by ethical professionals and with safety mechanisms, an accident is simply less likely, and all human errors are avoided.

2.2.3 Technology dependencies

Today more than ever, it is impossible to imagine the financial markets without seeing the technology behind the stock exchanges. From the history of the first electronic market to today's stock exchange, the digitisation of finance has been the thing that has changed the industry the most over the last few centuries. In barely 50 years, technological progress is such that it can no longer be dissociated from each other.

Financial markets are entirely digital, all stock exchanges have been deserted by traditional brokers. The New York Stock Exchange, the emblem of American capitalism, no longer hosts the slightest transaction within it since 2010. (Geneau, 2010) (Lash, 2011) A real change at that time, the New York Stock Exchange was relocated to the small town of Mahwah (New-Jersey), 60 kilometres from New York. (Lash, 2011) The data centre that was built there has the surface area equivalent of 5 football fields. However, the 37'000 square meters available are only used with a capacity of 20%. (Geneau, 2010) The 80% of the available space is not used directly for the stock exchange but is intended for rental, more precisely for high frequency traders that we will explain later. (Geneau, 2010)

While this relocation of the U.S. exchange remains in the same jurisdiction, the relocations of certain other exchanges have taken place in another country. The Paris exchange, for example, is not located in Paris or even in France. (La Finance Pour Tous, 2021) To find the servers of the French exchange, it is necessary to cross the English Channel. It is in a small town in England, on the outskirts of Basildon, that French servers, like those of many European stock exchanges, have found homes. (Canal+, 2015) About 40 kilometres from London, this town is home to a large percentage of all European transactions.

All these relocations are driven by a technological desire. Today, stock exchanges are no longer located according to the securities or assets they offer to trade. But they are in fact based on the technology available, the computer skills of the population, the available internet access, the available space. All these elements are studied before taking a decision to relocate to build huge data centres. Technological dependency is a certainty, and political or geopolitical risks have too little influence on these relocations.

If technology decides the location of the exchanges, it also decides their fate. Thus, to demonstrate the strong technological dependence of finance and the importance of speed of access to stock exchanges, we can cite the case of the stock exchanges in Zurich in Switzerland and Singapore in Asia. Prior to the 2000s, the two financial centres were at the same level in terms of transaction volume. But less than 20 years later, Singapore handles twice as many transactions as Zurich. (Renault, 2016) One of the main reasons for this change is geography. While Switzerland is a small, landlocked country, Singapore has direct access to the sea, which has enabled it to develop an underwater communication infrastructure quickly. Burying submarine transmission cables is cheaper than having to build a land-based system. Since speed has been the key element for years, it is logical that Singapore's financial centre has managed to establish itself thanks to a technological advantage. (Renault, 2016) This is also the case for the stock exchanges in Tokyo or London.

It is considered that technology has become the key to the development of the global financial markets. If technology is so present, it is because the players have understood that it offers opportunities for additional profit. All it took was a technological revolution and human greed to profoundly transform all the world's stock markets.

2.2.4 AI comes into play

Algorithms are pieces of code designed to produce a result based on input parameters. In this sense, a trading algorithm allows a certain predetermined operation to be carried out in contact with an identified situation. In this way, they make it possible to invest automatically in the markets by determining a strategy. Regardless of the situation, the algorithms perform the same trades repeatedly to achieve a profit.

Some detractors of this new algorithmic trading industry argue that it is impossible to use an algorithm on past data to try to predict the future evolution of an asset and that the critical sense and intuition of a flesh-and-blood trader will not be replaced by a vulgar computer. Awareness and the ability to make decisions automatically based on data but without

following a pre-established pattern are exactly the elements that lead us to glimpse a new trading opportunity thanks to artificial intelligence.

BlackRock is the world's largest asset management company. In 2021, the assets under management of this company would amount to approximately 8'700 billion dollars. (Arte, 2020) This figure corresponds to more than 30 times the annual GDP of Finland and more than 3 times that of France. This company has a huge impact on the real economy and grants a power never seen for a private sector company. (Beales, 2020) The company is not considered a bank but only an independent asset manager. If we are talking about BlackRock, it is obviously to talk about the artificial intelligence of this company.

Aladdin is BlackRock's computer system. Located in a small town in Washington State, in Wenatchee to take advantage of lower electricity costs, this artificial intelligence monitors absolutely everything that happens in the world. (Arte, 2020) The slightest statement from ministers, weather events and even a company's popularity rating are considered when estimating the value of an investment. All without human intervention. Every week, the programme carries out more than 200 million calculations to advise investments. (Arte, 2020)

“In the tale, the street boy Aladdin gets rich with the help of a genius. At BlackRock, it is the Aladdin computer system that enriches the company. “ (Arte, 2020)

Aladdin is the acronym for Asset, Liability, Debt and Derivative Investment Network. This artificial intelligence currently manages all BlackRock's investments, i.e., approximately 8'700 billion. But that is not all, artificial intelligence is also available to all other investors who are able to put a price tag on it to benefit from sound advice. A total of 21'600 billion dollars would be managed by this computer system, which is approximately the GDP of the United States. (Wikipedia, 2021) It is more than 10% of all the world's wealth which is based on this artificial intelligence. (Wikipedia, 2020)

“This standardisation of world investment could amplify the domino effect of the next financial crisis. “ (Arte, 2020)

The integration of artificial intelligence in the world of finance seems to have started a long time ago. Today most people still seem to be unaware of the eminent importance of artificial intelligence in our capitalist economy. Although the investments and the cost of maintaining such a computer system are out of the ordinary, it is worth the effort. And BlackRock already seems to be one step ahead in this new investment method.

2.3 High-frequency trading

High Frequency Trading (HFT) differs from algorithmic trading in a few points. Although they both use algorithms to perform financial transactions without human intervention of any kind, high-frequency trading robots buy and sell at speeds that surpass any understanding of human reason. (Evans, 2021) More rational trading algorithms, on the other hand, seek to invest for longer periods of time by analysing market or asset fundamentals or numbers.

2.3.1 Speed factor

According to a famous American film, "There are three ways to win. Be the first, be the smartest or cheat" (Chandor, 2011) If cheating does not seem to be a viable long-term opportunity and being the smartest requires too much investment in learning, then the last solution is to be the first. High frequency traders have understood this and have founded a business model entirely based on speed.

Initially designed to avoid human error, trading algorithms gave birth to this new entrant. These new high-frequency trading algorithms can complete an entire investment process in a fraction of a second. Get the data, analyse, decide, execute, start over. Nobody performs these operations faster than high frequency trading robots. 0.000064 seconds is the time that a powerful computer would need to decide to execute a trade by analysing all the data. (Reed, 2021) In 1 second, a single high frequency trading algorithm would be able to process a total of 15'625 transactions. While at the same time, a human trader would spend several seconds or minutes processing a single trade. (Café du Forex, 2018) Thanks to the new performance of these computers and the democratisation of stock exchanges, these computers can overtake any seasoned broker, surpass all other algorithms, and impose their presence in the system. (Evans, 2021)

Speed has always been the key element in the development of high-frequency trading algorithms. For years, firms that execute these algorithms have invested heavily in trying to squeeze a few milliseconds out of their competitors. (Krugman, 2014) These companies mainly employ traders, mathematicians, quantitative engineers, and computer scientists, but also external companies to get the best connection speed to the stock exchange server. (Café du Forex, 2018)

A man named Dan Spivey working at Spread Networks is a perfect example of this American rush for speed. In 2010, he discovered a great investment opportunity. The New York and Chicago stock exchanges, some 1,300 kilometres apart, are financial centres

where several of the same companies are listed simultaneously. This means that a share of a company is traded on both stock exchanges at the same time. This dual listing offers the opportunity to arbitrage between prices.

To better understand the principle of arbitrage, let us imagine a share listed on two stock exchanges. If the price rises on one exchange, the exchange in question sends the information to the other exchange so that it can adjust the price of the asset. A problem arises if someone manages to transmit a transaction before the price is adjusted on the second exchange. This strategy is called price arbitrage.

The topography of this region is not conducive to creating a completely straight fibre optic line between New York and Chicago, particularly because of certain mountain ranges. This is in any case the observation made by the people who had built the communication network between the two stock exchanges at the time. The line used by the exchanges is therefore the most optimal and fastest only if we exclude the fact that passing through the mountains is a possibility.

Dan Spivey decided to take up the challenge and beat the current speed record between the two exchanges, which was 16 milliseconds round trip. With an investment of $300 million and 1,330 kilometres, he will manage to build the straightest line ever by passing through everything in his path and reach a speed of 13 milliseconds round trip between the servers of the two exchanges. (Papadopoulos, 2020) He will rent this line for millions of dollars to high-frequency trading companies for a few years but will end up being put out of business by a company that will use microwave relays and will manage to reach 9 milliseconds for a round trip. (Papadopoulos, 2020)

Figure 1 : Map of connections between New York and Chicago (Eric Budish, 2014)

Today, and still at a cost of tens or hundreds of millions of dollars, high frequency traders still dream of becoming the fastest on the stock market. (Gupta, 2015) (Krugman, 2014)

2.3.2 Risks and controversy

If the profits are real, the risks are significant. High-frequency trading constantly optimises all decisions made, speeds up order transmission and aims for absolute profit. So, it only takes one wrong move or one false rumour to derail an entire global trading system. Thanks to globalisation, not only the world is interconnected, but also the stock exchanges. The risks that a country agrees to take by not putting in place any rules affect not only one country but the whole system. So much so that in 2008, the American subprime crisis spread across the globe.

For 10 years now, the entire economy has been subject to rapid changes for no apparent reason. On May 6th, 2010, the shares of the technology company Accenture suddenly dropped from $40 to $0.01 in 3 minutes. (Gogerty, 2014) This drop corresponds to a negative movement of 99.99% in the stock market value of this company. Thus, in a few minutes, US$35 billion evaporated. (Gogerty, 2014) At the same time, a total of 1000 billion dollars went up in smoke over all the other companies listed on the stock exchange. (Gogerty, 2014) It is only a few minutes later that the share price will return to an adequate and comparable price to the one before. (Treanor, 2015)

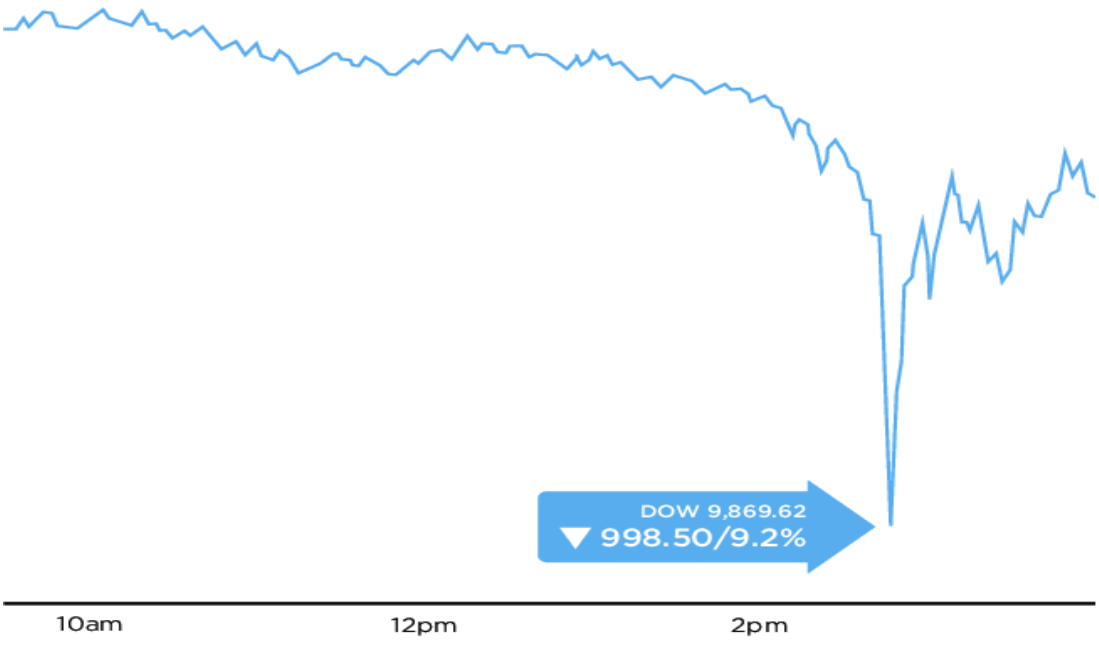


Figure 2 : Fall of the courses during the Flash Crash of May 6, 2010 (Doubleday, 2013)

It took several months to officially determine that the person responsible for this infernal downward spiral was in fact the high-frequency trading algorithms that had massively sold all their positions just because that is what everyone was doing at the stock exchange now. These sudden price drops that appear and recover within minutes are called flash crashes. In these situations, very real companies can see their stock market crashes suddenly without having the slightest chance to act. (Gogerty, 2014)

While prices may reach derisory lows, there are also assets that may react differently. (Treanor, 2015) This is the case of the Apple company which, on May 6th, 2010, suddenly reached a value of 100'000 dollars per share. (Gogerty, 2014) (Treanor, 2015) This valuation would imply that the company has a market capitalization equal to 93'200 billion dollars, an amount larger than the sum of the gross domestic product of all countries in the world. (Gogerty, 2014)

Although the values fell or rose at a great speed, there was no change in the real value of the companies nor in their valuation on that day. (Gogerty, 2014) High frequency algorithms would be responsible for these flash crashes for a very simple reason. They rely solely on price to detect buying and selling opportunities. The real or intrinsic value of a company is never considered. Too busy watching the micro differences between prices and taking the stake in a few microseconds, these high-frequency algorithms put our economy at risk for a very private gain. (Renault, 2016)

2.3.3 Market manipulation

"The market is rigged! " (L., 2014) This is what the author and former Wall Street trader Michael Lewis said live on CBS channel on March 31st, 2014. 7 years later, it is still the same observation that we can discover by listening to several former traders who were literally ejected by these new generation trading machines. (Held-Khawam, 2015)

When we go to the cinema, we stand in line and then when it is our turn, we buy our ticket to go and see the film. This could be compared with finance and trading before the advent of high frequency traders. Today, while you are waiting in line, people are passing you. They arrive at the cash register before you and buy all the tickets. Of course, they immediately sell them to you for a little bit more. You are still happy because you have been able to get what you were there for. That is exactly what happens on the financial markets these days. (Held-Khawam, 2015)

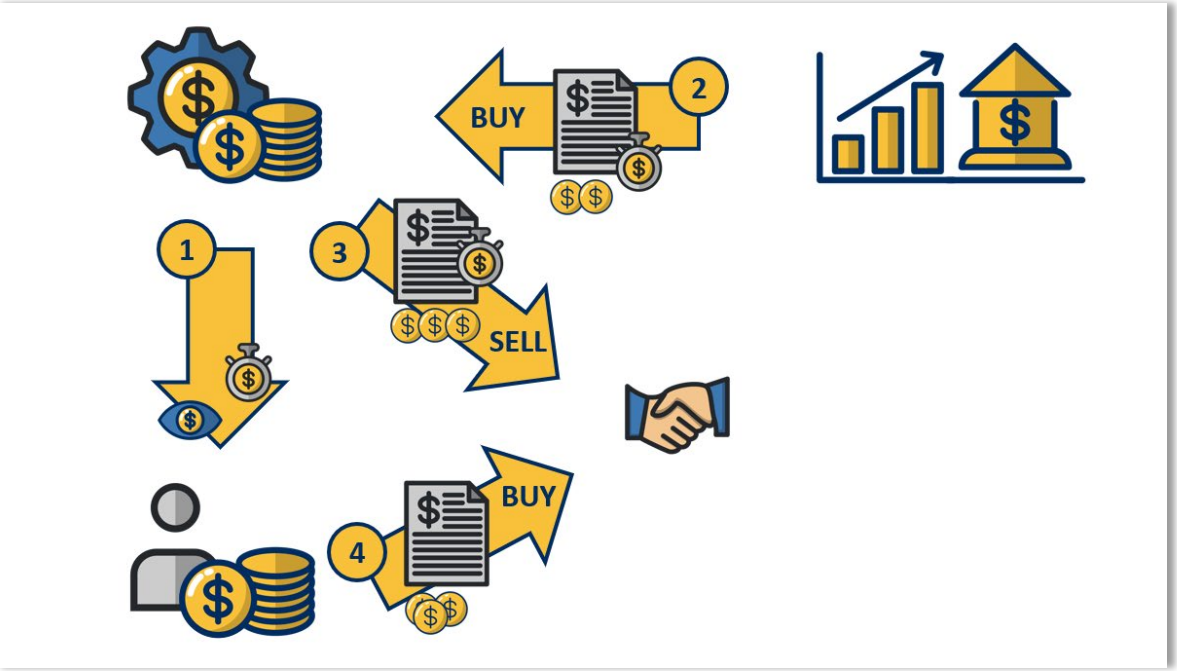


Figure 3 : The process of manipulating purchase prices on stock markets

This is what former Royal Bank of Canada traders would have discovered. (Canal+, 2015) Every time they tried to buy an order in the market, the available quantities would drop as soon as the order was valid but before they could buy anything. At first, they thought it was a coincidence, that someone was placing orders at the same time. Later, they discovered that the trigger was the buy or sell order they wanted to place in the market. So as soon as they placed an order, the available quantity would drop. In a demonstration of this manipulation, the traders entered orders without validating them, then waited a few more seconds to see if the quantities fell, but it was only when they validated the orders that the number of shares in the market disappeared. So, they were being overtaken by a tool that was able to identify their order and get ahead of them.

In his book "Flash Boys", Michael Lewis clearly states: "This is a scam. A legal one, to be sure. But a scam, nonetheless. A kind of technological hold-up, neither seen nor known... " (Held-Khawam, 2015)

The number of orders processed by these high-frequency algorithms is such that even the authorities can no longer cope with them. Tracks are being blurred and the markets are becoming more and more opaque. This opacity continues to benefit high-frequency traders, who can implement any strategy they wish. The only people who would be able to regulate, or at least implement adequate regulation, today are the high frequency traders themselves. (Canal+, 2015)

In addition, high-frequency trading companies have developed a system for regulating high-frequency trading. The problem is that these companies know in advance what can or cannot be detected in the markets. Trusting a burglar to install security cameras on his house is a bit like what the financial market authorities do with high-frequency trading. But did he really have other possibilities now? Poor regulation is perhaps better than no regulation at all.

2.3.4 Market impact

To understand whether high frequency trading has an impact on the health of financial markets, it is necessary to define the conditions required for a stock market to be considered healthy. First, it is necessary for the market to be liquid. That is, there must be buyers and sellers so that anyone can buy or sell an asset at any time. (Waikar, 2019) If the market were composed of only one seller, then it would be a monopoly. The price of the asset would not necessarily result in an average value to investors but simply the value at which the seller would value the asset. The more participants there are, the more liquid the market is called. This liquidity ensures that the execution price that can be obtained is the result of an implicit consensus between all the participants in the trade. The second parameter to be considered is the price of the asset. This must correspond to the real value of the company or the underlying. (Waikar, 2019) If the asset is quoted at 3 times the real value of the company, then the market is not considered as healthy and sustainable.

In this sense, high-frequency trading provides liquidity through a multitude of orders that are sent very quickly. Buyers and sellers can thus always find a participant who will agree to conclude a transaction.

On the other hand, and as demonstrated before, high-frequency traders can outperform standard brokers or algorithms. They are therefore free to influence the transaction prices of other participants. This is opposed to the healthy functioning of the market. Transactions should be concluded at the market price and not at the price imposed by a high-frequency algorithm. (Parker, 2019)

Looking further, the trading algorithms do not produce any value that is accounted for in the country's gross domestic product. They simply buy and sell assets to make large profits. If money is not created, then it comes from another source. All participants who invest in the market may be involved. With each transaction, small investors are easy prey for high frequency algorithms. (Parker, 2019)

Let us imagine an investor X who decides to invest in company Y. He buys a share for 4.05 euros. During his investment, a high-frequency trader overtook him and raised the price by 0.05 euros. A few weeks later, investor X sells his stock for 4.95 euros. Again, the algorithm came into play and managed to bring the price down by 0.05 euros. By doing the calculations, Investor X makes a profit of 0.90 euros and the high frequency algorithm 0.10 euros. Both players are winners, but investor X has been deducted part of his profit.

High frequency trading algorithms do not prevent other players, as many people think, from making successful trades. They simply take part of the profits. More precisely, they target micro-anomalies and rush into the loophole to rake in very low profits that multiplied with many large transactions allow these companies to make huge profits. (Canal+, 2015) High frequency trading is not fundamentally bad for the health of the markets, nor is it good for it either. It brings advantages that need to be recognised and disadvantages that should be better regulated to avoid further slippage. (Parker, 2019)

One idea, as Joshua Mollner explains, would be to set up a minimum delay between orders to slow down high frequency algorithms and thus reduce the undesirable effects of this practice. (Waikar, 2019) "It's not about making drastic, sweeping changes, small surgical changes, like the delays we're proposing, can have a big impact. The tiniest speed bumps can make a big difference." (Waikar, 2019) Perhaps a solution as simple as this would be effective and prevent the next flash crash.

2.4 System architecture

To understand how algorithmic trading systems work, we will study the architecture of such a system and then dissect the different elements that make it up.

"The automated trading system or Algorithmic Trading has been at the centre-stage of the trading world for more than a decade now. A "trading system", more commonly referred to as a "trading strategy" is nothing but a set of rules, which is applied to the given input data to generate entry and exit signals (buy/sell)" (Pachanekar, 2019)

While building a trading strategy may seem simple at first glance, it is not. The rules that define the actions that the algorithm will have to perform are very often studied and developed by professionals, quantitative analysts. (Pinto, 2012) These highly qualified people are very much in demand on the job market because they allow to create strategies that will make a lot of money later.

2.4.1 New system architecture

Traditional systems that existed before the emergence of algorithmic trading are no longer relevant. Due to too high latency and problems with their scalability, a new architectural system has emerged to solve the points that were problematic with the previous architecture. (Pinto, 2012)

The diagram below shows a typical architecture optimised for algorithmic trading. It is mainly composed of 3 different lanes: Application, Server, and Exchange. Each of these lanes has a specific purpose. The Application lane, equivalent to a user interface, allows interaction, visualization, and control of the Server part. The Exchange part is responsible for processing and routing transactions, but also for transmitting information on various financial assets. The Server side is the brains of the system. It is the most important part because it allows the application of a strategy studied and developed for the purpose of carrying out transactions on the stock exchanges.

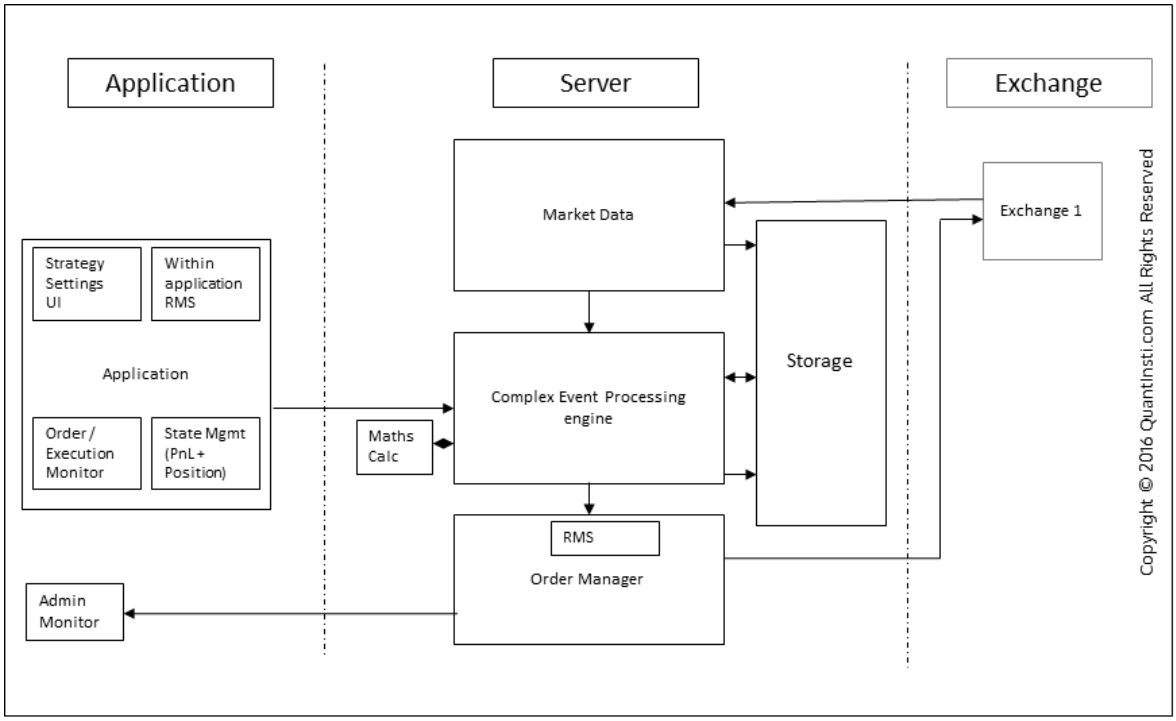


Figure 4 : Diagram of the architectural components of an algorithmic trading system (Pachanekar, 2019)

Since the Application part is mainly used as a user interface, we will not study its components further. The same applies to the Exchange part, which has only the purpose of conducting information or decisions. The Server part is the one we are interested in. It will make the decisions and apply an algorithmic trading strategy. It can be broken down

into 3 main components: Market Adapter (MA), Complex Event Processing Engine (CEP) and Order Routing System (ORS).

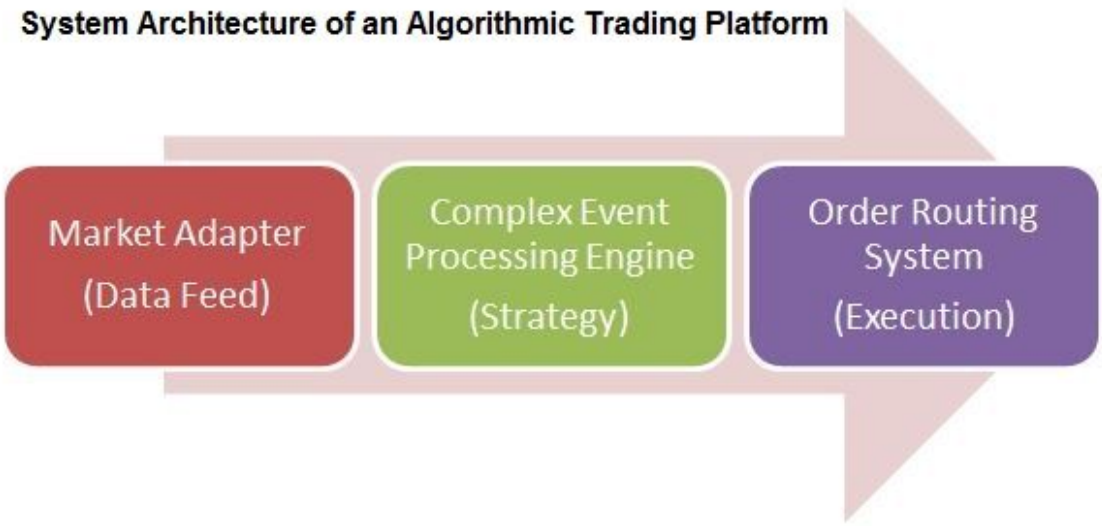


Figure 5 : Sequence of use of the architectural components of an algorithmic trading system (Pachanekar, 2019)

2.4.2 Market adapter

The market adapter is, as its name suggests, an adapter. This adapter allows you to communicate with the broker or directly with the stock exchange. Intermediaries or stock exchanges provide data so that brokers and traders can make informed investment decisions. This data is mainly live quotes.

The live quotes that are transmitted by these intermediaries are often not usable as they are, as pre-processing is necessary before our system can apply a strategy. The adapter therefore converts all data that is sent by these intermediaries so that it can be understood by our EPC.

2.4.3 Complex event processing engine

Once the data is converted by the MA, we can use it. The EPC is responsible for working with this data. This work requires various statistical calculations, comparisons with historical data and decision making for the generation of transactions. (Reid, 2014)

"A complex event is nothing but a set of incoming events. These include stock trends, market movements, news etc... Complex event processing is performing computational

operations on complex events in a short time. In an automated trading system, the operations can include detecting complex patterns, building correlations and relationships such as causality and timing between any incoming events. CEP systems process events in real-time. " (Pachanekar, 2019)

The CEP is the brain of our system. Most development work is usually concentrated on this part that can make all the difference. To better understand the importance of this component, we can make a small comparison with something well known. A car allows us to move from point A to point B. The body of the vehicle is important so that we can use it but useless if the engine is not running. It is the same with the CEP, if it does not work then the system has no reason to exist.

