**Universidade de Évora - Escola de Ciências e Tecnologia**

Mestrado em Engenharia Informática

Dissertação

# Trading Simulator with Real Market Data

Marcelo Bento Silva Feliz

Orientador(es) | Pedro Salgueiro

Vitor Beires Nogueira

Évora 2023

**Universidade de Évora - Escola de Ciências e Tecnologia**

Mestrado em Engenharia Informática

Dissertação

# Trading Simulator with Real Market Data

Marcelo Bento Silva Feliz

Orientador(es) | Pedro Salgueiro

Vitor Beires Nogueira

Évora 2023

A dissertação foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

| | | |
|---|---|---|
| Presidente | \| | Teresa Gonçalves (Universidade de Évora) |
| Vogais | \| | Pedro Patinho (Universidade de Évora) (Arguente)<br>Pedro Salgueiro (Universidade de Évora) (Orientador) |

Évora 2023

*To My Beloved Sister*

*This dissertation is dedicated to my dear sister, Vitoria, whose strength and resilience in the face of adversity have been a constant source of inspiration. As you navigate the challenging path of life, may this work stand as a symbol of hope and a reminder that, even in the darkest moments, we can achieve greatness. Your unwavering love and support have carried me through this journey, and I dedicate these achievements to you with all my heart.*

# Acknowledgments

I would like to express my heartfelt gratitude to those who made this dissertation possible, marking the culmination of a significant chapter in my academic journey.

To my advisors, Pedro Salgueiro and Vitor Beires Nogueira, your guidance and unwavering support have been instrumental in the successful completion of this work.

To my parents, for their unconditional love, constant support, and the values they instilled in me. Without your encouragement, this achievement would not be possible.

To my beloved girlfriend, Carina, your understanding, encouragement, and love have been a continuous source of inspiration.

To my friends, who stood by me through the highs and lows, thank you for the laughter, emotional support, and for sharing this journey with me.

To Anchorage Digital, for providing the research opportunity and resources needed for this work. Your collaboration was indispensable in the development of this research.

To all who contributed in any way to this project, my heartfelt thanks. This achievement is the result of the collective efforts of many, and I am immensely grateful to each one of you.

Thank you.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

**OEMS** Order Execution Management System

**REST** Representational state transfer

**API** Application Programming Interface

**FX** Foreign Exchange

**NASDAQ** National Association of Securities Dealers Automated Quotations

**NYSE** New York Stock Exchange

**BSE** Bombay Stock Exchange

**NSE** National Stock Exchange (India)

**LSE** London Stock Exchange

**IDE** Integrated Development Environment

**OANDA** Online And Negotiable Dealing Automation

**CSV** Comma-Separated Values

**PoW** Proof of Work

**PoS** Proof of Stake

**SMA** Simple Moving Average

**ANNs** Artificial Neural Network

**HFT** High-frequency Trading

**DPoS** Delegated Proof of Stake

**PBFT** Practical Byzantine Fault Tolerance

**PoA** Proof of Authority

**PoET** Proof of Elapsed Time

**PoV** Proof of Vote

**PoT** Proof of Trust

# Abstract

## Trading Simulator with Real Market Data

This dissertation investigates cryptocurrency trading simulator enriched with historical market data as a testing tool, covering foundational financial principles, blockchain's impact, trading intricacies, and the role of trading systems and market data. It includes a thorough review of existing literature, emphasizing real market data.

The simulator, featuring three core modules, allows rigorous testing with trusted market data and efficient code structure. It outlines data collection and processing methods, addressing sources, sample sizes, reliability, and ethics.

This implemented simulator offers a realistic evaluation environment, interacting with a database module and supporting parameter customization. Rigorous testing validates its accuracy and reliability for system assessment.

The dissertation concludes by outlining the future of cryptocurrency trading simulator research, focusing on improving accuracy and functionality, with the potential for significant contributions to the field.

**Keywords:** Cryptocurrencies; Trading; Order book; Market data; Backtesting

# Sumário

## Simulador de negociação com dados de mercado reais

Esta dissertação investiga um simulador de negociação de criptomoedas desenvolvido com dados históricos de mercado como ferramenta de teste, baseando-se em princípios financeiros fundamentais, impacto da blockchain, complexidades da negociação, o papel dos sistemas de negociação e dos dados de mercado. Inclui uma revisão completa da literatura existente, com ênfase nos dados do mercado.

O simulador, apresenta três módulos principais, permite testes rigorosos com dados de mercado confiáveis e estrutura de código eficiente. O mesmo delineia métodos de recolha e processamento de dados, abordando fontes, tamanhos de amostra, confiabilidade e ética.

O simulador implementado oferece um ambiente de avaliação realista, interagindo com uma base de dados e suporta a personalização de parâmetros. Testes rigorosos validam a sua precisão e confiabilidade para a avaliação do sistema.

A dissertação conclui sumarizando o futuro de cripto simuladores, focando-se na melhoria de precisão e funcionalidade, com o potencial de contribuir significativamente para o campo.

**Palavras chave:** Cryptocurrencies; Trading; Order book; Market data; Backtesting

# 1

# Introduction and Motivation

Cryptocurrency buying and selling have experienced exponential growth in recent years, attracting the attention of businesses and investors. According to recent statistics from CoinMarketCap [1], the total market capitalization of cryptocurrencies has surpassed $2 trillion USD in 2021, a significant increase from just $17.7 billion in 2017. As a result, it is becoming increasingly important to create trading simulators using real market data to test and understand market conditions.

The objective of the present work is to enhance the testing infrastructure at Anchorage Digital, where I had the opportunity to work while pursuing my dissertation. In collaboration with Anchorage Digital, this work involves developing and implementing a sophisticated trading simulator powered by real market data. This simulator aims to simulate various market conditions and assess the system's responsiveness and performance under different scenarios. While the primary goal is not to create or optimize trading strategies, this work will focus on evaluating the trading system's infrastructure and ensuring its stability, reliability, and scalability within the cryptocurrency trading market.

Key components of the research endeavor include the creation of an order book that subscribes to live market data feeds, the development of a trade placement interface enabling the entry of trades with

[1] https://coinmarketcap.com/

diverse types and parameters, and the implementation of a replay framework that records and injects market data into the simulator. This integration with Anchorage Digital's existing test suite will enable seamless compatibility and facilitate comprehensive system evaluation.

An essential feature of the trading simulator is its capacity to act as a robust debugging tool. The ability to rewind time and analyze past market data allows for meticulous investigation in the event of any operational issues. This feature empowers Anchorage Digital to identify, understand, and rectify problems by thoroughly examining specific events, patterns, or anomalies that may have influenced the system's behavior.

Furthermore, the work presented in this thesis aims to continually upgrade the trading simulator to provide additional functionalities. This includes enhancing backtesting and historical analysis capabilities to evaluate the performance of different aspects of the trading infrastructure. Finally, scalability and performance optimization will be prioritized to ensure the simulator can handle large volumes of market data and simulate high-frequency trading environments.

In conclusion, this study endeavors to elevate the testing and understanding of cryptocurrency trading through the development of a trading simulator using real market data. By focusing on evaluating the trading system's infrastructure, stability, and scalability, as well as serving as a powerful debugging tool, the simulator aims to contribute to the overall efficiency of the cryptocurrency trading market. The continuous upgrading of the simulator will further enrich its functionalities and align with Anchorage Digital's objectives.

## 1.1 Motivation

Cryptocurrency trading has witnessed a remarkable surge in popularity and market value over the past few years, attracting significant attention from businesses, investors, and researchers alike. As the total market capitalization of cryptocurrencies surpasses trillions of dollars, it becomes increasingly vital to develop robust tools and systems to understand and navigate this dynamic market.

The motivation behind this dissertation stems from the need to enhance the testing and infrastructure of cryptocurrency trading systems, specifically focusing on Anchorage Digital's trading platform. While the primary objective is not to develop new nor to enhance existing trading strategies, this work aims to contribute to the efficiency and effectiveness of the trading market by creating a comprehensive trading simulator using real market data.

By simulating various market conditions, our research seeks to understand how Anchorage Digital's trading infrastructure responds and performs under different scenarios. This exploration will provide valuable insights into the system's strengths, weaknesses, and areas for improvement.

Moreover, a fundamental aspect of this research is to establish an order matching logic that accurately fills or rejects orders, allowing the calculation of performance measures and facilitating comparisons with other systems. By measuring and analyzing key performance indicators, we can identify areas of optimization and contribute to the continuous improvement of Anchorage Digital's trading platform.

Additionally, this work recognizes the significance of a robust debugging tool within the trading system. Hence, the developed trading simulator will enable retrospective analysis, allowing users to trace and investigate events in case of issues or anomalies. This retrospective capability will enhance the system's reliability and provide valuable insights into the causes of any problems encountered.

Ultimately, the work presented in this dissertation aims to advance the understanding and testing of cryptocurrency trading systems, specifically within Anchorage Digital, by leveraging real market data and a comprehensive trading simulator. By enhancing the infrastructure, identifying areas for improvement, and

offering a valuable debugging tool, this research contributes to the overall efficiency and effectiveness of cryptocurrency trading, benefiting Anchorage Digital and all of their costumers.

# 2

# Background

*This chapter serves as a comprehensive guide, providing insights into key topics that form the foundation of financial trading expertise. From blockchain technology's profound influence to the intricacies of cryptocurrency trading, trading strategies, and the vital role of trading systems and market data, this chapter offers an in-depth exploration of the intricacies of modern finance. Additionally, it delves into crucial aspects such as order types, the significance of order books, backtesting strategies, and the potential future developments and regulatory influences that shape the financial trading industry.*

In this chapter, we explore fundamental aspects of financial trading, from blockchain technology to trading systems and market data.

## 2.1 Blockchain

Blockchain refers to a technology where data is kept in a distributed ledger and participants can write, read, and verify transactions recorded in that ledger. However, it's not possible to modify or delete operations on

the transactions and other information that are in the ledger. Cryptographic primitives and protocols support and secure the Blockchain, e.g., digital signatures, hash functions, etc. Their function is to guarantee that the transactions are integrity-protected, authenticity-verified, and non-repudiated so they can be recorded in the ledger. Blockchain has several desirable features for its users, such as decentralization, autonomy, integrity, immutability, verification, fault tolerance, anonymity, auditability, and transparency. With these features, Blockchain technology has attracted a lot of attention in recent years [GY22].

One of the key aspects of blockchain is its decentralized nature. Unlike traditional centralized systems, where a single entity has control over the data, blockchain operates on a network of computers (nodes) that collectively maintain and validate the ledger. This decentralization not only reduces the risk of a single point of failure but also enhances security by making it exceedingly difficult for malicious actors to manipulate the data [GY22, NGHS17].

Autonomy in the context of blockchain refers to the ability of participants to interact directly without the need for intermediaries. Smart contracts, which are self-executing contracts with the terms of the agreement directly written into code, enable automatic and tamper-proof execution of agreements, further enhancing autonomy and reducing reliance on middlemen [GY22, DP17].

The integrity and immutability of blockchain stem from its design. Once a transaction is recorded on the blockchain and confirmed by the network, it becomes a permanent part of the chain of blocks. Each block contains a reference to the previous block, creating a chronological and unbroken chain of transactions. This design ensures that altering any past transaction would require the consensus and computational power of the majority of the network, making fraudulent changes practically infeasible [Pal19, ANA23].

Verification and transparency are inherent to blockchain due to its public nature. Anyone can examine the entire history of transactions on the blockchain, promoting trust and accountability. This transparency also plays a crucial role in industries where provenance and traceability are essential, such as supply chain management and food safety [GY22, Pal19].

The fault tolerance of blockchain is a result of its distributed nature. Even if a subset of nodes fails or becomes compromised, the network as a whole can continue to operate and validate transactions. This robustness makes blockchain especially resilient to attacks and system failures. [GY22]

Anonymity, while not an inherent feature of all blockchains, can be achieved in certain implementations. By using cryptographic techniques, users can transact without revealing their real-world identities. This has applications in privacy-focused transactions and systems [GY22, Pal19].

Auditability is facilitated by the transparent nature of blockchain. All transactions are visible and can be audited in real-time, simplifying compliance processes and reducing the need for complex and time-consuming reconciliation procedures [GY22, Pal19].

