Accounting automation for cryptocurrencies transactions

How to build an interface that connects a smart contract to a classic accounting tool

6th of July 2023

Authors

Aaron Pozzi

Management

Marco Pecoraro

Management

Premton Avdyli

BD/CRM

Erik Schiess

Operations

Patrizia Leonardi

Publication

SMART CONTRACTS LAB

Abstract

Traditional double-entry bookkeeping is the base of today's accounting. Despite its impact on the business world, it is vulnerable to manipulation and because of its nature, as well as its lack of standardization, it is also time-consuming and error-prone. The combination of the double-entry bookkeeping method with blockchain technology is called triple-entry bookkeeping. It could be a possible solution for the addressed problems, by providing a decentralized and transparent system with tamperproof transaction records and partwise automatization.

The following paper emphasizes the importance of double-entry bookkeeping for accurate financial reporting and highlights the limitations of manual accounting systems. It briefly introduces blockchain accounting, related topics, and a summary of the Swiss accounting software Banana+. Furthermore, the report discusses Swiss accounting standards applicable to companies utilizing blockchain technology by covering Swiss Value Added Tax (VAT) and corporate income tax. Finally, it proposes a Python-based interface solution, which connects blockchain transactions done on the Polygon Testnet with the accounting platform Banana+. The code facilitates the exchange rate conversion of blockchain transactions into Swiss francs and provides tax calculations. Detailed instructions for modifying and executing the code are provided. The outcome of the code execution is three CSV files that can be imported into the accounting software Banana+, intending to make blockchain transactions accountable.

Table of Contents

List of Figures

List of abbreviations

BCP	Blockchain Presence AG
CIT	Corporate Income Tax
FTA	Federal Tax Administration
POW	Proof-of-Work
POS	Proof-of-Stake
SCL	Smart Contracts Lab
SMEs	Small and Mid-Cap Enterprises
VAT	Value-Added Tax

1. Introduction

Traditional accounting systems face significant challenges due to their reliance on centralized databases and intermediaries. These inherent vulnerabilities expose them to errors, fraud, and manipulation, compromising the integrity of financial records (Demirkan et al., 2020). Additionally, the process of reconciling transactions among various parties is often complicated and prone to errors, primarily because different accounting platforms are employed (Zenko, 2022). Another critical concern in accounting is the substantial costs associated with auditing, particularly for large organizations (Imhoff, 2003).

To address these issues and revolutionize the field of accounting, the emergence of blockchain technology offers promising solutions. By implementing a decentralized and transparent system, blockchain offers a tamper-proof ledger where all transactions can be securely recorded and stored (Atlam et al., 2018). This enhanced transparency and accountability mitigate the risks associated with fraud and errors. Moreover, the integration of smart contracts further streamlines and automates accounting processes, reducing the reliance on manual interactions, and enhancing efficiency and accuracy.

However, despite the potential benefits, blockchain accounting faces various challenges due to its early stage of development (Dai & Vasarhelyi, 2017). These challenges encompass the high implementation costs, the absence of standardized practices, the lack of a regulatory framework, the adoption rate, and the emergence of new potential forms of fraud. To provide a comprehensive examination of blockchain accounting, this paper will specifically focus on crypto companies, which already store their transactions on the blockchain. We will base our findings and proposed solution on an already-existent code created by Andrea Giambonini, a former employee of Blockchain Presence AG (BCP). BCP is a low-cost smart contract oracle with on-chain authentication (Blockchain Presence AG, n.d.). Our aim is to provide a clear guideline to crypto companies on how to benefit from a direct link between blockchain and an accounting platform. We will address two main hypotheses:

H1: The code presented in the solution will completely automate the accounting process.

We will verify this hypothesis by analyzing the results obtained by running the Python code presented in Chapter 6.2. Our goal is to propose a fully automated Python interface where an accounting software like Banana is directly connected to a blockchain network like Polygon Testnet “Mumbai”. This approach creates an almost foolproof system for the import of blockchain transactions, which then can be further processed in the accounting software.

H2: Our paper provides clear and precise guidance on how to apply our idea to other crypto companies.

The second hypothesis is that we want to provide clear and precise guidance for or target companies. This should enable them to profit from the numerous advantages of blockchain technology while minimizing its pitfalls. We will verify this by looking at the advantages and

disadvantages that our paper provides by trying to be as objective as possible in the conclusion and limitation chapter.

The paper is divided into seven parts. The first chapter contains the introduction, the following chapter the methodology, followed by the third chapter “Theory review”, which is divided into basic accounting, blockchain accounting, and use cases. The fourth chapter is dedicated to Banana, a simple accounting software that will be part of the proposed solution. The fifth chapter deals with accounting standards and regulations, pointing out important legal aspects of the accounting world and current trends of standardization, while the sixth chapter contains the description of a fully automated Python interface, with the complete code attached in the appendix. Finally, the conclusion and a brief outlook on future perspectives are part of the seventh chapter.

2. Methodology

To answer the research question, we posed in the introduction, we decided to use a different approach depending on the chapter under discussion. For the more theoretical chapters: Basic Accounting, Blockchain Accounting, and Practices Standardization, we selected some of the most popular research databases, including Google Scholar, ScienceDirect, and Swisscovery to gain access to pertinent papers. The search was conducted using the keywords such as "blockchain", "accounting" and "Swiss accounting standards". In addition, the bibliography of relevant research papers was examined to identify further sources. For the chapter focusing on the explanation of a basic accounting method, the reference website Banana.ch was mostly used as it already contained all the information we were looking for. Our attention was limited to the inclusion of English-language literature and some references in German regarding Swiss accounting standards.

In the section dedicated to our solution, since we are presenting a Python code that unifies a smart contract and all the transactions made, we had to search for some information from previous works. We found an interesting code made by Andrea Giambonini in the BCP framework (Giambonini, 2021) and we adapted it by adding some new details and features. To ensure readability for people without a technical background we first explain simply what the code does and what is needed to change to make it work for any company. Important to notice here is that the solution is strictly connected with the smart contract. One should also change function names because every smart contract is defined in its own way by the creator. Therefore, it is important to recall the correct names of the function in the Python file. The second explanation is contained in the appendix. Here we tried to directly translate what is each function doing and how the different code lines are connected. Overall, this document aims to provide readers with a clear understanding of the general concept of accounting, the associated problems, and a practical solution.