Many people underestimate the work involved in updating this component. An efficient CEP is a CEP that is continuously maintained with new strategies. "No single strategy can guarantee everlasting profits. Hence, quants are required to come up with new strategies on a regular basis to maintain an edge in the markets. " (Pachanekar, 2019)

There are many algorithmic trading strategies that are popular. They can be used as a basis for development or used as is. We will go through the most popular strategies in the next chapter. However, it is quite clear that the development of these strategies is a business and that to expect a substantial gain, it is necessary to call upon qualified quantitative analysts.

2.4.4 Order routing system

If the MA and EPC are used to make investment decisions, a component is needed to execute the decisions. The role of the ORS is precisely to communicate, in a language that intermediaries can understand, the willingness to execute a transaction in the financial markets. The type of transactions, quantities and all other information relating to the orders are transmitted by the EPC to the ORS, which then carries out the transactions based on the information that has been previously issued.

The ORS is only able to execute transactions but may also use strategies to do so. There are a few execution strategies that are popular. These will be discussed in one of the following chapters.

"Since automated trading systems work without any human intervention, it becomes pertinent to have thorough risk checks to ensure that the trading systems perform as

designed. The absence of risk checks or faulty risk management can lead to enormous irrecoverable losses. Thus, a risk management system (RMS) forms a very critical component of any automated trading system." (Pachanekar, 2019) The Risk Management System (RMS) is generally located in the ORS. This position enables it to verify all decisions that are taken before the final transmission of the order. This is an important operation that makes it possible to monitor the proper functioning of the RMS.

Once passed through the MA, the CEP and the RMS, the buy or sell order is transmitted by the ORS to the exchange so that the transaction can take place. The time it takes for these transactions to take place ranges from milliseconds to seconds. The robustness of the infrastructure, the speed of transmission and the speed of information processing play an important role in the speed of placing an order. (Pinto, 2012)

2.5 Analytical strategy discovery

In this section we will look at how to find trading opportunities. These opportunities are usually identified through optimised trading strategies. Each strategy uses data to provide a clear buy signal and the direction of the position to be taken. That is, it decides whether to buy or sell a financial instrument at a given time using an algorithm. (Seth, 2020)

These algorithms can be very short or very complex depending on the strategy being considered. But in any case, these algorithms, which allow people to earn millions and many more per year, are very well protected and are only accessible with significant financial means. The strategies that will be discussed are the most common in trading algorithms.

2.5.1 Trend-following strategies

This strategy is one of the most widely used, for an obvious reason: It is the simplest!

To illustrate this strategy, we will use a metaphor. Let us take the example of fashion, do we need to know where it comes from to follow it? Another example with electricity, do we need to know how it is produced to use it? The answer to both questions is generally no. The Trend-following strategy is based on the same principle. We do not need to know why the price of a share rises or falls if we know the direction of the movement of financial assets.

We will have understood that, thanks to the analysis of the trend of a share or stock, we can invest without making any prediction on the future price or without researching the tool we

buy on the market. The tool does not matter, whether it is the stock of a wheat company or the shares of the world's largest bank, only the direction of the financial asset is analysed and makes it possible to make gains. (Covel, 2021)

This strategy takes advantage of crowd psychology. If someone starts to sell, then the person next door will tend to want to sell in turn and so on. The sheep effect, as it is rightly called, is indeed present in the financial markets. And conversely, if the stock starts to rise, people will want to buy that asset. In this scenario, the trader or the trading algorithm that detects the trend early enough can open a position and start cashing in on the purchased assets as soon as the trend moves favourably against the trader's prognosis. In this technique, the key to success is simply being able to identify a trend early enough and ride the wave. The main question now is how a trading algorithm can read, understand, and determine a trend from the data it receives.

Let us start by explaining a technical indicator that will be used in this strategy: the simple moving average. This indicator, also abbreviated as SMA or MA, is commonly used in the world of finance and trading. Its calculation is relatively simple. Let us take the SMA20 indicator as an example, it calculates the average of the last 20 prices on the desired time scale. In the same way, the MA200 indicator calculates the average of the last 200 prices.

The technique known as "Trend Optimised Golden Cross Strategy" is a strategy that can be used to obtain buy and sell signals for all types of financial instruments. It works mainly with 3 conditions. If the price is above the average of the last 200 prices and the average of the last 9 prices is above the average of the last 50 prices, then the computer must buy. (Coin Rule, 2020) If the average of the last 9 prices goes below the average of the last 50 prices, then the computer must sell. (Coin Rule, 2020) The graphs (Figure 6, Figure 7) below show the appearance of these conditions.

Figure 6 : Chart with signals for the technique on Apple assets (Modified screenshot)

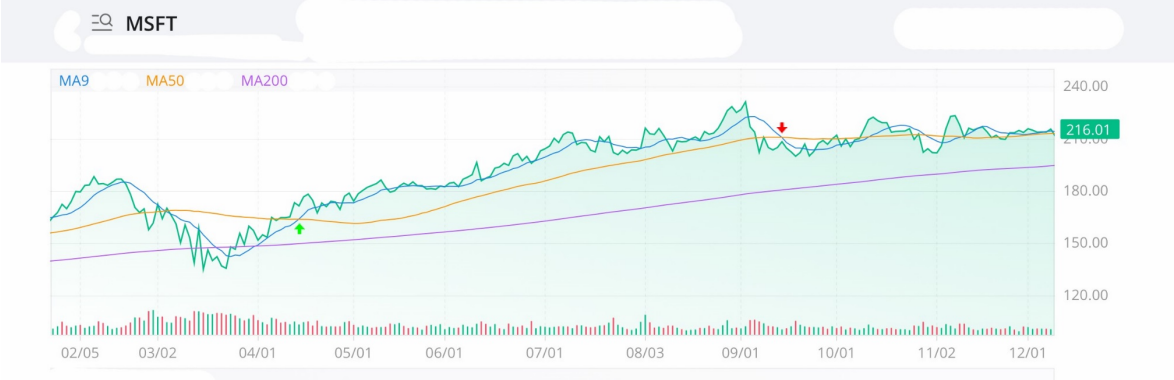


Figure 7 : Chart with signals for the technique on Microsoft assets (Modified screenshot)

This technique can be implemented very easily and works perfectly in theory. Although some important parameters are still missing, we could very well imagine a robot trader with this kind of strategy.

2.5.2 Arbitrage opportunities

This strategy is singularly simple to explain but more complicated to implement.

To help understand this strategy, we will use a comparison with a fruit and vegetable market. A customer arrives and wishes to buy 20 kilos of courgette for 2.89 euros per kilo. We hear the rumour and decide to go hunting. We quickly find a seller who sells his courgettes for 2.84 euros per kilo and who is willing to sell us the 20 kilos we need. We buy at 2.84 euros and immediately sell the 20 kilos at 2.89 euros. Using this strategy, we made a profit of 0.05 per kilo (selling price - buying price), i.e., a total profit of 1.00 (profit per kilo * volume). This gain is derisory but what is important to understand is that we took no risk to make a profit. Since we carried out both transactions simultaneously, the investment time is very short or even non-existent. This very short investment time and the absence of risk is what characterises this strategy.

We have understood how this strategy works in a fruit and vegetable market, now let us look at the same thing in the financial markets. Some instruments are traded on several stock exchanges at the same time. The aim is to find price differences in the markets and exploit this loophole. If both transactions are executed simultaneously and a price difference exists, a capital gain is realised without taking any risk.

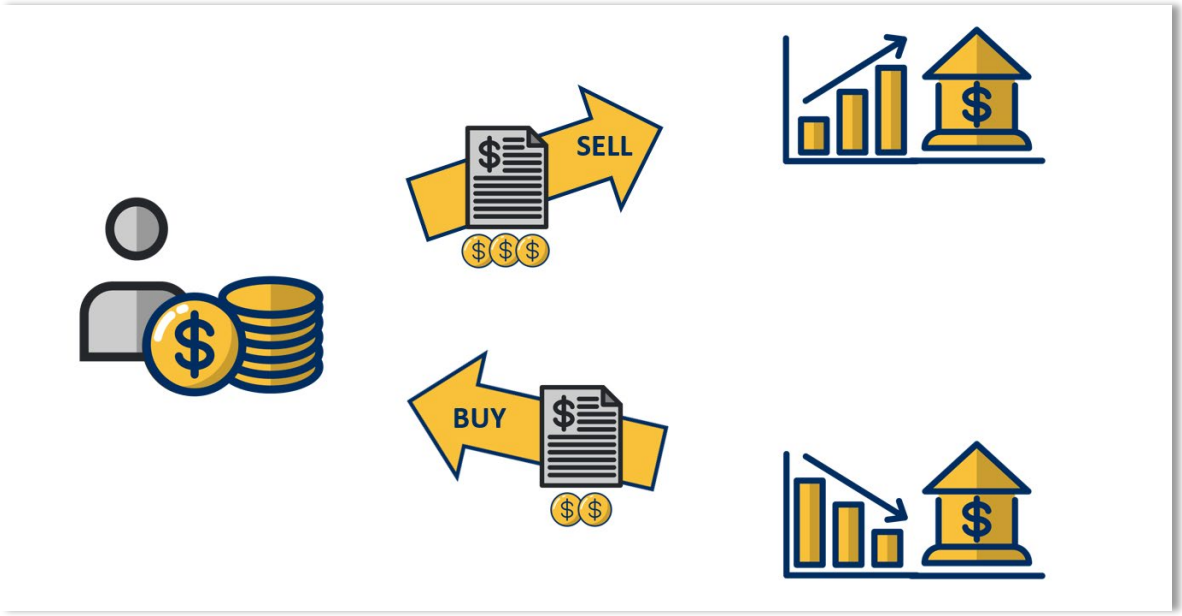


Figure 8 : Practice of simultaneous buying and selling at different prices

The main elements to be considered to successfully implement this strategy are: Speed, because we do not want prices to change until our orders are executed. And simultaneous execution, because we need to be sure that we have what we are selling, and that the other person is buying. (Corporate Finance Institute (CFI), 2021)

To complicate this strategy a bit, we will implement two variables that are missing in our first example but are essential. The first variable is the exchange rate. Since financial instruments are traded on several stock exchanges in several countries or continents, it is very likely that the currency of the prices will be different each time. It is therefore necessary to compare prices in the same currency. (Broking, 2020) The second variable is the cost of executing transactions. Traders generally pay buying and selling fees on the instruments they trade, and it is the same for trading algorithms (Broking, 2020). In short, it is necessary to compare the values of different instruments considering these two important elements. (Corporate Finance Institute (CFI), 2021)

This strategy works in theory, but as we can reasonably expect, such differences on known instruments are rare. To find larger differences, it is rather necessary to turn to options or derivatives exchanges. These stock markets that sell options, for example, can present significant opportunities. (Corporate Finance Institute (CFI), 2021)

An option is a contract that gives the purchaser the right to buy or sell a specific security at a predetermined price before the maturity of the contract. Now imagine that the price of the underlying asset, the specific security to which the option relates, suddenly changes. In this scenario, the value of the option will also change, but perhaps not as suddenly as the

security itself. This scenario would also allow us to exploit this difference in arbitrage to make a gain.

2.5.3 Index fund rebalancing

This strategy can offer definite returns for those with the right information.

To illustrate this strategy, we will start by simplifying things. Let us take an investor X who decides to invest 1,000 euros on the stock market with the absolute desire to own 50% of the shares of company A and the other 50% of company B. One year after his investment, he looks at his share portfolio. The share of company A has risen by 20% and that of company B has fallen by 20%. The total sum of his portfolio is therefore still 1,000 euros, but it is composed of 60% of the shares of company A and 40% of the shares of company B. Investor X is not happy with the allocation of his money. He wants to have half of his portfolio in each share. He therefore decides to sell the excess 10% of the shares of company A and buy the missing 10% of the shares of company B. His portfolio is now rebalanced.

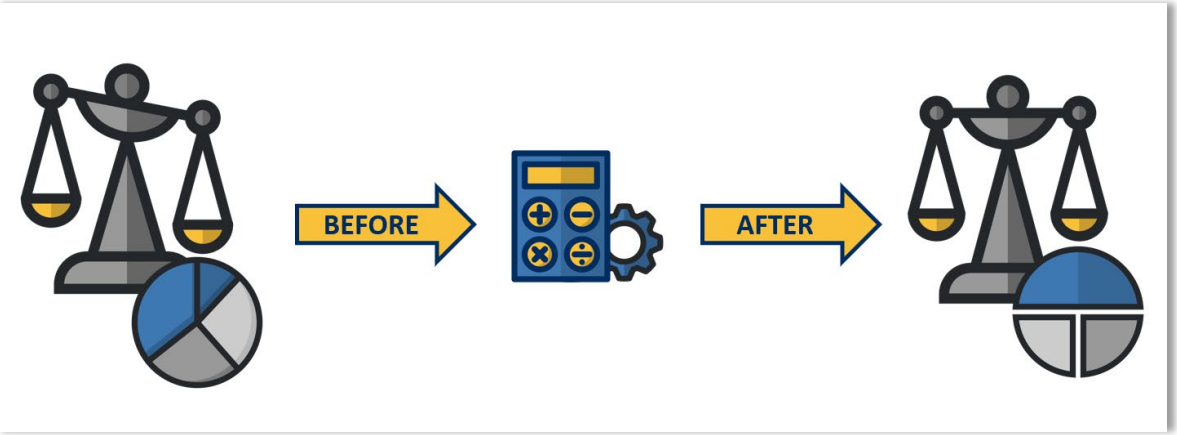


Figure 9 : Stock market portfolio before and after rebalancing

While the above example may seem normal, it is in fact an opportunity for other people in the markets. If we had known Investor X's strategy and when he was going to update his portfolio before he executed the trades, we would have been able to determine which shares he was going to sell, and which shares he was going to buy. Knowing this information, we could therefore have anticipated price movements.

It is not possible to know the strategy of all small investors like Investor X above. It does not matter anyway, since the volume he trades is not large enough to impact the market and make the price change. But it is a different story when it comes to stock market indices or public investment funds. These two instruments use the same rebalancing technique to

adjust the weight of each position either according to the real economy or according to the risks accepted. (Chen, Rebalancing, 2020)

It is relatively easy for everyone to know the components of an index. Similarly, the components of a mutual investment fund as well as the applied strategy is usually published to convince investors. All that is missing is the date on which all transactions will be carried out. For the major US indices, for example, the dates are officially published and accessible to everyone, i.e., the third Friday at the end of each calendar quarter. (Black Rock, 2020) For investment funds, they have the possibility to do so at their discretion but generally adapt their dates to the official index dates to replicate the allocation changes within their portfolio.

If we have a life insurance or simply a pension fund invested in the financial markets, we have normally had to make a choice based on the returns chosen. This choice that is made is in fact the admitted percentage of each asset class according to the risk and the expected return. The operation is the same for these elements, which must also adjust the weighting of the asset class according to past performance to maintain a level of risk equal to the investor profile. (Chen, Rebalancing, 2020)

It is very easy to use these rebalancing movements to your advantage and implement it in a trading algorithm. But you still need to know the information and be able to rely on it blindly.

2.5.4 Mathematical model-based strategies

Mathematical models have always had something fascinating about them, and this strategy draws heavily on them.

There are several mathematical concepts that can be used to predict events. Descriptive statistics, probability theory, linear algebra, linear regression, and many other calculation methods can be effective for algorithmic trading. (Thakar, Essential Mathematical Concepts for Algorithmic Trading, 2020) Whether central tendency, dispersion measure, mean, median, mode, variance, or deviation measures, all these descriptive statistics tools can be used to predict the price or direction of a financial instrument with varying degrees of success. (Thakar, Essential Mathematical Concepts for Algorithmic Trading, 2020) Although effective in theory, all these mathematical models rely on past performance to estimate the future trajectory. These trading models that try to reproduce past movements are not always certain and must be implemented with a sharp critical eye.

And what would happen if it were no longer necessary to know whether the market value of an asset will rise or fall in the next few hours or days to make a profit? Traders have been using a similar concept for a long time to enable them to trade without worrying about the direction of the market. This is the Delta-neutral strategy, a proven mathematical model.

To understand how this strategy works, it is necessary to understand how options and the hedging they provide work. Let us take the example of a call option on company X. This is a contract that gives the holder the right to buy a share, in the future and until the expiry of the contract, at a pre-determined price. In the same way, a put option gives its owner the right to sell the underlying at a predetermined price. The underlying is the item that can be bought or sold through this contract. Exercise, i.e., the transaction to buy or sell the underlying, is not mandatory. It is up to the holder of the right to decide whether he wishes to benefit from the option or whether he wishes to let the contract expire instead.

Before starting to explain the strategy, it is necessary to know one last parameter that will be used. The delta of an option is the rate of change in the price of an option relative to a one-unit change in the price of the underlying asset. (Chen, Delta Neutral, 2020) For example, if a call option has a delta of 0.4 and the underlying increases by 3%, the performance of the option will be equal to the performance of the underlying multiplied by the delta of the option. It is also important to be aware that the delta decreases over time and changes in line with the price development of the underlying asset. (Chen, Delta Neutral, 2020)

Let us take the following concrete scenario (Forex OX, 2018) , a share X with a price of 110.00 euros and a call option on share X with a strike price of 110.00 euros, a statistical volatility of the underlying of 8% and a residual maturity of 30 days. The estimated prices for this option are as follows:

UNDERLYING PRICE (EUR)	OPTION PRICE (EUR)	OPTION DELTA
108.00	2.14	-0.42
109.00	1.43	-0.58
110.00	0.91	-0.42
111.00	0.53	-0.28
112.00	0.28	-0.16

Table 1 : Table with price of the underlying, option price and option delta

Investor Y wishes to purchase 2 shares X at 110.00 euros and 5 options at 0.91 euros. In this situation, the delta of the shares is equal to the number of shares purchased, i.e., 2.00. And the option delta is equivalent to the option delta multiplied by the number of options

purchased, i.e., -2.10 (-0.42 * 5). In this situation, the delta has been neutralised and is almost neutral, i.e., equal to 0.00. (Chen, Delta Neutral, 2020)

Now what would happen if the share price increased by 2.00 euros. We will calculate the profit or loss of our investor Y. First, he would pocket the difference of 4.00 euros on the share price ((112.00-110.00) * 2). However, the options he bought have lost value, they are now worth only 0.28 euros each. He has therefore lost an amount of 3.15 euros ((0.28 - 0.91) * 5) in total. If we add all the investments together, our investor still gains 0.85 euros (4.00 - 3.15).

And now let us imagine that the share price decreases by 2.00 euros. The loss on the shares would be 4.00 euros ((108.00-110.00) * 2). At the same time, these options would have won and would now be worth 2.14 each, i.e., a profit of 6.15 euros ((2.14-0.91) * 5). The total result of his strategy would then be a profit of 2.15 euros (6.15 - 4.00).

As it has been shown, this strategy is a winner regardless of whether the stock rises or falls if the trades have a neutral delta. (Forex OX, 2018) The key point of this strategy is the volatility of the asset. If the asset price rises or falls, the investor makes money. However, if the price does not change, the value of the option will decrease over time and the investor will lose money. This technique is based on a precise mathematical model that can be easily implemented in a trading algorithm. All you need to do is to have all the initial information to make all the calculations necessary for the decision to be made automatically.

2.5.5 Trading range (Mean Reversion)

This strategy is based on the principle that the ups and downs of the markets are only temporary states that sooner or later return to their average value.

Let us take the example of a stock that sees its market value rise sharply. When looking for the reason behind this price variation, our investor X finds no valid reason either on the stock itself, on the sector or on the market in general. He therefore decides to bet on a fall in the price in the coming weeks. This bet is a perfect illustration of the concept of this strategy, where a value continuously fluctuates by periodically passing through its average value. (Chen, Mean Reversion, 2021)

Larry Connors has developed a relatively simple analysis strategy that considers enough variables so that the signals generated allow him to take advantage of periods in which an asset is overbought or, on the contrary, oversold. (Davis, 2021) Its strategy does not identify

the beginning of a deviation from the average value but rather anticipates a rapprochement of this value. (StockCharts, 2021)

To master the strategy, it is necessary to know the use of the moving average (MA) as well as the relative strength index (RSI). We will therefore look in detail at the RSI. This indicator measures the variation of prices to evaluate the overbought or oversold conditions of an asset. (Fernando, Relative Strength Index (RSI), 2020) The RSI is a value between 0 and 100 that moves from one end of the range to the other depending on the asset's situation. If the value is greater than 70, then it is overbought, and conversely if the value is less than 30, then it is oversold.

Larry Connors' strategy uses 4 conditions to determine a buy signal and a sell signal. The buy signal is generated when the price is above the moving average over the last 200 periods and the RSI indicator is below 10, while the sell signal appears when the opposite occurs. (StockCharts, 2021) That is, when the price is below the moving average over the last 200 periods and the RSI indicator is above 90. The graphs (Figure 10, Figure 11) below illustrate the signals that can be obtained using this strategy.

Figure 10 : Chart with signals for the technique on Facebook assets (Modified screenshot)

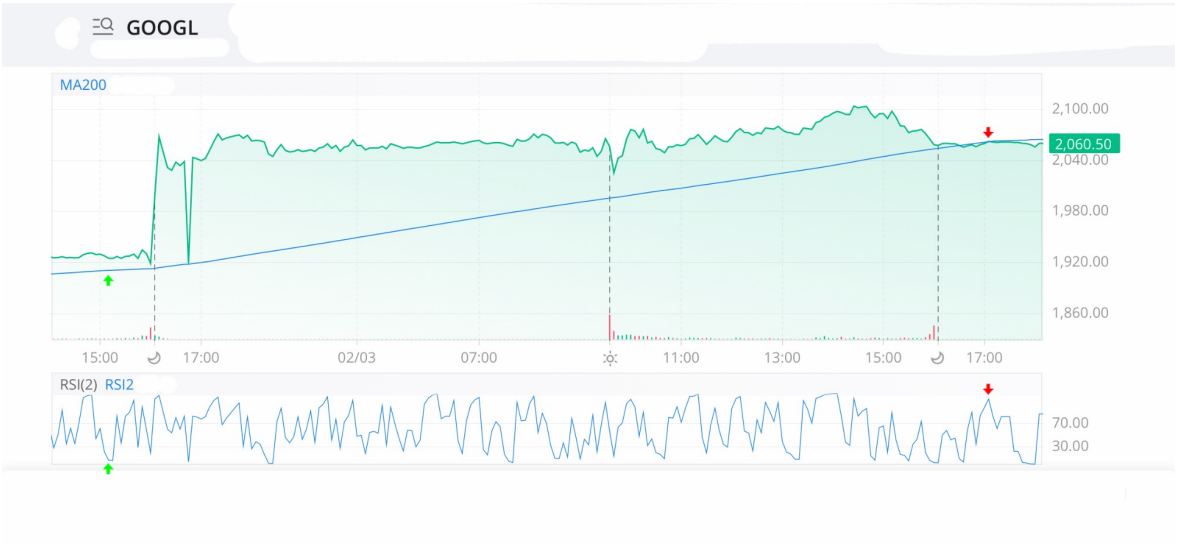


Figure 11 : Chart with signals for the technique on Google assets (Modified screenshot)

It is necessary to pay attention to the indicators used to determine opportunities in the stock markets but even more so with this strategy. If a stock is under-listed, then it seems reasonable to assume that its price will rise to become the new long-term average value. The same is true for a stock that is over-listed. It is necessary to know if events have occurred and if these are likely to modify the average value of the instrument soon. (WH Self Invest, 2021)

Apart from these exceptions, this strategy is relatively simple to implement in a trading algorithm. But it is possible to go even further by incorporating analysis of the perceived value of the asset, intrinsic value, market and sector events and news about the company. It should be noted that most strategies do not consider these elements, which are essential to correctly distinguish justifiable increases in the face of irrational behaviour linked to the psychology of market players. Instead of using these elements, it is also possible to consider the state of supply and demand to improve the strategy. (WH Self Invest, 2021)

2.6 Execution strategy discovery

In this section, we will look at the different techniques used to optimise the execution of transactions using algorithms. It is easier to optimise execution algorithms if the total amount of the order is large. In this way, the order can be fragmented to allow for smooth trading without market impact. (Seth, 2020) Execution algorithms can be complex depending on the purpose of the order. We will only see the most popular techniques, but the list is not exhaustive.

2.6.1 Volume-weighted average price (VWAP)

This strategy uses a specific technical indicator that is considered a trading benchmark.

To understand the strategy, it is therefore important to know this indicator. The Volume Weighted Average Price (VWAP) is used to determine the average price of an asset over a given period. It can be compared to the moving average but there is a big difference. We will try to understand this difference with an example.

To illustrate this comparison, we will use a more meaningful example (LiteForex, 2020). Let us take a market gardener who sells his vegetables on the market. He offers crates of 50 courgettes and each courgette costs 2.0 euros. The price of the basket is therefore 100.0 euros, and the average price is 2.0 euros. He now decides to include another variety of courgette in this crate. This new variety costs 3.0 euros and the crate is now composed of 50 courgettes at 2.0 euros and 10 courgettes at 3.0 euros. If we calculate the average price of a courgette, it will be 2.5 euros ((2.0+3.0)/2). Regardless of the number of each variety of courgette or the total number of courgettes in the crate, the average price does not consider the number or volume of each courgette in the crate. In this scenario the average price is the moving average indicator.

Now let us look at things differently, we will calculate the total price of the crate and then divide this number by the number of courgettes. With this formula we find an average equal to 2.17 euros ((50*2.0 + 10*3.0)/60 = 2.1666...). This average is more representative of reality, but we must not forget that it is a weighted average that takes volume into account. (LiteForex, 2020) In the formula, we use 3 times variables linked to volume, i.e., the number of each variety and the total of courgettes. This formula, which includes not only price but also volume, is the perfect example to introduce the new VWAP indicator.

Figure 12 : Diagram with basket of courgettes and average / weighted average price

In the same way that the indicator worked perfectly for courgettes, it also works very well for measuring a trend in the evolution of the price of a financial asset. On a graph, this indicator represents a simple line that oscillates up and down over time. It indicates a reference price used by many investors and traders to make decisions about the optimal time to execute an order.

This indicator has two main distinct uses. It can be used by investors who wish to buy or sell a position with a large volume. In this case, investors will try to open the position to buy when the price is lower than the indicator and will open the position to sell when the price is higher. By using this indicator, large volume investors can avoid having too strong an impact on the market. The aim is to restore the price to get closer to the indicator and avoid amplifying a possible gap already present. (Fernando, Relative Strength Index (RSI), 2020) For investors with less volume and for traders, this indicator can be used to confirm an identified trend. In this sense, it is an important tool to validate a decision already taken. For example, an investor will prefer to buy when the price is higher than the indicator but will prefer selling when the price is lower than the indicator value. (Fernando, Volume Weighted Average Price (VWAP) Definition, 2021)

However, it is important to bear in mind that this indicator is only valid for a single day. That is, it cannot be studied over a long period of time. Each day, it is necessary to calculate from zero and avoid including previous values in order not to distort market reality. (LiteForex, 2020) Another flaw is that it is based solely on historical values and therefore cannot predict future market events.

This strategy, although it has several problems, is still easy to understand to implement in a trading algorithm. Moreover, it can be a very effective tool in many situations.

2.6.2 Time weighted average price (TWAP)

This strategy differs from the previous one in that it does not consider volume but only time as an additional unit of measurement.

To understand how this strategy works, we must first understand how this indicator is calculated. First, the average price for each day is calculated. To perform this operation, it is sufficient to calculate the average of the opening price, the highest price, the lowest price, and the closing price for a single financial asset. Once this value is calculated for several days, it is again sufficient to calculate an average of all these values. This indicator is simpler to calculate than the VWAP since it does not consider volume but simply the time horizon as an additional variable. (Thakar, Time-Weighted Average Price (TWAP) in Financial Markets, 2020)

This indicator has many uses. To illustrate the main use (Thakar, Time-Weighted Average Price (TWAP) in Financial Markets, 2020), imagine an investor X who wants to buy 1'000 shares of a company, this corresponds to a very large volume. To do so, he consults the offer volumes on this stock. There are currently only 150 shares available at the Time Weighted Average Price that the investor has previously calculated. If he issues a buy order for 1,000 shares, he will unbalance the market and the price will rise. (Trading Tuitions, 2017) He absolutely wants to avoid this situation, so he decides to split his order into several orders of a smaller value. He will therefore be able to buy the 150 shares available now, then throughout the day, he will gradually send all the other orders up to the 1'000 shares he initially wanted to buy. At the end of the day, he will have managed to buy all the assets he wanted at the best price without having caused a price rise because his order was too big for the volume traded of this company's share.