The blockchain technology can be classified into three categories: public or permissionless blockchain, private or permissioned blockchain, and consortium blockchain: 1) Public blockchains are an open network where anyone can participate, make transactions, and validate them without needing permission. They are decentralized and trustless networks that rely on consensus algorithms to maintain security and consensus, such as Proof of Work (PoW), Proof of Stake (PoS), or Delegated Proof of Stake (DPoS). Bitcoin and Ethereum are examples of public blockchains.; 2) Private blockchains are restricted networks in which participation, transactions, validation, and access to the ledger are exclusively available to a certain group. It's a centralized blockchain that uses as consensus algorithms Practical Byzantine Fault Tolerance Practical Byzantine Fault Tolerance (PBFT), Proof of Authority Proof of Authority (PoA), and Proof of Elapsed Time Proof of Elapsed Time (PoET), among others.; 3) Consortium blockchains are a hybrid type between the public and private ones. In this case, we can have several organizations controlling the consensus process

instead of a single one. The consensus algorithms used include Practical Byzantine Fault Tolerance PBFT, Proof of Vote Proof of Vote (PoV), Proof of Trust Proof of Trust (PoT), etc [AOJAF21, YA22, SJ19].

Some of the main mechanisms used to reach consensus in blockchain are: Proof of Work PoW is the first blockchain algorithm; it was created in 1993 and applied to the Bitcoin paper in 2008. Miners compete to solve complex mathematical puzzles, and the first one to solve one gets the right to add a new block of transactions to the blockchain. The advantage of the PoW algorithm is its high security and significant degree of decentralization. However, its main disadvantage is greater energy and resource consumption. Therefore, this algorithm is not indicated for a big and fast-growing network that requires a huge number of transactions per second [AOJAF21, YA22, SJ19].

Proof of Stake PoS was created as an alternative of PoW due to its large computing power consumption problem. In PoS the users don't have to solve the mathematical problem to achieve consensus; on the other hand, users only need to use cryptocurrency as a stake to achieve consensus. Creating new blocks and validating transactions are based on the number of cryptocurrency tokens they hold and are willing to "stake" as collateral. However, it is more energy-efficient compared to PoW, but raises concerns about centralization, especially if a small number of users hold a significant portion of tokens [AOJAF21, YA22, SJ19].

Delegated Proof of Stake DPoS is a variation of PoS and was proposed to improve PoS security by relying on stakeholder votes to pick block producers or witnesses; thus, it solves the concerns about centralization in PoS. The DPoS consensus mechanism doesn't require mining or complete node verification, making it simple and efficient. Instead, it is validated by a limited number of witness nodes and is also more power-efficient than PoW and PoS. Various versions of DPoS are currently in use by many blockchains like BitShares, EOS and Lisk [AOJAF21, YA22, SJ19].

Practical Byzantine Fault Tolerance PBFT, was proposed in 1999 to solve the problem of General Byzantine, a classic problem in distributed computing and cryptography. PBFT ensures that nodes in a network can reach a consensus on the order of transactions even when some nodes are malicious or fail. The maximum number of malicious nodes, f cannot exceed 1/3 of the total number of nodes; otherwise, it is required by PBFT. The process of this algorithm is divided into 3 phases: pre-prepare, prepare, and commit. In the pre-prepare phase,the leader proposes a block of transactions to the other nodes. Each node verifies the proposal and then broadcasts its agreement or disagreement with the proposed block. In the prepare phase, nodes that agree with the proposed block broadcast a "prepare" message. A node waits until it collects enough prepared messages from other nodes before moving on to the next phase. In the commit phase, it broadcasts a "commit" message. When a node receives enough commit messages from other nodes, it considers the block accepted. In each phase, a node advances to the next phase if it receives votes from more than two-thirds of all nodes. There are many variants of this algorithm, which is implemented by several blockchains like Hyperledger [AOJAF21, YA22, SJ19].

Besides the consensus mechanisms stated above, there are several others (e.g., Proof of Elapsed Time, Proof of Burn, Proof of Weight, Proof of Capacity, among others). Different blockchain projects choose consensus algorithms based on their specific goals, scalability requirements, security considerations, and the trade-offs they are willing to make between decentralization, energy efficiency, and performance. The choice of consensus algorithm is a critical decision in the design and operation of a blockchain network. By using consensus algorithms, the blockchain is seen as a trust machine, giving its users security among untrusted nodes [AOJAF21, YA22, SJ19].

In conclusion, the multifaceted features of blockchain make it a transformative technology with applications across various domains. Its potential to reshape industries, enhance security, streamline processes, and establish trust in a digital world continues to drive its widespread adoption and ongoing development.

### 2.1.1 Cryptocurrency

Cryptocurrency, which can be simply defined as digital currency or virtual currency, is a medium of exchange that functions like money but, unlike traditional currency, is independent from national borders and central banks. In other words, cryptocurrencies are utilized to facilitate online transactions without the need for a central intermediary to process them. The transactions are digitally recorded on a public ledger. Cryptocurrencies are exchanged through a peer-to-peer network, and the system ensures and documents consensus for any given exchange.For example, popular cryptocurrencies include Bitcoin (BTC), Ethereum (ETH), and Ripple (XRP). For any given coin held by the network, there is an evergrowing chain of code that tracks the transactions executed by the coin. Digital wallets store the cryptocurrencies, which can be connected to or disconnected from the internet. While internet-connected wallets can offer a convenient way to access cryptocurrencies, they are subject to cybersecurity risks [MAN+16, R21, MP18].

In turn, blockchain is the technology that sustains cryptocurrencies, but its application is not limited to just cryptocurrencies. However, cryptocurrencies are good use cases for blockchain technology, which makes full use of its advanced features. The blockchain technology used by cryptocurrencies, such as Bitcoin, is an open, distributed ledger that records transactions. With these characteristics, there is no need for a trusted third party, and solves the double-spending problem. When a transaction is agreed upon between users, the majority of computers in the network must verify that the transaction is valid before it can be added to the digital ledger. Blockchain technology operates without a central authority. Furthermore, the records that are created are irreversible and cannot be duplicated or changed [CGW17, PS21].

As stated before, different cryptocurrencies employ various consensus algorithms, such as Proof of Work PoW or Proof of Stake PoS, each with its own set of advantages and limitations. However, the adoption of cryptocurrencies and blockchain technology has not been without challenges. Regulatory uncertainties, scalability concerns, and environmental debates surrounding energy-intensive PoW mechanisms have raised questions about the sustainability and widespread acceptance of these technologies [R21, CGW17, PS21].

Although there are benefits to transactions with cryptocurrency, such as the use of a decentralized platform and the ability to transact 24 hours a day, seven days a week, there are also some risks to be concerned about, such as the risks associated with cybersecurity [R21].

In conclusion, by leveraging cryptographic techniques, cryptocurrencies provide a secure and decentralized means of conducting transactions in the digital realm. This innovative approach challenges conventional financial systems that rely on intermediaries like banks and governments. Blockchains and cryptocurrencies have become two of the most pressing topics in the financial industry.

## 2.2 Trading

Trading is a well-known concept that refers to any voluntary exchange of goods or services between several parties. In financial terms, trading refers to an investment strategy used to explore financial markets that are not in equilibrium, with the aim of taking maximum advantage of market deviations [Hay, EVDHM05, L+04].

Cryptocurrency, on the other hand, refers to digital money that has no physical representation, doesn't require a financial institution, and has no association with authority. Cryptocurrencies have emerged in the recent past as important financial software systems, gaining the attention of investors and companies. The first cryptocurrency, Bitcoin, was created in 2009 and refers to a peer-to-peer digital exchange system that uses cryptography to generate and distribute currency units. The value of cryptocurrencies isn't based on a country's economy; instead, it is based on the security of an algorithm that traces all the transactions

[MSH+16, Mil18, HU21].

When combined, these two concepts give rise to cryptocurrency trading, which refers to the act of buying and selling cryptocurrencies with the intention of making a profit. Cryptocurrency trading can be characterized in three aspects: object, operation mode, and trading strategy. The object of cryptocurrency trading is the asset being traded, which is "cryptocurrency." The mode of operation of cryptocurrency trading is determined by the method of transaction in the cryptocurrency market. Finally, a trading strategy in cryptocurrency trading, formulated by an investor, is an algorithm that defines a set of predefined rules to buy and sell on cryptocurrency markets [FVB+22, MSH+16].

## 2.3 Trading System

A cryptocurrency trading system, mostly used in exchanges, acts as a hub for buying and selling digital currencies. Users place orders to buy or sell cryptocurrencies, and the system matches these orders efficiently. It maintains a dynamic list of buy and sell orders for each cryptocurrency pair, showing the prices and quantities, helping traders make informed decisions.

The system provides crucial market data, such as price charts, trading volume, and historical data, enabling traders to analyze market trends. Users typically need a digital wallet to store their cryptocurrencies, and many systems offer integrated wallets for convenience.

Security is a top priority, with measures like two-factor authentication and encryption in place. Advanced users can use Application Programming Interface (API)s for automated trading strategies. Exchanges also maintain liquidity pools to ensure smooth trading.

Traders should be aware of fees and commissions associated with trading, and they can interact with the system through user-friendly interfaces. Regulatory compliance, including identity verification, is crucial, and customer support is available for assistance [BTNS19].

## 2.4 Orders

There are several different types of orders to buy or sell assets, we will discuss the most common and straightforward types: Market Orders and Limit Orders. Market orders refer to an order to buy or sell an asset at the current best available price. Market orders are a good option when the goal is to execute the trade immediately. Market orders are subject to potential fluctuations between the broker's receipt and execution of the trade. For example, larger orders take longer to fill and, if large enough, can actually move the market on their own. A market order is generally appropriate when you think a stock is priced right, when you are sure you want a fill on your order, or when you want an immediate execution. Thus, a market order is typically adequate for buyers who believe a stock is priced correctly or require immediate execution, however, despite having guaranteed execution, the purchase may not be made at the price the buyer wanted [Sch, Inv].

On the other hand, a limit order is an order to buy or sell a stock at a specific price or better. A buy limit order can only be executed at the limit price or lower, while a sell limit order can only be executed at the limit price or higher. A limit order is suitable when the buyer believes they can buy or sell at a lower or higher price than the current quote. Limit orders provide investors with greater control over their trades' buying and selling prices. It's important to keep in mind that if the market price never falls within the limit order guidelines, the investor's order may fail to execute. The asset may reach a target price, but there may be insufficient liquidity to fulfill the order when its turn comes [Sch, Inv].

## 2.5 Market Data

Market data refers to the transmission, in real time, of trade-related data. It contains a variety of data including price, bid and ask quotes, and market volume. Market data is available across several global markets, including stocks, indices, forex, and commodities. It is used to assess the worth of various assets. Traders use market data to get as much information as possible in order to calculate market risk.

One potential improvement to the use of market data is the development of more advanced tools and techniques for analyzing it. For example, machine learning algorithms could be used to analyze market data in real time and identify patterns that are not immediately obvious to human traders. This could lead to more accurate predictions of market trends and better trading decisions. Another area for improvement is the integration of market data with other sources of information, such as news and social media. By analyzing market data in the context of external events and trends, traders could gain a more complete picture of market conditions and make more informed decisions.

Market data has some advantages; it is generated in real time, which means that it can be used to make quick but informed trading decisions. It is also used to access historical prices, which are a crucial part of technical analysis and can be useful when creating a strategy for future trades. Market data is easily accessible and typically separates market pricing data from other information, however some market data providers also offer fundamentals. In terms of actions, this can include market valuations, company performance reports, and reference data.

However, it is important to be aware of the potential drawbacks of market data. There is always the risk of latency or lags in the delivery of information, especially since the information can come from trading venues all over the world. Therefore, it is important to ensure that the data provider is reliable and has the capabilities to provide high-speed access to accurate market data. Additionally, market data can be complex and difficult to interpret, especially for novice traders. As a result, it is important to have a good understanding of the underlying concepts and to be able to use the data effectively in order to make informed trading decisions [VT21, AD20, BBP06, Spa].

## 2.6 Order Book

The order book refers to a list of orders from both buyers and sellers for a specific asset, showing the prices and the volumes that both parties are willing to buy or sell at. Originally used on stock exchanges, order books are now becoming popular in cryptocurrency trading and are simply fundamental for any asset exchange.

Order books are mostly digital, providing traders and financial analysts with real-time market pricing information. As a critical tool in cryptocurrency trading, the order book is widely used by traders worldwide. Moreover, the order book can match automatically according to traders' preferences. Every order book has a depth level, which refers to the number of price levels available at a particular time in the book. The depth level can be fixed, and orders beyond that depth are ignored or rejected, or it can contain unlimited levels. To choose the depth level, there are some tests that can be executed to determine which depth level is more suitable.

The characteristics of the order book vary depending on the specific asset, but they typically have several components, including the buyer's side (bid) and seller's side (ask). Some order books use the terms "bid" and "ask" to describe the buyers' and sellers' orders, respectively. The buyer's side is typically displayed on the left, while the seller's side is on the right, often colored green and red, respectively. The prices displayed indicate the value interest of both sides, while the offers in the order book are represented by a

table of numbers showing the prices and total amounts from both sides. This visual representation enables traders to quickly gain an overall understanding of market demand and supply. In every stock trade, there is a difference between the asking and the bidding price for every asset; this difference is called the spread. Therefore, the spread represents the difference between the highest price a buyer is willing to pay and the lowest price a seller is willing to accept. In simpler terms, if an asset has a small difference between the buying (bid) and selling (ask) prices, it is considered more liquid. On the other hand, if the spread between bid and ask prices is large, the asset is considered less liquid. In essence, assets with more buyers and sellers have tighter spreads, making them easier to trade efficiently.