3. Theory Review

3.1. Basic Accounting

In this theory review, the history and basic concepts of accounting will be examined. Afterward, potential problems will be pointed out, which can arise with basic manual accounting systems.

The double-entry system, which was invented by the Italian merchant Pacioli (1494) at the end of the 15th century, is the underlying concept of accounting (Sangster & Scataglinibelghitar, 2010). It requires that every entry to an account have a corresponding and opposite entry to a different account. Each account thus possesses two fundamental operations: debit and credit (Hayes, 2021). Credit decreases an asset and increases a liability, while debit leads to the opposite outcome. These two fundamental operations can be translated into the basic accounting equation (Hayes, 2021):

$$Assets = Liabilities + Equities$$

This equation represents the relationship between the resources of a company (assets) and the claims against those resources (liabilities). The resulting difference between assets and liabilities is the owner's equity and represents their residual income (Fernando, 2023).

The double-entry system of accounting is essential for ensuring accurate financial reporting, as it provides a systematic and organized way of recording financial transactions. Without it, it would be impossible to maintain the accounting equation transparently and prepare reports like balance sheets, income statements, etc. (AccountingTools, 2022). Thus, the double entry system allows businesses to organize, record and thus analyze financial transactions. It serves as a useful tool and enables businesses to monitor and understand the financial situation of the organization. While accounting technology and methods have evolved, the underlying principles of the double-entry system have remained the same (Hayes, 2021).

Today, transactions in a double-entry manner are carried out manually, using either fully manual bookkeeping or digital accounting software, which is becoming the standard. Fully Manual accounting involves the use of paper ledgers, journals, and other analog methods to keep track of upcoming transactions (Britannica, n.d.) and is less and less used due to modern digital book-keeping alternatives, which enable businesses to keep track of all transactions in a digital manner, leading to less manual labor and partwise automation (alexandersloan, 2023).

Although both systems may be sufficient for small businesses with a low volume of transactions, both can lead to the addressed problems below, due to their manual nature, which gets worse proportionally to the number and complexity of accountable transactions.

The first problem is the difficulty to keep track of a growing number of transactions (Testbook Edu Solutions Pvt. Ltd., 2023). The volume of transactions can quickly become overwhelming, making it challenging to keep accurate records. Also, wages for the growing number of accountants are skyrocketing. Thus, this problem is not only leading to incorrect records or missed transactions but also to smaller profits due to growing wage costs (Ellis, 2022).

Another aspect of accounting is reconciliation. This process involves reconciling transactions between different parties, which can be time-consuming and error-prone because each organization uses its own accounting platform and accounting software, which is often incompatible (Tuovila, 2022). Also, there are numerous accounting standards, intermediaries, and legal and regulatory boundaries in the accounting process, which create a highly complex working environment and high costs of legal consultation (Botzem & Quack, 2009).

A third problem with manual accounting is that it can be challenging to maintain the security and confidentiality of financial information since it is either stored offline in paper ledgers or journals, or in centralized databases, which both become a centralized point of failure in case of fraud or manipulation. Also, paper-based ledgers, journals, or hard drives can be easily lost, stolen, or damaged, putting sensitive financial information at risk. This can result in financial fraud and theft, which can cause significant financial losses for a business. Despite security measures, this problem remains important due to more and more sophisticated security breaches (Alawida et al., 2022).

The last problem of manual accounting is the risk of human error or manipulation. Recording transactions manually can be tedious and time-consuming, also leaving space for conscious manipulation. Such error or manipulation results in inaccuracies and mistakes in financial statements, wrong decision-making, or financial mismanagement (Testbook Edu Solutions Pvt. Ltd., 2023).

In conclusion, accounting is an essential aspect for businesses to monitor and understand their financial information. The basic double-entry system leads to the well-known accounting equation and is thus crucial for accurate financial reporting and information transparency. But unfortunately, due to its manual nature, it can create several problems. Most importantly the risk of human error, overwhelming amounts of information, security and confidentiality breaches. Thus, it is time for more sophisticated accounting systems, that address the problems mentioned and rely not only on the double entry system but on the newest technological developments such as blockchain technology or machine learning methods. This next step in the evolution of accounting is called “triple-entry accounting” and has the potential to revolutionize the accounting industry. Following Fraser (1993), this concept was first introduced by a professor at Carnegie Mellon University Yuji Iljiri during the 80s. In more recent years, the idea was expanded to blockchain technology, which as a third component in the accounting process links the books and transactions together in a tamperproof ledger (Rana, 2020). In the “triple-entry” world, all accounting entries are cryptographically sealed, which prevents manipulation reliably and makes faults and human-made errors retraceable. Smart Contracts automate certain features and thus streamline the accounting process and reduce human-made errors. Also, since the transactions are