This scenario clearly demonstrates the advantage of splitting an order. The investor can thus place a large order but still trade within the limit of the offer available on the market. This strategy can be perfectly used by a trading algorithm that will even be able to split orders into smaller portions to minimise the effects that such a large order can have. (Trading Tuitions, 2017) The algorithm can track the price and supply throughout the day, thus determining the ideal times to make a trade. (Thakar, Time-Weighted Average Price (TWAP) in Financial Markets, 2020) During this time, the investor will be free to choose his next investment.

Another advantage of this solution is the automatic price smoothing. (Trading Tuitions, 2017) If an investor wanted to place a medium sized order on the market, he would have to choose when or define a condition to trigger the order. In any case, he takes the risk that his execution price will not be the best if the price falls just after his investment. In the case of a split order, orders will be concluded throughout the day. This will have the effect of smoothing the average purchase price over the day. The algorithm will therefore generally be able to benefit from a lower average execution price than if it had decided to buy everything at the same time.

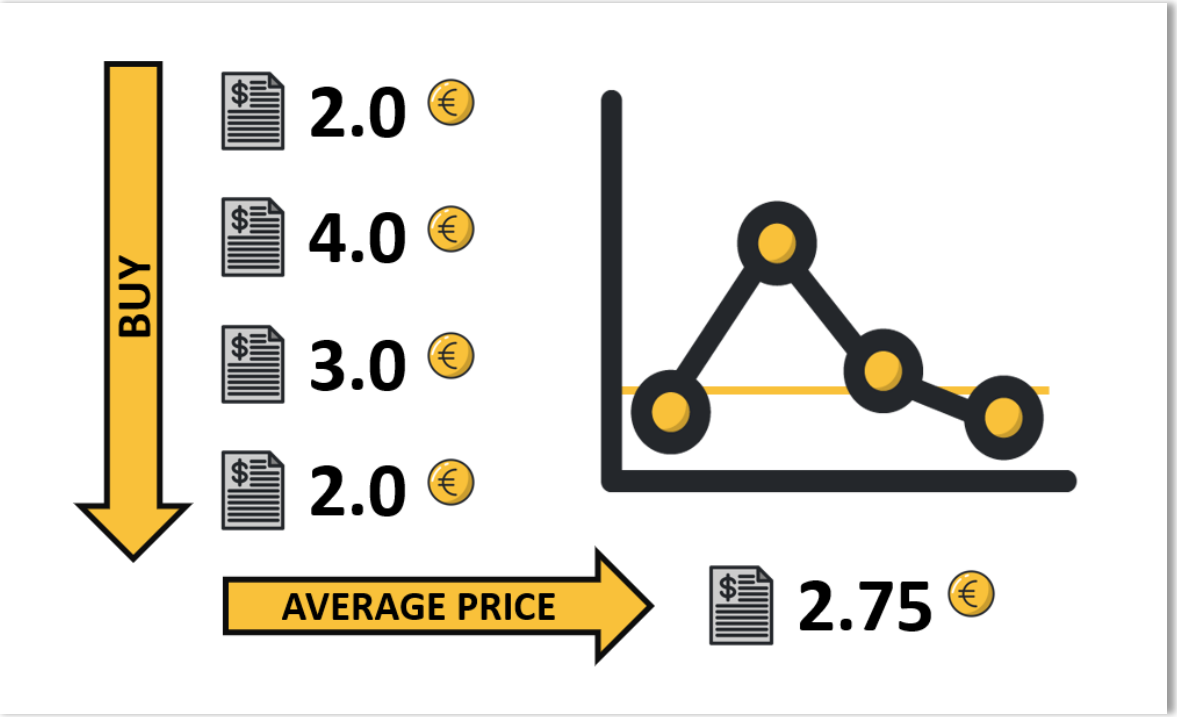


Figure 13 : Diagram with demonstration of price smoothing for multiple purchases

Even if this strategy is easy to understand and implement, it presents a concrete risk of spying. All market participants can see the orders placed. Therefore, if the algorithm always issues an order of the same value and always at the same time interval, other investors will be able to anticipate new orders and possibly use this information to take advantage of them. Generally speaking, and in the past, the signals of this indicator have proven to cover a large proportion of the risks. (Thakar, Time-Weighted Average Price (TWAP) in Financial Markets, 2020)

2.6.3 Percentage of volume (POV)

This strategy is used to control the execution of large market orders by targeting only a portion of the total volume available in each period.

To understand how this strategy works, we will take the example of our investor X who wants to invest in a financial asset. The average volume on the asset is around 20'000 securities per hour. Our investor wants to buy 100'000 shares of this company, but he knows that it is not possible to issue a single purchase order for 100'000 shares. He is indeed right; such a purchase order would change the current supply and demand situation and push prices upwards. Our investor would still like to carry out this important transaction but at a normal average price. The solution offered by this Percentage of Volume strategy allows him to carry out his transactions without having a visible impact on the market. (Blaze Portfolio, 2019)

The strategy works very simply as a function of volume. Let us say we divide the opening day of the stock market into several small fractions of time. It is now possible to calculate the average volume for these periods. It is then in relation to the volume of each period that the trading algorithm will decide the number of shares to buy. Thus, the algorithm uses a certain percentage to calculate the value of the order. (Bershova, 2012) (Blaze Portfolio, 2019) If the algorithm wants to operate in the most delicate way possible, it will use a small percentage to perform the calculations. Orders will thus be executed throughout the day or even during the week depending on the volume available.

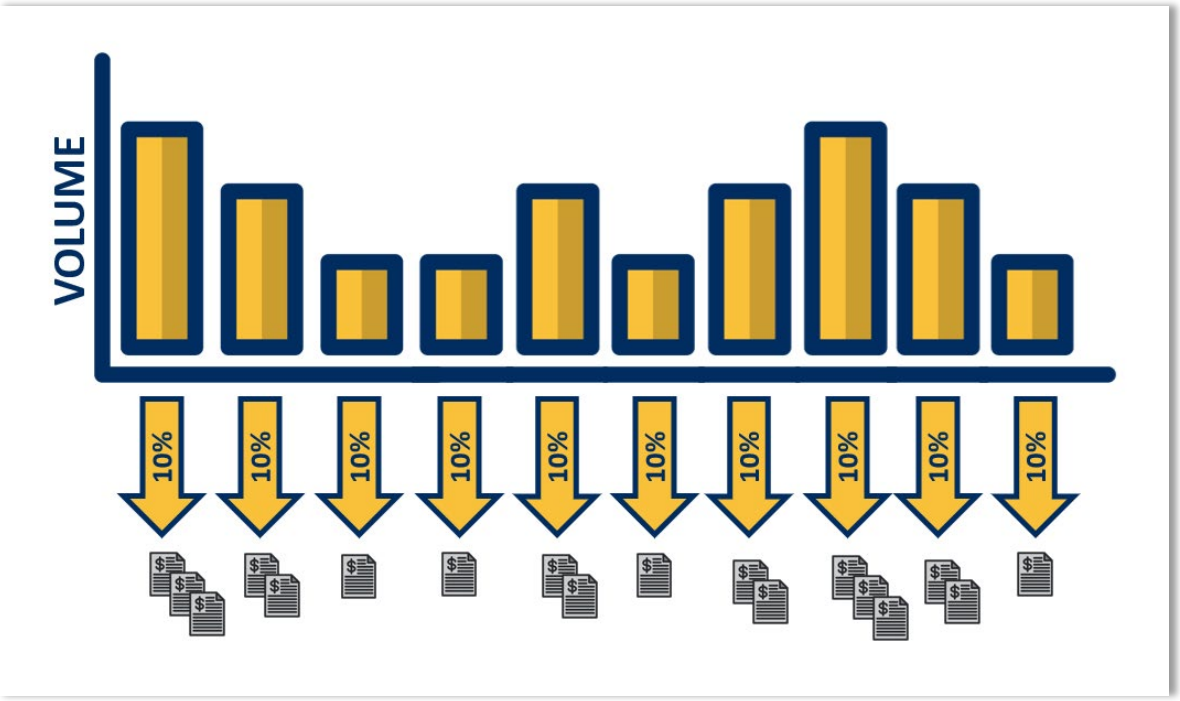


Figure 14 : Diagram with purchase corresponds to the volume traded on the market

The higher the percentage, the more aggressive the algorithm will be and the more likely it will have a visible impact on the market. If, on the contrary, it uses a low percentage, then it will be able to buy securities while following the normal evolution of volume. (Bershova, 2012)

The concrete advantages of this strategy are the absence of impact on the market as well as a certain price smoothing which could be beneficial in the medium/long term. (Blaze Portfolio, 2019)

This strategy is relatively simple to implement since it is only necessary to know the current volume of the stock. However, one must be aware that there may be small variations since the amount that the algorithm will allocate will depend on the volume multiplied by the percentage but that securities cannot be fragmented at the time of purchase. The number of securities purchased will be calculated by the algorithm, but the decimal part will be dropped.

2.6.4 Implementation shortfall

This strategy is mainly used to control the execution of transactions at the best market price when taking a position.

To clearly understand the concept, let us illustrate the example (Hayes, 2020) with a cook who wants to buy apples to put apple-based dishes on his menu. This person cannot go directly to the apple seller, so he sends the waiter to find out the price of the apples per kilo. On his return, the waiter announces the price to the cook. A kilo of apples costs 1.60 euros. The cook finds this price advantageous and decides to place an order for 100kg. To complete the transaction, the waiter must return to the seller. During the discussion between the waiter and the apple seller, the seller explains to him that he has just received a large order from a competitor and therefore no longer has the quantity needed to complete the transaction. He also explains that if he is prepared to spend 1.65 euros per kilo, then he will do business with him. The server accepts and carries out the transaction. The next day, the apples are delivered to the restaurant. The cook must pay the price of the order, the transport costs, and the packaging costs of the apples. In total, he will pay 1.74 euros per kilo.

In the scenario below, we see that the price the cook decides to deal with is different from the price he finally gets in the end, considering taxes. This difference is called an implementation shortfall. (Inc., 2005) It often occurs in scenarios like this or whenever there is an exchange of services. Commissions, fees, and taxes in addition to the price difference make up this slippage. (Hayes, 2020)

In all sales scenarios, this difference can be visible, and it is also the case when we look at trading. When an investor X decides to buy a financial asset, he should consider that the

market is still likely to move slightly until the order is finally executed. He should also consider that commissions, fees, or taxes may apply. The difference between the price at the time of the decision and the price at the time of execution constitutes the possible slippage. (FX Algo News, 2021)

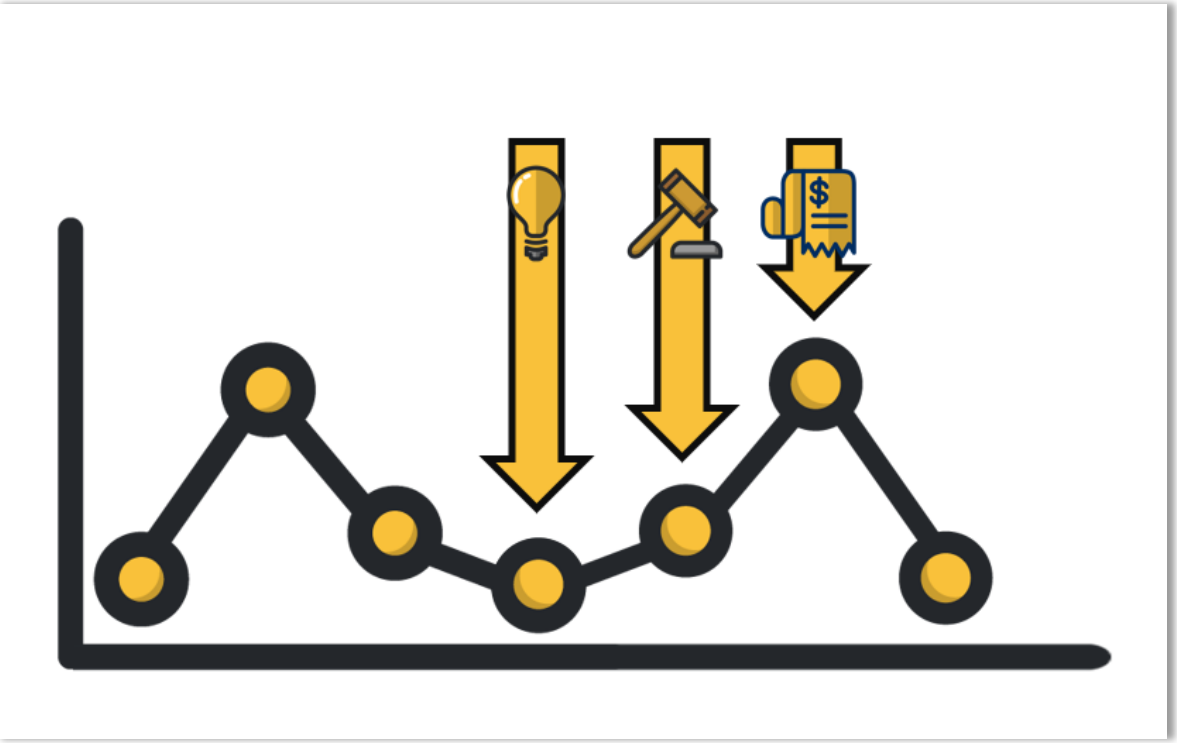


Figure 15 : Graphic to demonstrate the shortfall implementation process

An algorithm that buys on the markets is also subject to this constraint. Although it is faster than a human, prices are likely to move very quickly and therefore a slippage should never be excluded. To counter this phenomenon, it is possible to use buy orders with a limit. (United States government, 2021) This means that the order will be executed but only at a price below the limit. With an order of this type, the algorithm is certain not to buy at a price higher than the price it has itself defined. While this technique has a great advantage at first sight, we should also think that the order could never be executed if the market price never goes below the limit before the order expires. In the latter scenario, this is an investment opportunity that would have been missed by the algorithm because of an overly rigid pricing policy. (Hayes, 2020)

If, contrary to the previous execution, the algorithm does not define a price. The order will be executed at the time of reception on the market at the market price as it stands, this type of order is called a market order. (United States government, 2021) In this scenario the algorithm is certain to conclude the transaction but not necessarily at the same price as when it made the investment decision.

The same problems arise when one wants to sell assets. Both types of orders are possible. However, finding oneself with undesirable assets because of a sell limit order that has never been executed is much riskier than missing an opportunity. For this reason, it is rather advisable to use a market order to quickly sell a position at any price in case of strong price fluctuations. Sell limit orders should only be used with great caution and in rather stable market conditions. (Hayes, 2020)

To implement this strategy in an algorithm, it is necessary to choose between the price difference one is prepared to accept between the time of the decision and the time of purchase or possibly accept the fact that the order might not be executed. It is possible to let the algorithm itself make the least risky decision by using the volatility of the asset as an indicator as well.

2.6.5 Unusual trading strategies

Beyond the popular strategies we have already mentioned, there are a multitude of other possibilities.

Let us take the example of Investor X who makes several trades a day but fails to become profitable. On the other hand, there is Investor Y who consistently wins in the markets. Investor X begins to get jealous and decides to spy on the other investor's trades. He extracts the other investor's strategies from the data he has collected. Once the strategies are in place, he starts investing using this new strategy and very quickly becomes profitable.

While this may seem like intellectual property theft, it is not. The truth is that all market players have access to the order book. (Markman, 2021) This order book is a summary of all the orders sent to the market. Thanks to this order book, which is public, it is impossible to know who carried out the transactions, but it is possible to see all the transactions anonymously. (Mathur, 2019)

Figure 16 : Scenario with an algo sniffing to copy the opponent's strategy

A trading algorithm with a standard strategy, as we have seen above, executes orders based on market data that is accessible to everyone. It is therefore possible, using this same data and the order book, to determine whether market participants always make the same decisions in the face of the same market situations or whether an order appears periodically. Algorithms that try to find out what is happening on the other side of the market are called sniffing algorithms.

These algorithms are very complex, but if they are well trained, they can guess very accurately the strategies of other players in the market. (Markman, 2021) They generally work with built-in intelligence or machine learning that allows them to identify upcoming trades based on previous trades. (Mathur, 2019)

If the strategy is known in advance, then the basic algorithm can become vulnerable. Let us imagine that an algorithm X executes orders for 1000 shares every hour. If algorithm Y finds the strategy, which a human could also easily do with such a simple strategy, it can influence the share price just before the arrival of the competitor's order. In this way, algorithm Y will make a profit by using the strategy of algorithm X against him.

An algorithm that discovers a competitor's strategy will have an advantage even if it does not try to make a profit against him. For example, it can predict future price movements based on orders that will be placed in the future. In this way, he will have access to a crystal

ball that will allow him to know when to invest and when to withdraw. He can also copy his competitor's strategy if he realises that all previous trades were winning. (Markman, 2021)

It is important to know that strategies like this exist. Being predictable makes you vulnerable. This is the case in many situations, but it is an even bigger problem in a stock market populated by greedy people. These algorithms are not easy to set up, they must be smart enough to detect adverse strategies but also brilliant enough to know how to turn this advantage into concrete financial gain.

3 Development plan

The work that will be carried out in this thesis will be the realization of an algorithmic trading software. The development of this software will allow the application of the knowledge and skills acquired during the research work. The previous theoretical part of this chapter is important for a good understanding of the next part. Therefore, it is strongly recommended to have a good understanding of the theoretical part before starting this practical part. If any of the concepts covered in the development are unclear, it is advisable to refer to the explanations given in the previous section.

The development will serve as a demonstration. As mentioned earlier, this work will not be used for profit but only for educational purposes. A fictitious trading account will be used to avoid financial risks. Before setting up a system like this one, it is relevant to draw the attention of all readers to the risks[1] of this activity. Furthermore, all data and information used will be available free of charge and freely available to allow others to duplicate this software without any initial investment. Persons who wish to do so are advised to use a fictitious trading account as well. Trading is a professional activity that requires specific knowledge just as other jobs require it.

All the parts required to set up the software will be studied in the following chapters. It should be noted that the graphical part will not be treated as a chapter, but its development, at least the important parts, will be discussed in the context that corresponds to the use of the graphical interface. Initially, the architecture that has been put in place will be discussed and illustrated in concrete terms with several diagrams that will allow the logical flow of a transaction to be understood. After this phase, the preparation of the environment will be discussed. The necessary hardware resources will not be too high. The aim is that it should be possible to run the software on a standard laptop with an average internet connection.

---

[1] « All data and information contained in this work is for information purposes only and does not constitute investment advice or service, solicitation, offer or recommendation to buy, sell or otherwise deal in any of the assets mentioned throughout this thesis.

All information in this document is without any guarantee as to its accuracy or completeness. The authors, appraisers or all persons involved in the preparation of this thesis cannot be held liable in any way whatsoever for any direct or indirect loss or damage arising from the use of this information, nor for any risks associated with the malfunctioning of the software or the presence of viruses.

The attention of the investor is expressly drawn to the fact that an investment in the financial assets mentioned in this work presents risks, as the investments made may go up or down. Past performance is no guarantee of future performance. The performance shown does not take into account commissions and fees charged on individual transactions. » Adapted from (Investissements Fonciers S.A., 2021)

The methods of accessing the necessary data will then be explained. Market data, i.e., live quotes for financial instruments will be obtained through the broker with whom a fictitious trading account has been created. Technical analyses will be retrieved with the help of a third-party REST API available free of charge. All this data will then be processed and converted before being made available to the other components of the algorithmic trading system.

Certainly, the most important part, and the one that makes all the difference between a system that makes profits and one that does not, is the implementation of the strategy. This is the brain of the software. This component will use the data extracted in the previous step to determine whether a transaction should be made. In the algorithmic trading industry, it is usually quantitative analysts or mathematicians who establish the strategies. Without access to such skills, a simple strategy will be used to allow the system to make decisions automatically and without any human assistance.

If the system that enables decisions to be made is important, it is because the strategy it contains is crucial. To ensure that the strategy in place does not fail, it will be tested on historical data. Unfortunately, it will not be possible to carry out this simulation directly with the broker who only offers this service in return for a fee. However, to have an estimate of the profitability expected with this strategy, it will be tested on an online platform, specialised and free of charge. The strategy that will be simulated during this test will be the same one that will be implemented within the algorithmic trading software. Thanks to the simulation on historical data, it will be possible to get an idea of the frequency of transactions and past profitability. However, it is important to keep in mind that the simulation on historical data is by no means a guarantee for the future growth of the financial markets.

Once the strategy has been set up and verified that it could work, the components that allow transactions to be created and transmitted directly to the broker will be implemented. The automatic execution of all the decisions taken by the strategy will be dissected and the risk management system will be studied and put in place since this will be the last time the data will be checked before being definitively transmitted for execution.

When all these steps have been completed, the system should be functional and should be able to execute transactions without any assistance. It will then be time to see what the software's actual capabilities are and to prove that all the elements put in place throughout this development work together in the expected way. Once proof of operation has been obtained, the evaluation part will allow the results and capabilities of the software to be discussed in more detail.

The final objective of this software will be to be able to carry out transactions instead of a human. The speed of the analysis will be evaluated in comparison with a person performing the same operations. However, the evaluation of speed is complex as it is highly dependent on the available hardware resources and the speed of network access. Finally, performance will be evaluated. While considering that this activity generally requires the involvement of a quantitative analyst or mathematician, the objective will be to achieve a positive performance.

4 System development part

In this development part, we will see in detail how to build a working algorithmic trading software. From the architecture of this software, through the backtesting of the strategy that has been implemented to the proof of operation, all the steps will be explained so that they can be reproduced by the reader. The initial set-up, obtaining the data, implementing the strategy, and automatically executing the trades will be the focus of development throughout this chapter. Parts of the code will be repeated in this document to facilitate the understanding and assimilation of the software. As for the software and the source code, it will be available to everyone on GitHub with an MIT license.

4.1 Software overview

Before we start the actual development, we will look at the important elements of the software that we will implement in the rest of this report. All the software will be built from scratch, our goal will be to obtain an engine capable of following a predefined strategy to place orders on the financial markets.

First, we will look at the graphical framework that has been laid down for the development. We wanted to have something that was uniform and elegant throughout the user's use of it. This graphic framework that we came up with defines elements such as colours, fonts and little things that give character and quality to our software. Once we had this framework, we even went so far as to imagine a logo that corresponds to our software. This emblematic shark was chosen because it is an animal that is fast, knows how to spot its prey from a great distance and is above all capable of biting hard with its powerful jaw when it sees an opportunity. These characteristics will also be the ones we can expect to find in the software we are going to develop. As for the title that has been chosen, even if one might think that it refers to a bull market, it is simply a tribute to the American film of the same name released in 1968.

Another important element that we have emphasised and will emphasise again in this report is the desire to use only free tools. Companies that use algorithmic trading software, automatic trading, or even high-frequency trading, spend huge amounts of money to create systems that allow them to make more and more profit. We wanted to prove that it is not necessary to invest any money to build a system that allows us to place orders automatically on the financial markets.

The user will be an important factor in this software. He will have the possibility to configure the strategy directly through an interface that we will set up for that. He will have the possibility to change the instrument to be monitored but also to slightly modify the behaviour of the strategy. These minor modifications include the entry signals, the sale signals, and the type of order to be used. In addition to this, all the information needed to follow the evolution of the portfolio will be provided to the user through the graphical interface that will be developed. The user will therefore be free to follow the evolution of his algorithm in real time. He will be able to visualise information specific to the application of the strategy by the algorithm, follow the evolution of his demo account and all the transactions carried out will be displayed on the screen without delay.

The user will have to make sure that all the parameters correspond to these expectations and then he will be free to launch the algorithm to start monitoring the selected instrument. If any signals appear then they will be immediately picked up by the software which will perform all the necessary operations to take advantage of them. The user will not need to perform any manual approval or other operations once the system is started. Upon shutdown and to avoid being left with positions in the account, all unexecuted orders will be cancelled, and open positions will be immediately closed at the market price. A clearly visible stop button is always available so that the user can quickly shut down the algorithm in case of sudden market movements or malfunctions.

In the following chapters, we will see how to carry out all the steps to create our own software. This work can be taken up by any reader for further study or research on the subject. Although we would prefer to keep this work within a didactic framework, this development is free to be used by anyone, anywhere, anytime.

4.2 Software architecture

To summarise the architecture of the software that we are going to use for the implementation of the software, it is sufficient to look at figure 4 of the theoretical part about the architecture. But to better understand why this architecture is efficient and effective, we must observe the flow of operations that take place successively in such a system. We will start by defining the tasks that will be performed by the software in detail.

First, we will have to connect our software to a broker or rather to the broker's API services. The software will need to be able to communicate to place orders or obtain information that will later allow it to place those orders. For this purpose, we will need a package to

automatically manage the connection and communication between the two systems. We will call this package "ConnectionHandling".

Next, we will need all the elements that will allow us to process the data, analyse the data, decide, and then execute that decision. These are the 3 components that can be seen in figure 5 previously discussed in the theoretical part. These 3 packages will therefore be added to our architecture. The "MarketDataHandling" package will process the incoming information in relation to the instrument, sorting it before passing it to another independent process. Then the "EventProcessingHandling" package will receive all the necessary information from the previous one and will analyse the prices, variations, and trends to eventually find a buy or sell signal. Finally, the "ExecutionHandling" package will allow our system to be autonomous in the execution of transactions. It will be responsible for establishing buy or sell orders and transmitting them directly to the stock market. By the word directly, we are ignoring the communication packet, but it is very clear that all incoming and outgoing communications will pass through this packet before being transmitted or received.

To make sure our system does not have a short memory, we have even given it a "StorageHandling" packet. Even if the name is explicit, we will still look at its utility. It will mainly oversee saving received information, keeping track of transactions, and allowing our system to display the data on the screen. Any parameter changes or data will first pass through this package before being used by any other component. It will act as a separator between the trading algorithm and the user interface. Finally, and as we expected, a "UserInterface" package will allow us to interact with the software. From modifying the strategy to displaying the data, it is this package that will be responsible for presenting the data and allowing us to manipulate the engine. This package will be the one and only way for the user to interact with the components.

Now that we know the 6 packages that are needed for development, we can easily implement the architecture that will be used by our software as shown in the diagram below.