By using an order book, traders can benefit from understanding the market and gaining as much information as possible about a specific asset. This knowledge can inform trading decisions and help traders stay ahead of the competition. However, it's important to keep in mind that order books can be volatile, and traders need to have a clear strategy and risk management plan in place when using them [Ken, Roş09, CST10].

## 2.7 Backtesting

The term "backtesting" is used in several different ways in finance. Most commonly, backtesting refers to either 1) an assessment of the hypothetical historical performance of a suggested trading strategy or 2) the evaluation of financial risk models using historical data on risk forecasts and profit and loss realizations [Chr08].

Using backtesting, a trader can simulate a trading strategy by using historical performance, which allows for an analysis of risk and profitability before trading with their own capital. Based on the results of backtesting, traders can modify their strategy accordingly. To be truly efficient, it's important to choose a sample of data from a relevant time period that reflects several market conditions. The selected historical data should be representative in order to be as accurate as possible. By doing so, traders can understand whether the results of the backtest represent a fluke or a good negotiation. The use and good correlation between out-of-sample testing and forward performance testing determine the viability of the trading system.

All quantified ideas can be backtested, but a programmer may be required to code the idea into the trading platform's language. To help the trader, the programmer can make it possible to change the system, such as by using the simple moving average Simple Moving Average (SMA) crossover system. In this case, the trader could change the lengths of both sides to backtest which lengths would have a better performance based on historical data and improve their strategy.

Despite the benefits of backtesting in predicting the viability and profitability of a trade, traders have to be careful and cautious in the development of the system. For example, different data sets should be used in backtesting to understand the true performance and viability of the data, and traders should avoid bias in making decisions. In-sample testing should be followed by backtesting with data from another time period to see the correlation between them. If the results are similar, it's more likely that they are valid [Cam05, HL15].

## 2.8 Potential Future Developments on financial tradings

Over the last few years, there has arisen an exponential interest in the use of machine learning algorithms for financial trading; one popular method is the use of artificial neural networks Artificial Neural Network (ANNs). Machine learning algorithms use complex mathematical models to analyze large amounts of financial data and are capable of identifying patterns and trends that may be difficult for human traders to detect. These algorithms automatically improve themselves by finding patterns in existing data, without

explicit instructions. On the other hand, artificial neural networks, unlike traditional modeling, are valuable because they include a self-training component, so the algorithm can learn from past data to make predictions about future market movements.Besides ANNs, other machine learning techniques have been applied in financial trading and have shown results in predicting market trends, identifying anomalies, and detecting fraud, such as decision trees, random forests, and support vector machines [Tsa, VF09, FVB+22].

The use of big data analytics and cloud computing has also had a significant impact on financial trading. These approaches have the ability to process and analyze massive amounts of data in real-time. This way, traders can make more informed decisions and identify opportunities more quickly.

In the area of financial trading, there has been an increase in the popularity of cryptocurrency trading. Cryptocurrencies (e.g., Bitcoin, Ethereum, and Litecoin) have gained significant attention from investors due to their decentralized nature and potential for high returns. However, trading cryptocurrencies comes with unique challenges, such as high volatility and limited liquidity. As a result, new trading strategies and algorithms are being developed specifically for cryptocurrency markets [FVB+22, Ata11].

The state of the art in financial trading is constantly evolving, with new technologies and approaches being developed and refined. This constant evolution is likely to have a significant impact on the financial industry, transforming the way trading is conducted and the types of opportunities that are available to investors.

## 2.9 Impact of new technologies and approaches on financial trading

Over the last few years, the financial industry has seen an incredible increase in regulations. Because of the global financial crisis of 2008, regulators around the world have created new rules and requirements with the objective of bringing transparency, justice, and stability to financial markets. These regulations have had a significant impact on financial trading, from the types of investments to the amount of leverage used.

One important area of regulation is the use of automated trading systems (algorithmic trading). The potential risk of these systems has been raising some concerns, mainly among regulators, like their potential use for market manipulation or their ability to trigger large-scale market movements. As a result, many jurisdictions have created new rules and requirements for algorithmic trading, such as required testing and risk control [Ata11, FVB+22].

On the other hand, high-frequency trading High-frequency Trading (HFT) strategies are another area of regulation. HFT concerns a type of algorithmic trading that uses sophisticated algorithms to execute trades at extremely high speeds, often in milliseconds or microseconds. One positive aspect is that HFT can facilitate trades at a lower cost; however, the speed of HFT could put other market participants at a disadvantage. Regulators have expressed concerns about the potential risks of HFT, such as the potential for increased market volatility or the potential for market abuse. Thereby, many jurisdictions have introduced new rules and requirements for HFT, such as mandatory registration and reporting [Jon13, BBHK19, Ata11].

Other areas like dark pools are also included in the regulations; these are private trading venues where traders can buy and sell securities without interacting directly with the broader market. Regulators are expressing some concerns about potential risks, like market fragmentation or the possibility of conflicts of interest. Because of that, some rules and requirements for dark pools are being raised, such as mandatory reporting and disclosure [Ata11, JPSW19, BRW22].

## 2.10 Conclusion

In conclusion, this chapter has provided a foundational understanding of key concepts in financial trading, from the transformative power of blockchain technology to the dynamics of cryptocurrency trading and the critical role of trading systems. We've explored various aspects, including consensus algorithms, order types, market data, order books, and backtesting. Moreover, we've delved into the potential future developments and regulatory influences shaping the financial trading industry.

As we move forward in this dissertation, the next chapter will build upon this foundation, focusing on the practical application of these concepts. We will explore in depth how a crypto simulator serves as a valuable testing tool in the ever-evolving world of financial trading. By investigating the functions, advantages, and real-world use cases of such a simulator, we aim to provide valuable insights that contribute to the ongoing development of efficient and secure trading strategies. By bridging the theoretical understanding outlined in this chapter with the practical insights to come, we strive to offer a holistic perspective on the role of crypto simulators in shaping the future of financial trading.

# 3

# State of art

*This chapter reviews the literature and research on trading simulators, focusing on those that uses real market data. It explores the functions, features, and value of incorporating actual market data, as well as evaluates the simulation engines and algorithms used in leading trading simulators. This works aims to expand our knowledge of trading simulators and their potential as tools for improving trading efficiency.*

For testing and assessing trading strategies across a variety of financial markets, including cryptocurrency, trading simulators have grown in popularity. Before implementing their methods in the real world, traders can test them in a risk-free setting by simulating trades using historical market data. Recent developments in data processing, machine learning, and other technologies have fueled the creation of trading simulators. Simulators have improved in sophistication and accuracy as a result, giving traders a strong tool for enhancing their trading success.

The literature and research on trading simulators will be reviewed in this chapter, with a focus on those that use real market data. We will review the functions and features of current simulators, discussthe value of incorporating actual market data, and assess the simulation engine and algorithm that we utilized to create our own trading simulator. We expect that this research will expand our knowledge of trading simulators and their potential as a tool for raising trading efficiency.

## 3.1 Real Market data Simulators

In this section, we examine TradingView, QuantConnect, Coinigy, and Backtrader, offering insights into their capabilities and how they contribute to the world of trading. TradingView, known for its real-time data and technical analysis tools, appeals to a wide audience. QuantConnect empowers users to create and test trading strategies using Python and C#, making it a valuable resource for algorithmic traders. Coinigy focuses on cryptocurrency trading, simplifying portfolio management, and offering advanced charting tools. Lastly, Backtrader, an open-source Python framework, is tailored for backtesting and trading algorithm development.

### 3.1.1 TradingView

Including stocks, futures, Foreign Exchange (FX), and cryptocurrencies, TradingView is a web-based platform that offers real-time data and charts for a variety of financial markets. The platform is a well-liked tool for technical analysis because it provides sophisticated charting capabilities and a wide range of technical indicators [Tra, Tay21].

The portal offers current information and quotes from numerous international exchanges, including National Association of Securities Dealers Automated Quotations (NASDAQ), New York Stock Exchange (NYSE), Bombay Stock Exchange (BSE), National Stock Exchange (India) (NSE), and LSE London Stock Exchange (LSE). Users have access to news feeds, historical data, and real-time streaming quotes. Additionally, TradingView offers an effective charting tool that enables users to design unique charts using a range of chart styles, periods, and technical indicators. More than 100 technical indicators and drawing tools are available on the platform to help users evaluate market movements and spot trading opportunities [Tay21, AK18, Tre].

TradingView features a sizable and vibrant community of traders who communicate their thoughts and insights via the social network of the platform. Users can share charts, debate trading techniques, and follow other traders. Users of TradingView may also create alerts that send them notifications when certain market conditions are satisfied. Price levels, technical indications, and news events can all trigger alerts [Tay21, AK18, Tre].

MetaTrader, Interactive Brokers, and TD Ameritrade are just a few of the brokers and trading platforms that TradingView offers interaction with. Both free and premium membership options are available from TradingView, with the paid plans including extra features like more data and more sophisticated charting tools [Tra].

Overall, because of its sophisticated charting features, real-time data, and vibrant trading community, TradingView is a well-liked platform among traders and investors. The software gives users a wide range of tools for researching market patterns and spotting trading opportunities.

### 3.1.2 QuantConnect

QuantConnect gives users the resources and tools they need to create, test, and use trading strategies. For creating trading algorithms, the platform enables the usage of Python and C# programming languages. Users can test their trading methods against historical market data using the backtesting engine offered by QuantConnect to assess their performance and improve their algorithms [IZO17, Tay21, Her22, HSK20, Spö20].

To test their trading techniques, users get access to more than 15 years worth of financial data from international markets, including equities, futures, options, and FX. Users of the platform can write, test, and debug their algorithms using an integrated development environment Integrated Development Environment (IDE) that is offered by the platform. A code editor, debugger, and notebook interface for data exploration and analysis are all included in the IDE. Using interfaces with a variety of brokerage accounts, such as Interactive Brokers, Online And Negotiable Dealing Automation (OANDA), and Tradier, users can apply their algorithms in actual trading situations after a strategy has been evaluated and improved through backtesting. Additionally, the QuantConnect platform features a sizable and vibrant community of algorithmic traders and engineers who communicate through the forum [Tay21, Her22].

To assist users in learning about algorithmic trading and advancing their abilities, QuantConnect provides a variety of instructional materials, such as video lessons, documentation, and webinars. The platform offers a cloud-based infrastructure for high-speed data processing and analysis, allowing for quick and effective backtesting and deployment of trading algorithms [IZO17, Qua].

QuantConnect is an all-encompassing platform that offers traders and developers the instruments and resources required to create, test, and implement trading strategies. It is an excellent resource for traders of all skill levels due to its emphasis on community and education [Tay21, Her22].

### 3.1.3 Coinigy

A cloud-based application called Coinigy gives cryptocurrency dealers the tools they need to manage their portfolios and carry out deals on several exchanges. Users of the platform can keep an eye on their holdings, follow their gains and losses, and examine market patterns across many exchanges using a single dashboard that is provided. Users of Coinigy can trade on many cryptocurrency exchanges using a single interface. On exchanges like Binance, Coinbase, Kraken, and Bitfinex, among others, users can make orders. Additionally, Coinigy offers a portfolio management tool that enables users to keep track of their holdings across several wallets and exchanges. The tool offers real-time data on asset allocation, profit and loss, and portfolio value [Coi, TEA, Goo].

Users can utilize the platform's sophisticated charting and technical analysis capabilities to examine market patterns and spot trading opportunities. With Coinigy, you may access more than 75 technical indicators and numerous chart types. Additionally, the portal offers news feeds and real-time market data from a variety of sources, such as Bloomberg, Reuters, and CoinDesk. Users can personalize their news feeds to keep up with the most recent bitcoin market movements. Additionally, Coinigy provides a trading bot that enables users to automate their trading methods. The bot can be set up to carry out trades in accordance with preset criteria and rules [Coi, TEA, Goo].

Both free and premium subscription plans are available through Coinigy, with the paid plans including extra features like access to more trading pairs and more sophisticated technical analysis tools. In conclusion, Coinigy is a complete trading platform for cryptocurrencies that gives users access to a number of exchanges, cutting-edge charting tools, portfolio management, and trading automation capabilities [Coi, TEA, Goo].

### 3.1.4 Backtrader

Backtrader is an open-source Python framework for backtesting and trading algorithm development. Using historical data, it offers a straightforward and adaptable interface for developing and testing trading strategies. The platform has many features that make it a top pick among quantitative traders [Bac, WYJ+20].