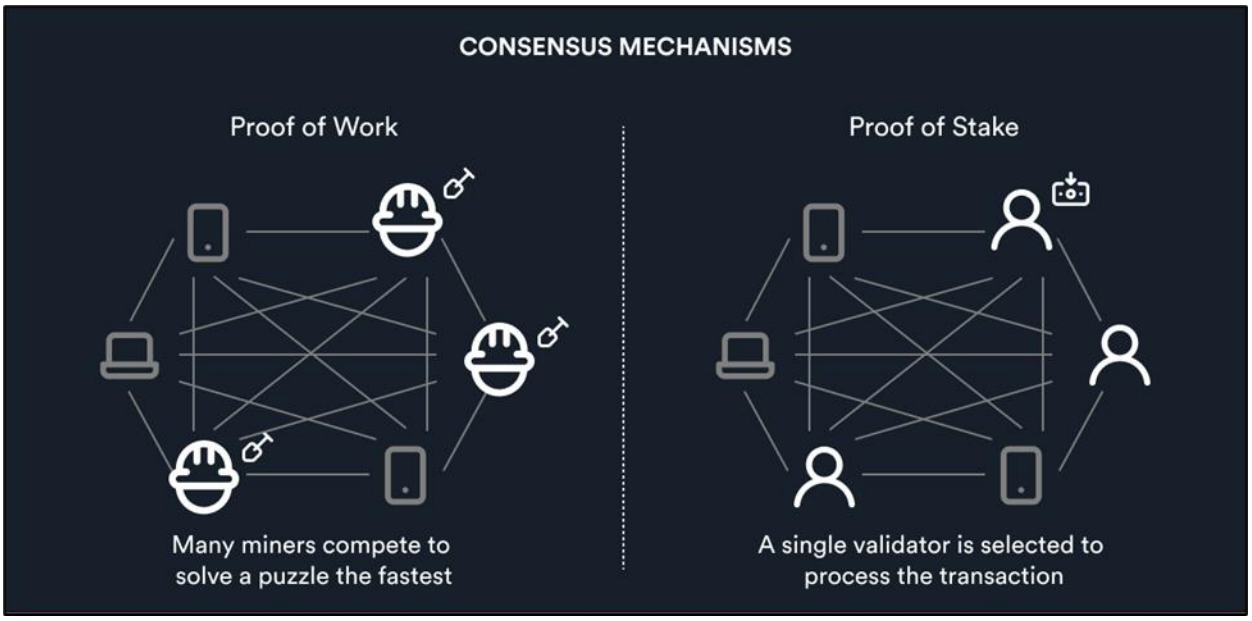
written on one blockchain, there is no need for separate accounting software. Thus, accounting data can be easily shared between separate firms, auditors, and authorities, making fraud nearly impossible and reducing the number of required intermediaries. And since blockchain technology does not rely on centralized databases, accounting databases do not become centralized points of failure in the future.

Unfortunately, since blockchain technology is still in the development phase, this next step in the accounting world has not arrived yet. But it offers a promising solution for the future, gaining more and more popularity in the accounting world.

3.2. Blockchain Accounting

The previously mentioned acceptance of blockchain technology has already led to the increasing use of it in accounting as well as to a significant transformation in the way financial transactions are recorded and maintained (Adams, 2020). The classic accounting system differs from Blockchain accounting, which is a system that uses a decentralized digital ledger to record financial transactions to create secure and transparent accounting records. It operates through a network of nodes, where each node has a copy of the ledger. The nodes use a consensus mechanism; transactions are verified and validated by the nodes in the network, creating a reliable system of record-keeping (Deloitte, 2017). Depending on which Blockchain the accounting is based on we either have a proof-of-work (POW) or proof-of-stake (POS) consensus mechanism. Traditional accounting relies on centralized systems, where financial data is stored in a single location and managed by a central authority. Blockchain, on the other hand, is decentralized, meaning that it is not controlled by a single entity. Instead, all participants in the network have a copy of the ledger and are responsible for validating transactions (ACCA, 2022). It ensures precise transaction accounting. The graphic below illustrates the typical consensus mechanisms which are being used for Blockchain accounting. In POW the mentioned nodes are miners, in POS they're validators.

Figure 1: Two different consensus mechanisms operating on nodes (miners/validators)



Source: Own illustration

One of the main advantages of Blockchain accounting is that it allows for real-time recording of transactions. This means that financial data is always up to date, providing an accurate and comprehensive view of a company's financial position at any given time (FreshBooks, 2023). This is very important as it is fundamental for businesses to make informed decisions, prepare for tax season, and secure funding. Maintaining up-to-date financial records is much more crucial for

SMEs (Small and Mid-Cap Enterprises) to manage their cash flow effectively, track expenses and identify potential financial issues before they become significant problems. Without accurate financial statements, it can be challenging to identify areas where expenses can be reduced, profits can be increased, and create forecasting budgets (Visram, 2021).

Furthermore, several other benefits come with Blockchain-based accounting; one of the benefits of blockchain accounting is its ability to reduce the risk of fraud and errors. The decentralized nature of blockchain technology makes it difficult to tamper with or alter data (Originstamp, 2023). Moreover, the use of smart contracts and automated processes enhance the accuracy and consistency of data in blockchain-based accounting systems (ACCA, 2022). In addition to increased security and transparency, the use of blockchain technology in accounting can also reduce costs and streamline processes. Blockchain technology reduces the need for intermediaries in transactions, thus reducing costs and increasing efficiency (Kunselman, 2021). The use of blockchain technology can also make the auditing process more efficient and accurate (Tan & Low, 2019).

Despite these benefits that speak for the use of Blockchain accounting, there are also some potential drawbacks to blockchain accounting. One of the major concerns is the lack of standardization in the use of blockchain technology in accounting (Originstamp, 2023). Furthermore, there are concerns regarding data privacy and security, which must be addressed for the widespread adoption of blockchain technology in accounting (Deloitte, 2017). Blockchain accounting is still difficult to implement in real-world use cases, but there are ways to take advantage of the benefits which Blockchain Technology brings, without the need to implement the whole accounting process through a smart contract. An example of this would be our solution which we implemented for this paper; our team has created a link between a smart contract and the accounting platform Banana+ to simulate transactions while maintaining the classic double-entry system accounting, the execution of which we will discuss hereafter.

4. Basic accounting system

In the context of digital transformation and the evolution of technology, the financial management of companies is undergoing significant change. This chapter explores basic accounting systems, which are the essential foundation for the financial management of businesses. Understanding the principles and procedures of such a system is crucial to properly recording, classifying, and analyzing financial transactions. An in-depth look at the innovative Banana accounting software and how it integrates into this framework, offering advanced solutions for efficient and accurate financial management, will be conducted.

The accounting system is a set of procedures and rules used to record and classify all financial transactions of an organization to measure its financial performance (Ghasemi et al., 2011). An accounting system that will be studied in depth is Banana+.

The Banana+ system is an accounting software developed in Switzerland that helps companies manage their finances efficiently and effectively (Banana.ch SA, n.d.-a). This system is intuitive to use and offers a wide range of features, including bookkeeping, financial report generation, and invoice management (Banana.ch SA, n.d.-a). In addition, the Banana+ system offers an intuitive and easy-to-use user interface, making it accessible even for those without formal accounting training. In this way, companies can use the software without necessarily having to hire specialized accounting personnel.