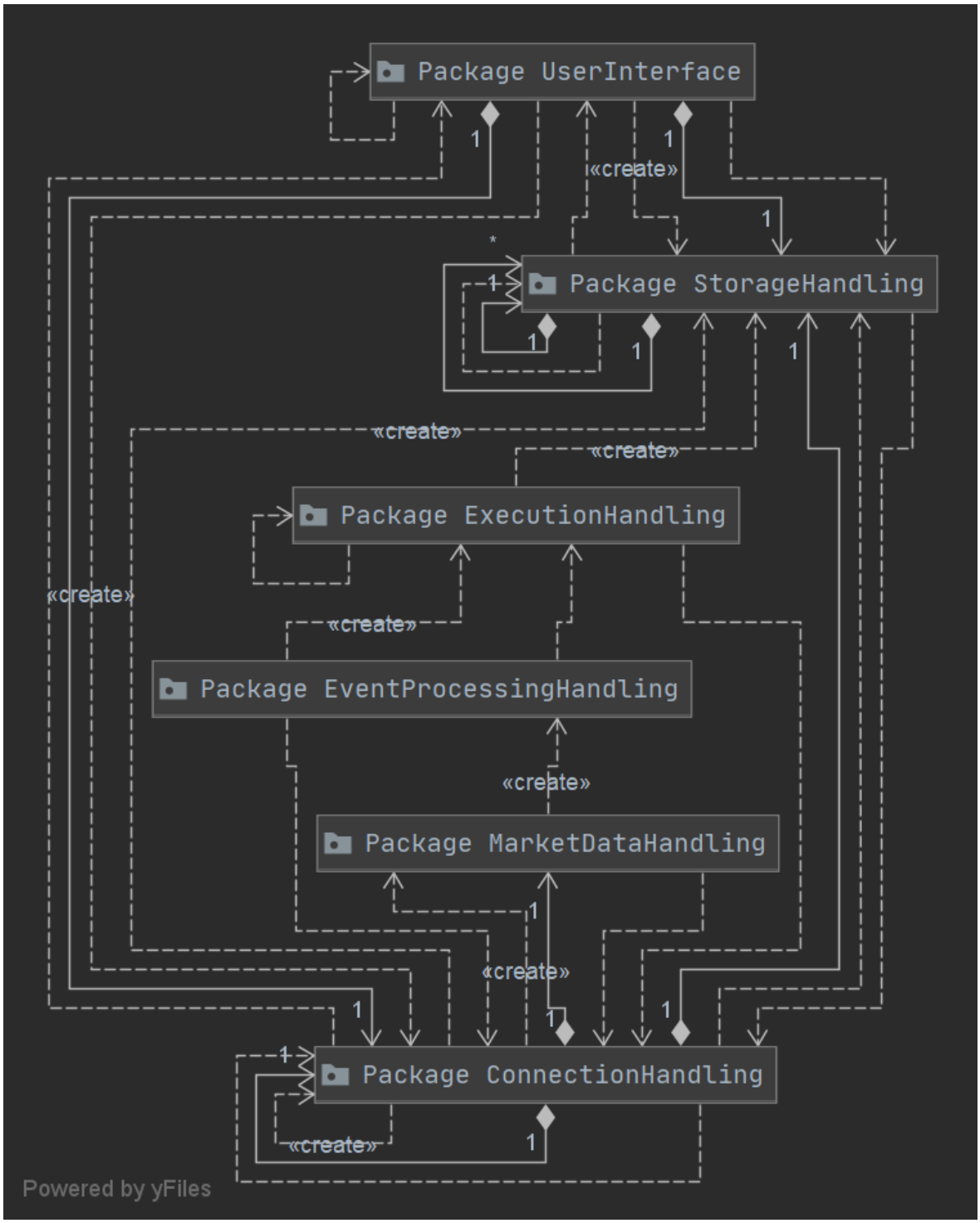


Figure 17 : Diagram of all packages (Modified screenshot)

If the previous explanation is not clear, it is strongly recommended to read again the theoretical part concerning the architecture used in this kind of software. In the following, we will study each package independently to understand the work that has been done on each of these parts. Note that the architecture and organisation of the "MarketDataHandling", "EventProcessingHandling" and "ExecutionHandling" packages will be studied in the chapters dedicated to them in this document.

As a reminder, the "ConnectionHandling" package allows a fluid communication between the software and the Rest API service provided by our broker. To simplify understanding, we will use an analogy. Just as we might find in a human, we will need ears to listen, a mouth to speak and a brain to manage the exchanges. Each component is distinct in our architecture. The "InputAdapterIB" component, for example, listens for incoming streams and distributes them correctly to other components, most often to storage. The "OutputAdapterIB" component is comparable to the mouth since it is responsible for forwarding the requests to the broker. It mainly receives information from the brain and then communicates directly with the Rest API. The last but by far the most important component is the brain. In our software, the brain corresponds to the "TwsIB" component. It is the brain that gives the order to start or stop a connection and is responsible for always keeping an eye on the mouth. In the following figure, the contents of this package are detailed to illustrate the explanations.

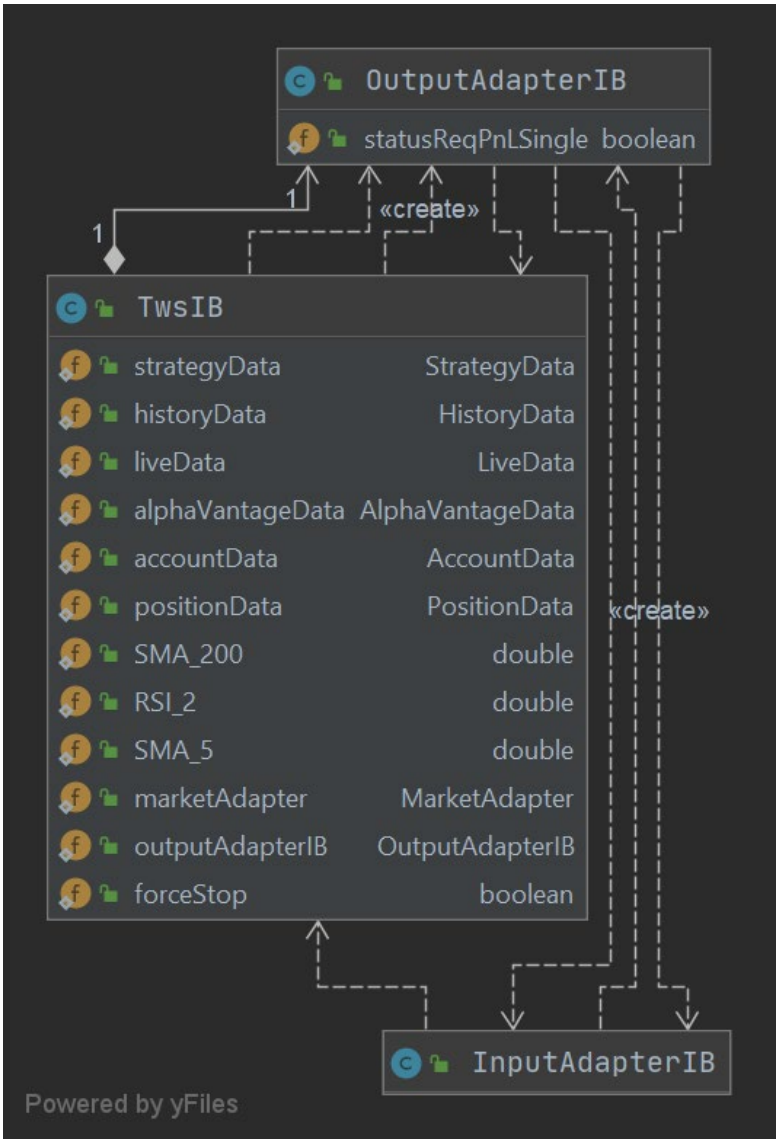


Figure 18 : Diagram of the package "ConnectionHandling"

(Modified screenshot)

To continue, we need to detail the contents of the "StorageHandling" package. This package is mainly responsible for making information available to all other components. All components that have the address of these components can modify or obtain data according to their needs. This package is built around a "Subject" interface. This interface is a component of something much larger as it represents part of the observer pattern that was introduced to simplify data exchange. - For those who are a little confused about this programming pattern, it allows an Observer to subscribe to a Subject to receive notification of changes to the underlying data and thus update the data without the need to make unnecessary periodic calls. - Each class that represents data offers the possibility to other components to subscribe, modify and obtain the same data. These classes are easily identified by their implementation of the Subject interface. Among all these classes, some use custom objects to facilitate data administration, but they work on the same principle as all the others. The following figure provides a schematic of an explanation to better understand it.

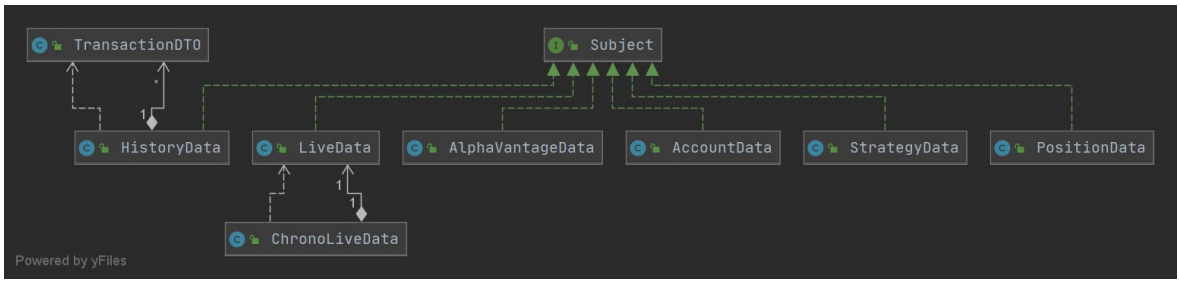


Figure 19 : Diagram of the package "EventProcessingHandling" (Modified screenshot)

Finally, we will discuss the "UserInterface" package. Even if this package only provides access to the background of our software, it is still important and will be briefly discussed. As in any architecture or software using Java Swing, there is a Jframe and Jpanels. In our architecture the "JFrameBTWS" component is the window or space that will host all the other screens. No graphic component comes directly from this class, which only integrates the panels and transmits the important information. Since we need to dress up our window, a background screen has been specially designed for this purpose. The "BackgroundScreen" component facilitates the integration of the navigation into the application since it must always remain static and available. In addition to these navigation options, it also offers window management with various "...Action" components that it maintains.

While these two components allow us to set the window and navigation, the screens that we see as a user are still missing. 8 different screens are used to manage the display. They are controlled by the main Jframe and are easily identified by their notation "...Screen900X" where the X is the screen number. These screens all inherit the "AbstractScreen"

component, which is used to gather important but common elements. And most of them implement the "Observer" interface which is part of the programming pattern we saw earlier. In effect, this means that they can be aware of changes in the storage components and update themselves.

The "Component" and "STATIC" packages will not be discussed as they only serve to group together the tools or fixed values that are used by most screens. They allow a centralisation of static data as well as a grouping of objects used for display. The following figures perfectly illustrate the user interface architecture that we have just explained.

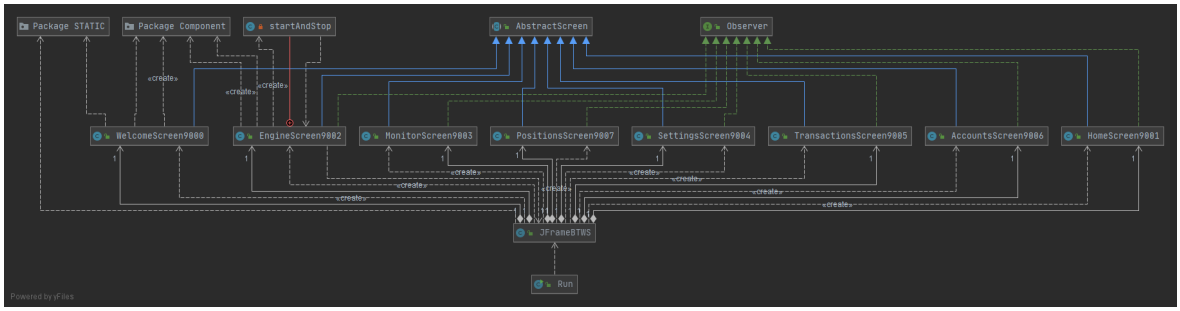


Figure 20 : Diagram of the package "UserInterface" Part 1 (Modified screenshot)

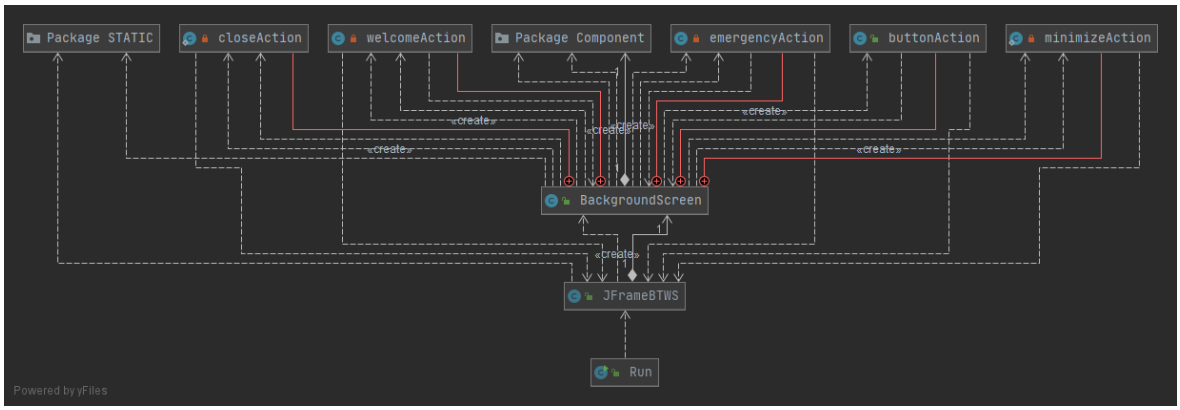


Figure 21 : Diagram of the package "UserInterface" Part 2 (Modified screenshot)

Now that we have seen the outline of the architecture we will be using, it is time to move on to the initial configuration part to set up our working environment and create all the dependencies we will need later.

4.3 Initial setup

We can now move on to the initial configuration of the working environment. Before we begin it is important to keep in mind that this setup has been designed to use only free data or tools that are generally available to most readers. In this way, a no-cost replication of this project setup is possible for almost anyone without prior investment.

To start with, we selected a broker, i.e., a broker to place our orders and get the price of the instruments in real time - In our search, we were careful to get a free tool, a reliable service, Rest API access but also a good documentation. At the end of the search, it was agreed that the Interactive Brokers platform met all the points that were important. - We will therefore need to download the trading workstation of this broker to run it on our laptop. To do this, you will find the installation executable at this link: https://www.interactivebrokers.com/en/index.php?f=16040. Once the application is installed, we need to create a demo account (Paper Trading). This usually only requires an email address to log in and get started. This will give us access to a trading environment with a large amount of money without taking any real risk as all transactions will be made with this demo account.

Once the demo account is created and connected, we will have to activate the settings to be able to communicate with the trading workstation from our software. To do this we need to go to the application > In the "File" tab on the top left > Select "Global configuration" from the drop-down list > Select "API" > Select "Settings". We should now be on the settings that allow us to manage the Rest API settings. On this page we change the values to match the following figure and then apply and save the changes before exiting this window. When running our software, this application must be open on the desktop of the computer and in connected mode.

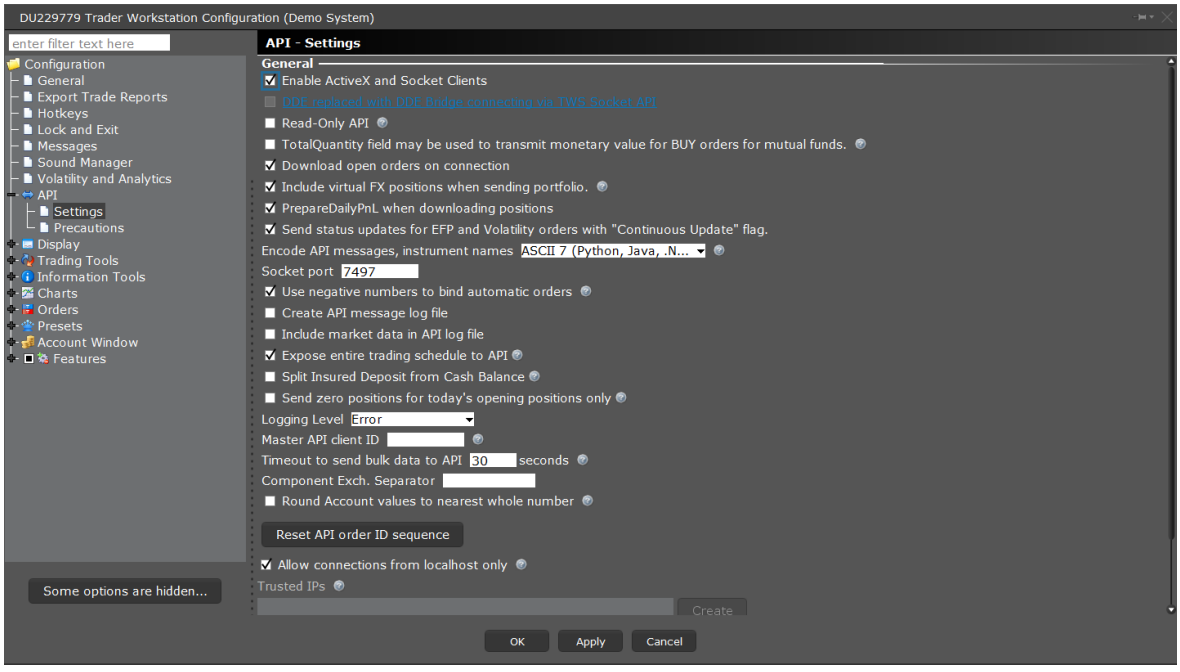


Figure 22 : Screenshot of the trading workstation screen showing the API settings (Modified screenshot)

The application we have just installed allows us to obtain a lot of information, but the demonstration mode is a bit restrictive. Therefore, and to stay in a free environment, we will use another Rest API for all the technical analysis we will need later. These technical indicators will mainly be used to determine buy and sell signals for our instruments. Alpha Vantage offers a reliable and free service that we will use for this. An API key must be created before we can start. To do this, we go to https://www.alphavantage.co/support/#api-key and create an API key for free. Once this is done, we make sure to save it in a safe place. We will not use it until later. The API key we have just obtained only allows for 5 requests per minute and 500 per day but it is sufficient for the strategy model we have chosen.

Now that we have all the third-party elements back, we can start setting up our IDE (Integrated development environment). We will use IntelliJ for this project, but you are free to use any IDE if you know the ropes. For the details, we have centralised our dependencies by creating our project with Maven and we are using Java 8. These features are popular and should easily be implemented by most readers with some Java knowledge. Before we formally start coding, we still need to install some dependencies in our "pom.xml" file. We will need the Interactive Brokers dependency, a JSON adapter and JUnit to perform basic unit tests.

To start, get the "TwsApi.jar" file either from http://interactivebrokers.github.io/# (Preliminary installation must be done) or from the repository of this project (Faster and easier). We will use version 9.76 of the dependency that we took care to deposit inside our project. We will therefore update our "pom.xml" file by integrating the Interactive Brokers API but also by integrating the "json-simple" and "junit-jupiter-engine" dependencies which will be downloaded automatically. Once the file has been updated, do not forget to synchronise it so that all the changes are considered by our IDE. The "pom.xml" file should look like the following if we have followed the previous instructions correctly.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>BullittTWS</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
```

```
                <configuration>
                    <source>8</source>
                    <target>8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

    <dependencies>

        <dependency>
            <groupId>com.ib</groupId>
            <artifactId>interactive-brokers-api</artifactId>
            <version>9.76</version>
            <scope>system</scope>
            <systemPath>${project.basedir}/IB/TwsApi.jar</systemPath>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.googlecode.json-
simple/json-simple -->
        <dependency>
            <groupId>com.googlecode.json-simple</groupId>
            <artifactId>json-simple</artifactId>
            <version>1.1.1</version>
        </dependency>

        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.5.2</version>
            <scope>test</scope>
        </dependency>

    </dependencies>
</project>
```

The initial setup is rather easy, and we have done all the necessary operations for a quick start. In the following parts we will dive straight into the code and see how the components we have already seen in the architecture part work. We will use everything that has been retrieved in this part but also the implementation that has been done. Therefore, it is strongly recommended that you have done the previous steps before continuing reading. If any operations have not been performed successfully, it is possible that the system simply cannot be started.

4.4 Market data stream

In this section we will try to obtain the market data that we will use later. These data are more exactly the real time price of the instrument we are monitoring and the latest technical analysis available for the same instrument. As already explained earlier, we have chosen to keep things free rather than simple. Therefore, we will get the technical analysis from another provider than our broker. To briefly summarise the situation, the real prices will be obtained from Interactive Brokers and the technical analysis from AlphaVantage. It should

be noted that calculating this data ourselves would have been possible. However, these calculations would have required too much preparation time when launching our algorithm. For this reason, the data is extracted directly from third party suppliers.

To begin with, we will need to know all the components that will be used in the data retrieval step. The "MarketAdapter" component will be used for real-time price management while the "AdapterManager" component will manage the technical analyses. More specifically, it will contain the "AdapterSMA" and "AdapterRSI" tools that will be needed to obtain the data. Its main role is to manage the queries and to update the data according to the results obtained. To better illustrate the architecture of the "MarketDataHandling" package, the following figure is a representation of its contents.

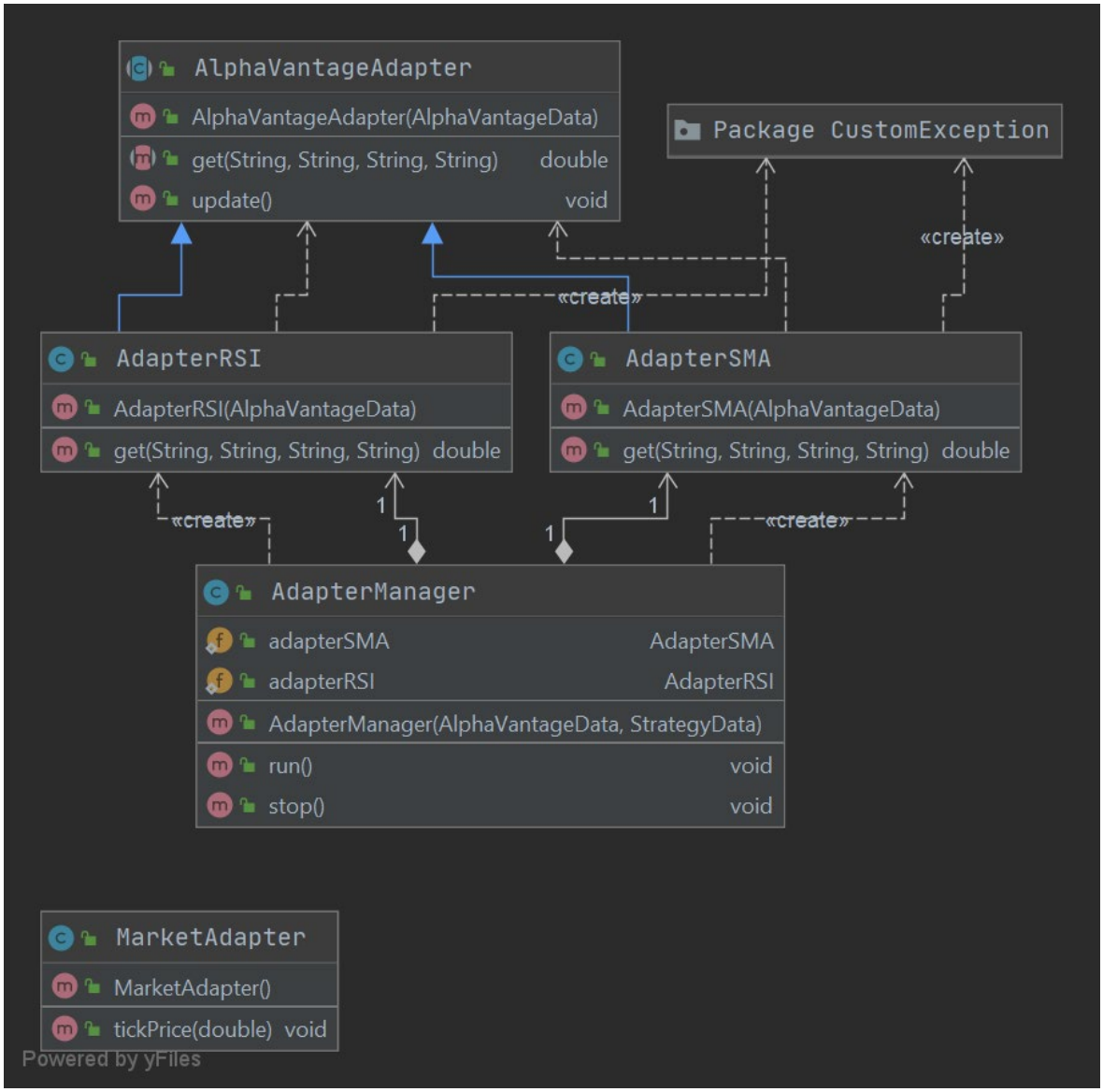


Figure 23 : Diagram of the package "MarketDataHandling" (Modified screenshot)

Now that we have an overview of the organisation of our package, we will see how the operations to obtain this data are carried out. First, we will retrieve the real time prices. To do this, we need to make a request to our broker first. The onReqMktData() method allows us to give it the necessary instructions and the instrument we want to monitor so that it can start sending us the data in real time. This instruction is issued by the "ConnectionHandling" package.

```
public void onReqMktData() {

    if (idOnReqMktData != 0){
        onCancelMktData();
    }

    idOnReqMktData = TwsIB.getNextValidID();
    m_client.reqMktData(idOnReqMktData, initializeContract(), "221",
false, false, new ArrayList<>());
}
```

Once we have made this request, we have to wait for our provider to start sending us the necessary information. As a reminder, we mentioned the ears and mouth as part of the “ConnectionHandling” package. Now that we have asked a question, we need to listen to the answer. The tickPrice() method of our API communication package will do this.

```
@Override
public void tickPrice(int tickerId, int field, double price, TickAttrib
attribs) {
    if (field == 2){ // askPrice
        TwsIB.marketAdapter.tickPrice(price);
    }
}
```

This first method listens to all the information transmitted in relation to the price, filters the useful data and transmits it to our "MarketAdapter" component. This new component has a method that allows it to receive each price transmitted by the API and then decide what to do with it. To clarify the situation, our component will look to see if the price is valid and then will request an analysis at each time interval that has been selected by the user. It is the tickPrice() method of this component that is used to perform these operations.

```
public void tickPrice(double price) {
    /* RETURN IF PRICE IS WRONG */
    if (price == -1.0){
        return;
    }

    /* FORWARD PRICE ACCORDING TO TIMESCALE */
    long currentMS = System.currentTimeMillis();
    if ((currentMS - referenceMS) >= delayMS){
        referenceMS = currentMS;
        System.out.println("> ANALYSIS DONE AT "+new
```

```
Time(System.currentTimeMillis()).toString());
        new StrategyEngine(price); //Will check for signal
    }
}
```

Once the analysis is requested, our "EventProcessingHandling" package will take care of the rest. We will see this package in the next chapter but before that we still need to retrieve the technical analyses that will be used by this package when transmitting the price. In this new task to be performed the "AdapterManager" component will be the one responsible. Its mission is rather simple, it will obtain the instrument that the user wants to be monitored, obtain new data, and then update this data so that it can be used by the other packages. This workflow will be executed every minute until the system is completely shut down to ensure that the data is always up to date. The time interval of 60 seconds is not a setting, but it is the interval that our supplier proposes. Therefore, there is no point in doing this every 15 second if the data only changes every 60 seconds. The code for this method will not be illustrated in this document but can be viewed online.

The method that is used by the "AdapterManager" component to update the data is in its dependencies. It has one dependency. Each dependency is a component specialised in retrieving a technical indicator. As we will only be using two different technical indicators for our analysis, we only need 2 specialised components to retrieve the SMA and RSI. If these indicators are not clear, it is strongly recommended to review the theoretical part that deals with this subject. Each component in charge of retrieving an indicator inherits from the "AlphaVantageAdapter" component which defines the API key to use and the get() method to implement. The latter is the skeleton that allows the retrieval of the indicator.

```
/* ABSTRACT MAIN METHOD - GET TECHNICAL INDICATOR */
public abstract double get (String symbol, String interval, String
time_period, String series_type) throws MarketClosedException,
OverloadApiUseException, NoNetworkException, MissingApiKeyException;
```

Whether it is the "AdapterSMA" or "AdapterRSI" component, both implement the same skeleton, but each perform a query to obtain a distinct indicator. Taking the SMA indicator as an example, this is retrieved by implementing the get() method within the corresponding component, i.e. "AdapterSMA". This method checks that the API key provided by the user is valid, performs a web request, receives a result in JSON format, extracts the most recent value and then simply returns this result. To better illustrate this process, here is the code for this method.