Backtrader provides users with a straightforward and adaptable API that enables them to design unique trading strategies. Comma-Separated Values (CSV) files, Yahoo Finance, and Interactive Brokers are just a few of the many data sources that are accessible through the platform. Additionally, the platform offers numerous timeframes, enabling traders to test their methods on various timescales. Trading methods can be tested using past data thanks to Backtrader's sophisticated backtesting features. Traders can assess the effectiveness of their strategies using the platform's extensive range of performance metrics. These indicators include Maximum Drawdown, Sharpe Ratio, Sortino Ratio, and others [Dav, Bac].

Additionally, the platform offers live trading features that let users put their plans into action immediately. Multiple brokers, including Oanda, and Alpaca, are supported by Backtrader. Market orders, limit orders, stop orders, and other order types are available on the platform. Furthermore, Backtrader offers a selection of visualization tools so that users may see the outcomes of their backtesting. The software provides many different chart kinds, including line charts, candlestick charts, and more [Dav, Bac].

Overall, Backtrader is a strong and adaptable platform for backtesting and the creation of trading algorithms. The platform offers comprehensive backtesting capabilities, live trading capabilities, a straightforward and adaptable API, and visualization tools. Since it is open-source and has a wide range of capabilities, it is a preferred option among quantitative traders [WYJ+20].

## 3.2 Conclusion

After all the trading simulators have been reviewed, it is evident that each platform has advantages and disadvantages of its own. While the majority of platforms provide strong backtesting engines to test trading methods, not all of them provide a complete solution to test an enterprise's trading infrastructure.

Any trading plan must include the trading system, which is the method utilized to acquire or sell assets. Testing the trading system, however, can be difficult, particularly for businesses that need a tailored solution to meet their particular trade infrastructure. To make sure that their trading system is operating as intended in such circumstances, traders may need to build their own testing infrastructure.

To test their trading system in a real-world setting is one of the key reasons some developers may choose to create their own trading simulator. By doing this, developers can pinpoint potential problems with their infrastructure for trading and enhance the functionality of their system. This is crucial for businesses who need a tailored solution to meet their unique requirements and have sophisticated trading infrastructure.

The ability to completely manage the testing environment, cost-effectiveness, and a competitive edge are just a few benefits of building an unique trading simulator. Having a special testing infrastructure allows traders to thoroughly test their trading system and perhaps even find new trading possibilities that may not be accessible through other trading simulators. Therefore, for traders who seek a customized solution that fits their particular objectives and aids them in achieving lucrative trading, building a customized trading simulator can be a useful investment.

# 4

# Methodology and Design

*In this chapter, we present the methodology and design of a cryptocurrency trading , comprising three key modules: the database, populate, and the simulator. These modules facilitate rigorous testing of the trading system using real market data from trusted sources. The modular design streamlines code structure and enhances user-friendliness. Our implementation harnesses Python, Golang, websockets, the BigQuery database, and efficient testing frameworks. Detailed insights into each module will be provided in later chapters.*

The primary goal of this project is to further enhance the testing infrastructure of the trading system at Anchorage Digital by allowing simulation of different market conditions and assessing how the system reacts to them. By leveraging real market data, the simulator facilitates the exploration of advanced trading strategies and in-house back testing. This capability is critical to meet the demands of a high-frequency cryptocurrency trading platform that operates 24/365.

## 4.1 Methodology

The overall methodology for building the trading simulator with real market data involves several key steps, as seen in the Figure 4.1. Firstly, we will identify and gather real market data from reliable sources such as

Coinbase, Kraken, and other dealers using an Order Execution Management System (OEMS) that exposes a Representational state transfer (REST) API. The collected data will undergo a validation process to ensure the accuracy of the number of bids and offers for each trading pair. This validation step is crucial to maintain data integrity within the simulator and ensure the fidelity of market conditions.

One of the primary objectives of the simulator is to serve as a robust testing tool for the existing trading system. To achieve this, we will simulate various market conditions using the real market data obtained. By generating repeatable results, we can compare the output of the trading system against the expected outcomes. This comprehensive testing process will help identify any discrepancies or issues in the functionality and performance of the trading system, allowing for timely improvements and risk mitigation. The simulator will have the capability to execute trades using different order types, simulate price slippage, and generate detailed execution reports. It will also support the capturing and replaying of production market data, enabling the simulation of specific scenarios for in-depth analysis. Moreover, we will provide flexibility within the simulator to inject custom market data, enabling internal teams to reproduce specific market conditions and produce repeatable results for the targeted test cases.


Figure 4.1: Overall system architecture

## 4.2 Risks of Using Third-Party Data Sources

There are inherent risks involved when relying on the accuracy and dependability of the data provided when using third-party data sources. [Liaa] Although we intend to verify the quantity of offers and bids for each trading pair, it's crucial to take other aspects of data quality into account as well, such as potential biases or sluggish updates. It becomes increasingly important to assess and track the effectiveness and consistency of data from each source as the simulator grows and incorporates data from more dealers. We will put several strategies into place to lessen these risks. To ensure the reliability of the data sources, ongoing evaluation will be done. Additionally, methods for data validation will be used to find and address any inconsistencies or anomalies in the market data. To improve data integrity and lessen the impact of potential errors, cross-referencing data from various sources can also be taken into consideration.

## 4.3 Design

The trading simulator will follow a modular design approach, consisting of different components that work together to provide the desired functionality. The primary component within the simulator will be responsible for executing trades, simulating market conditions, and generating comprehensive reports. This component will handle the core logic of the simulator, including order execution, position management, risk management, and strategy implementation. Importantly, communication between the simulator and the user is facilitated through websockets.

Furthermore, a separate module will be developed specifically for populating the database with real market data. This module will retrieve data from the OEMS and store it in the BigQuery database. By

separating this functionality into its own module, we ensure a clear separation of concerns and maintain the focus of the simulator component solely on trading simulation aspects. To facilitate the testing of the trading system, a dedicated testing module will be created. This module will interact with both the trading simulator and the trading system to validate their functionality and performance. It will simulate various trading scenarios, execute trades within the simulator, and compare the expected results with the actual outcomes generated by the trading system. By adopting a modular design, we can achieve better code organization, maintainability, and reusability. Each component will have well-defined responsibilities, enabling independent development, testing, and potential future enhancements.

1. **Populate Module**:

   - The populate module focuses on populating the database with market data. It fetches, validates, and formats data from external sources, which is then stored in the database for subsequent simulation.
   - This module plays a critical role in keeping the database up-to-date and ensuring that the simulator has access to accurate and relevant market information.

2. **Database Module**:

   - The database module is primarily responsible for storing and managing historical market data obtained from reliable sources. This includes data retrieval, storage, and maintenance.
   - It ensures data integrity, security, and efficient access, allowing other components to interact with market data seamlessly.

3. **Simulator Module**:

   - The simulator module is the core of the cryptocurrency trading system, responsible for simulating the behavior of a real trading environment.
   - It interacts with the database module to access historical market data.
   - This component also manages trade execution and performance analysis/evaluation.

By clearly defining and segregating these responsibilities among the three modules, we enable independent development and testing of each component. This separation of concerns promotes a well-organized and scalable system design, making it easier to maintain and extend the cryptocurrency trading simulator as new requirements or enhancements arise. It also reduces the risk of unintended side effects when modifying one part of the system, as each module operates with a well-defined scope.

## 4.4 Implementation Tools and Technologies

The trading simulator will be implemented using a number of tools and technologies, which offer various advantages and considerations. The trading simulator is built on a foundation of Python for the Populate and Simulator modules, Golang for the Simulator module and trading system, and the BigQuery database. Python's adaptability, extensive library ecosystem, and ease of integration make it a versatile choice for developing and integrating different components quickly. Additionally, Python is a widely adopted language and is already used in the company's tech stack, allowing for seamless collaboration and knowledge sharing within the development team [KSL+20, She15].

Golang, known for its performance advantages, will be leveraged in the trading simulator, especially in handling high-frequency trading scenarios. Golang's efficiency and concurrency capabilities enable the system to process large amounts of data quickly, resulting in improved performance and responsiveness [Mos]. Furthermore, Golang is gaining popularity in the industry, and its usage aligns with the company's tech stack, ensuring compatibility and familiarity for the development team.

The BigQuery database will serve as the main repository for storing and managing market data. BigQuery's scalability and ability to handle enormous amounts of data make it an ideal choice for the trading simulator. With BigQuery, the system can efficiently process and analyze large datasets, facilitating data-driven decision-making [nw]. Moreover, as BigQuery is already used in the company's tech stack, integrating the trading simulator with existing infrastructure will be seamless, enabling better collaboration and utilization of resources.

One advantage of using these tools is that they are already part of the company's tech stack. This familiarity and prior experience with the tools reduce the learning curve for the development team and allow for efficient development and maintenance of the trading simulator. The team can leverage existing knowledge, best practices, and resources, streamlining the development process.

However, it is important to acknowledge certain potential considerations. Python, although versatile, may have specific performance limitations in comparison to lower-level languages. Yet, given the context of this project and its focus on collecting market data, stringent performance requirements may not be applicable, rendering this a non-significant drawback. Careful optimization and effective utilization of libraries can alleviate any associated concerns. Similarly, Golang, while highly efficient, may entail a steeper learning curve for developers less acquainted with its syntax and paradigms. Ensuring adequate training and support will be crucial to ensure a smooth adoption process.

In addition to these core technologies, websockets will be used for efficient data streaming, enabling real-time order and execution updates within the simulator. Retrieving market data from dependable sources will be made possible by the OEMS API. This API will offer the required endpoints so that simulation applications can access historical and possibly current market data.

## 4.5 Usage and Flexibility

The trading simulator will primarily support the trading of crypto assets, specifically trading pairs. Internal teams will have the flexibility to define and implement their own trading strategies within the simulator. This versatility allows traders to explore and evaluate their strategies in a controlled environment before deploying them in real-market scenarios. An interactive and user-friendly interface can easily be implemented afterwards to facilitate strategy configuration, execution, and result analysis.

## 4.6 Conclusion

The methodology and design of the trading simulator with real market data encompass key steps such as data collection, validation, and simulation of market conditions. The primary objective is to enhance the testing infrastructure and assess the performance of the trading system under various scenarios. By adopting a modular design approach, the simulator can be easily maintained, expanded, and integrated with the existing trading system, while building upon the existing tech stack.

The chosen tools and technologies, including Python, Golang, and the BigQuery database, offer several advantages for developing the trading simulator. Python's adaptability, extensive library ecosystem, and

compatibility with the company's tech stack allow for quick development and integration. Golang's performance advantages and growing industry adoption make it ideal for handling high-frequency trading scenarios. BigQuery's scalability and data processing capabilities provide a robust solution for storing and managing market data.

Furthermore, websockets will be utilized to enable efficient data streaming and real-time market updates within the simulator. The OEMS API will serve as a reliable source for retrieving market data from dependable sources of exchanges or dealers. Leveraging these tools, the trading simulation platform can provide a comprehensive and flexible environment for testing.

It is important to mitigate the risks associated with third-party data sources through continuous assessment, data validation, and cross-referencing techniques. These measures will ensure the accuracy and reliability of the data used in the simulator, enhancing the validity of the simulation results.

Overall, the implementation of the trading simulator with the selected tools and technologies, including Python, Golang, websockets, and the OEMS API, aligns with the objectives of the project. The modular design, integration with the existing trading system, and the company's familiarity with the chosen tech stack contribute to a robust and efficient trading simulation platform.

# 5

# Data Collection and Processing

*This chapter provides an overview of the data collection and processing methods used to develop the system. It discusses the sources and methods of data collection, sample size determination, data types collected, reliability measures, encountered challenges, tools and technologies employed, data processing procedures, ethical considerations, and documentation for reproducibility. In this chapter groundwork is established for the subsequent chapters.*

Chapter 4 explores the crucial aspect of data collection and processing methods used to enable the testing of the trading system. By employing robust methodologies, this chapter brings some insight about the accuracy and reliability of the collected data, allowing for comprehensive evaluation of the trading system's performance. Key elements covered in this chapter include the identification of data sources and the selection of appropriate collection methods. Additionally, the determination of the sample size, the types of data collected, and measures taken to ensure reliability and validity are discussed. This Chapter also highlights encountered challenges and limitations, along with the tools, technologies, and procedures employed for data processing and cleaning. Ethical considerations and documentation practices for transparency and reproducibility are also addressed. The insights provided in this chapter lay the foundation for the subsequent chapter, which focuses on implementing and evaluating the trading system using the collected data.

## 5.1 Data Sources and Collection Methods

The data used for this project was obtained from secondary sources, exchanges like coinbase and kraken, through an API. These sources were selected based on their reputation and the availability of market snapshots, which provided the required data for constructing an order book.