One of the main features of the Banana+ system is its ability to integrate with other software, allowing companies to customize the system to their specific needs. Banana+ can be integrated with various technologies, including blockchain technology. This system already employed blockchain technology in 2002, being the first in the world in the commercial field (Banana.ch SA, n.d.-d).

The use of blockchain in this system can enable companies to record their financial transactions securely and immutably on the blockchain. In fact, manipulating records in an attempt to falsify or delete them is virtually impossible (Schmitz & Leoni, 2019). This ensures the integrity and authenticity of the information over time (Banana.ch SA, n.d.-d).

In particular, a company that operates in the blockchain industry and uses this technology for its financial transactions can easily import data from these transactions into Banana+ using methods such as importing data from a CSV file. This file must be compatible with the format required by Banana+ regarding the position of the information, more precisely the columns of the CSV file have to be the same as those in Banana+. However, it can be customized according to business needs, such as classifying financial transactions or automatically assigning accounting categories (Banana.ch SA, n.d.-c). The file is imported using the data import function in the software.

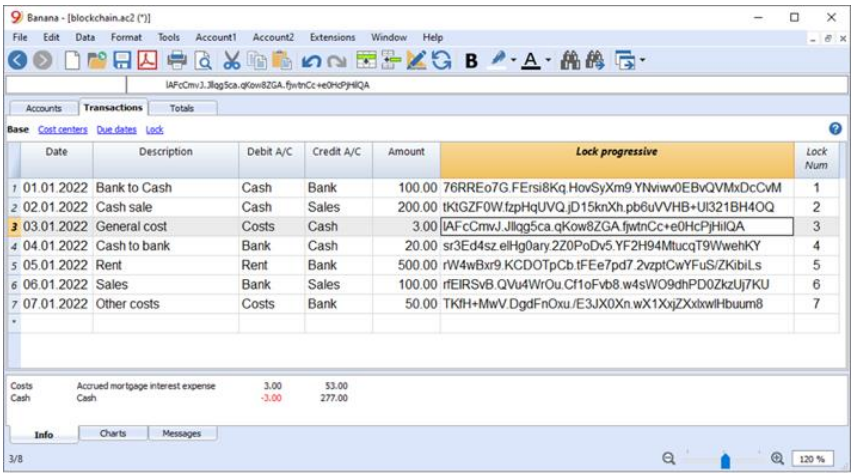
Once the import process is completed, the transaction data will be automatically integrated into Banana+'s accounting system, making transaction information available. In summary, financial transactions made on the blockchain by the company, such as paying creditors or purchasing

cryptocurrencies, can be automatically recorded on Banana+, thanks to the connection between the blockchain and the accounting software. In this way, the company can keep track of its financial activities accurately and transparently, avoiding data entry errors and without having to manually record each transaction.

Furthermore, the Banana+ accounting system using this technology ensures the security of accounting data entered into the system. More precisely, the system allows working in a personalized way, changing the data at will until everything is in place and afterward the command is then used to lock the movements (Banana.ch SA, n.d.-d).

This is where blockchain technology takes over, where for each transaction its seal (also referred to as Hash) is calculated and the movement can no longer be changed (Banana.ch SA, n.d.-d). In this system, the seals are contained and displayed in the "progressive lock" section and each seal is numbered.

Figure 2: Banana+ Accounting with locked movements



Source: Banana.ch SA (n.d.-d)

As shown in Figure 2, each transaction is represented by its own seal. Banana+ accounting works like Bitcoin, so each seal includes both data regarding its corresponding transaction and the seal from the previous row (Banana.ch SA, n.d.-d). This means that when changes are made to a data item, its seal and all subsequent seals will change. For example, if in line number 3 the amount of that transaction is changed, the digital seal of line 3 and subsequent lines shown in the lock progressive column will be different.

Banana+ offers this movement release feature to allow flexibility to the business, such as the need to generate reversals for incorrect entries. This does not reduce security, as it is the responsibility of the manager to verify records, discard invalid ones, and ensure high data quality (Banana.ch SA, n.d.-d).

To determine whether a collection is intact, the seals are recalculated down to the last element and compared with the original seal. If they are the same, the collection is intact. A firm can prove that

the accounting data have not changed by keeping copies of the seals (Banana.ch SA, n.d.-d). The auditor will be able to verify that no changes have occurred.

To conclude, the integration of blockchain technology into Banana+ enables businesses to manage their financial assets and track transactions securely and transparently on the blockchain. This makes Banana+ an ideal choice for companies and start-ups that operate in the field of blockchain technology and want a reliable and easy-to-use accounting system.

5. Practices standardization

This chapter discusses the challenges of implementing blockchain technology in accounting and proposes a solution to address these obstacles. The first section of the text examines the problems that impede the adoption of blockchain technology in accounting, such as the need for confidentiality, the potential for fraud, limited verification capability, and regulatory challenges. The second section presents a proposed solution that circumnavigates these obstacles by connecting the blockchain transaction with an accounting platform. The text also explains the accounting standards that Swiss companies working with blockchain must adhere to, including the rules on balance sheets, income statements, financial statement notes, and cash flow statements. Finally, the text explains the Swiss Value Added Tax (VAT), corporate income tax, and exchange rates applicable to Swiss blockchain companies.

5.1. Blockchain Accounting Problems

Although blockchain accounting entails undoubtedly numerous benefits, its implementation is still facing problems. As Yu et al. (2018) argued, it is improbable that in the short term blockchain would have a great impact on accounting processes. Therefore, they offer a different vision about blockchain focusing on using this new technology as a platform to voluntarily disclose internal data. The authors claim that blockchain technology does not eliminate the risk of fraud since the firm could insert manipulated raw data. However, this issue may be addressed by conducting audits that will prioritize analyzing the rationality and legitimacy of business operations rather than preventing accounting fraud (Yu et al., 2018). Consequently, the reduction of risk and costs is only partially addressed by an accounting system implemented on the blockchain. In addition, a new level of technical expertise is required. Financial accountants need to be able to verify the authenticity of source documents and ensure the validity of smart contracts utilized in blockchain accounting. Consequently, higher employee qualifications would be needed.