```
/* GET Simple Moving Average Indicator */
public double get(String symbol, String interval, String time_period,
String series_type) throws MarketClosedException,
```

```
OverloadApiUseException, MissingApiKeyException, NoNetworkException {

    if (API_KEY == null || API_KEY.length() != 16){
        throw new MissingApiKeyException(); // APIs KEY IS NOT AVAILABLE
OR WRONG
    }

    try {
        /* REQUEST DATA */
        URL url = new
URL("https://www.alphavantage.co/query?function=SMA&symbol=" + symbol +
"&interval=" + interval + "&time_period=" + time_period + "&series_type="
+ series_type + "&apikey=" + API_KEY);
        HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
        conn.setRequestMethod("GET");
        conn.connect();

        int responsecode = conn.getResponseCode();

        if (responsecode != 200)
            throw new RuntimeException("HttpResponseCode: " +
responsecode);
        else {
            Scanner sc = new Scanner(url.openStream());
            StringBuilder inline = new StringBuilder();
            while (sc.hasNext()) {
                inline.append(sc.nextLine());
            }
            sc.close();
            conn.disconnect();

            JSONParser jsonParser = new JSONParser();

            try {
                Object obj = jsonParser.parse(inline.toString());

                DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
                LocalDateTime localDate = null;
                double SMA = 0.0;

                JSONObject skr = (JSONObject) obj;
                Map analysis = ((Map) skr.get("Technical Analysis:
SMA"));

                /* FIND MORE RECENT ANALYSIS */
                for (Map.Entry pair : (Iterable<Map.Entry>)
analysis.entrySet()) {
                    LocalDateTime tempLocalDate =
LocalDateTime.parse(pair.getKey().toString(), formatter);

                    if (localDate == null) {
                        localDate = tempLocalDate;
                        SMA = Double.parseDouble((String) ((JSONObject)
pair.getValue()).get("SMA"));
                    }

                    if (tempLocalDate.isAfter(localDate)) {
                        localDate = tempLocalDate;
                        SMA = Double.parseDouble((String) ((JSONObject)
pair.getValue()).get("SMA"));
                    }
```

```
                }
                return SMA;
            } catch (ParseException e) {
                e.printStackTrace();
            } catch (NullPointerException e) {
                throw new OverloadApiUseException(); //LIMIT APIs CALL
REACH
            }
        }
    } catch (UnknownHostException u) {
            throw new NoNetworkException(); //NO NETWORK AVAILABLE
    } catch (IOException e) {
        e.printStackTrace();
    }
    throw new MarketClosedException(); //THE MARKET CLOSED
}
```

This method may seem complex at first glance, but it is only used to obtain values that will be necessary for the following. The same is true for the RSI indicator, which is retrieved by a different component but whose implemented method varies only slightly from the one we have just studied. These values are stored in the brain of our software, i.e., in the "TwsIB" component of the "ConnectionHandling" package, so that other components can access them easily. Now that we have retrieved all the information we will use later, we can move on to what to do with the data.

4.5 Strategy implementation

To begin this section on strategy implementation, it is crucial to understand the importance of speed. Let us take an example with a metaphor, hunting. The principle is simple, wait for an animal to come and shoot it. Sometimes the animal will be stationary but most of the time it will be moving. It is therefore important to decide quickly before the animal disappears for good. In the financial markets, similar situations can be found, although traders only chase opportunities and not live animals. Still, they must decide before the buying or selling opportunity disappears.

Now that we have understood that we need to develop an effective but quick strategy, we can start to visualise this package which will be quite light as it contains only one component. After all, it also has only one role and it is by far the most important role. The following figure shows the "EventProcessingHandling" package.

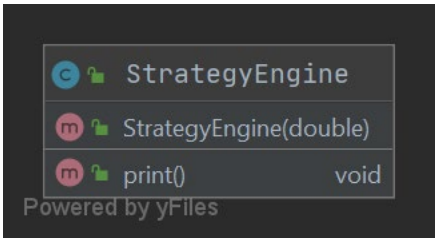


Figure 24 : Diagram of the package "EventProcessingHandling" (Modified screenshot)

We will not talk about the exchanges within the package, but rather we will look in detail at the "StrategyEngine" component which allows all the decisions to be made using the data we have previously obtained. This class is mostly private, the only public attribute it contains is its constructor. It is only when the class is instantiated that it is activated. As soon as it has fulfilled its mission, it becomes obsolete and will therefore be deleted automatically by the process. We will therefore call the constructor of this StrategyEngine() component with the current price as the only parameter. The component will then take care of retrieving the technical indicators it needs from the algorithm's brain, check if these indicators are valid, update the user interface and then proceed to apply the strategy. The code below illustrates the operation of the component in the application of the strategy that will be studied later.

```
public StrategyEngine(double price){
    this.price = price;

    /* TECHNICAL INDICATORS */
    SMA_200 = TwsIB.SMA_200;
    RSI_2 = TwsIB.RSI_2;

    /* NOT 0.0 DATA VERIFICATION */
    if(price != 0.0 && SMA_200 != 0.0 && RSI_2 != 0.0) {

        long startMs = System.currentTimeMillis();

        /* UPDATE liveData */
        TwsIB.liveData.incrementAnalysis();
        TwsIB.liveData.setCurrentPrice(price);
        TwsIB.liveData.update();

        […]

        […]

        /* UPDATE liveData */
        long stopMS = System.currentTimeMillis();
        TwsIB.liveData.setAnalysisTime(stopMS - startMs);
        TwsIB.liveData.update();
    }
}
```

This process is very short and allows the focus to be on the analysis part, which is the important part of this component. To implement effective conditions, a strategy is needed. We could have already guessed which strategy was going to be implemented by looking at the data we took care to collect from the beginning. If we remember well the strategies we mentioned in the theoretical part, Larry Connors' RSI(2) strategy uses all the indicators we now have in our possession. It is exactly this Mean Reversion technique that we will implement in this algorithmic trading software. It is important to understand this strategy well before diving into its implementation.

As a reminder, here is an extract about the strategy: "Larry Connors' strategy uses 4 conditions to determine a buy signal and a sell signal. The buy signal is generated when the price is above the moving average over the last 200 periods and the RSI indicator is below 10, while the sell signal appears when the opposite occurs. (StockCharts, 2021) That is, when the price is below the moving average over the last 200 periods and the RSI indicator is above 90".

These 4 conditions are used to determine buy and sell signals. In fact, 2 conditions are needed for each type of trade. Note that further work based on Larry Connors' strategy assumes that the strategy is more profitable if the determining bounds for the RSI are placed within 5 units of the limit. Because this assertion depends very much on the instrument being monitored as well as on the state of the market, we have provided a certain margin of precision for the user. The user can therefore adjust the value of this variable by specifying a certain level of precision in the trading software settings.

If we now focus on buy signals, we can easily implement these two conditions. If the conditions are not met then nothing will happen, conversely if the conditions are met then a trade should be generated. This is exactly what our component will do. In a positive scenario, it will update the interface and instruct another packet to perform the transaction on the markets. The code below illustrates the application of the strategy for buy signals.

```
/* CONDITION : price > SMA(200) && RSI(2) < 10 */
// Adapt RSI_LIMIT_BUY to accuracy using strategy_data
double accuracy = TwsIB.strategyData.getAccuracy();
double RSI_LIMIT_BUY = 10 - (accuracy*5);
if (price > SMA_200 && RSI_2 < RSI_LIMIT_BUY){

    /* UPDATE liveData */
    TwsIB.liveData.incrementOrder();
    TwsIB.liveData.update();

    /* PLACE BUY ORDER */
    OrderManager orderHandler = new OrderManager();
    orderHandler.placeBuyOrder(price);
    System.out.println("> BUY ORDER HAS BEEN PLACED NOW");
    print();
}
```

This process is used to determine if buying opportunities exist but in some situations there will also be scenarios where it is better to bet on a negative market movement i.e. to go short. In the same way as the previous process, sell signals are generated by a similar process but with reverse conditions and it is a sell order and not a buy order that will be issued. The code below illustrates the triggering of a sell signal as well as the actions taken following this triggering.

```
/* CONDITION : price < SMA(200) && RSI(2) > 90 */
// Adapt RSI_LIMIT_SELL to accuracy using strategy_data
double RSI_LIMIT_SELL = 90 + (accuracy*5);
if (price < SMA_200 && RSI_2 > RSI_LIMIT_SELL){

    /* UPDATE liveData */
    TwsIB.liveData.incrementOrder();
    TwsIB.liveData.update();

    /* PLACE SELL ORDER */
    OrderManager orderHandler = new OrderManager();
    orderHandler.placeSellOrder(price);
    System.out.println("> SELL ORDER HAS BEEN PLACED NOW");
    print();
}
```

These two conditions conclude this section on the "StrategyEngine" component. Even if it is relatively short and concise, this component is the “raison d'être” of the system. The whole architecture is oriented around it. If the basic strategy is not good, it also buries any hope of profit. To check that this basic strategy works, we will test it individually in the next chapter. This will be tested outside our application to ensure that our code or interpretation of the strategy in the code does not bias the actual results of this strategy.

4.6 Strategy backtesting

In the previous step, we took care to set up our strategy. In the software we have built so far, it is this strategy that will be decisive. It is therefore in our interest to test the strategy outside our software to ensure that it is not faulty in any way. Here again, we have found a free application that would allow us to carry out this test phase before continuing with the development of our software. This application is used directly online, and all the calculations will be done on a remote server so as not to use the resources of our machine. The following link https://www.quantreex.com/trading# gives access to the application we are talking about. It is necessary to identify yourself beforehand, but a free account will be sufficient for the tests we are going to carry out.

Once connected to this service, we will need to define the strategy to be tested. To better understand the interface of the application, we advise all readers to follow the guided tour which is rather quick and efficient. The strategy that will be used has already been mentioned many times in the theoretical part as well as the part about the implementation of the strategy. In case of doubts, it is advisable to refer to these two parts. Now we will try to implement the 4 conditions that were discussed before. If we remember well what was done before, our strategy should look like the following figure.

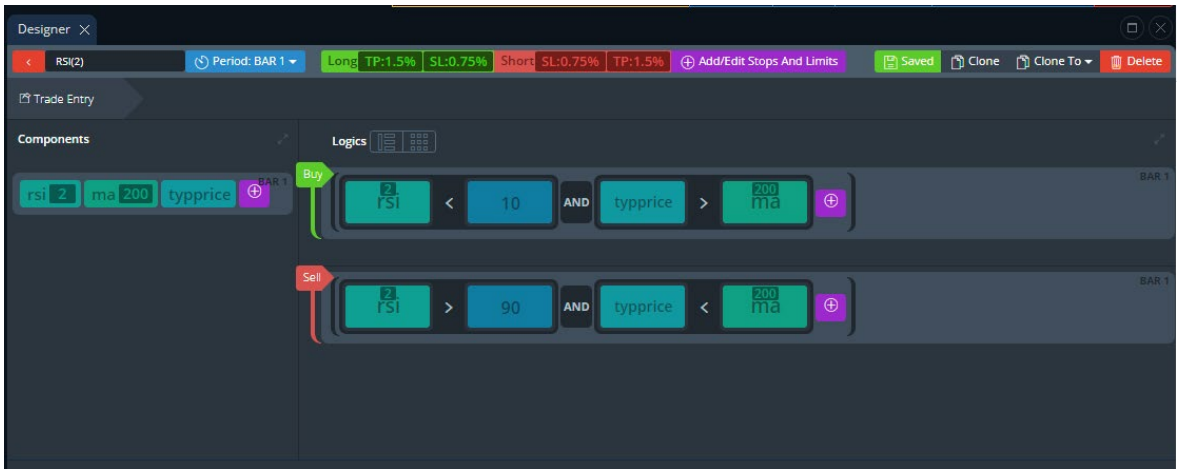


Figure 25 : Definition of the strategy for backtesting (Modified screenshot)

Note in passing that we had to define exit limit prices. In our case, this corresponds to the take profit and stop loss limits that the user is free to set as he wants. In this test phase, we selected arbitrary values. These values correspond to a small price variation but also limit the risks we will take. The following figure shows the output configuration we have made.

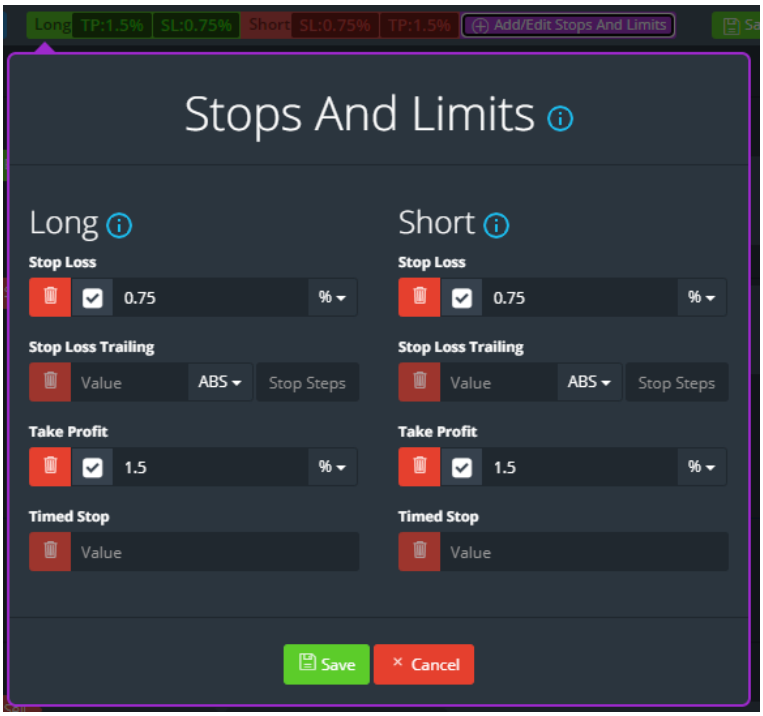


Figure 26 : Definition of automatic exit conditions for backtesting (Modified screenshot)

The set-up is now complete and all that remains is to carry out a series of tests. It is important to understand that not all instruments behave in the same way when faced with this strategy. If some currency pairs obtain convincing results, this will not be the case for other currencies and even for some stocks. For information, we have tested this strategy over an interval of 1 minute, which is approximately what will be applied in our software.

For this test phase, we pre-selected instruments that generally performed well against this strategy. These are the following currency pairs: AUD/JPY, AUD/NZD, AUD/CHF. These currency pairs, but especially the Australian dollar (AUD), reacted well to the conditions we elaborated earlier. Out of curiosity, we also tested this strategy on some stocks such as Facebook, Alphabet, and Intel. These instruments, even if they will not be available in the software, seemed to meet our expectations rather well in the preliminary testing phases.

We performed several tests for each instrument. In total for each of our 6 instruments, we will have performed 3 tests over different periods. The first test concerns Q1 and Q2 2020 (01.01.2020 - 30.06.2020), the second test concerns Q3 and Q4 2020 (01.07.2020 - 31.12.2020) and the third test concerns the last 180 days. It is important to note that this time horizon covers the coronavirus crisis as well as the current market structure. In the end 18 independent tests were conducted to verify the strategy, the table below summarises the performance we obtained for our different tests. As for the complete analyses that were obtained for each of these 18 tests, they can all be found in the appendices of this document.

	Q1-Q2 2020	Q3-Q4 2020	180 Days	AVG
AUD/JPY	10.85% Appendix 01	0.24% Appendix 02	5.93% Appendix 03	5.67%
AUD/NZD	-3.93% Appendix 04	-1.07% Appendix 05	1.94% Appendix 06	-1.02%
AUD/CHF	12.60% Appendix 07	-2.41% Appendix 08	-0.88% Appendix 09	3.10%
Facebook	-1.49% Appendix 10	4.69% Appendix 11	10.08% Appendix 12	4.42%
Alphabet	-46.99% Appendix 13	6.82% Appendix 14	11.22% Appendix 15	-9.65%
Intel	19.67% Appendix 16	-1.41% Appendix 17	40.40% Appendix 18	19.55%
AVG	-3.10%	2.29%	22.90%	7.36%

Table 2 : Aggregation of all backtesting results

The table above helps to make clear that a strategy cannot always be a winner. The market is unpredictable, and every opportunity comes with some risk. According to the analysis we have carried out, it is possible to expect between 20 and 120 transactions per 6-month period. But it is not the number of trades that matters, nor the profit per trade or per instrument, but the result of all the aggregated trades. These few instruments that are mentioned above fit this strategy well now, but this may change in the future. We have therefore found instruments to place our orders on for the next few months but monitoring and verification will be necessary if we wish to continue to run this system in the coming years. For our trading software, we will use this trading strategy, which we think is acceptable given the expected performance.

Before concluding this part of the tests, one last detail about this strategy is worth mentioning. Most of the trades that are made on these instruments are losing trades. In each test, we only obtained 30-40% of winning trades. This is not a major hindrance to the implementation of this strategy because when we lose, we lose small. But when we win, we win big.

4.7 Automatic trade execution

We saw earlier how to get the data and how to detect buy and sell signals. Our software is therefore capable of observing everything that happens, but it lacks the ability to intervene in the markets autonomously to take advantage of all the opportunities. In this section, we will look at how to equip our software with an automated order placement system. This means that no intervention on our part will be necessary and this is what characterises automatic trading.

The "ExecutionHandling" package will take care of placing orders once our system has detected an opportunity. No intelligence is really needed, its role is only to create the trades and execute them according to the information it receives. To do this, it will be equipped with a component that will be responsible for creating the order and then transmitting it, but since no errors will be tolerated, an order verification component will also be implemented. These two components will be sufficient for our system to be autonomous and to carry out transactions for us. The following diagram illustrates the organisation of this package.

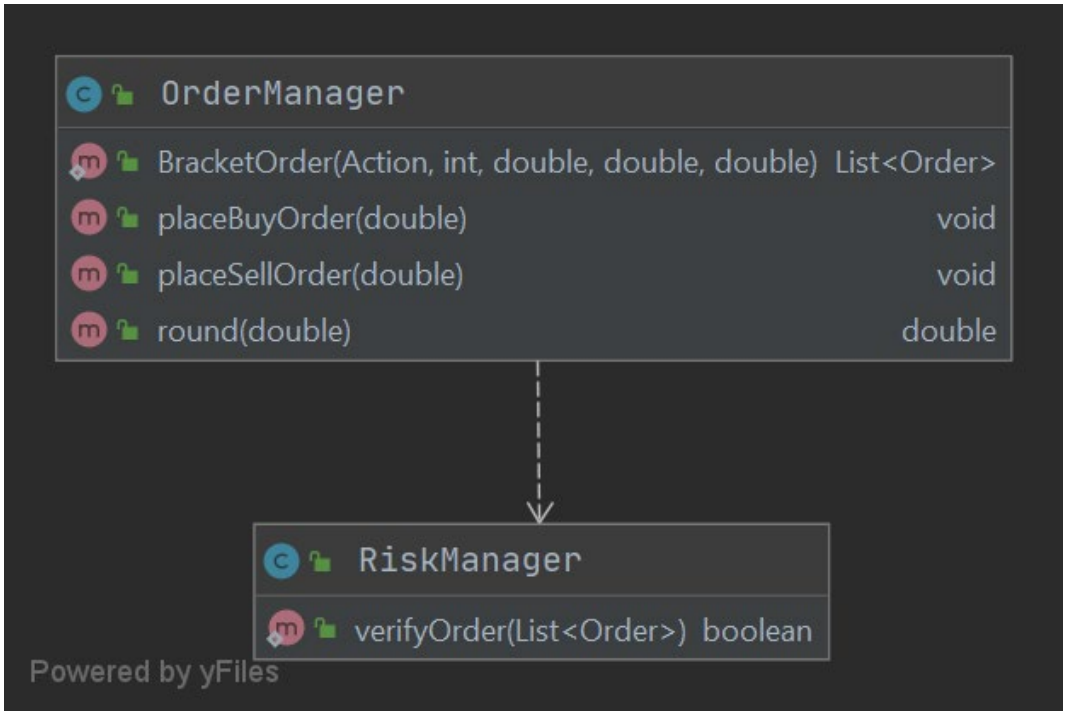


Figure 27 : Diagram of the package "ExecutionHandling" (Modified screenshot)

Now that we have an overview, let us take a closer look at the transaction process. Once the opportunity has been detected, it will automatically be passed to the “OrderManager” component. The information that is passed to this component is very limited, in fact only the price at which the opportunity was identified is passed as a parameter to the placeBuyOrder() method. This method is of course only invoked if it is a buying opportunity. In a reverse scenario, the placeSellOrder() method will be called. These two methods are the gateway to this component and are very similar. They are responsible for calculating the information needed to generate the order. The quantity, the order type, the take profit limit, and the stop loss limit are calculated using the data saved in the storage components. Once this data is available, two other methods are called internally to complete the process. These are the BracketOrder() and placeBracketOrder() methods which we will look at a little later. The code below represents our placeBuyOrder() method and allows us to better understand the workflow.

```
/* CREATE BUY ORDER */
public void placeBuyOrder(double price) {
    /* Adapt quantity to multiplier using strategy_data */
    int multiplier = TwsIB.strategyData.getMultiplier();
    int quantity = 20000 * multiplier;

    /* Adapt order to type using strategy_data */
    String order = TwsIB.strategyData.getOrder();
    double limitPrice = 0;
    switch (order){
        case "Market":
            limitPrice = -1;
            break;
        case "Limit":
            limitPrice = price;
            break;
    }

    /* Adapt takeProfitLimitPrice to strategy_data */
    double takeProfit = TwsIB.strategyData.getTake_profit();
    double takeProfitLimitPrice = price * (1.0+(takeProfit/100.0));

    /* Adapt stopLossPrice to strategy_data */
    double stopLoss = TwsIB.strategyData.getStop_loss();
    double stopLossPrice = price * (1.0-(stopLoss/100.0));

    /* Adapt stop loss to SMA_5 => stopLossPrice > SMA(5) */
    double SMA_5 = TwsIB.SMA_5;
    if (stopLossPrice > SMA_5){
        stopLossPrice = SMA_5;
    }

    /* Get bracket order */
    List<Order> orders = BracketOrder(Types.Action.BUY, quantity,
round(limitPrice), round(takeProfitLimitPrice), round(stopLossPrice));

    /* Transmit bracket order */
    placeBracketOrder(orders);
}
```

In the same way that this method calculates the data for a buying opportunity, the placeSellOrder() method does so for selling opportunities. In fact, only the calculation of the take profit and stop loss limits differs. We can understand this better by looking at the method code below. For the sake of clarity, only the code that differs from the previous method is shown here.

```
/* CREATE SELL ORDER */
public void placeSellOrder(double price){

    […]

    /* Adapt takeProfitLimitPrice to strategy_data */
    double takeProfit = TwsIB.strategyData.getTake_profit();
    double takeProfitLimitPrice = price * (1.0-(takeProfit/100.0));

    /* Adapt stopLossPrice to strategy_data */
    double stopLoss = TwsIB.strategyData.getStop_loss();
    double stopLossPrice = price * (1.0+(stopLoss/100.0));

    /* Adapt stop loss to SMA_5 => stopLossPrice < SMA(5) */
    double SMA_5 = TwsIB.SMA_5;
    if (stopLossPrice < SMA_5){
        stopLossPrice = SMA_5;
    }

    /* Get bracket order */
    List<Order> orders = BracketOrder(Types.Action.SELL, quantity,
round(limitPrice), round(takeProfitLimitPrice), round(stopLossPrice));

    /* Transmit bracket order */
    placeBracketOrder(orders);
}
```

Now that we know how the essential data for order creation is obtained, we will look in detail at how this is done. The BracketOrder() method is used to generate orders for all opportunities, whether they are buying or selling opportunities. It takes all the information we have obtained before and creates an order list. This order list includes the main order, the take profit order, and the stop loss order. They are all linked to each other through the main order ID. This method is not complex, but it is important to be able to trade on the markets with our broker. It is responsible for assigning the values we have previously calculated to the orders that will be placed. The code below perfectly describes the creation of the 3 orders we have talked about so far.

```
/* CREATE BRACKET ORDER (PARENT, TP, SL) */
public static List<Order> BracketOrder(Types.Action action, int quantity,
double limitPrice, double takeProfitLimitPrice, double stopLossPrice) {

    /* GET ID FROM TWS IB */
    int parentOrderId = TwsIB.getNextValidIDOrder();

    /* PARENT ORDER */
    Order parent = new Order();
```

```
    parent.orderId(parentOrderId);
    parent.action(action);
    parent.totalQuantity(quantity);
    /* Adapt order type using strategy_data */
    switch (TwsIB.strategyData.getOrder()){
        case "Market":
            parent.orderType(OrderType.MKT);
            break;
        case "Limit":
            parent.orderType(OrderType.LMT);
            parent.lmtPrice(limitPrice);
            break;
    }
    parent.transmit(false);

    /* TAKE-PROFIT(TP) ORDER */
    Order takeProfit = new Order();
    takeProfit.parentId(parentOrderId+1);
    takeProfit.orderId(parentOrderId+2);
    takeProfit.action(action.equals(Types.Action.BUY) ? Types.Action.SELL
: Types.Action.BUY);
    takeProfit.totalQuantity(quantity);
    takeProfit.orderType(OrderType.LMT);
    takeProfit.lmtPrice(takeProfitLimitPrice);
    takeProfit.transmit(false);

    /* STOP-LOSS(SL) ORDER */
    Order stopLoss = new Order();
    stopLoss.parentId(parentOrderId+1);
    stopLoss.orderId(parentOrderId+3);
    stopLoss.action(action.equals(Types.Action.SELL) ? Types.Action.BUY :
Types.Action.SELL);
    stopLoss.totalQuantity(quantity);
    stopLoss.orderType(OrderType.STP);
    stopLoss.auxPrice(stopLossPrice);
    stopLoss.transmit(true);

    /* SET OF ORDERS */
    List<Order> orders = new ArrayList<>();
    orders.add(parent);
    orders.add(takeProfit);
    orders.add(stopLoss);
    return orders;
}
```

Now that we have the list of orders we need to transmit, all that remains is to proceed with the actual transmission. As we have seen in the first two methods, the list of orders is transmitted using a last additional method, the placeBracketOrder() method. This method simply checks the orders and then passes them on to the “ConnectionHandling” package, which will give the order to our API. The code below shows all the operations that are performed in this method.

```
/* PLACE ORDER */
private void placeBracketOrder (List<Order> bracketOrder) {
    /* VERIFY BRACKET ORDER */
    if (RiskManager.verifyOrder(bracketOrder)) {
        /* TRANSMIT ORDER TO MARKET */
        for (Order o : bracketOrder) {
            TwsIB.outputAdapterIB.onPlaceOrder(o);
        }
        /* ADD TRANSACTIONS TO HISTORY */
        TwsIB.historyData.addTransactions(new
TransactionDTO(bracketOrder));
        System.out.println("> The risk management system agreed a
transaction.");
    } else {
        System.out.println("> The risk management system blocked a
transaction.");
    }
}
```

We have seen how orders are transmitted and when they are verified by our system. Before concluding this section, we will look at the process that allows our system to verify the orders before they are finally transmitted. It is the verifyOrder() method which will receive the list of all the orders in parameter before taking charge of the control. This check ensures the integrity of the orders. If the list of orders does not conform to what we expect from the system for any reason, then the opportunity will be abandoned. The method code below allows you to see what reasons the system would have for not taking a position on an opportunity.