To retrieve the data, we established a subscription to an OEMS API, which delivered market data at an interval of 1 second. This subscription-based connection ensures a continuous flow of real-time market data, enabling us to capture dynamic market trends and fluctuations. This granularity was initially chosen to facilitate system development, although it can be adjusted in the future based on efficiency considerations.

However, one remarkable feature of the OEMS API is its ability to fetch data from specific markets and trading pairs. This allows us to focus our data collection efforts on particular segments of interest, ensuring that the dataset is tailored to our research objectives. It should be noted that some variations in the data arrival time were observed, occasionally deviating from the expected 1 second interval. These discrepancies were handled by considering the received data as the closest available representation and did not significantly impact the simulator's functionality.

A sample of the raw data received from the OEMS API is provided below in 5.1for reference:

Listing 5.1: Sample of raw data

```
"data": [
{
  "Symbol": "BTC-USD",
  "DepthType": "Price",
  "LiquidityType": "Indicative",
  "ExchangeTime": "2023-03-06T14:57:41.443000Z",
  "SystemTime": "2023-03-06T14:57:41.444069Z",
  "Bids": [
    {
      "Price": "22436.00",
      "Size": "0.04496237"
    },
    {
      "Price": "22432.51",
      "Size": "0.03352651"
    },
  ],
  "Offers": [
    {
      "Price": "22433.66",
      "Size": "0.01000000"
    },
    {
      "Price": "22433.68",
      "Size": "0.02228873"
    },
  ],
  "Markets": {
    "coinbase"}
}]
```

Here is the description of each key from the data above (Listing 5.1):

- `"Symbol": "BTC-USD" (String):`
  This Key indicates the trading pair symbol, which in this case is "BTC-USD," representing the exchange of Bitcoin (BTC) for US Dollars (USD).

- `"DepthType": "Price" (String):`
  This Key specifies the depth type, indicating that the data represents the price levels in the order book.

- `"LiquidityType": "Indicative" (String):`
  This Key indicates the liquidity type, suggesting that the data provides indicative or estimated values rather than actual trade data.

- `"ExchangeTime": "2023-03-06T14:57:41.443000Z" (String/DateTime):`
  This Key represents the timestamp of when this data was retrieved from the exchange. It includes the date and time in UTC format.

- `"SystemTime": "2023-03-06T14:57:41.444069Z" (String/DateTime):`
  Similar to the "ExchangeTime," this Key represents the timestamp, but it is associated with the system's time for recording the data.

- `"Bids": [...] (Array of Objects):`
  This Key starts a section of data related to the bid side of the order book. It includes an array of bid prices and sizes, where each bid is represented as an object with "Price" and "Size" attributes.

  - `"Price": "22436.00" (Float):`
    This represents the price at which buyers are willing to purchase Bitcoin (BTC) in USD.
  - `"Size": "0.04496237" (Float):`
    This represents the quantity or size of Bitcoin (BTC) available for purchase at the corresponding bid price.

- `"Offers": [...] (Array of Objects):`
  This Key starts a section of data related to the offer (ask) side of the order book. It includes an array of offer prices and sizes, where each offer is represented as an object with "Price" and "Size" attributes.

  - `"Price": "22433.66" (Float):`
    This represents the price at which sellers are willing to sell Bitcoin (BTC) in USD.
  - `"Size": "0.01000000" (Float):`
    This represents the quantity or size of Bitcoin (BTC) available for sale at the corresponding offer price.

- `"Markets": { "coinbase" } (Object of Strings):`
  This Key specifies the market or exchange where this data is sourced from. In this case, it indicates that the data is from the Coinbase exchange.

A sample of the processed data is provided in the Table 5.1bellow.

| Row | Price | Size | Market | Trading_Pair | Data | Side |
|---|---|---|---|---|---|---|
| 3501 | 28250.69 | 0.07160561 | Coinbase | BTC_USD | 2023-03-30 16:35:22.324000 UTC | Bids |
| 3502 | 28250.68 | 0.00464593 | Coinbase | BTC_USD | 2023-03-30 16:35:22.324000 UTC | Bids |
| 3503 | 28247.95 | 0.02288867 | Coinbase | BTC_USD | 2023-03-30 16:35:22.324000 UTC | Bids |
| 3504 | 28247.93 | 0.04 | Coinbase | BTC_USD | 2023-03-30 16:35:22.324000 UTC | Bids |
| 3505 | 28247.91 | 0.01 | Coinbase | BTC_USD | 2023-03-30 16:35:22.324000 UTC | Bids |
| 3506 | 28253.15 | 0.012611 | Coinbase | BTC_USD | 2023-03-30 16:35:22.324000 UTC | Offers |
| 3507 | 28253.63 | 0.00054901 | Coinbase | BTC_USD | 2023-03-30 16:35:22.324000 UTC | Offers |

Table 5.1: SnapShot of the Data Base

## 5.2 Sample Size Determination and Data Selection

To determine the sample size for the trading simulator, we focused on the depth level of the order book, which refers to the number of offers and bids included in the data set. Through extensive testing, we evaluated different depth levels to identify the optimal balance between data quantity, system efficiency, and storage requirements.

After conducting various tests, it was observed that a depth level of 10 generally provided a sufficient quantity of offers and bids to fulfill the orders fed into the simulator. This depth level was carefully chosen based on the analysis of simulated trading scenarios and statistical data.

In scenarios where a higher depth level was tested, such as 15 or 20, the increase in data volume did not offer significant improvements in the simulation results. In fact, it led to increased storage requirements without significantly enhancing the accuracy or realism of the simulation.

Moreover, it is important to consider the time frame and the number of orders involved when determining the optimal depth level. In situations where a small time frame with a high influx of orders is anticipated, a higher depth level may be more suitable to ensure an accurate representation of market dynamics.

By selecting a depth level of 10, we strike a balance between obtaining sufficient data for realistic simulations and managing storage requirements effectively. It is important to note that the system has been designed to allow for easy modification of the depth level in the future, should the need arise based on evolving market conditions or specific testing requirements.

Overall, the decision to use a depth level of 10 is justified by the analysis of simulated trading scenarios, statistical considerations, and the need to optimize system performance and storage utilization while still capturing the essence of market dynamics.

## 5.3 Types of Data Collected

The collected data comprises a blend of both quantitative and qualitative attributes, each playing a crucial role in shaping our analysis. These attributes encompass:

1. Price: This attribute reflects the monetary value associated with a specific order, aiding in the calculation of order costs and financial analyses.
2. Size: Size pertains to the quantity or volume of an order, a vital factor for understanding the extent of market activity and order dynamics.
3. Market: The market attribute identifies the specific market or trading platform from which the data

originates. It helps contextualize the data within different trading environments.

4. TradingPair: This attribute denotes the pair of assets being traded, offering insights into asset correlations and trading relationships.

5. Exchange Time: Exchange time functions as a timestamp, marking the exact moment when the data point was received. It facilitates chronological analysis and temporal correlations.

6. Side: Side indicates whether an entry is a bid (buying) or an offer (selling) in the order book. It provides a directional context to market activity.

Collectively, these attributes collaboratively underpin the construction of the order book and drive essential calculations within our simulator. For instance, Price and Size enable us to compute order costs, while Market and TradingPair ensure accurate association with relevant trading contexts. Exchange Time assists in temporal analysis, and Side clarifies the nature of entries in the order book.

## 5.4 Ensuring Reliability and Validity

Several measures were implemented to ensure the reliability and validity of the data collection methods. Data validation tests were conducted to verify the correctness and completeness of the received data. If any discrepancies or missing parameters were detected during the testing process, the respective data samples were discarded, and is given the option to obtain new data. This mechanism helped minimize errors and inconsistencies in the collected data.

## 5.5 Challenges and Limitations

During the data collection process, one challenge encountered was the initial lack of information regarding the market associated with each offer and bid. This limitation was addressed by modifying the communication process and sending separate requests for each market data, ensuring that the obtained data corresponded to the specific market. By overcoming this challenge, the simulator could accurately associate each order with the appropriate market.

## 5.6 Tools, Software, and Technologies

Python served as the core programming language for the development of the data collection module. Working in tandem with a suite of libraries, including `json`,`bigquery`,`sys`,`os`, and `datetime`, Python facilitated a seamless array of tasks essential to our data collection process. Additionally, it's worth exploring the intricacies of the OEMS and its relationship with BigQuery. A standout feature of OEMS lies in its ability to fetch data efficiently from an eclectic array of external sources, championing punctuality and precision in our dataset. However, it's crucial to highlight that our quest for data enrichment purely stems from reality, devoid of any intervention or enhancement. The innate real-time essence of our data acquisition, seamlessly achieved through the resourceful application of OEMS, aligns with the very rationale behind embracing BigQuery. This strategic embrace of BigQuery serves as our conduit to shape this real-time tapestry into structured datasets, empowering our research with a data reservoir for future explorations and analyses.

## 5.7 Data Processing and Cleaning

To prepare the collected data for analysis, several processing steps were undertaken. We can see the overall process in Figure 5.1. Initially, the data value representing the time the data was received was discarded, as it was not found to be useful for simulation purposes, we decided, however, to use the Exchange time attribute (the real time of the order). The data rows were then sorted based on the Exchange time, which allowed for chronological analysis and seamless integration with the simulator. Although slight discrepancies in time values from different markets were observed, they did not significantly affect the analysis conducted in the project. However, it is recommended to adjust the throttle and decrease the update rate to mitigate such discrepancies in future implementations.

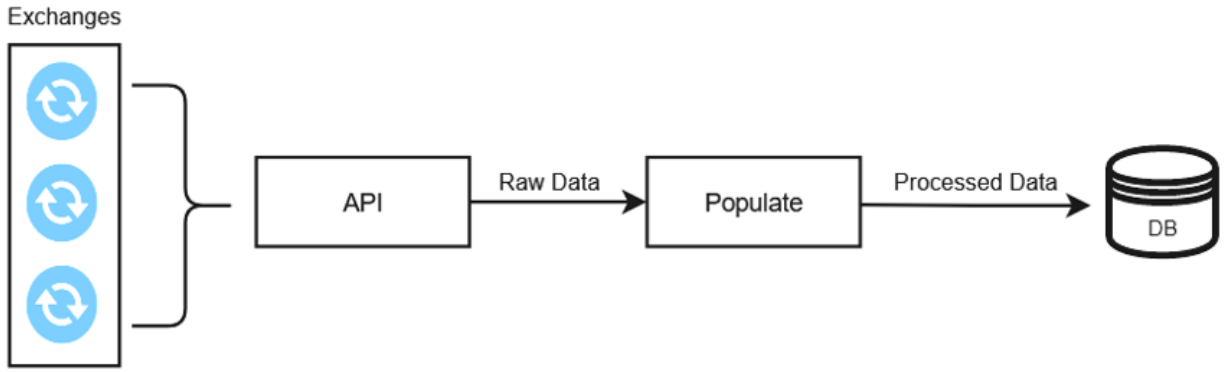


Figure 5.1: Overall data collection module architecture

## 5.8 Security consideration

To bolster the security and privacy of accessing Market data, critical measures were implemented to protect sensitive API credentials and secure database access.

To prevent unauthorized access to API credentials, they were removed from the code and securely stored as environment variables. This separation ensures that sensitive data remains inaccessible to unauthorized parties, aligning with the company best practices for API credential management.

For secure database access (BigQuery), authentication was established using a google account. Leveraging Gmail's robust security features, this approach adds an extra layer of protection to safeguard financial data from potential threats.

By adhering to these security considerations, the system fortifies its defenses against unauthorized access, data breaches, and privacy violations. These measures demonstrate a commitment to ethical data handling practices, ensuring the confidentiality and integrity of sensitive financial information. Continuous monitoring and periodic security audits further contribute to sustained protection and the identification of emerging security concerns.

## 5.9 Documentation and Reproducibility

To enable reproducibility and transparency, the data collection and processing procedures were thoroughly documented. This documentation includes the description of data sources, API endpoints, data collection methods, data processing steps, and any modifications made during the implementation. This documentation provides a comprehensive guide for future researchers or practitioners who wish to reproduce or extend the project.

## 5.10 Conclusion

This Chapter detailed the data collection and processing methods employed to gather the necessary data for testing the trading system using a simulator. The sources, collection methods, sample size determination, data types collected, reliability and validity measures, encountered challenges, utilized tools and technologies, data processing steps, ethical considerations, and documentation procedures were discussed.

# 6

# Simulator Implementation

*This chapter presents the implementation of a comprehensive trading simulator designed to enhance the testing infrastructure of the cryptocurrency trading system. The simulator leverages real market data to create a realistic environment for evaluating the trading system's performance. It interacts with a database module, generates execution reports based on real market data, and allows for customization of parameters. Rigorous testing and validation ensure accuracy and reliability. The simulator proves to be a valuable tool for identifying strengths, weaknesses, and areas for improvement in the trading system, guiding future developments and optimizations.*

The purpose of this chapter is to provide a detailed account of the implementation of the trading simulator developed as part of this dissertation. The simulator aims to enhance the testing infrastructure of the cryptocurrency trading system, with a specific focus on Anchorage Digital's trading platform. By creating a comprehensive trading simulator that uses real market data, this study contributes to the efficiency and effectiveness of the trading market by enabling thorough testing and evaluation of the trading system's performance.