Coyne & McMickle (2017) conclude that an immediate possibility of accounting using blockchain is infeasible. They identify three obstacles that impede the use of blockchain in accounting: first, the need for confidentiality, which makes public blockchains unsuitable; second, the potential for firms to manipulate private blockchains retroactively; and third, the limited transaction verification capability provided by the blockchain (Coyne & McMickle, 2017). Finally, even Pawczuk et al. (2019) reported that there are obstacles to the adoption of triple-entry accounting such as regulatory challenges, potential security risks, and uncertainties regarding the return on investment.

Regulatory challenges as reported are a key factor in blockchain accounting adoption and they are a difficult hurdle to overcome. According to Gerard Brennan, an auditing and blockchain expert, “there is an urgency for consensus, regulations, and standards” (CPA.com, 2019). In a survey conducted by Gauthier & Brender (2021), many participants emphasized the absence of auditing standards that cater to modern technologies. In addition, the time needed to issue new standards ranges from at least 5 to 10 years. It does not fit the rate at which new technologies are growing.

In the same study, the authors find a growing demand and interest in blockchain accounting meaning that auditors and accountants are aware of the incredible potential of blockchain accounting.

5.2. Our Approach

Despite all these challenges, it would be a missed opportunity to disregard the immense potential of blockchain technology in revolutionizing accounting processes. Therefore, we circumnavigated the issues of a direct link with a smart contract proposing a connection coded in phyton between the transactions on the blockchain and an accounting platform, in our case Banana +. We will analyze our solution more in detail in Chapter 6 while following are reported all the accounting standards that Swiss companies that work on the blockchain should adhere to.

5.2.1. Swiss standards for blockchain reporting

All the commercial bookkeeping and accounting rules applicable to all companies in Switzerland are included in Article 957 ff. of the Swiss Code of Obligations (Bundesrecht, 2023). They rule the structure of the balance sheet, income statement, and the content of financial statements notes.

However, larger entities that undergo regular audits are required to include a cash flow statement, disclosures in the notes to the financial statements, and a management report in their financial reporting. In addition, listed companies, cooperatives with over 2,000 members, and foundations that are regularly audited must prepare a separate financial statement according to an accepted standard, such as IFRS or Swiss GAAP FER. However, they may be exempt from this requirement if a consolidated financial statement is prepared in accordance with an accepted standard (PWC, 2015).

With our proposed solution we are transmitting the blockchain transaction with Banana+, which helps us to create financial statements that are compliant with the abovementioned standards.

5.2.2. Value-added Tax

The Swiss Value Added Tax (VAT) is a consumption tax overseen by the Swiss Federal Tax Administration, applicable to domestic goods and services. The standard VAT rate in Switzerland is 7.7%, which is the lowest in Europe (Swiss Confederation, 2023b). In Switzerland the Swiss Confederation levies VAT on supplies of goods and services rendered by taxable persons in Switzerland (domestic tax), the acquisition of supplies from enterprises with their place of business abroad by recipients in Switzerland (acquisition tax), and the import of goods (import tax) (LawyersSwitzerland.com, 2023). Swiss blockchain companies are naturally included in this list even if they are gaining their income in cryptocurrencies. In fact, they obtain payment tokens that should be considered equivalent to legal tender (swisstaxexpert.com, n.d.).

However, the location of the customers that are paying in those tokens cannot be clearly defined due to the anonymity characteristic of blockchain technology. This raises a problem in the VAT calculation.

Fortunately, other than the standard method there exist other tax reporting ways to address these issues. One of them is the so-called “Saldosteuersätzes Method”. It simplifies reporting to the Federal Tax Administration (FTA) as there is no need to calculate previous taxes (Eidgenössische Steuerverwaltung, 2018). Under this reporting method, the tax liability is determined by multiplying the gross turnover (including taxes) by the authorized balance rate set by the FTA without any need to identify the source of the customer. The applicable rate for each sector or activity is determined by the ordinance on the value of the balance rates. However, taxpayers should meet the following conditions to calculate their tax using this method: the annual taxable turnover (including VAT) does not exceed CHF 5.005 million, and the tax payable does not exceed CHF 103,000 per year. This is calculated by multiplying the total taxable turnover by the balance rate applicable to the corresponding sector or activity (Eidgenössische Steuerverwaltung, 2018).

As reported before, the applicable rate depends on the sectors or activities that the company provides. For the category “Databases: sale of data and information of all kinds”, where oracles should lay, it has been agreed to be 5.9% since the first 01.01.2018 (Swiss Confederation, 2017). It has already ruled a slight raise in the VAT rate from 01.01.2024 (Swiss Confederation, 2023a).

In the proposed solution, as explained in the following chapter, our code can subtract the VAT rate and give a post-tax amount that can be inserted directly into the software Banana+.

5.2.3. Corporate Income Tax and Exchange Rates

In Switzerland, any cryptocurrency income must be included in the taxable income in the amount of the equivalent in Swiss francs. Other crypto activities that must be included in the taxable income are mining, staking, and airdrops (Deloitte, 2022).

Switzerland has a unique tax system when it comes to income taxation, which is comprised of potentially three different taxes: Federal Income Tax, Canton Income Tax, and Municipal Income Tax. Federal Income Tax is the same tax rate across the country, whereas Canton Income Tax and Municipal Income tax vary depending on the taxpayer's residence. The direct federal corporate income tax (CIT) is imposed at a fixed rate of 8.5% on post-tax profits. As a result, CIT is eligible for tax deductions, thereby decreasing the taxable income and leading to an approximate direct federal CIT rate on pre-tax profits of 7.83% (PWC, 2023). Unlike some countries, Switzerland does not offer specific tax breaks for cryptocurrencies. Due to crypto gains being tax-exempt for private investors, they cannot deduct crypto capital losses (Vontobel, 2022). However, those who qualify as self-employed traders or a business may be able to deduct crypto capital losses to reduce their tax bill (Koinly, 2023). Individuals who are self-employed in the field of cryptocurrency trading or operate crypto-related businesses are subject to Capital Gains Tax, which is applied to the profits generated from the sale or exchange of crypto assets. The tax rate can reach up to 7.8% (Rue, n.d.). In conclusion, the tax laws governing crypto in Switzerland are unique and require

careful attention to ensure that taxpayers pay the least amount of tax possible. However, our solution will not be able to report automatically corporate income taxes. Further research and implementations are still needed. This point is covered extensively in Chapter 7.