```
public static boolean verifyOrder (List<Order> bracketOrder) {

    /* Order Integrity Verification */
    if (bracketOrder.size() != 3){
        return false;
    }

    Order parent = bracketOrder.get(0);
    Order tp = bracketOrder.get(1);
    Order sl = bracketOrder.get(2);

    if (parent.orderId()+1 != tp.parentId()){
        return false;
    }

    if (parent.orderId()+1 != sl.parentId()){
        return false;
    }

    if (parent.totalQuantity() != tp.totalQuantity()
            || parent.totalQuantity() != sl.totalQuantity()){
        return false;
    }

    if (parent.orderType() == OrderType.LMT
            && TwsIB.strategyData.getOrder().equals("Market")){
        return false;
    }
```

```
    if (parent.orderType() == OrderType.MKT
            && TwsIB.strategyData.getOrder().equals("Limit")){
        return false;
    }

    if (tp.lmtPrice() == 0
            || sl.auxPrice() == 0){
        return false;
    }

    if (parent.getAction().equals("BUY")
            && tp.lmtPrice() <= sl.auxPrice()){
        return false;
    }

    if (parent.getAction().equals("SELL")
            && tp.lmtPrice() >= sl.auxPrice()){
        return false;
    }

    return true;
}
```

In this sequence of conditions, if one of them is positive then the whole process stops, and our system will start looking for an opportunity again. We now know everything that goes on behind the display screen of our application. It is now time to look at how to start it up and how to start trading the stock markets automatically. We will see all these points and the proof that this software works the way we wanted in the next chapter.

4.8 Proof of work

In this chapter, which will conclude the development part, we will focus on the implementation of the software we have built. For this we will need all the elements we have studied so far. It is therefore imperative that you have followed the previous sections to understand what we are going to do in this final stage. To do this, it is necessary to have the source code of the software or the executable. It is possible to copy the source code directly from the official GitHub repository or to find the corresponding JAR file also available in the same repository. We will assume that everyone who now wants to run the software on their own machines has the executable or is able to compile the source code to start the program.

Before we start our actual software, we will start the trading workstation and log in. This will be necessary to make the connections at the start of the software and then get all the data about our account and the assets we will be monitoring. We have already discussed all the steps to be taken in the initial setup section, so we will simply repeat them once again. If we have followed the instructions, we should have a screen that looks like the following figure.

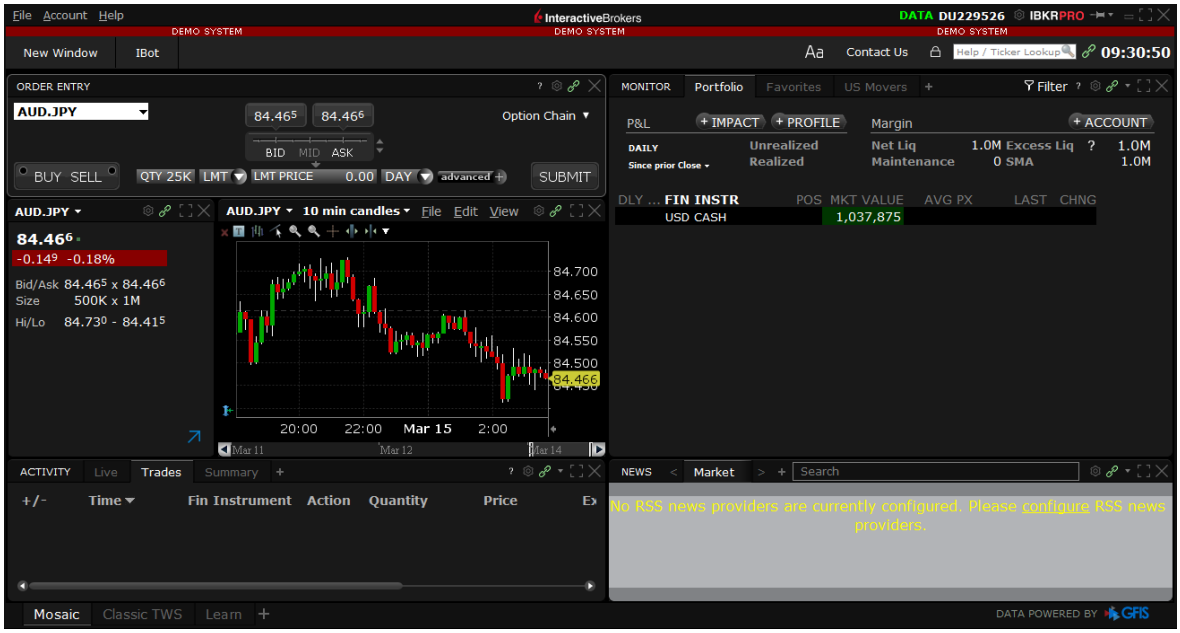


Figure 28 : Screenshot of the trading workstation screen showing the state before starting (Modified screenshot)

Now we can start our software. The reader should also be able to do this thanks to all the information we have given so far. The following figure represents the welcome screen and means that the program is ready.

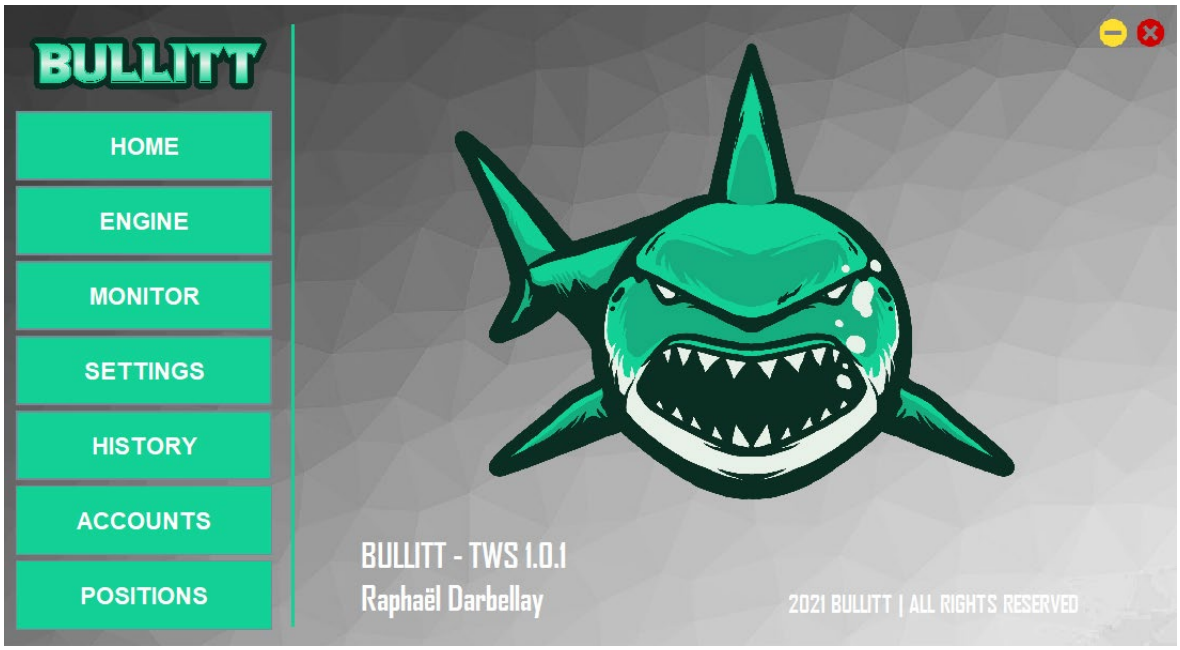


Figure 29 : Screenshot of the "Welcome" screen of the software (Modified screenshot)

Clicking on the "HOME" button takes us to the home page of this software. Here we see the list of steps that are necessary for start-up and monitoring, as well as some basic data that will be useful for monitoring the evolution of the algorithm once it is running. The following figure shows the home screen we are talking about.

Figure 30 : Screenshot of the "Home" screen of the software before starting (Modified screenshot)

Before officially starting the algorithm, we will check and adjust the parameters. To do this, we go to the "SETTINGS" tab. Here we can change the instrument we want to monitor, but also the precision, the timescale, the multiplier, the take profit and stop loss limits and the type of order used. If we have not correctly understood what these values correspond to, which have been previously detailed in this document, it is not recommended to continue but rather to go back and read them. It is very dangerous to start a trading algorithm without knowing exactly what we are doing. Once all the changes have been made, do not forget to save the values by pressing the "GO" button. The following figure shows the interface for entering this data.

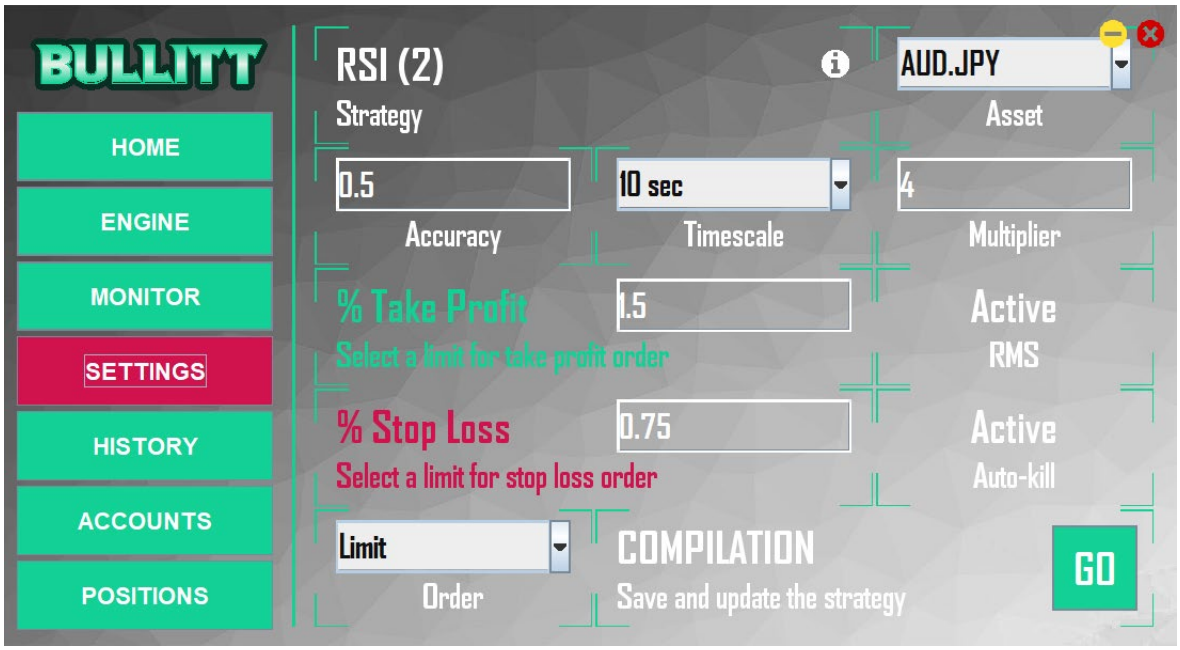


Figure 31 : Screenshot of the "Settings" screen of the software (Modified screenshot)

To finish, we are going to carry out a last small adjustment. We have in our possession an API key that we took care to get at the beginning of the development part, it is now time to use it. To do this, we can stay on the "SETTINGS" page. Next to the strategy we have a white information button. If we click on this button, the system will ask us to enter the API key we have. When entering the key, we should remember to exclude any spaces that may be in front of or behind the string. The following figure shows the input field for the API key.

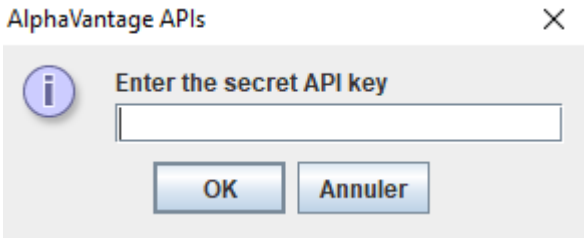


Figure 32 : Screenshot of the "Request API" screen of the software (Modified screenshot)

We are ready for take-off! Let us buckle up and go to the "ENGINE" tab. This is where we will start the algorithm. We check that the data on this screen is the data we want to start the algorithm with, and that the API key is active. If this is not the case or if we want to change any information, we can simply go back and make new changes. In our scenario, we will start the algorithm with the data we see in the following figure.

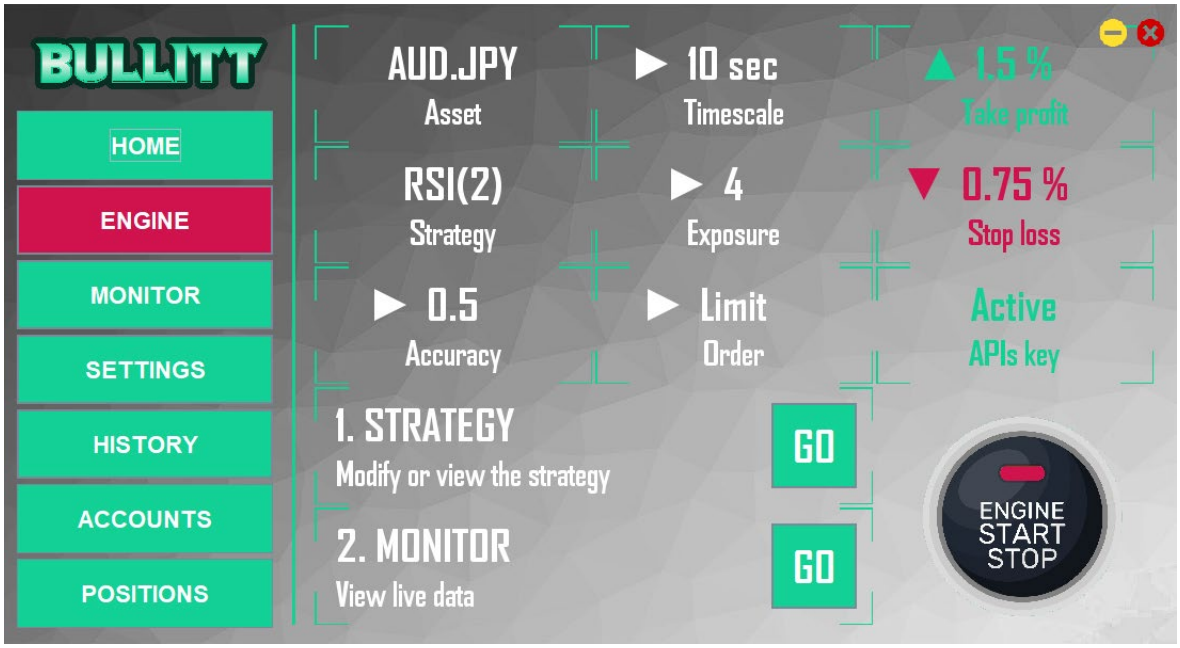


Figure 33 : Screenshot of the "Engine" screen of the software before starting (Modified screenshot)

3, 2, 1, Let us go! To start, we click once on the "START" button. This button will change from red to green to indicate its status and an immediate stop button will appear in the top left corner of our application. This button is specifically designed to stop the system immediately if needed. In the following figure, you can see the two changes that occurred after the start-up.

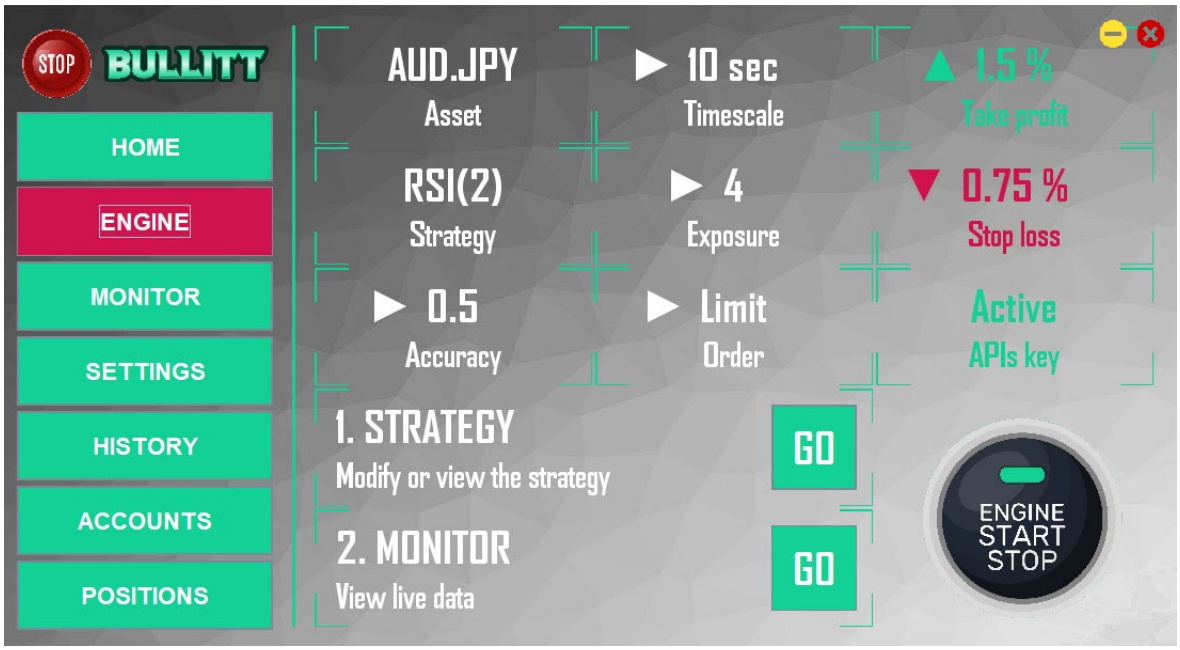


Figure 34 : Screenshot of the "Engine" screen of the software after starting (Modified screenshot)

Now that everything has been launched, we enter the monitoring phase. To do this, we move to the "MONITOR" tab. On this new screen we see values. These values are updated by our software to reflect the actual value of the portfolio or algorithm. If we have reached this point, it is likely that no further explanation is necessary, as the field names are self-explanatory. However, a small clarification is required at this point. The values for prices are calculated according to the price at the start of the algorithm and not since the beginning of the opening of the markets. The following figure shows the data we have obtained since we started.

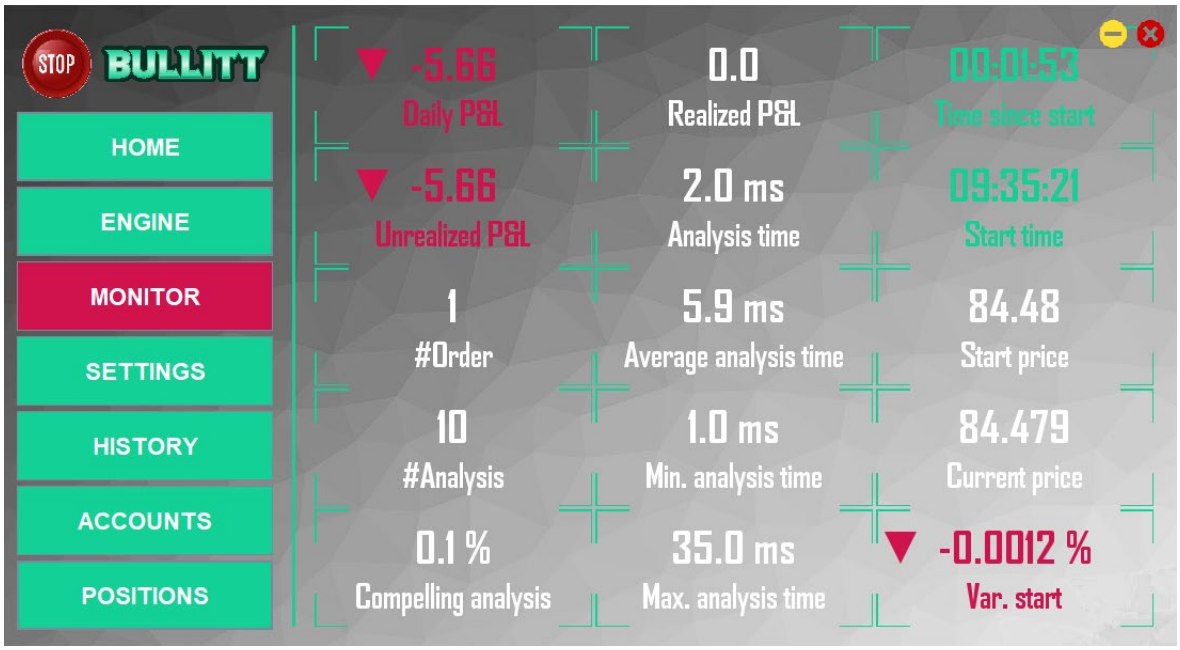


Figure 35 : Screenshot of the "Monitor" screen of the software (Modified screenshot)

We have seen in the previous figure that an order was executed by our system. We will now switch to our trading workstation to see how the transaction went. In the following figure, you will see the 3 orders that have been placed by our software but are still pending because the limit price has not yet been reached.

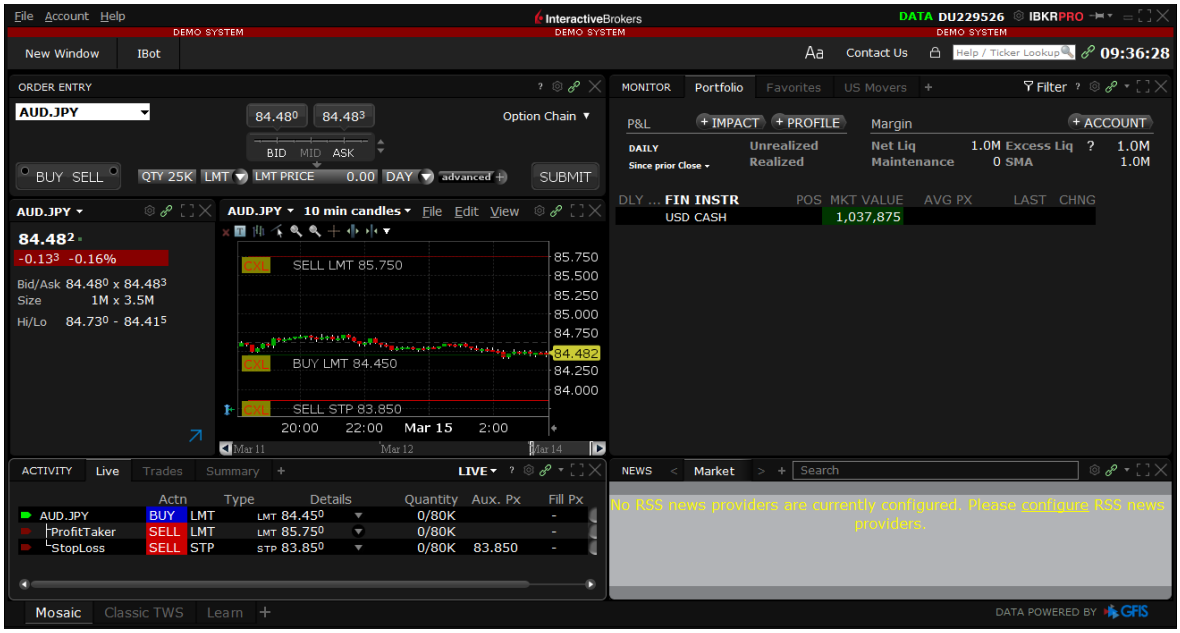


Figure 36 : Screenshot of the trading workstation screen showing the state before execution (Modified screenshot)

Finally, our order was executed a few moments later. (We deliberately changed the limit to illustrate this tutorial, otherwise we would have had to wait for the order to trigger itself based on the price, but this might never have happened). If we look at the status of our account

again, we will see that 1 of the 3 orders transmitted has been executed. The other two orders are pending as they are exit orders that will be triggered depending on the market evolution. The following figure illustrates what has just been explained.

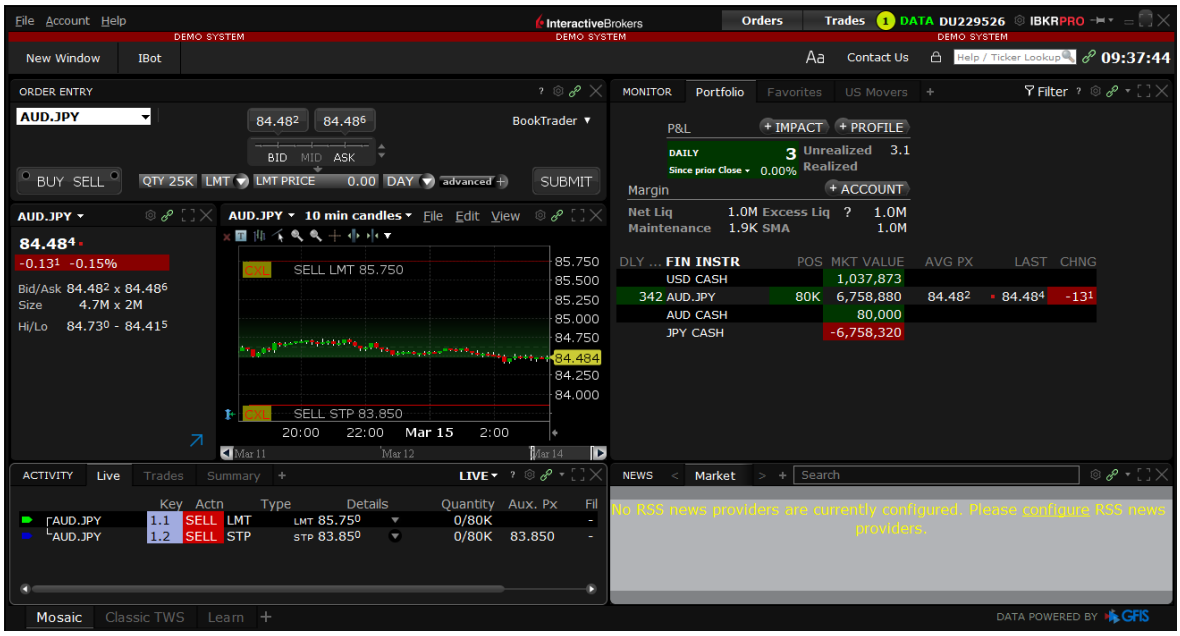


Figure 37 : Screenshot of the trading workstation screen showing the state after execution (Modified screenshot)

Everything seems to be going well, so now it is time to see if our software also provides us with the relevant information. So, we go to the "HISTORY" tab. Here we can see that we have all the information corresponding to the transaction we have just carried out. The following figure confirms this.

Figure 38 : Screenshot of the "History" screen of the software (Modified screenshot)

To get information about the state of our account, we can easily go to the tab "ACCOUNTS" which will give us useful data. On this screen we find the amount of funds available, the total value of our account, our buying power and the necessary margin that is blocked on the account. Other information on the evolution of the account is also available in the right column. All this information comes from our broker and is updated automatically in case of changes. The following figure shows the details of these values.

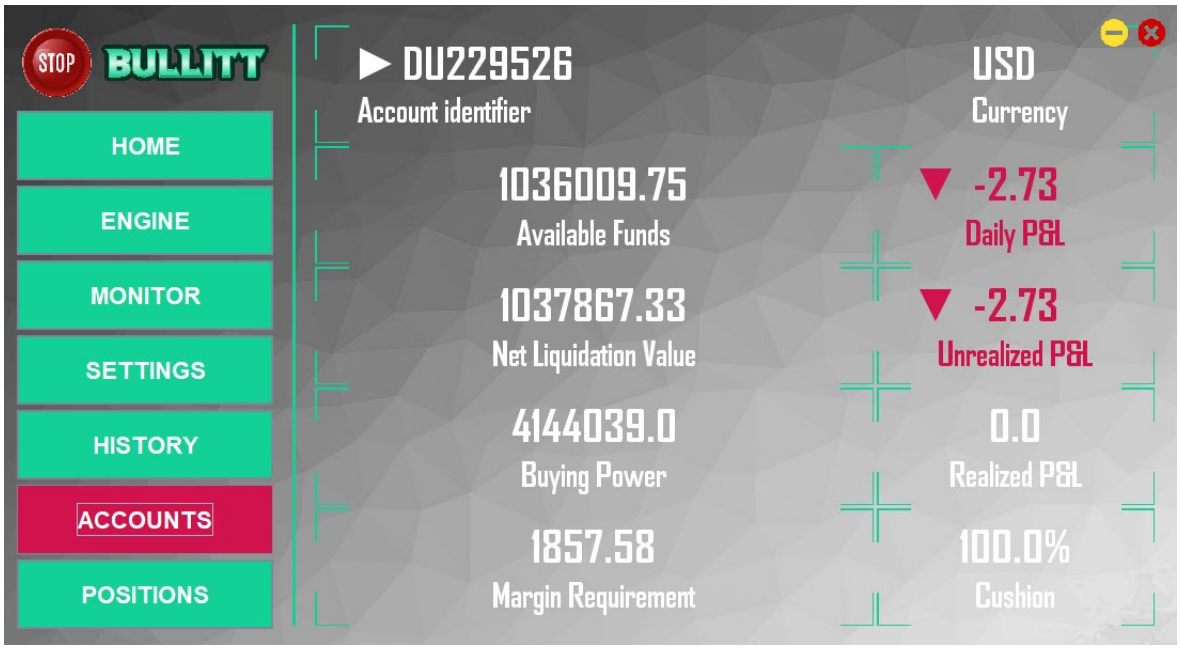


Figure 39 : Screenshot of the "Accounts" screen of the software (Modified screenshot)

There is one last screen that we have not yet explored together, the "POSITIONS" screen. We go there straight away. Here we have various information that corresponds to the instrument we are currently monitoring. The volume, average unit cost, value, type as well as the instrument-specific development data. As before, these values are continuously updated when changes occur. The following figure shows this information.

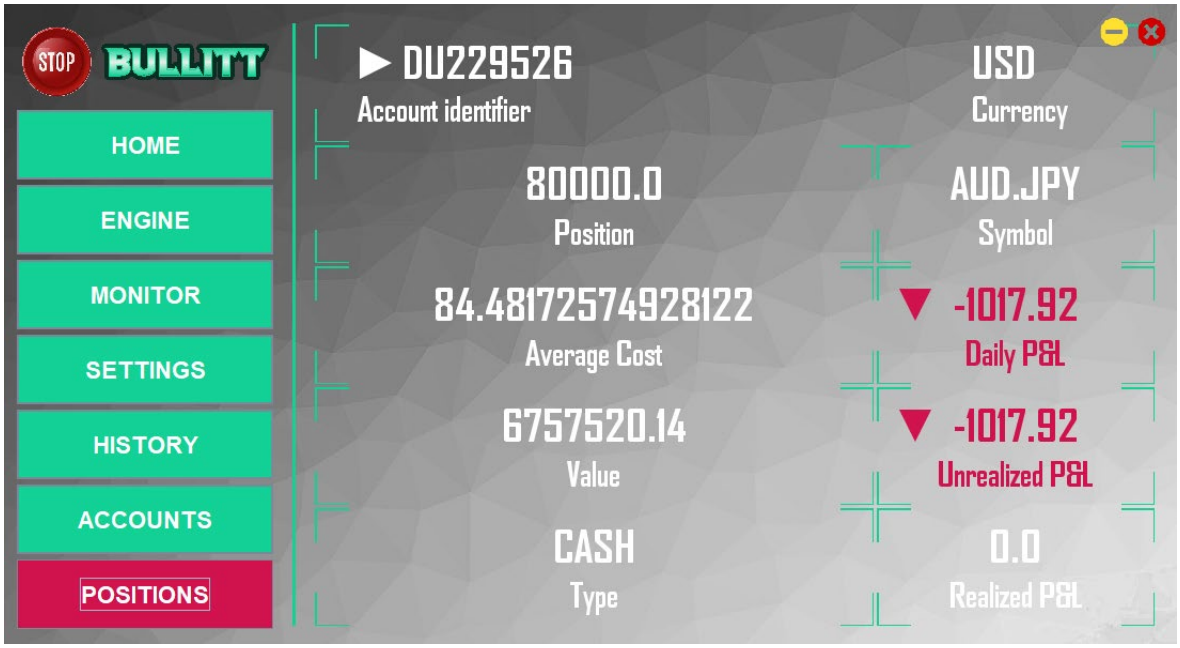


Figure 40 : Screenshot of the "Positions" screen of the software (Modified screenshot)

We talked about the home screen at the very beginning and mentioned that it also had some basic information. Now that our software has been launched, we can go back and check that this is also updated. This data includes the number of buys, sells, difference, exposure in the monitored currency and the direction of our exposure. We can see from the following figure that the data is up to date and that it corresponds to the transactions that have been made.

Figure 41 : Screenshot of the "Home" screen of the software after starting (Modified screenshot)

Knowing how to start is good. Knowing how to stop is better. Just like in the casino, it is necessary to know how to take your losses or your winnings before you end up plucked. To stop the software, we have seen a big red button in one of the corners of each screen since the start-up. It is on this button that it will be necessary to click to finish the process. When stopping our software will automatically close all open positions and delete any orders that have not been executed so far, especially stop loss or take profit orders. Once the red button is clicked, it disappears and the status in the "ENGINE" tab becomes red again as shown in the following figure.

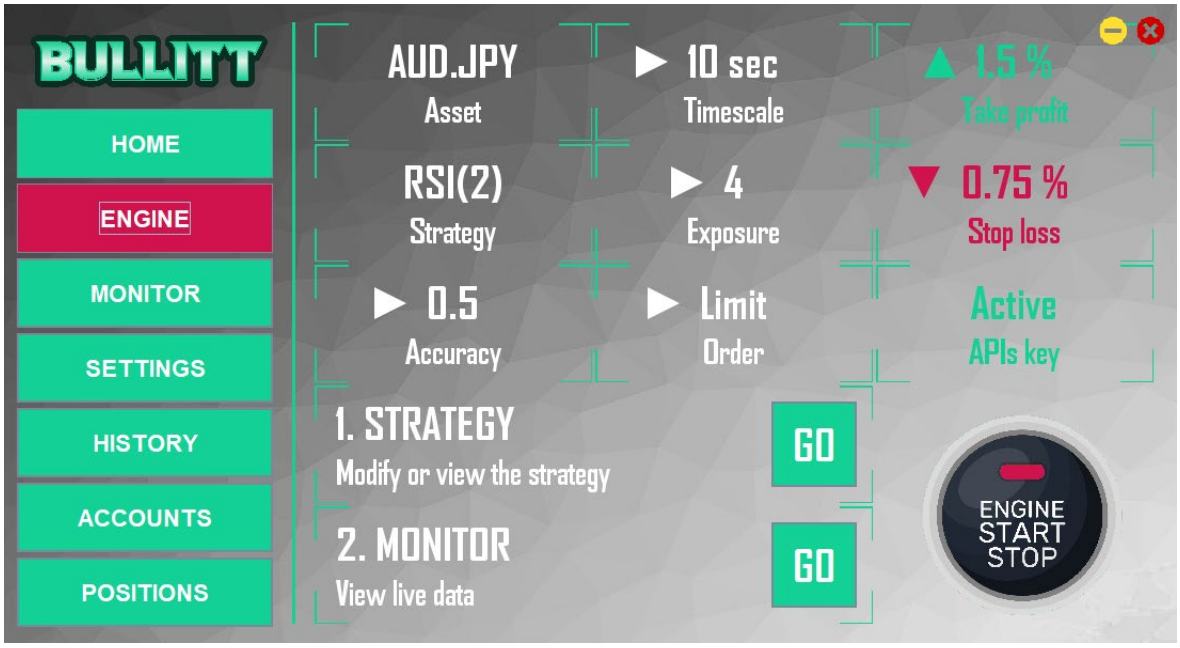


Figure 42 : Screenshot of the "Engine" screen of the software after stopping (Modified screenshot)

Before ending this chapter, we will check that our software has reset the positions and orders to zero with our broker. To do this, we will switch to the trading workstation one last time. We will see in the following figure that there are no more active orders and that all positions are at 0.

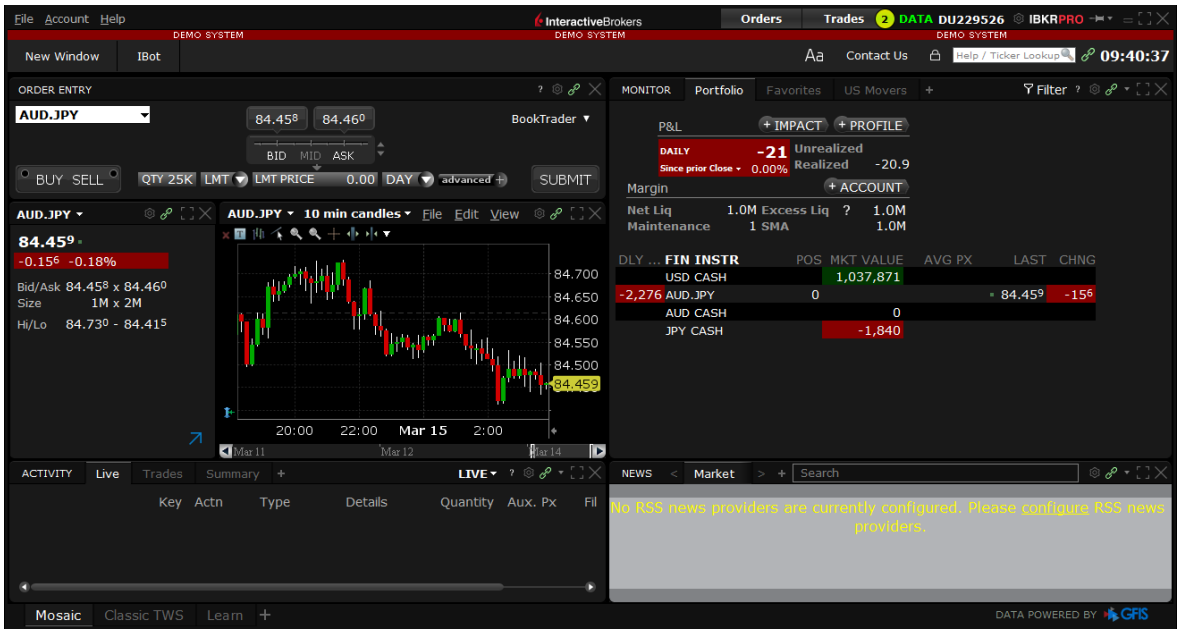


Figure 43 : Screenshot of the trading workstation screen showing the state after stopping (Modified screenshot)

Once the algorithm is stopped, all screens are not automatically reset. The data is still visible for analysis or manual export to a trading book. Only after the algorithm is restarted will the values be reset. The user is free to start the algorithm as many times as he wants with different instruments each time if he wishes.

If the application is closed and then reopened, there is no need to redefine the strategy or API key, all these values are stored locally on the user's PC to allow a smooth user experience. If these files are deleted or the user runs the application on a new computer, the data will have to be re-entered before the algorithm can be run.

The use of this software is, by virtue of its MIT licence, free. However, we would like to remind you that this software has been developed for educational purposes and that it is strongly advised not to use a real account if the user does not have the necessary knowledge of the stock markets or the programming language used. We decline in advance any responsibility as for the use which could be made of this software.

5 Evaluation

After talking about the theory and implementing algorithmic trading software together, it is now time to be a bit critical and evaluate everything that has been achieved so far. We will detail the results obtained, discuss the ethics, evaluate the thesis before ending with the knowledge that has been gained during this process.

5.1 Overview of results

As a reminder, the main objective was to build a trading algorithm capable of placing orders autonomously on the stock markets. The underlying objectives were also to find out if such a machine could beat a real trader in terms of speed and performance. It has taken some time to get to this point, but at this stage readers would normally already have an answer or an opinion on some of the objectives we posed at the beginning of this research work.

First, and as we have reasonably demonstrated, the software we have built throughout this thesis can place orders following a simple strategy that had been predetermined. It can follow the market, obtain technical analysis, and make decisions on its own. These decisions are then executed automatically by placing orders on the exchanges and instruments that are targeted. The objective is therefore fully achieved. It should be noted, however, that our algorithm is only able to target 105 currency pairs to apply our strategy. While this may seem like a lot at first glance, we would have liked to try and trade stocks as well and not just currencies as we managed to do. This proved to be off topic because live quotes for stocks require a real account and a paid subscription for each exchange. While this would have been possible, it was against our original idea to be able to allow anyone to set up an algorithm in this way. We wanted to offer a software that did not require any financial investment but above all that avoided all people to take risks on the stock markets.

Our second objective was speed. This is automatically measured each time an analysis is performed. In our tests, we found that the speed of our algorithm (About 6 milliseconds on average) is much faster than a real trader if they perform the same operations. This objective was imperative and very logical. It would have been pointless to have an algorithm to perform analysis slower than a human. Even though our analyses are quite fast, we are still not playing in the high-frequency trading class, which reaches much better speeds. This analysis speed only considers the analysis and calculations that are made to find an opportunity once all the information is in our possession and without considering the transmission of orders. These two elements have been excluded from the average because they are too dependent on the hardware used and the network available. We should keep

in mind that we are running the trading algorithm on our own computer with a Wi-Fi connection whereas professionals in the field use dedicated servers close to the stock exchanges and directly connected by the latest fibre optics. It would therefore have been pointless to compare our performance with that of these trading giants. From a neutral point of view, the speed we obtain simply cannot be obtained by a traditional trader, so it is logical that the algorithm beats the trader on this point.

Finally, our third objective was to obtain a better performance than a traditional trader. It is difficult to say whether this objective has been achieved or not because the two players are simply not playing in the same league. A traditional trader invests differently from an algorithm, for example, he can understand the current market situation, whereas the algorithm only invests by considering the numbers. To take this into account, one would have to turn to artificial intelligence, but this is not within the scope of this research project. In the end, based on all the research we have done, we know that a trading algorithm can beat many traders. The only requirement is that it has sufficient resources. Hardware resources are important, but what is even more important is the ability to use mathematicians or quantitative analysts to develop complex and optimized strategies to take advantage of opportunities in the markets. Unfortunately, without these valuable resources at our disposal, we do not believe our trading algorithm can beat all traders all the time. The strategy used is the reason for this and is one of the major development opportunities of this work as we will see later in this report.

5.2 Trustworthiness and ethical viewpoints

Before continuing further in the evaluation of the work done, we will dwell a little on the reliability of the research and the ethical point of view of our software. This chapter will allow us to have a little subjective view about our thesis.