## 6.1 Design and Architecture

The developed simulator is a sophisticated tool designed to mimic real-world trading scenarios and provide a controlled environment for testing the trading system. It leverages real market data to replicate the dynamics of financial markets, generating realistic order book updates and trade executions. By simulating different market conditions, users can assess the impact of various factors on the system's performance. This unique approach using real market data adds an extra layer of realism and accuracy to the simulation.

The simulator retrieves real market data from the database, allowing for realistic order book updates and trade executions within the simulation. This integration enables the simulator to mimic actual market dynamics and evaluate the trading system's performance under various scenarios.

Figure. 6.1 outlines the high-level flow of an order as it moves through different modules in your trading system, including communication with the database, orderbook management, and user feedback in the form of an execution report. The flow is simplified for illustration, and the actual implementation may involve more detailed components and interactions. Follows a description of each step of the process:

- Receive Order: The order is initially received from the user.
- Order Processing Part1: This module processes the incoming order and prepares it for further processing.
- Database Module: This module communicates with the database to retrieve market data necessary and sends it to the Order Book module in case this is the first order or there are orders waiting to be executed.Otherwise, it sends the data to the order processing module.
- Orderbook Module: The orderbook module handles creating the order book and provides information to process waiting orders (e.g., limit orders).
- Process Waiting Orders: This module manages any waiting orders, such as limit orders that are pending execution and updates the order book.
- Order Processing part2: This module at this phase will now processes the incoming order and update the final order book.
- Execution Report: This is the final module, and the one that communicates with the user. Here an execution report is generated to provide feedback to the user.

## 6.2 Integration with other Modules

The simulator module interacts seamlessly with the database module, which populates the database with real market data like we can see in Figure 6.2 . Serving as a server, the simulator receives orders from the trading system and responds with execution reports. This integration ensures accurate testing and evaluation by facilitating a mooth communication between the simulator and the trading system.

## 6.3 Order Book Generation and Execution

Within the simulator, algorithms are implemented to generate execution reports based on real market data fetched from the database. The simulator creates an order book as we can see in Table 6.3 for a specific

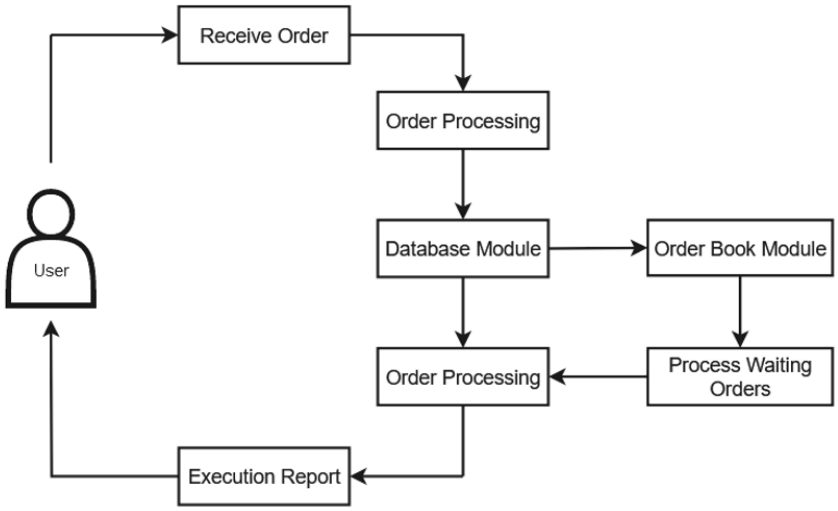


Figure 6.1: Working modules of the simulator

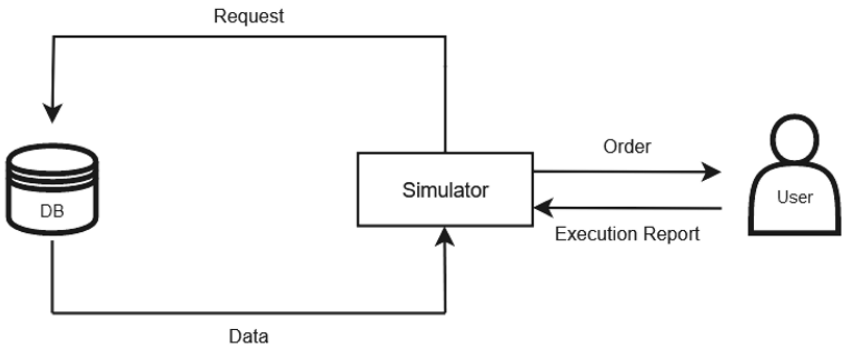


Figure 6.2: Simulator Modules

time period and executes the received orders, updating the order book accordingly. The internal clock of the simulator advances with each order, allowing for continuous order book updates and order filling. Active limit orders are also monitored and executed as appropriate.

The algorithm used in the simulator leverages the created order book and the received orders to execute trades. When a buy order is received, the algorithm identifies the corresponding sell orders in the order book and fills the buy order by removing the appropriate quantity of currency being used from the order book. The execution report generated by the simulator provides the total price required to fill the order, reflecting the dynamics of real-world trading.

## 6.4 Technology Stack

The simulator is implemented using Python, websockets, and the BigQuery database. Python provides the necessary flexibility and efficient data processing capabilities for developing complex trading algorithms. Websockets enable real-time communication between the simulator and external data sources, such as the BigQuery database that stores historical market data. The use of unique IDs generated by the uuid package ensures accurate tracking and identification of each order.

Trading_Pair: USD – BTC
Date: 2023-03-30 16:35:22.324000 UTC

| Market | Size | Bids | Offers | Size | Market |
|---|---|---|---|---|---|
| Coinbase | 0.07160561 | 28250.69 | 28253.15 | 0.01261132 | Coinbase |
| Coinbase | 0.00464593 | 28250.68 | 28253.63 | 0.00054901 | Coinbase |
| Coinbase | 0.02288867 | 28247.95 | 28253.64 | 0.0296442 | Coinbase |
| Coinbase | 0.004 | 28247.93 | 28259.41 | 0.02663954 | Coinbase |
| Coinbase | 0.01 | 28247.91 | 28259.43 | 0.00426119 | Coinbase |

Table 6.1: Order book created using information from the data base

## 6.5 Customization and Documentation

Customizability is a key feature of the simulator, allowing users to tailor the simulation to their specific needs. Parameters such as order types, order book size, dataset selection, market data, and execution rules can be customized. A comprehensive documentation file accompanies the simulator, providing clear instructions on how users can specify their desired settings and experiment with different scenarios. This documentation ensures ease of use and promotes efficient customization and experimentation.

## 6.6 Testing and Validation

Rigorous testing and validation procedures are implemented to ensure the accuracy and reliability of the simulator. Historical market data are used to verify that the simulator produces expected results. Performance metrics are measured, and unit tests are conducted to assess the fidelity of the simulator, validating the accuracy of order book updates, order filling, and execution reports. By conducting thorough unit testing, the simulator's fidelity and reliability are verified, ensuring reliable results for testing and evaluation.

## 6.7 Reporting and Analysis

While testing of the trading system is still ongoing, the simulator already generates comprehensive reports 6.1 and analysis for the tested orders. Key performance indicators, risk metrics, and trade statistics are presented to evaluate the trading system's performance. Detailed analysis is performed to identify areas for improvement, assess the impact of different parameters, and gain valuable insights into the behavior of the trading system under various market conditions.

As part of future improvements, advanced machine learning techniques can be incorporated into the simulator. By applying machine learning algorithms to historical market data, the simulator can enhance the accuracy of order book generation and improve trade execution. These advancements would further refine the realism and effectiveness of the simulator, contributing to more comprehensive testing and evaluation of the trading system.

## 6.8 Limitations and Future Improvements

The simulator, although providing a realistic environment, has some limitations. It may not capture all intricacies of market behavior and participant interactions. However, future improvements can be made to enhance the algorithm used for calculating execution reports. This may involve incorporating advanced

techniques such as machine learning to improve order book generation and integrating more sophisticated risk management features.

## 6.9 Order and Execution Report

Within the context of the cryptocurrency trading simulator, the order and execution report format is pivotal for replicating real-world trading scenarios. These formats adhere to the company's established conventions and play a crucial role in the simulation process. This section provides an overview of the order and execution report formats, their significance, and the key variables they encompass.

### 6.9.1 Order Format

The order format used in 6.1 represents a simulated trading order submitted to the cryptocurrency trading system within the simulator. It follows the company's established conventions and includes essential information such as:

Listing 6.1: Order Request

```
{
    "reqid": 498374466,
    "type": "CustomerNewOrderSingle",
    "ts": "2023-07-06T17:01:00.610687Z",
    "data": [
    {
      "Symbol": "BTC-USD",
      "ClOrdID": "23746327467323ry7y7yrf374y",
      "Side": "Buy",
      "Currency": "BTC",
      "OrderQty": "2",
      "OrdType": "Market",
      "Strategy": "",
      "TimeInForce": "FillOrKill",
      "TransactTime": "2023-07-06T17:01:00.610687Z",
      "Markets": null,
      "Group": "ws:e87cb45c05ad389d58904ea398345c24b50f46c15d412d...",
      "AllowedSlippage": "0"
    }]
}
```

Here is the description of each key from the data above Listing 6.1:

- `"reqid": 498374466,` (Integer)
  This Key indicates the request ID associated with this order.
- `"type": "CustomerNewOrderSingle",` (String)
  This Key specifies the type of data, which is "CustomerNewOrderSingle."
- `"ts": "2023-07-06T17:01:00.610687Z",` (String/DateTime)
  This Key represents the timestamp of when this data was recorded, including the date and time in UTC format.

- `"data": [` (Array)
  This Key marks the beginning of a section containing data.

  - `"Symbol": "BTC-USD",` (String)
    This Key indicates the trading pair symbol, which is "BTC-USD," representing the exchange of Bitcoin (BTC) for US Dollars (USD).
  - `"ClOrdID": "23746327467323ry7y7yrf374y",` (String)
    This Key specifies the Client Order ID associated with this order.
  - `"Side": "Buy",` (String - "Buy"/ "Offer")
    This Key indicates the side of the order, which is "Buy."
  - `"Currency": "BTC",` (String)
    This Key represents the currency involved in the order, which is Bitcoin (BTC).
  - `"OrderQty": "2.0",` (float)
    This Key specifies the order quantity, which is "2."
  - `"OrdType": "Market",` (String - "Market"/"Limit")
    This Key indicates the order type, which is "Market."
  - `"Strategy": "",` (String)
    This Key represents the strategy associated with the order, which is empty in this case.
  - `"TimeInForce": "FillOrKill",` (String - "FillOrKill"/"FillAndKill")
    This Key specifies the time-in-force for the order, which is "FillOrKill."
  - `"TransactTime": "2023-07-06T17:01:00.610687Z",` (String/DateTime)
    This Key represents the transaction timestamp, including the date and time in UTC format.
  - `"Markets": null,` (String)
    This Key indicates the markets or exchanges related to this data, which is null in this case.
  - `"Group": "ws:e87cb45c05ad389d58904ea398345c24b50f46c15d412d0f671e66b766247d39",` (String)
    This Key specifies the group associated with this data.
  - `"AllowedSlippage": "0.0"` (float)
    This Key represents the allowed slippage for the order, which is "0."

These variables ensure consistency with the company's existing order format, enabling the simulator to mimic real-world trading order submissions accurately.