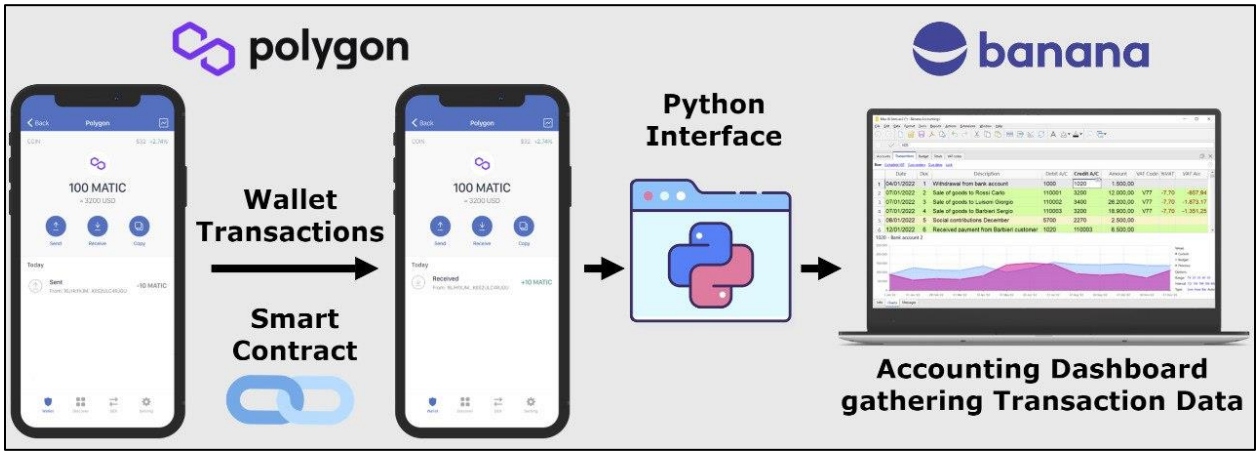
It is now widely recognized that one of the biggest problems of cryptocurrencies is their volatility. This can cause several issues for our target companies while accounting for revenues and expenses, especially in the choice of the exchange rate. Different approaches can lead to huge differences in accounting numbers. We decide to use a consistent approach that avoids manual work in order to minimize error probability. Thanks to a link with a digital asset data platform, our code will automatically convert the cryptocurrency into Swiss francs. It will look for the closing exchange rate at the transaction date on the Coingecko platform, and convert the amount automatically (CoinGecko, 2023).

6. Our solution

Our solution is based on a code created by Andrea Giambonini in the framework of Blockchain Presence AG (Giambonini, 2021). Our aim was to provide a new version of the Python code in order to automatize the transmission of data regarding blockchain transactions that happens through a smart contract. Transactions are performed on the Polygon Testnet called Mumbai, which allows us to simulate them without any risk or expense, as the MATIC tokens which we use in the Testnet have no monetary value. We decided to implement the solution using the data present on the smart contract created by Blockchain Presence AG because we had more data available than creating them by ourselves with a new smart contract. However, it is important to note that the application of this idea is not immediate for other startups. As already explained in the methodology chapter, this Python code is closely related to the smart contract. Each smart contract is defined differently, so if one would want to replicate the same code for a completely different smart contract, it should also relate carefully to it by calling the different functions as set out in the smart contract (more precisely as defined in the .json file, in the case of our solution it is called contract_abi.json).

In order to better understand the process involved behind the idea, Figure 2 below shows how the Python interface allows us to gather all the MATIC transactions that have been done in the Mumbai Testnet and automatically transfer them into Banana+ through the received CSV files.

Figure 3: The solution in a nutshell



Source: Own illustration

6.1. Python code Explanation

One problem that many companies encounter is that of collecting data on transactions in cryptocurrencies. Indeed, it is not possible to automate the transmission of data such as hashes, information on the sender/receiver, the conversion of a cryptocurrency into the desired currency (in our case into CHF), the cost of transactions (Gas), and other important data (see Figure 2). Indeed, one could also do the entire data transfer process manually. The advantage of a manual

method is that no coding knowledge is needed, but only knowing how to use a system such as Polygonscan or Etherscan and copying the data to an Excel table is enough. The huge disadvantage, however, is the amount of time consumed in doing this transcription, which can become unsustainable if large numbers of transactions have to be reported. In our case with transactions on the blockchain this can become a big problem with a manual system since in the blockchain world due to the high volatility there are numerous transactions per day. Moreover, the costs are not only in terms of time but also in terms of money. It is enough to think how much money could be wasted on repetitive, error-prone work. Therefore, the advantages of an automatic method outweigh its disadvantages.

The creation of the Python code makes it possible to automate the transmission of transaction data from the smart contract to the accounting system of one's choice. As you can see in the next chapter the code produces as output 3 comma-separated files (.csv) that can be read with any accounting program. In our case, we chose to use Banana+ since it is one of the most widely used in Switzerland and has a simple and easy-to-use layout. The last passage is then to import the data into the Banana file (.ac2) that contains all the journal entries created specifically for the company.

It is very important to emphasize that the functioning of the code presented in Appendix A goes hand in hand with the proper functioning of Python. Before deploying the code, it is important to install the desired program (e.g., Visual Studio Code) and all the packages required to run it. In addition, the code must be modified in the points listed below (see Appendix A for the number references):

1) Change the WebSocket URL
2) Change the address of the smart contract SCL_ADDRESS
3) Change the Coingecko reference if you need to have the data in another currency (e.g., in USD)
4) Modify the start block to make the code work from the point you want to start by inserting the preferred one in the file .txt called “startblock”
5) Change the function names with something unrelated to Smart Contracts Lab (SCL)

When running the code, at some point there will be a question popping up about VAT which needs to be answered with “y” for yes or “n” for no. Then a second question will be shown but this time related to the desired delimitator number to take into consideration, the answer should be a “,” or “;” depending on the preference. After this passage, the three CSV files should be created in the pc environment and then they can be opened with the program that is best suited. The last step should be to import the data in the CSV files into the environment in which the accounting transactions are recorded. In our case, these transactions are imported into an AC2 file, which is the specific file for accounting in Banana+.