First, we want to talk about the reliability of our research. The research done for the theoretical part and especially about the history of trading turned out to be easy. Mainly because many documents, reports and articles were available and mostly all went in the same direction. It is therefore possible to assume that this first historical part is more than reliable. The situation was not the same for the more recent information that we had to discover about trading and trading software. This recent information was generally not found in books but rather in articles. These articles were fewer in number and some contradicted each other. So, we have tried to be thorough and objective with all this information, we really hope that this work only embodies the reality as it is possible to see it in our society. Given the number of sources collected, we believe that this part is also strongly reliable and can

easily be used for further research. Regarding the software and its development, it was not difficult to find valuable information in articles or in the documentation of the various brokers we consulted. The specialised blogs were also very helpful in verifying what we had already read in other sources. Once again, we believe that this document and the sources used are reliable.

Secondly, we wanted to talk about an ethical point of view. The foundation of our society is based on adding value. To illustrate this, let us take the example of a baker. He transforms and assembles products that he sells at a higher price in his shop. It is the same for the consulting company that sells its services to companies. In most existing businesses today, we can easily distinguish a form of added value that is added at each stage. Therefore, we can find the value added tax. This tax has been present on every good or service consumed for a long time and we contribute to it almost every day without necessarily realising it.

In the world of trading, it is different. Buying an instrument and selling it a few hours, days or months later brings absolutely no value to that instrument. There is no value-added tax involved. If people can make profits, it is usually because others are making losses. Trading is in no way a system for adding value but only for moving existing value. Take the case of high frequency trading, we saw in the theoretical part that it allows a part of the profit to be taken when a transaction is made. This profit comes directly from the investor. Here again, wealth has been moved, not created.

Many critics try to convince that these companies, which hire people with a lot of talent, are mobilising people who could be much more useful elsewhere and bring real added value to society. For example, a talented mathematician could analyse the onset of heart attacks to allow them to be detected several days in advance instead of allowing rich companies to become even richer. This example is not the only one, but merely serves to illustrate the view of these critics.

From a general point of view, not all trading and high-frequency trading companies are bad, it is just that they are primarily focused on generating their own wealth. Similarly, they do not necessarily do what is allowed in the markets but everything that is not prohibited. This powerful industry is the cause of many scandals and we will not enter this debate by giving our point of view. But thanks to the objective views provided so far, every reader should be able to make his or her own decision about the ethics of this field.

5.3 Thesis research

Our thesis is structured in several parts. These parts have given rhythm to our work during the last months. They include the theoretical part, the development, and the detailed explanation of the whole software. These three parts form the hard core of our work and minimal elements were then grafted onto this core.

The theoretical part presented at the beginning of this report has allowed us to concretely gain valuable knowledge in the realisation of the objectives. Even if all the things we mentioned in this first chapter were very vast, they nevertheless allowed us to have a global vision of the environment and to build a trading algorithm by considering several aspects and not only the software side. From a critical point of view, it would have been possible to present less elements in this first part but from another point of view, it is important to have a global vision in the development of such a software.

The development part was successfully carried out while respecting the deadlines that we had previously set ourselves. The development plan was precise, and this allowed us to have a common thread to which to relate when problems arose. It took a long time to get going, partly because of the API of our broker, which we had to learn before we could start to imagine a plausible architecture. The rest of the implementation went on continuously without major downtime due to problems. The final goal of our thesis was reached when we managed to get our algorithm up and running and prove that it could place orders on the market autonomously.

The explanation of the software was one of the main parts. We wanted to give everyone the keys to be able to use or duplicate this project but also to go into enough detail so that readers really understand the workings of such a system. Now we are convinced that having development skills in the Java programming language is essential before starting to implement such a project. However, in our opinion, understanding the code is sufficient to be able to duplicate the code while being able to be aware of how the software works. We strongly advise against setting up a system like this for people who are beginners in programming, but in no case does prior knowledge of finance seem to be necessary. From an external point of view, we believe in our project and think that it could be reused for further research, notably for the elaboration of a more complex trading strategy as we will see in the section dedicated to this subject.

Overall, our work is what could reasonably be expected when we started thinking about this topic. We have taken care to design each part to facilitate the reader's assimilation. This is

one of the reasons why this report, which is certainly precise and summarises a lot of work, is so long. We did not want to deliberately come up with such a long document, but our desire to get to the bottom of things did not really allow us to be as concise as we would have liked from the start.

5.4 Learning experience

After a lot of effort, sleepless nights developing the software and cups of coffee, one thing is for sure; we have learned a lot from this work.

As for the reader, the theoretical part allowed us to assimilate the algorithmic trading environment in a concrete way. We have read and become familiar with many articles and books that would have been worth a longer look. Even so, this research has been very interesting. We have gained new knowledge about finance even though this was not the main purpose of this work. By already knowing a little about the stock market and how it works, we were able to delve into the emergence of algorithmic trading in detail. Understanding the advantages and disadvantages of these systems was also very exciting. We were able to see how finance has changed dramatically in recent years with all the new technologies. We also became aware of the abuses that are made with high frequency trading and how it was possible to manipulate the market, rather too easily in our opinion. After fully understanding how these new entrants in algorithmic trading software work, we were able to learn more about the architecture that is often used. This was one of the most complex but at the same time one of the most exciting points, as it was the one that most affected computer science and technology. In the last part we looked at the strategies used or at least the basic strategies that could be used. This part proved to be easily digestible. Although traders often use complex terms to define simple things, we still managed to find concrete and easier examples with situations outside the industry.

Then, the development part allowed us to apply all that we had learned previously. Having really seen the trading algorithm in all its facets allowed us to be efficient in this phase. Developing software from scratch was not an easy task. The hardest part was to imagine a structure, a reliable architecture that could match what we were going to do. In the end, the research we did on the subject for the writing of the theoretical part was a great help. The rest of the development still required a lot of research on the internet. Knowing the Java programming language gave us an advantage, but we still acquired many new skills. The need for simple and fast code so that the algorithm is also simple and fast forced us to rethink a lot of what we knew. These two requirements often forced us to act in a different, unconventional way. We had to balance between good programming practice and efficiency

in the code. For obvious reasons of simplicity and comprehensibility, we applied some programming patterns, but these patterns often have the unfortunate disadvantage of making the code heavy. Taking this into account, we managed to find compromises. This has been very beneficial in terms of learning as we have in a way moved away from development as we knew it before starting this project.

Finally, the documentation of the API of our broker, endless but really detailed, allowed us to add very easily functionalities that we did not even imagine at the beginning of the development. Once again, this documentation was beneficial in terms of learning. We honestly think that this is the first time we have seen such a long and precise document. So, reading this has allowed us to see what is being done in this industry at a professional level. The accuracy of the API was not an easy thing because we had no general functions but very specific ones. Often this was useful, but on other occasions we really could have got shorter methods.

Overall, apart from the finance skills we learned, we really learned a lot about developing software from scratch. We had to do all the necessary tasks whether it was connectivity, design, user experience and of course the software itself. We believe that the software that has been built in this project is of good quality and demonstrates all the skills that have been accumulated over the 3 years of study in software design.

Although we wanted to continue to develop new features, we focused on those that were most important to us to meet the deadlines. We had time to add most of the features we had imagined along the way, but software is never really finished and always benefits from updates after its development. Some of these will be mentioned in the section that follows this chapter.

6 Conclusion

This final chapter concludes the long-term work carried out in the framework of this project. In this sense, we will hear the author's opinion on the last elements that have not necessarily been addressed throughout this report. This opinion will serve as a conclusion. Before formally concluding, we will also mention the many developments or improvements that can be made in the future. These further developments include all those elements for which we did not have time or for which additional expertise was needed.

This last part is about my view of the subject, the industry, and my work. I have used the polite "WE" in all sections and chapters to include the readers as well as all the authors of the research I have read and consulted while writing this report. This chapter is solely about my own opinion and I will therefore use 'I'. This will be the only place in this document where I will allow myself to give my opinion independently. This choice not to include my opinion earlier in this document was made to keep as much objectivity as possible about all the work and writing that has been done so far.

First, I wanted to explain why I am so passionate about this topic and what factors led me to choose it over another. To understand this motivation, you only must go back a little in my professional career. I started by working in the banking industry. More precisely, I did a 3-year training in one of the biggest banks in Switzerland but also worldwide. This training gave me a very strong taste for finance, and I never stopped loving this field. This is still the case today.

After this first training, I decided to turn to a future profession; IT. In lack of knowledge in this very vast field, I logically decided to follow a new training in computer management. This training has been very enriching from a professional but also from a personal point of view. While still being very much involved in finance, I started to learn a lot of things in new technologies. When it came to choosing the subject of this work, it was obvious to me that I wanted to do something at the intersection of these two fields. Algorithmic trading was that perfect intersection, and I was very keen to put my knowledge and skills to work for all those amateurs who would be willing to build such a system in a free and fun way.

I think this topic is more than relevant in the context of today's society. Trading but especially algorithmic trading is something that many people "think" they know. It is not uncommon to find advertisements on social networks or on the internet promising huge gains from trading the stock market. Some even sell or offer investment recommendations on social networks. Becoming rich through trading overnight does not exist, or at least does not happen any

more often than winning the lottery jackpot. This myth deserves attention because it is problematic for Sunday investors who improvise themselves as professional traders. We must never forget that trading is a profession. A profession that is studied, learned, and practiced like any other profession. In the end, just as you cannot become a doctor overnight, you cannot become a trader either. As far as I am concerned, after 5 years of trading on the stock markets, I cannot even claim to be an experienced trader.

Certainly, the trading industry is full of opportunities to make money, and even a lot of money. While this may convince people to get into trading, it is important to note that about 90% of investors in the stock market lose money. This means that it is the other 10%, usually professionals, who win. I cannot tell you not to do it just as I cannot tell you to do it. I just want to point out that not knowing what you are doing in the stock market can result in the total loss of the investment made. A quote from Mark Twain that I like is “It's not what you don't know that gets you into trouble, it's what you know for sure that isn't true.” In fact, this quote applies perfectly to the stock market.

The trading industry is an area that will continue to amaze people as it represents the capitalism of our modern world. With the power of trading firms, high-frequency trading and even banks, any profound changes become impossible to implement by our governments. If change is not an option, then perhaps it is time to put in place regulations to limit the excesses that are being carried out as with the arbitrage we saw earlier. And even if no regulations are put in place, the simple fact of becoming aware of this could have an impact. For my part, before I started writing this thesis, I was only vaguely aware of the excesses of a few crazy traders or greedy companies. I am also convinced that it is the same for most investors who put money into the stock market nowadays.

One last thing I wanted to mention that I think is important. There are many such software programs sold on the internet. They promise miraculous returns in a very short time without any knowledge of finance or computers. Moreover, they are relatively cheap compared to the expected performance. The question is, why are they being sold if their returns are so high? The reality is that the algorithms that work very well are so well guarded that only a few people know about them. The cause of this is rather obvious; the multiplication of the same algorithm in the financial markets would inevitably lead to a decrease in performance on all replicas of it. In general, most of what you can find on the internet is not worth it. Let us never forget that during the gold rush, it was not the miners who made their fortune. It was the people who sold the picks and shovels...

6.1 Idea for further research proposal

To conclude this report, we will discuss ideas for a future research proposal. Despite the investment of time and energy in this work, there are some improvements that simply have not had time to be addressed and could be considered in the continuation of this project. We have identified some ideas for improvement or development that would add value to the software, and we will discuss them here.

To begin with, it would have been interesting to have a graph with the price evolution curve. This would have been particularly useful to quickly visualise when a purchase or a sale is made. To do this, it will be necessary to have a specific component in the drawing to be able to mark the evolution on a screen dedicated to this. This idea was not kept in our project because it was not a major part of the software at the time of development. But now that we have finished it, we think it can add value for the user.