### 6.9.2 Execution Report Format

The execution report format in 6.2 represents the outcome of a trading order within the simulator. This format adheres to the company's conventions and contains critical details, including:

Listing 6.2: Execution Report

```
1 {
2     "Timestamp": "2023-04-03 14:45:00+00:00",
3     "User": "user",
4     "Symbol": "BTC-USD",
5     "OrderID": "127-4tyj174017-24tt0127-490fdg274127402",
6     "ClOrdID": "53746-gdf7432784-8g4g-754857-g211765f66",
```

```
    "SubmitTime": "",
    "ExecID": "",
    "Side": "Buy",
    "TransactTime": "2023-04-03 14:45:00+00:00",
    "OrdeStatus": "Executed",
    "OrderQty": 2.0,
    "OrdType": "Market",
    "Currency": "",
    "LeavesQty": 0,
    "CumQty": 2.0,
    "AvgPx": 28093.646441026944,
    "TimeInForce": "FillOrKill",
    "CumAmt": 0,
    "DecisionStatus": "",
    "AmountCurrency": "",
    "MarketAccount": "dealer",
    "TotalPx": 56187.29288205389
}
```


Here is the description of each key from the data above Listing 6.2:

- `"Timestamp": "2023-04-03 14:45:00+00:00"`, (String/DateTime)
  This Key represents the timestamp of the execution report, including the date and time in UTC format.
- `"User": "user"`, (String)
  This Key specifies the user associated with the execution report.
- `"Symbol": "BTC-USD"`, (String)
  This Key indicates the trading pair symbol, which is "BTC-USD," representing the exchange of Bitcoin (BTC) for US Dollars (USD).
- `"OrderID": "127-4tyj174017-24tt0127-490fdg274127402"`, (String)
  This Key specifies the Order ID associated with the order.
- `"ClOrdID": "53746-gdf7432784-8g4g-754857-g211765f66"`, (String)
  This Key represents the Client Order ID associated with the order.
- `"SubmitTime": ""`, (String)
  This Key indicates the submission time of the order, which is empty in this case.
- `"ExecID": ""`, (String)
  This Key specifies the Execution ID, which is empty in this case.
- `"Side": "Buy"`, (String - "Buy"/ "Offer")
  This Key indicates the side of the order, which is "Buy."
- `"TransactTime": "2023-04-03 14:45:00+00:00"`, (String/DateTime)
  This Key represents the transaction timestamp, including the date and time in UTC format.
- `"OrderStatus": "Executed"`, (String - "Executed"/"PartiallyExecuted"/"Failed"/"Canceled")
  This Key indicates the status of the order, which is "Executed."

- `"OrderQty": 2.0,` (Float)
  This Key specifies the order quantity, which is a floating-point number, "2.0."
- `"OrdType": "Market",` (String)
  This Key indicates the order type, which is "Market."
- `"Currency": "",` (String)
  This Key represents the currency involved in the order, which is empty in this case.
- `"LeavesQty": 0,` (Float)
  This Key specifies the quantity of the order that remains unfilled, which is an integer, "0."
- `"CumQty": 2.0,` (Float)
  This Key represents the cumulative quantity filled, which is a floating-point number, "2.0."
- `"AvgPx": 28093.646441026944,` (Float)
  This Key specifies the average price at which the order was executed.
- `"TimeInForce": "FillOrKill",` (String - "FillOrKill"/"FillAndKill")
  This Key indicates the time-in-force for the order, which is "FillOrKill."
- `"CumAmt": 0,` (Float)
  This Key represents the cumulative amount for the order, which is an integer, "0."
- `"DecisionStatus": "",` (String)
  This Key specifies the decision status, which is empty in this case.
- `"AmountCurrency": "",` (String)
  This Key indicates the currency associated with the amount, which is empty in this case.
- `"MarketAccount": "dealer",` (String)
  This Key specifies the market account, which is a string, "dealer."
- `"TotalPx": 56187.29288205389` (Float)
  This Key represents the total price for the execution, which is a floating-point number.

These variables align with the company's established execution report format, providing insights into the outcome of each order and ensuring that the simulator operates consistently with existing trading practices.

## 6.10 Conclusion

The implementation of the simulator has proven to be a valuable tool in testing and evaluating the trading system's performance. By providing a realistic and controlled environment, the simulator enables the identification of strengths, weaknesses, and areas for improvement. Through simulating a wide range of market environments, the simulator enhances the trading system's robustness and effectiveness in real-world scenarios. The insights gained from the simulator will guide further developments and optimizations of the trading system.

In conclusion, this chapter has provided a comprehensive overview of the simulator implementation, highlighting its design, integration with the database module, order book generation and execution, technology stack, customization options, testing and validation procedures, reporting and analysis capabilities, as well as its limitations and potential future improvements. The simulator serves as a powerful tool for testing and evaluating the performance of cryptocurrency trading systems, contributing to the advancement of the trading market.

# 7

# Performance Evaluation and Validation

*This chapter delves into the performance evaluation and validation of the developed cryptocurrency simulator. The evaluation encompasses trade execution time and result consistency, while validation focuses on response reliability and reproducibility. The chapter outlines optimization strategies to enhance trade execution time and discusses the simulator's consistent response generation. Limitations and future enhancements are considered, alongside a reflection on the project's conclusion. The chapter concludes by emphasizing the simulator's value as a testing tool for assessing trading system performance.*

This chapter presents the performance evaluation and validation process conducted for the developed cryptocurrency trading simulator. The objective of this evaluation is to assess the simulator's functionality, reliability, and its suitability as a testing tool for the trading system. The evaluation metrics include trade execution time, result consistency, and response accuracy.

## 7.1 Performance Evaluation

The performance evaluation focuses on assessing the simulator's efficiency and responsiveness under various conditions. While the simulator's accuracy is not a primary concern, its ability to handle trading scenarios and provide prompt responses is of paramount importance.

### 7.1.1 Trade Execution Time

One of the key performance indicators we are closely monitoring is the trade execution time. This metric measures the duration it takes for the simulator to process and execute a trade order. Our initial tests have indicated that the response time for each individual order is around 4000 milliseconds, primarily due to the time taken for accessing the database.

### 7.1.2 Optimization Strategies

We had devised a sophisticated optimization strategy with the intention of significantly enhancing this response time. The core idea behind this strategy is to preload all the necessary data from the database right at the start of the testing phase. By doing so, we can subsequently process orders and responses without the need for repeated database access. Additionally, we are considering the approach of collecting all incoming orders for simultaneous processing and delivering all execution reports together once the processing cycle concludes. These adaptations are anticipated to result in a substantial reduction of the response time to less than 10 milliseconds for each order, thus markedly boosting the overall performance of the simulator.

It's important to note that the decision not to implement these optimization measures in the current phase was driven by various factors. Time constraints played a role, as well as the need to prioritize other objectives that held more immediate significance.

Also important, it's worth emphasizing that even more impressive efficiency improvements are well within the realm of possibility. These enhancements could potentially yield substantial benefits. However, achieving their full potential would necessitate a more comprehensive investment of time and resources. Consequently, these enhancements are earmarked for a subsequent development phase that has not yet been explored. This decision arises from the current allocation of resources and the overarching priorities of the project.

## 7.2 Validation

The validation process focuses on ensuring the simulator's reliability and consistency in producing consistent results. Given that the simulator is intended for testing the trading system by replacing the cryptocurrency market, the accuracy of the results is not a primary concern. Instead, the objective is to maintain response consistency and reproducibility.

### 7.2.1 Result Consistency

During the validation process, various tests were performed using specific orders, which included running the platform multiple times and observing the results. The simulator consistently produced the same results

for each test, achieving a response consistency rate of 100%. This outcome is aligned with the simulator's design objective of providing repeatable responses.

### 7.2.2 Response Correctness

While response consistency was effectively achieved, the evaluation phase did not extensively focus on testing the deep-seated correctness of the responses. However, given the simulator's robust design and meticulous alignment with real-world market data, there is a strong expectation that response accuracy will be upheld with a high degree of confidence.

### 7.2.3 Communication with the Trading System

In addition to the performance evaluation and validation, it's essential to shed light on the communication aspect between the cryptocurrency trading simulator and the trading system itself. This communication is a critical component of the overall functionality and reliability of the simulator.

The simulator communicates with the trading system through websockets, allowing it to send trade orders, receive execution reports, and access historical market data. While the communication protocol was carefully designed and implemented, it's important to note that, due to the scope of this phase, the full spectrum of possible interactions was not explored.

Despite the limitations imposed by the project's scope and resources, it's worth highlighting that the communication between the simulator and the trading system through websockets worked as intended within its defined parameters. The simulator successfully transmitted orders, received execution reports, and retrieved market data, enabling comprehensive testing of individual trade scenarios.

However, it's essential to acknowledge that further exploration of complex communication scenarios, such as handling a high volume of concurrent users or simulating extreme market conditions, may be required in future development phases. These scenarios would demand more extensive testing to ensure that the communication framework remains robust and reliable under diverse conditions.

In conclusion, while the scope of communication testing was limited in this phase, the cryptocurrency trading simulator's ability to effectively communicate with the trading system through websockets is a testament to its reliability as a testing tool for assessing individual trading system performance. Future work will focus on expanding and refining this communication framework to accommodate more complex scenarios and ensure its seamless functionality under a wider range of conditions.

## 7.3 Limitations and Future Directions

The evaluation process also highlighted certain limitations that may impact the simulator's performance and validation scope. One notable limitation is the availability of historical data that may not comprehensively cover all market scenarios. Addressing this limitation requires the identification of relevant data sets, potentially with input from testers, to ensure the simulator is robust in handling a wide range of scenarios.

Another notable aspect to address is the testing of multiple users connected to the simulator simultaneously. However, it should be noted that the simulator is primarily designed as a testing tool for assessing individual trading system performance. Therefore, the testing of multiple concurrent users was not a focus in this phase, as it does not align with the primary use case. Instead, our emphasis was on evaluating the

simulator's ability to process individual orders efficiently and reliably.

# 8

# Conclusion and Future Work

*This chapter explores the potential avenues for future research and development of the cryptocurrency trading simulator. Building upon the insights gained from the performance evaluation and validation discussed in Chapter 7, this chapter delves into the exciting possibilities for expanding, enhancing, and refining the simulator's capabilities. The primary objectives of this future work are to improve accuracy, scalability, and functionality, making the simulator an even more invaluable tool for assessing trading system performance.*

## 8.1 Introduction

In the preceding chapters, we have delved into the development, performance evaluation, and validation of our cryptocurrency trading simulator. Through these endeavors, we have successfully achieved the objectives set for this dissertation. Our simulator now stands as a valuable testing tool, ready to be utilized for assessing the trading system's performance. However, it's crucial to acknowledge that the effective utilization of this tool for testing the trading system depends on the collaborative efforts of the development team within our institution responsible for the trading system. Chapter 7 highlighted the

significance of rigorous data validation and testing, emphasizing the efficiency and responsiveness of the simulator in producing consistent results.

Building upon the insights gained from our performance evaluation and validation, it paves the way for future work and enhancements to our cryptocurrency trading simulator. As we conclude our discussion of past endeavors, we transition into the exciting realm of possibilities that lie ahead. In this chapter, we will explore avenues to address limitations, improve accuracy, and expand the simulator's capabilities for testing trading system performance.

Our objective is clear: to provide the financial institution with a robust and invaluable tool for testing the complex trading system in place. The journey towards achieving this goal has just begun, with ample room for further improvement and development in the future, where innovation and excellence converge to shape the future of cryptocurrency trading simulation.

The following areas of future work are pivotal in shaping the continued evolution and effectiveness of our cryptocurrency trading simulator, ensuring that it remains a valuable tool for assessing trading system performance and adapting to the dynamic landscape of cryptocurrency trading.

## 8.2 Future Work

In this section, we will discuss several key areas where the simulator can be enhanced:

### 8.2.1 Communication Enhancement

To further enhance the simulator's capabilities, consideration will be given to expanding and refining the communication framework between the simulator and the trading system. As of the current phase, the communication between the two systems is in its initial stages. While it is possible to send orders from the trading system to the simulator and receive execution reports, no extensive testing of the trading system using the simulator has been conducted yet.

The next critical step in the development process is to effectively use the simulator for testing purposes. Only after rigorous testing and utilization of the simulator within simulated high-frequency trading scenarios can we confirm that the communication is working as intended. This phase will involve comprehensive testing of the trading system's responses to the simulator's inputs and evaluating the simulator's ability to accurately use and calculate orders received by the trading system.

By focusing on communication enhancements and the subsequent practical application of the simulator, we aim to create a robust and reliable testing tool that can seamlessly interact with the trading system, ultimately providing valuable insights into its performance and reliability. This iterative approach ensures that both communication and functionality are thoroughly validated and refined.

### 8.2.2 Algorithmic Improvements

To enhance the simulator's accuracy, we plan to incorporate advanced algorithms that mirror real-world order book liquidity dynamics. This will enable the simulator to provide a more realistic representation of market conditions, thereby improving the quality of test results.

### 8.2.3 Transaction Fee Modeling

The development of a transaction fee model that closely resembles real trading fees is essential. This will allow users to evaluate trading strategies with a more accurate assessment of their costs, providing a more comprehensive testing environment.

### 8.2.4 Data Coverage Expansion

One significant limitation brought to light during the evaluation is the availability of historical data that may not comprehensively cover all market scenarios. To address this limitation, the identification of relevant data sets, potentially with input from testers, will be essential. Expanding the range and diversity of historical data sources will ensure that the simulator is robust in handling a wider spectrum of market scenarios. This expansion will contribute to more comprehensive and accurate testing outcomes.