In conclusion, it is important to note that this code is based on the Polygonscan and Coingecko APIs. Therefore, if something does not work, it may also be due to the link that the code has with them. As far as Polygonscan is concerned, the API allows us to extract all transaction data and later enter it into the CSV file. As for the Coingecko API, it allows us to establish a link with the

exchange rates between the cryptocurrency and the reference currency we want to use, in our case the Swiss franc. This last API has a rate limit of 10-30 calls per minute, meaning that if you exceed the request limit you will be blocked, and the code interrupts the data gathering (CoinGecko, 2023). Therefore, it is extremely important that you pay attention to the number of requests for the APIs.

6.2.Results

As already explained before, the outputs that we obtain are three CSV files that can be read with the program that is best suited. In our case we used Banana+. Important to notice here is that all transactions that are reported are referred to the "relay" and not to the "new commitment" made on the smart contract. In fact, the "relay" is the last step that is needed to complete the process, basically, it validates the block headers and verifies the transaction by looking at the other chain of headers (Daneshpajooh, 2022). Therefore the "relay" allows the transaction to be completed and to transmit the money flow to the receiver of the payment in our case.

The results obtained and presented in this section refer to the period of transactions that goes from the 27th of March till the 16th of May. During this period the SCL smart contract conducted various operations, and for this reason, we decided to report them with our code. To ensure better readability only a few of the transactions are reported. All the transactions are available in Appendix B. In the next three sub-chapter, the different files that are created with the Python code are presented, with important values commented on to enhance the readers' understanding of the output.

6.2.1. Output 1

Figure 4: Banana+ CSV Output 1

	Date	Doc	Description	Accoun-tDebit	Accoun-tCredit	AmountCur-rency	Exchan-geCurr-	Vat-Code	ExchangeRate	Amount
1	27.03.2023		0x18f78ff502	1027	3001	47050000.0	gwei	F3	0.983356572882	0.04785
2	27.03.2023		0x18f78ff502	1027	2330	2950000.0	gwei	F3	0.983356572882	0.00300
3	05.04.2023		0x76466493	1027	3001	47050000.0	gwei	F3	0.968804682255	0.04857
4	05.04.2023		0x76466493	1027	2330	2950000.0	gwei	F3	0.968804682255	0.00304

Source: Own illustration

As you can see in the highlighted row, there are a lot of parameters that one needs to consider, when reporting the details of a crypto transaction. In this case, a transaction of the amount of 0.05 MATIC was done between two wallets.

Date: the date is extremely important since every journal entry must refer to the date when the transaction was made.

Description: the description, in this case, is the hash number, which contains the important information regarding the transaction, and it serves as proof for the validation (Coinbase, n.d.). In fact, the hash number is unique for each transaction.

Account Debit: the account Debit in the highlighted transaction is 1027, which stands for "cryptocurrencies account", this number is used in Banana+ to simplify the transmission of the data between different files.

Account Credit: the account Credit in the highlighted transaction is 3001, which stands for "Earnings from cryptocurrencies transactions", this number is used in Banana+ to simplify the transmission of the data between different files. The second row has the same hash, so it refers to the same operation, but in this case, the Account Credit is 2330 because it stands for the VAT account. As explained in Chapter 5, also a company operating in the crypto market must report the Value Added Tax.

Amount Currency: this column contains the amount of money that was transferred in the transaction.

Exchange Currency: this column contains the specification of the currency of the amount of money described in the column before. In this case, the currency is Gwei. The reporting is done in Gwei to facilitate the transformation in CHF in the last passage because in the API the data are presented in Wei. To ensure better clarity, hereby the conversion table is presented:

Figure 5: Matic Conversion Table

Wei (10^{-18})	**Gwei (10^{-9})**	**Matic**
1000000000000000000	1000000000	1

Source: Polygonscan (n.d.)

VAT Code: this column contains the code F3, which represents the VAT code for sales subject to rate 1. (Reference 200, 322). This code is in use since 01.12.2018 (Banana.ch SA, n.d.-b). In our case with the Saldosteuersatz Method is 5.9% as it is explained in Chapter 5.2.2.

Exchange Rate: the presented number in this column represents the exchange rate obtained from the Coingecko API. More specifically, the 0.98335 in the highlighted row indicates how many MATIC corresponds to 1 CHF.

Amount: the amount is in CHF, and it is obtained by dividing 47'050'000 Gwei by the exchange rate. Before doing that, it is necessary to multiply the Exchange rate times 10^9 as you can see in Table 1 before since the AmountCurrency is in Gwei.

The data are luckily automatically presented thanks to our code, but we recommend verifying them also manually in order to see if some error occurred. After this verification process, it is time to import the data into the Banana+ environment, created specifically to include all the journal entries. The result is presented here below:

Figure 6: Banana+ Journal entries as represented in the AC2 file

	Date	Doc	Description	Debit A/C	Credit A/C	Amount Currency	Exchan-geCurr-	VAT Code	ExchangeRate	Amount CHF
1	27.03.2023		0x18f78ff50	1027	3001	47050000.00	gwei	F3	0.983356573	0.04785
2	27.03.2023		0x18f78ff50	1027	2330	2950000.00	gwei	F3	0.983356573	0.00300
3	05.04.2023		0x7646649:	1027	3001	47050000.00	gwei	F3	0.968804682	0.04857
4	05.04.2023		0x7646649:	1027	2330	2950000.00	gwei	F3	0.968804682	0.00304

Source: Own illustration

As you can see the journal entries are similar to the ones presented before with the CSV file. The only difference is that they are now automatically linked to all accounts, which allows the creation of an income statement and balance sheet immediately. An overview of this feature can be seen in Appendix B.

In conclusion, it is important to note, however, that when creating the initial file by defining the type of accounts and business model of the company, attention should be paid to how many numbers after the comma are to be considered in the "Amount CHF" entry. Indeed, if one does not pay attention to this, the Banana+ software rounds to two digits automatically, which can be a big problem especially when recording extremely small amounts as in our example.