Secondly, one of the weak points of our work is the strategy used. Due to lack of resources and knowledge, we used a basic algorithm that works but could be optimised. This optimisation alone would probably take the same amount of time as it took us to do this work but could be very interesting. It could for example be done by someone with a mathematical background as part of a bachelor or master thesis. This idea is not only about computer science but also requires other skills. A person who started in the world of computer science and is currently studying finance could be the ideal person to make this change.

Another problem we faced was the impossibility of doing algorithmic trading as we wanted to on stocks. As a reminder, this possibility would have cost money and we did not pursue it. Even though it would have to be paid for, it would be a useful major development. The person who would like to do this development could use the work that has been done so far without any cost. This modification would require a financial investment in the development as well as the use of a real trading account. For this reason, this idea would be outside the scope of this project and we mention it only because it is a possibility, we do not encourage it in any way.

Currently our system can observe one instrument at a time. To observe several instruments, it is therefore necessary to have access to several machines. One idea for further development of our project would be to be able to monitor several instruments at the same time. This modification would make it possible to capture more opportunities at once and increase the overall performance of the system. To do this, a global refactoring of the code

would be necessary but someone without knowledge of finance would easily be able to do it based on the work we have done. This could be interesting to be able to compare the performance between several assets at the same time.

Before finishing, we tested the strategy on historical data and tested our software on real data, but we never managed to test our software on historical data. The main reason is that a real account and a paid subscription to historical market data would have been necessary. If the optimisation of the strategy should take place in the future, then this modification would make sense. As it stands and with a demo account, the tests as we have conducted them seem to be sufficient. Again, this change would use a real account and would therefore be risky, which is why we cannot encourage it.

Anyone is free to use this work for further research. Nevertheless, we encourage users to stay within the framework that has been applied in the first place. That is, risk-free algorithmic trading for educational purposes. If developments should be made within this framework, we would be willing to collaborate to the best of our ability to help the research succeed. We also remind you that this software has an MIT license. This means that it can be used by anyone, anytime, anywhere and for anything, but that any research that will be done based on this project must also be licensed under the MIT license.

References

Albanese, C. (2015, September 29). *Forex scandal drives shift to algo trading*. Retrieved February 2021, from Financial News London: https://www.fnlondon.com/articles/forex-scandal-drives-shift-to-algorithmic-trading-20150929

Alti Trading. (2021). *Partie 1 – Histoire du trading : du forum romain à la création de la bourse de New-York*. Retrieved from Alti Trading: https://www.alti-trading.fr/histoire-du-trading-du-forum-romain-a-maintenant-partie-1/

Arte (Director). (2020). *Ces Financiers qui Dirigent le Monde - BlackRock* [Motion Picture]. Retrieved from https://www.youtube.com/watch?v=voSty1nfU-Q

Attac-Québec. (2010). *La Bourse contre la vie.* (February, Trans.) Editions MultiMondes.

Beales, R. (2020, July 6). *BlackRock is the new Goldman Sachs*. Retrieved April 2021, from BlackRocksBigProblem: https://blackrocksbigproblem.com/blackrock-is-the-new-goldman-sachs/#:~:text=BlackRock%20has%20been%20called%20%E2%80%9Cthe,in%20assets%20under%20direct%20management

Bershova, N. &. (2012). High-Frequency Trading and Long-Term Investors: A View from the Buy-Side. *Journal of Investment Strategies*, 2. doi:10.2139/ssrn.2066884

Black Rock. (2020, June 30). *Equity index rebalances*. Retrieved February 2021, from Black Rock: https://www.blackrock.com/au/intermediaries/ishares/equity-index-rebalances

Blaze Portfolio. (2019, February 25). *INTRODUCTION TO TRADE EXECUTION ALGORITHMS*. Retrieved February 2021, from Blaze Portfolio: https://blazeportfolio.com/blog/introduction-to-trade-execution-algorithms-2/

Borderie, A. (2009). *Les places financières internationales au lendemain de la crise.* RB édition.

Broking, A. (2020, September 7). *Angel Broking*. Retrieved April 2021, from HOW TO IDENTIFY ARBITRAGE OPPORTUNITY: https://www.angelbroking.com/knowledge-center/online-share-trading/arbitrage-opportunities

Café du Forex. (2018, April 25). *TRADING HAUTE FRÉQUENCE : CHIFFRES ET FONCTIONNEMENT*. Retrieved February 2021, from Café du Forex: https://www.cafeduforex.com/actualites/trading-haute-frequence-chiffres-fonctionnement

Canal+ (Director). (2015). *Nouveaux Loups de WallStreet* [Motion Picture]. Retrieved from https://www.youtube.com/watch?v=vOzH7Aj2MOA&t=2510s

Chandor, J. C. (Director). (2011). *Margin Call* [Motion Picture].

Chen, J. (2020, February 22). *Delta Neutral*. Retrieved February 2021, from Investopedia: https://www.investopedia.com/terms/d/deltaneutral.asp#:~:text=Delta%20neutral%20is%20a%20portfolio,of%20the%20position%20to%20zero.

Chen, J. (2020, April 26). *Rebalancing*. Retrieved February 2021, from Investopedia: https://www.investopedia.com/terms/r/rebalancing.asp

Chen, J. (2021, February 1). *Mean Reversion*. Retrieved February 2021, from Investopedia: https://www.investopedia.com/terms/m/meanreversion.asp

Coin Rule. (2020, July 30). *Trend Optimised Golden Cross Strategy*. Retrieved February 2021, from Coin Rule: https://coinrule.com/help/knowledgebase/trend-optimised-golden-cross-strategy/

Corporate Finance Institute (CFI). (2021). *Arbitrage*. Retrieved February 2021, from Corporate Finance Institute (CFI): https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/arbitrage/#:~:text=Arbitrage%20is%20the%20strategy%20of,equivalent%20assets%20with%20differing%20prices.

Covel, M. (2021). *Trend Following Theory by Michael Covel*. Retrieved February 2021, from Trend Following: https://www.trendfollowing.com/trend/

Davis, J. (2021). *6 Best Mean Reversion Trading Strategies(Stocks)*. Retrieved February 2021, from Johndeo Research: https://www.johndeoresearch.com/mean-reversion-trading-strategies/

Doubleday, R. (2013, April). *6 May 2010 US 'Flash Crash'.* Retrieved February 2021, from Research Gate: https://www.researchgate.net/figure/6-May-2010-US-Flash-Crash_fig2_276274908

Eric Budish, P. C. (2014, December). *The High-Frequency Trading Arms Race: Frequent Batch Auctions as a Market Design Response.* Retrieved February 2021, from Institut Louis Bachelier: The High-Frequency Trading Arms Race: Frequent

Evans, H. (2021, February 18). *The Role of High-Frequency and Algorithmic Trading*. Retrieved February 2021, from Velvetech: https://www.velvetech.com/blog/high-frequency-algorithmic-trading/#:~:text=The%20core%20difference%20between%20them,it%20a%20far%20superior%20option.

Fernando, J. (2020, November 17). *Relative Strength Index (RSI)*. Retrieved February 2021, from Investopedia: https://www.investopedia.com/terms/r/rsi.asp

Fernando, J. (2021, January 17). *Volume Weighted Average Price (VWAP) Definition*. Retrieved February 2021, from Investopedia: https://www.investopedia.com/terms/v/vwap.asp

Forex OX. (2018). *Construire une stratégie Delta-neutre*. Retrieved February 2021, from Forex OX: https://fr.forexox.com/constructing-delta-neutral-strategy-12242

FX Algo News. (2021). *Implementation Shortfall algo from HSBC*. Retrieved February 2021, from FX Algo News: https://fxalgonews.com/articles/implementation-shortfall-algo-from-hsbc.html#:~:text=NAME%20OF%20THE%20STRATEGY%3A%20IMPLEMENTATION%20SHORTFALL&text=To%20achieve%20its%20objective%2C%20the,the%20time%20interval%20between%20them.

Geneau, D. (2010, April 15). *Nyse Euronext se lance dans la colocation de centre de données*. Retrieved February 2021, from L'AGEFI Hebdo: https://www.agefi.fr/asset-management/actualites/hebdo/20160706/nyse-euronext-se-lance-dans-colocation-centre-193456

Ghosh, S. D. (2019). *Learn Algorithmic Trading.* Packt Publishing.

Gogerty, N. (2014). *The Nature of Value.* Columbia University Press.

Gupta, A. (2015, June 2). *History of Algorithmic Trading, HFT and News Based Trading*. Retrieved February 2021, from Quant Insti: https://blog.quantinsti.com/history-algorithmic-trading-hft/#:~:text=The%20Start%20of%20Algorithmic%20Trading,the%20late%201980s%20and%201990s.&text=HFT%20was%20able%20to%20execute,(HFT)%20has%20become%20widespread.

Hayes, A. (2020, June 23). *Implementation Shortfall*. Retrieved February 2021, from Investopedia: https://www.investopedia.com/terms/i/implementation-shortfall.asp

Held-Khawam, L. (2015, September 16). *Marché financier: un casino géant truqué qui nous gouverne*. Retrieved February 2021, from Citoyen du monde: https://philippefrossard.com/notables-et-pouvoirs/banquiers-productivistes/marche-financier-un-casino-geant-truque-qui-nous-gouverne/

Inc., I. (2005). *There are mainly three trading widely used trading benchmarks Œ Arrival Price, VWAP and Closing Price.* SIPC. Retrieved April 2021

Investissements Fonciers S.A. (2021). *Disclaimer*. Retrieved February 2021, from La Foncière: https://www.lafonciere.ch/disclaimer

Kissell, R. (2013). *The Science of Algorithmic Trading and Portfolio Management.* Elsevier Science.

Krugman, P. (2014, April 13). *Three Expensive Milliseconds*. Retrieved April 2021, from The New York Times: https://www.nytimes.com/2014/04/14/opinion/krugman-three-expensive-milliseconds.html

L., F. (2014, April 11). *Les «Flash Boys» déclenchent une curieuse tempête aux Etats-Unis*. Retrieved February 2021, from Le Temps: https://www.letemps.ch/economie/flash-boys-declenchent-une-curieuse-tempete-aux-etatsunis

La Finance Pour Tous. (2021, January 20). *Où se situe la Bourse de Paris ?* Retrieved February 2021, from La Finance Pour Tous: https://www.lafinancepourtous.com/outils/questions-reponses/ou-se-situe-la-bourse-de-paris/#:~:text=Le%20b%C3%A2timent%20de%20la%20Bourse,les%20titres%20boursiers%20en%20France.

Lash, H. (2011, April 8). *NYSE Euronext finishes move to Mahwah data center*. Retrieved April 2021, from Reuters: https://www.reuters.com/article/exchanges-nyse-mahwah-idUSN0821552320110408

Lemettre, J.-F. (2009, February). Quel futur pour l'industrie boursière ? Analyse d'un processus de transformation. *Innovations*(30), pp. 157-188. doi:10.3917/inno.030.0157

LiteForex. (2020, October 28). *VWAP Indicator: what is VWAP in trading and how to use it*. Retrieved February 2021, from LiteForex: https://www.liteforex.com/blog/for-beginners/best-technical-indicators/vwap-indicator-what-is-vwap-in-trading-and-how-to-use-it/

Markman, S. (2021). *Guerillas, snipers, sniffers, stealth bombers, welcome to the weird world of trading algorithms*. Retrieved February 2021, from Trader's Nest: https://www.tradersnest.com/blog/guerillas-snipers-sniffers-stealth-bombers-welcome-weird-world-trading-algorithms/

Mathur, N. (2019, February 14). *A Quick look at ML in algorithmic trading strategies*. Retrieved April 2021, from Packt: https://hub.packtpub.com/a-quick-look-at-ml-in-algorithmic-trading-strategies/

MJV Team. (2019, May 5). *Algo Trading: how algorithms are impacting the financial market*. Retrieved February 2021, from MJV: https://www.mjvinnovation.com/blog/algo-trading/

Pachanekar, R. (2019, November 28). *Automated Trading Systems: Architecture, Protocols, Types of Latency*. Retrieved February 2021, from Quant Insti: https://blog.quantinsti.com/automated-trading-system/

Papadopoulos, L. (2020, June 27). *There is a Secret $300 Million Cable Between New York and Chicago*. Retrieved February 2021, from Interesting Engineering: https://interestingengineering.com/video/there-is-a-secret-300-million-cable-between-new-york-and-chicago

Parker, T. (2019, June 25). *Has High Frequency Trading Ruined The Stock Market For The Rest Of Us?* Retrieved February 2021, from Investopedia: https://www.investopedia.com/financial-edge/0113/has-high-frequency-trading-ruined-the-stock-market-for-the-rest-of-us.aspx

Pinto, M. A. (2012). *Design and implementation of an algorithmic trading system for the Sifox application.* Porto, PT: Universidade do Porto. Retrieved March 2021, from https://core.ac.uk/download/pdf/302971773.pdf

Reed, E. (2021, February 2). *What Is High-Frequency Trading?* Retrieved February 2021, from Smart Asset: https://smartasset.com/investing/high-frequency-trading

Reid, S. G. (2014, May 17). *Algorithmic Trading System Architecture*. Retrieved March 2021, from Turing Finance: http://www.turingfinance.com/algorithmic-trading-system-architecture-post/

Renault, T. (2016, January 18). *Trading Haute-Fréquence : A la recherche de la vitesse de la lumière*. Retrieved February 2021, from Captain Economics: http://www.captaineconomics.fr/-trading-haute-frequence-a-la-recherche-de-la-vitesse-de-la-lumiere

Rousseau, A. B. (2013, August 14). *Les scandales financiers des traders français*. Retrieved February 2021, from Andlil Trader Inside: https://www.andlil.com/le-trio-infernal-des-traders-francais-ayant-fait-scandale-183180.html

Seth, S. (2020, March 7). *Basics of Algorithmic Trading: Concepts and Examples*. Retrieved February 2021, from Investopedia: https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp

Singh, M. G. (2009). *Electronic Exchanges.* Elsevier Science.

Stock Trading Warrior. (2021). *The Meteoric History*. Retrieved April 2021, from Stock-Trading-Warrior.com: http://www.stock-trading-warrior.com/History-of-Online-Stock-Trading.html

StockCharts. (2021). *RSI(2)*. Retrieved February 2021, from StockCharts: https://school.stockcharts.com/doku.php?id=trading_strategies:rsi2

Thakar, C. (2020, February 24). *Essential Books on Algorithmic Trading*. Retrieved February 2021, from Quant Insti: https://blog.quantinsti.com/books-algorithmic-trading/

Thakar, C. (2020, February 18). *Essential Mathematical Concepts for Algorithmic Trading*. Retrieved February 2021, from Quant Insti: https://blog.quantinsti.com/algorithmic-trading-maths/#Math-Concepts

Thakar, C. (2020, July 8). *Time-Weighted Average Price (TWAP) in Financial Markets*. Retrieved February 2021, from Quant Insti: https://blog.quantinsti.com/twap/

Trading Tuitions. (2017, April 20). *How to Calculate TWAP and use it for Trading?* Retrieved April 2021, from Trading Tuitions: https://tradingtuitions.com/how-to-calculate-twap-2/

Treanor, J. (2015, April 22). *The 2010 'flash crash': how it unfolded*. Retrieved April 2021, from The Guardian: https://www.theguardian.com/business/2015/apr/22/2010-flash-crash-new-york-stock-exchange-unfolded

United States government. (2021). *Types of Orders*. Retrieved April 2021, from Investor.gov: https://www.investor.gov/introduction-investing/investing-basics/how-stock-markets-work/types-orders

Vigaud, M. (2012, July 20). *Comprendre le Liborgate*. Retrieved April 2021, from Le Point: https://www.lepoint.fr/economie/comprendre-le-liborgate-20-07-2012-1487669_28.php#

Waikar, S. (2019, September 3). *Some High-Frequency Trading Strategies Can Damage the Stock Market's Health*. Retrieved February 2021, from KellogInsight: https://insight.kellogg.northwestern.edu/article/impact-of-high-frequency-trading

WH Self Invest. (2021). *TRADING STRATEGY: RSI 2P*. Retrieved February 2021, from WH Self Invest: https://www.whselfinvest.com/en/trading_strategies_31_RSI_2P.php

Wikipedia. (2020, November 3). *Aladdin (BlackRock)*. Retrieved February 2021, from Wikipedia: https://fr.wikipedia.org/wiki/Aladdin_(BlackRock)

Wikipedia. (2021, February 11). *Aladdin (BlackRock)*. Retrieved February 2021, from Wikipedia: https://en.wikipedia.org/wiki/Aladdin_(BlackRock)

List of figures

List of tables

Appendices (Backtesting of the strategy)

Appendix 1. Backtesting AUD/JPY [period Q1-Q2 2020] (Modified screenshot)

Appendix 2. Backtesting AUD/JPY [period Q3-Q4 2020] (Modified screenshot)

Appendix 3. Backtesting AUD/JPY [last 180 days] (Modified screenshot)

Appendix 4. Backtesting AUD/NZD [period Q1-Q2 2020] (Modified screenshot)

Appendix 5. Backtesting AUD/NZD [period Q3-Q4 2020] (Modified screenshot)

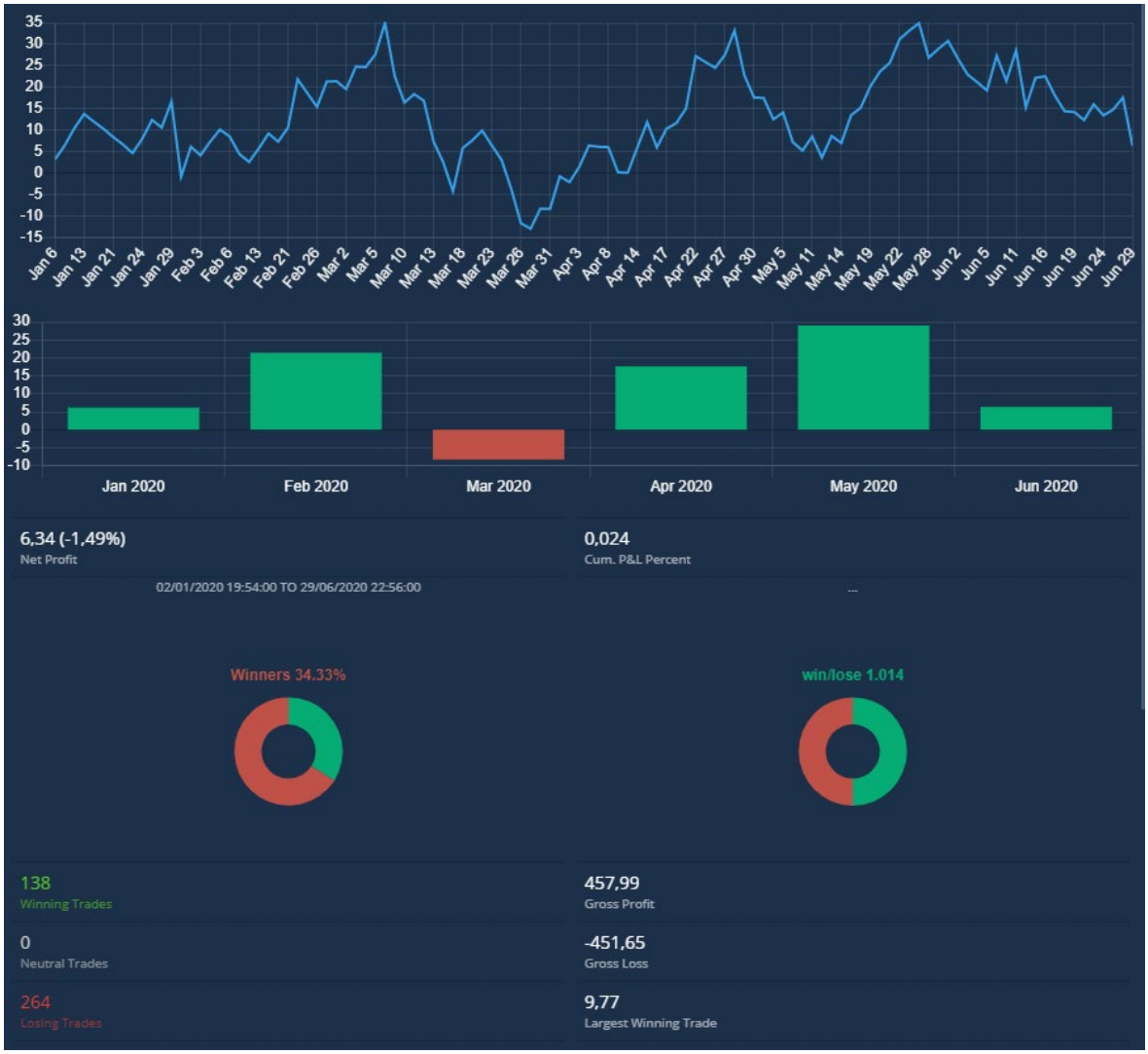
Appendix 6. Backtesting AUD/NZD [last 180 days] (Modified screenshot)

Appendix 7. Backtesting AUD/CHF [period Q1-Q2 2020] (Modified screenshot)

Appendix 8. Backtesting AUD/CHF [period Q3-Q4 2020] (Modified screenshot)

Appendix 9. Backtesting AUD/CHF [last 180 days] (Modified screenshot)

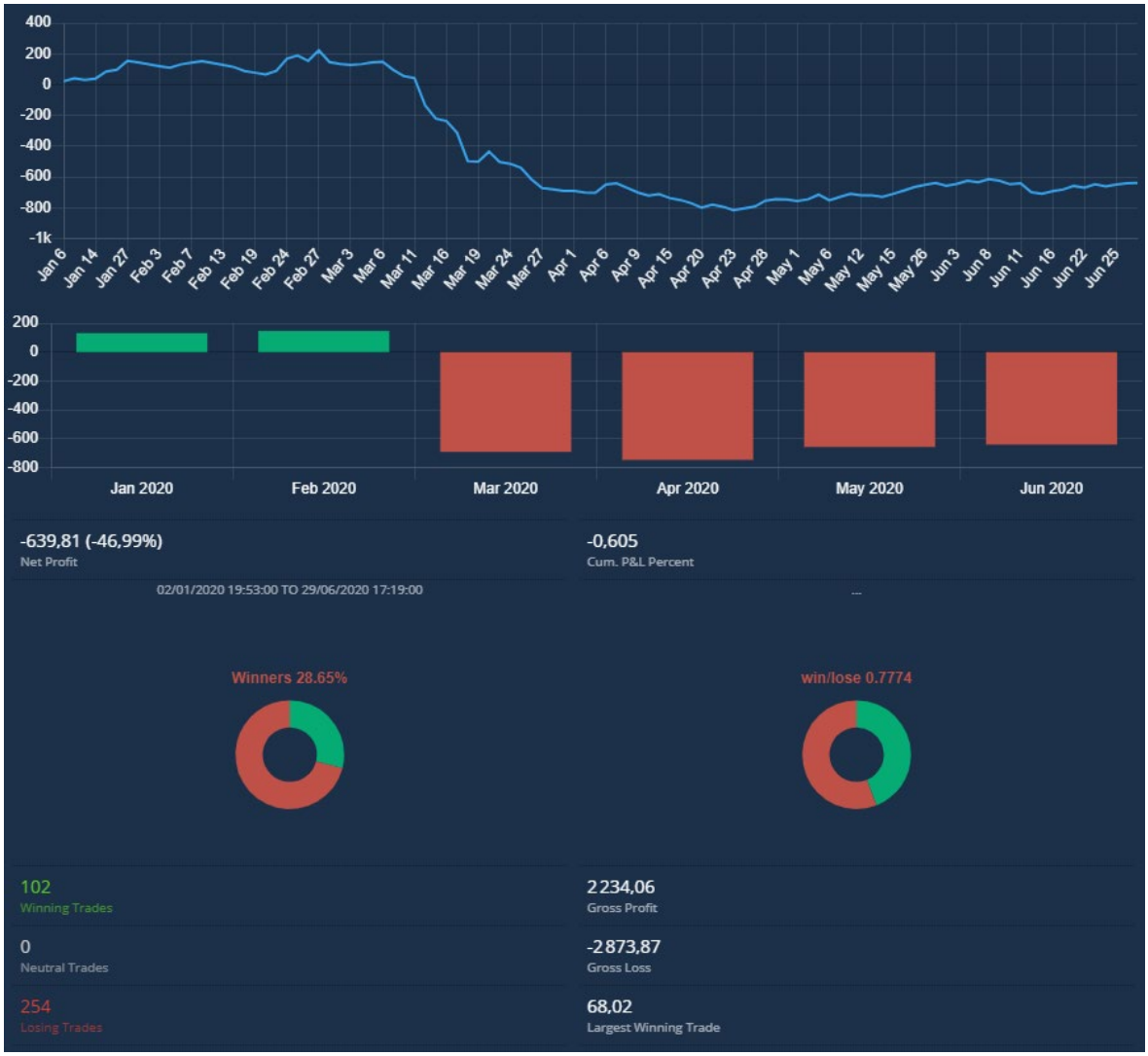
Appendix 10. Backtesting Facebook [period Q1-Q2 2020] (Modified screenshot)



Appendix 11. Backtesting Facebook [period Q3-Q4 2020] (Modified screenshot)

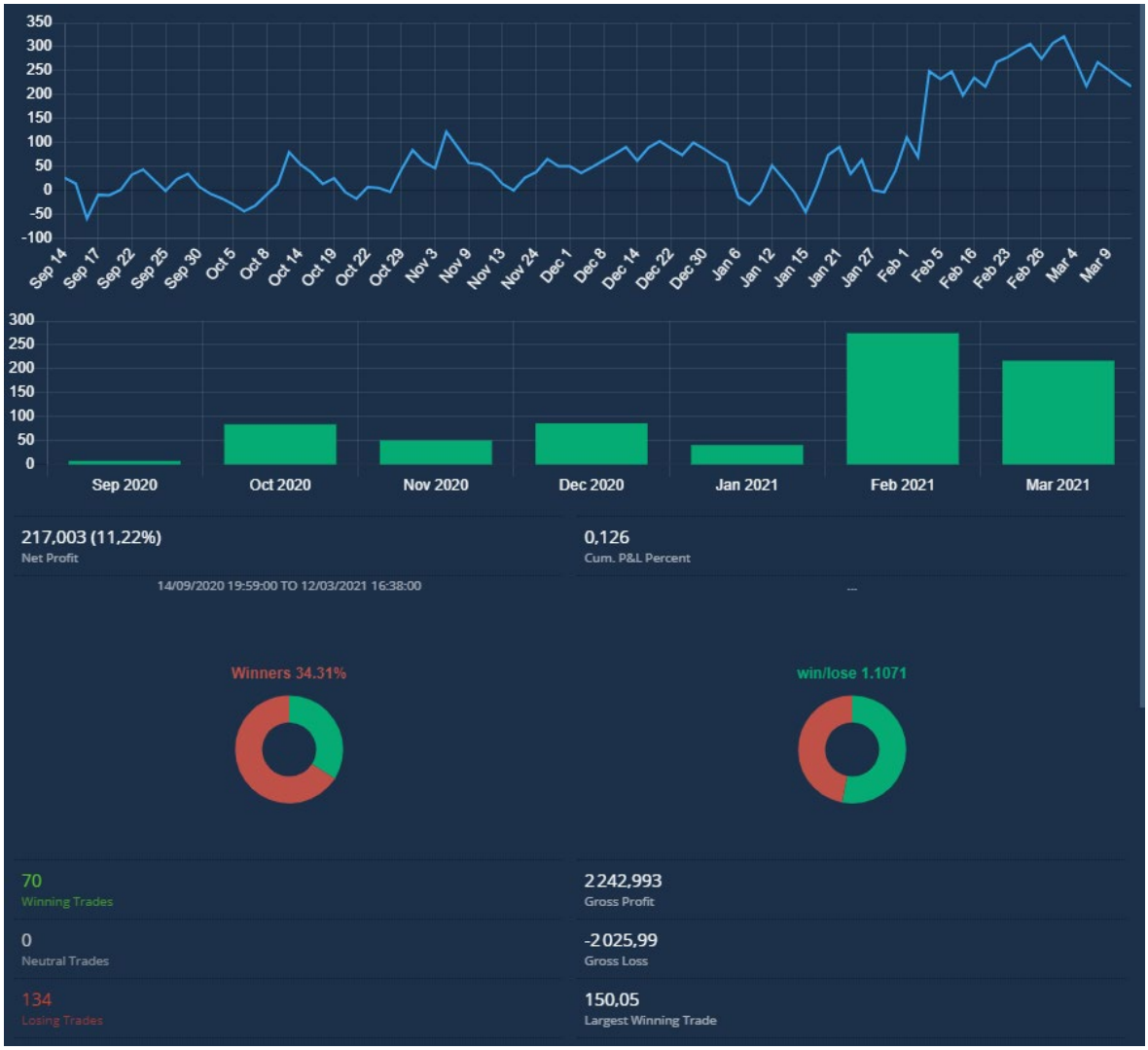
Appendix 12. Backtesting Facebook [last 180 days] (Modified screenshot)

Appendix 13. Backtesting Alphabet [period Q1-Q2 2020] (Modified screenshot)

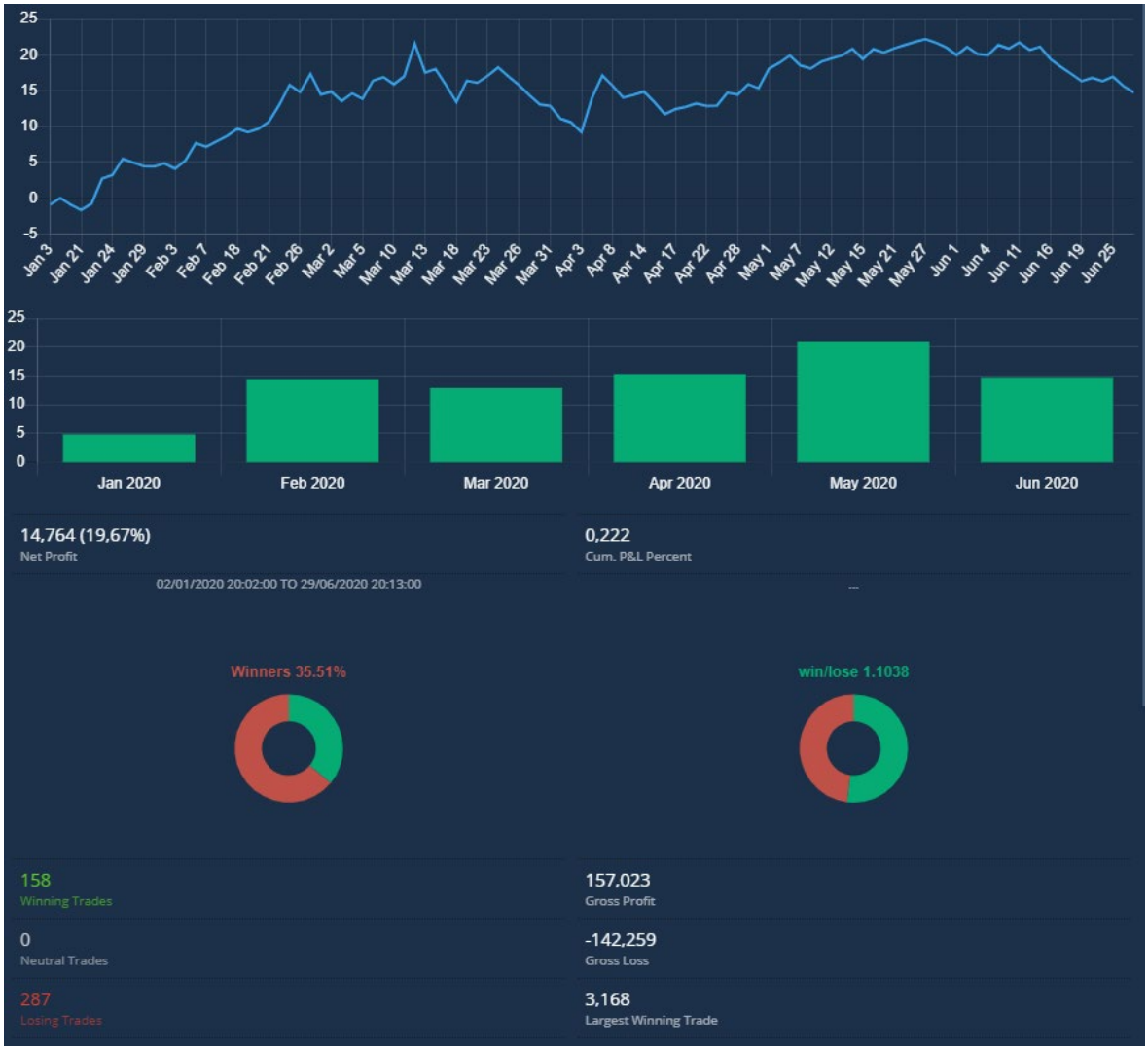


Appendix 14. Backtesting Alphabet [period Q3-Q4 2020] (Modified screenshot)

Appendix 15. Backtesting Alphabet [last 180 days] (Modified screenshot)



Appendix 16. Backtesting Intel [period Q1-Q2 2020] (Modified screenshot)



Appendix 17. Backtesting Intel [period Q3-Q4 2020] (Modified screenshot)

Appendix 18. Backtesting Intel [last 180 days] (Modified screenshot)