### 8.2.5 Multiple User Testing

While our primary focus has been on evaluating the simulator's ability to process individual orders efficiently and reliably, there is potential for future work in the realm of testing multiple users connected to the simulator simultaneously. This would involve assessing the simulator's performance and responsiveness under scenarios where multiple users are actively engaged. However, it is important to note that this direction may require careful consideration, as the simulator's primary use case is to evaluate individual trading system performance. Any future developments in this area will need to align with this overarching goal.

### 8.2.6 User Interface Enhancement

While a basic interface served the initial testing purposes, consideration will be given to developing a more advanced user interface. Such an interface could provide users with enhanced visualization tools, analytics, and a more intuitive user experience.

## 8.3 Scalability Improvements

### 8.3.1 Data Processing Optimization

To enhance scalability, we will focus on optimizing data storage and retrieval processes. Streamlining these operations will allow the simulator to handle larger datasets and more complex scenarios efficiently.

### 8.3.2 Parallel Processing

Feedback from users will continue to be instrumental in shaping future enhancements and ensuring that the simulator remains attuned to the evolving needs of the cryptocurrency trading community. User insights and suggestions will guide the refinement and development of the simulator, enhancing its usability and effectiveness.

## 8.4 User Feedback

Feedback from users will play a pivotal role in shaping future enhancements and ensuring that the simulator remains aligned with the evolving needs of the cryptocurrency trading community.

## 8.5 Ethical and Regulatory Considerations

As the simulator evolves, we will continue to monitor and address ethical and regulatory considerations that may arise. Ensuring compliance with evolving regulations and ethical standards is crucial for the simulator's credibility and usability.

## 8.6 Resource Allocation

It's important to note that the successful implementation of these future enhancements will require careful resource allocation in terms of time, personnel, and financial investments. Developing and implementing these improvements will be a phased process, with priorities aligned with the project's overarching goals.

# Bibliography


[AD20] Carol Alexander and Michael Dakos. A critical investigation of cryptocurrency data and analysis. *Quantitative Finance*, 20(2):173–188, 2020.

[AK18] Khalid Abouloula and Salah-ddine Krit. Using a robot trader for automatic trading. In *Proceedings of the Fourth International Conference on Engineering & MIS 2018*, pages 1–9, 2018.

[ANA23] Vizaad Ali, Azah Anir Norman, and Saaidal Razalli Bin Azzuhri. Characteristics of blockchain and its relationship with trust. *IEEE Access*, 11:15364–15374, 2023.

[AOJAF21] Kebira Azbeg, Ouail Ouchetto, Said Jai Andaloussi, and Laila Fetjah. An overview of blockchain consensus algorithms: Comparison, challenges and future directions. *Advances on Smart and Soft Computing: Proceedings of ICACIn 2020*, pages 357–369, 2021.

[Ata11] A Atak. The future of computer trading in financial markets. 2011.

[Bac] Backtrader. Welcome to backtrader! https://www.backtrader.com/. Accessed: 09/05/2023.

[BBHK19] Matthew Baron, Jonathan Brogaard, Björn Hagströmer, and Andrei Kirilenko. Risk and return in high-frequency trading. *Journal of Financial and Quantitative Analysis*, 54(3):993–1024, 2019.

[BBP06] Vladimir Boginski, Sergiy Butenko, and Panos M Pardalos. Mining market data: A network approach. *Computers & Operations Research*, 33(11):3171–3184, 2006.

[BM19] Abdullah Ahmed Omar Bahashwan and Selvakumar Manickam. A brief review of messaging protocol standards for internet of things (iot). *Journal of Cyber Security and Mobility*, pages 1–14, 2019.

[BRW22] Sabrina Buti, Barbara Rindi, and Ingrid M Werner. Diving into dark pools. *Financial Management*, 51(4):961–994, 2022.

[BTNS19] Amit Bauriya, Akshata Tikone, Pooja Nandgaonkar, and Kishor S Sakure. Real-time cryptocurrency trading system. *International Research Journal of Engineering and Technology*, 6:4845–4848, 2019.

[Cam05] Sean D Campbell. A review of backtesting and backtesting procedures. 2005.

[CGW17] David LEE Kuo Chuen, Li Guo, and Yu Wang. Cryptocurrency: A new investment opportunity? *The journal of alternative investments*, 20(3):16–40, 2017.

[Chr08] Peter Christoffersen. Backtesting. *Available at SSRN 2044825*, 2008.

[Coi] Coinigy. The coinigy story. https://www.coinigy.com/en/company/about/. Accessed: 03/05/2023.

[CST10] Rama Cont, Sasha Stoikov, and Rishi Talreja. A stochastic model for order book dynamics. *Operations research*, 58(3):549–563, 2010.

[Dav] Jignesh Davda. Backtrader for backtesting (python) – a complete guide. https://algotrading101.com/learn/backtrader-for-backtesting/. Accessed: 09/05/2023.

[DP17] Massimo Di Pierro. What is the blockchain? *Computing in Science & Engineering*, 19(5):92–95, 2017.

[EVDHM05] Robert J Elliott, John Van Der Hoek*, and William P Malcolm. Pairs trading. *Quantitative Finance*, 5(3):271–276, 2005.

[FVB+22] Fan Fang, Carmine Ventre, Michail Basios, Leslie Kanthan, David Martinez-Rego, Fan Wu, and Lingbo Li. Cryptocurrency trading: a comprehensive survey. *Financial Innovation*, 8(1):1–59, 2022.

[Goo] GoodCrypto. Coinigy review: A complete guide. https://goodcrypto.app/coinigy-review-a-complete-guide/. Accessed: 03/05/2023.

[GY22] Huaqun Guo and Xingjie Yu. A survey on blockchain technology and its security. *Blockchain: research and applications*, 3(2):100067, 2022.

[Hay] Adam Hayes. Trade definition in finance: Benefits and how it works. https://www.investopedia.com/terms/t/trade.asp. Accessed: 02/12/2022.

[Her22] Ilkka Herrala. Algorithmic evaluation of a technical analysis investment strategy. 2022.

[HL15] Campbell R Harvey and Yan Liu. Backtesting. *The Journal of Portfolio Management*, 42(1):13–28, 2015.

[HSK20] Ganesh Harke, Mikhail Shishlenin, and Suresh Koppisetti. Application of algorithmic trading strategies for retail investors. *Available at SSRN 3904097*, 2020.

[HU21] Robert Hudson and Andrew Urquhart. Technical trading and cryptocurrencies. *Annals of Operations Research*, 297(1):191–220, 2021.

[Inv] Investor.gov. Investor bulletin: Understanding order types. https://www.investor.gov/introduction-investing/general-resources/news-alerts/alerts-bulletins/investor-bulletins-14. Accessed: 27/03/2023.

[IZO17] Muhaimin Islam, David Francis Zielinski, and Obianuli Ebubechukwu Obiora. Trading system development. Technical report, 100 Institute Road, Worcester MA 01609-2280 USA, June 2017.

[Izz] Kamilia Izzrech. Trading with cryptocurrencies on electronic platforms.

[Jon13] Charles M Jones. What do we know about high-frequency trading? *Columbia Business School Research Paper*, (13-11), 2013.

[JPSW19] Thomas Johann, Tālis J Putniņš, Satchit Sagade, and Christian Westheide. Quasi-dark trading: The effects of banning dark pools in a world of many alternatives. 2019.

[Ken] Will Kenton. What is an order book? definition, how it works, and key parts. https://www.investopedia.com/terms/o/order-book.asp. Accessed: 03/12/2022.

[KSL+20] Selina Khoirom, Moirangthem Sonia, Borishphia Laikhuram, Jaeson Laishram, and Tekcham Davidson Singh. Comparative analysis of python and java for beginners. *Int. Res. J. Eng. Technol*, 7(8):4384–4407, 2020.

[L+04] Bob Litterman et al. *Modern investment management: an equilibrium approach*. John Wiley & Sons, 2004.

[Liaa] Garcia Liang. The risks of using third-party data. https://www.integrate.ai/blog/the-risks-of-using-third-party-data. Accessed: 02/06/2023.

[Liab] Garcia Liang. The risks of using third-party data. https://www.integrate.ai/blog/the-risks-of-using-third-party-data. Accessed: 02/06/2023.

[Lim] Binance Holdings Limited. coinmarketcap. https://coinmarketcap.com/. Accessed: 15/03/2023.

[MAN+16] Vivian A Maese, Alan W Avery, Benjamin A Naftalis, Stephen P Wink, and Yvette D Valdez. Cryptocurrency: A primer. *Banking Lj*, 133:468, 2016.

[Mil18] Monia Milutinović. Cryptocurrency. *Ekonomika*, 64(1):105–122, 2018.

[Mos] Nicolás Mosconi. Should you use golang? advantages, disadvantages & examples. https://www.devlane.com/blog/should-you-use-golang-advantages-disadvantages-examples. Accessed: 02/06/2023.

[MP18] Douglas R McKay and Daniel A Peters. Digital gold: A primer on cryptocurrency. *Plastic Surgery*, 26(2):137, 2018.

[MSH+16] Ujan Mukhopadhyay, Anthony Skjellum, Oluwakemi Hambolu, Jon Oakley, Lu Yu, and Richard Brooks. A brief survey of cryptocurrency systems. In *2016 14th annual conference on privacy, security and trust (PST)*, pages 745–752. IEEE, 2016.

[NGHS17] Michael Nofer, Peter Gomber, Oliver Hinz, and Dirk Schiereck. Blockchain. *Business & Information Systems Engineering*, 59:183–187, 2017.

[nw] 2nd watch. A high-level overview of google bigquery. https://www.2ndwatch.com/blog/high-level-overview-google-bigquery/. Accessed: 02/06/2023.

[P+21] CMA Panigrahi et al. Trend identification with the relative strength index (rsi) technical indicator–a conceptual study. *Panigrahi AK, Vachhani K, Chaudhury SK, Trend Identification with the Relative Strength Index (RSI) Technical Indicator–A Conceptual Study, Journal of Management and Research Analysis*, 8(4):159–169, 2021.

[Pal19] K Palanivel. Blockchain architecture to higher education systems. *Int. J. Latest Technol. Eng. Manag. Appl. Sci*, 8:124–138, 2019.

[PS21] Ingolf Gunnar Anton Pernice and Brett Scott. Cryptocurrency. *Internet Policy Review, Glossary of decentralised technosocial systems*, 10(2), 2021.

[Qua] QuantConnect. Documentation learn to use quantconnect and explore our features. https://www.quantconnect.com/docs/v2. Accessed: 03/05/2023.

[R21] James R. Cryptocurrency primer: A quick guide on cryptocurrency application and risks. *Trevor Schwartz investements*, 133:468, 2021.

[Roș09] Ioanid Roșu. A dynamic model of the limit order book. *The Review of Financial Studies*, 22(11):4601–4641, 2009.

[Sch] Charles Schwab. 3 order types: Market, limit and stop orders. Accessed: 27/03/2023.

[She15] Esther Shein. Python for beginners. *Communications of the ACM*, 58(3):19–21, 2015.

[SJ19] Kapil Sharma and Deepakshi Jain. Consensus algorithms in blockchain technology: A survey. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2019.

[Spa] John Spacey. What is market data. https://simplicable.com/new/market-data. Accessed: 02/12/2022.

[Spö20] Jan Spörer. Backtesting of algorithmic cryptocurrency trading strategies. *Available at SSRN 3620154*, 2020.

[Tay21] Yared Taye. *Trading and Investing Systems Analysis*. PhD thesis, Worcester Polytechnic Institute, 2021.

[TEA] THE INVESTOPEDIA TEAM. Coinigy definition. https://www.investopedia.com/terms/c/-coingy.asp. Accessed: 03/05/2023.

[Tra] TradingView. Tradingview. https://www.tradingview.com/. Accessed: 09/05/2023.

[Tre] John Treadle. Tradingview – a complete 2023 beginner's guide – your best platform for charting and technical analysis. https://tradingtools.net/tradingview-complete-beginners-guide/. Accessed: 09/05/2023.

[Tsa] Maria Tsarouva. How to use neural networks in financial trading and analysis. https://www.itechart.com/blog/neural-networks-in-trading/. Accessed: 31/03/2023.

[VF09] Bruce Vanstone and Gavin Finnie. Financial trading systems using artificial neural networks. In *Encyclopedia of Information Science and Technology, Second Edition*, pages 1532–1536. IGI Global, 2009.

[VT21] D Vidal-Tomás. An investigation of cryptocurrency data: the market that never sleeps. *Quantitative Finance*, 21(12):2007–2024, 2021.

[WYJ+20] Jun Wang, Qian Yang, Zhanlei Jin, Wenzhang Chen, Tiejun Pan, and Jian Shen. Research on quantitative trading strategy based on lstm. In *2020 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, pages 266–270. IEEE, 2020.

[YA22] Mohamad Z. Yusoff, J. and M. Anuar. A review: Consensus algorithms on blockchain. *Journal of Computer and Communications*, pages 37–50, 2022.