6.2.2. Output 2

Figure 7: Banana+ CSV Output 2

	hash	orderID	receiverA-ddress	DateTime	sen-der_PIN	senderID	commit-mentID	Relay_Status-Flag	GasObtainedF-orDelivery (GWei)	Unused_GasForDelivery (GWei)	commitment_fee (GWei)	Sender_Profit (Gwei)	ExchangeRate	Sender_Profit (CHF)
1	0x18f78f	266.0	0x067504	27.03.2023	0x81996l	1.0	1	True	14400000.0	10167000.00007055	100000000.0	60167000.00007055	0.983356572882	0.061185333641218690
2	0x76466	270.0	0x067504	05.04.2023	0x81996l	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	0.968804682255	0.062103622228607410
3	0x393c4	268.0	0x067504	05.04.2023	0x81996l	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	0.968804682255	0.062103622228607410
4	0x8537fc	269.0	0x067504	05.04.2023	0x81996l	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	0.968804682255	0.062103622228607410

Source: Own illustration

As you can see from Figure 3 above the file can be read directly with Banana+ as a CSV file. This file is only a data-gathering file and could be used as support when presenting the financial results. For example, the following columns are important:

ReceiverAddress: the address could be used for analysis that includes, which are the most important clients for the company, and which are the ones that do not contribute much to the

company's well-being. With this information, one can adapt the marketing of the company and propose a better client-oriented business for the future.

Sender_PIN: the same argument can be made for this column as above. In fact, by combining the two, as explained above, one can adapt to the needs of customers and the company in the best possible way.

GasObtainedForDelivery (Gwei): this column presents how much Gas (fee) is obtained for each transaction. It is a piece of important information that allows the company to keep track of the amount of money that it earned by doing a particular transaction.

Sender_Profit (CHF): this last column is important to see which sender is earning the most. This information could be used to offer a better service for the ones that are the most important clients and adapt to the business needs of the ones that are not the best senders.

6.2.3. Output 3

Figure 8: Banana+ CSV Output 3

	hash	orderID	receiverA-ddress	DateTime	sender_PIN	sender-ID	commit-mentID	Relay_Sta-tusFlag	Transacti-on_Fee (Gwei)	gasCostForDeliv-ery (GWei)	commitment_fee (GWei)	OrderCost (GWei)	ExchangeRate	OrderCost (CHF)
1	0x96720e	266.0	0x067504	27.03.2023	0x81996D1	1.0	1	True		14400000.0	100000000.0	114400000.0	0.983356572882	0.11633623362552911
2	0x187569	268.0	0x067504	05.04.2023	0x81996D1	1.0	1	True		14400000.0	100000000.0	114400000.0	0.968804682255	0.11808365720706608
3	0xb19124	269.0	0x067504	05.04.2023	0x81996D1	1.0	1	True		14400000.0	100000000.0	114400000.0	0.968804682255	0.11808365720706608
4	0x588c66	270.0	0x067504	05.04.2023	0x81996D1	1.0	1	True		14400000.0	100000000.0	114400000.0	0.968804682255	0.11808365720706608

Source: Own illustration

receiverAddress: the receiver address is important to adapt the business of the company and understand which are the most important receivers.

sender_PIN: the same argument is done for the receiver, also the sender helps to better adapt the needs of the clients to the needs of the company.

gasCostForDelivery (Gwei): this column presents how much Gas (fee) is used for each transaction. It is a piece of important information that allows the company to keep track of the amount of money that is lost by doing a particular transaction.

OrderCost_(CHF): this last column is important because it allows us to see how much the transaction had cost in francs. It gives a general overview of the costs. For instance, this data can be used as support when presenting the company's cost analysis.

7. Conclusion

Traditional bookkeeping systems face several challenges such as vulnerability to errors, fraud, manipulation, audit costs, and time loss. Blockchain technology presents a viable solution to address the vulnerabilities of traditional double-entry bookkeeping, offering a decentralized system that significantly improves transparency, accountability, and efficiency.

Our comprehensive investigation, comprising an extensive literature review and analysis of relevant studies, has provided valuable insights into the current state of knowledge in this field. By exploring the standardization of accounting practices and the adoption of blockchain technology in the accounting context, it was possible to identify the benefits that triple-entry bookkeeping can bring but also various barriers to its adoption, such as regulatory challenges and potential security risks.

Despite this, there is a growing demand and interest in blockchain accounting, which is an underdeveloped area. This study is therefore intent on focusing on this opportunity by proposing an innovative solution that automates the transmission of transaction data in order to create financial statements that comply with accounting standards. To guide our research, we formulated two specific hypotheses in Chapter 1.

The first hypothesis (H1) aimed to achieve complete automation of the accounting process using the code presented in our solution. The Python code produces 3 separate files that can be read by any accounting program in CSV format. All the files contain data relevant to the company's business. The first output file contains data regarding transactions performed on the blockchain, with the respective amounts, which is useful for defining credits and debits in accounting. The second file is a data-gathering file that could be used as support when presenting the financial results. The third file encloses information regarding transaction costs, relevant to the company's cost analysis, but also regarding customers to establish the most relevant ones. Regrettably, the objective of achieving fully automated accounting has not been realized, due to the need for additional implementations. Among these potential solutions, one could involve the incorporation of sender and receiver differentiation. Despite our rejection of H1, we recognize that we have established a solid foundation upon which to build a more advanced automated accounting system.

The second hypothesis (H2), aimed at providing clear and precise guidance to companies interested in improving automation in their accounting systems, has been confirmed. Through our research, we targeted to address the growing interest in blockchain-based accounting, recognizing that it is still an underdeveloped area with limited guidance available to companies. Due to the blockchain characteristic each smart contract is coded differently, this is the reason why our code is not directly applicable to other firms. Nevertheless, we thoroughly explained our solution, going through it step by step and recommending the necessary code adjustments to align with the smart contract utilized by the crypto company. This detailed approach provides a practical solution for implementing a more automated accounting system. We have successfully achieved the goal of H2 by providing an extensive and practical guidance framework for companies.

In addition, we presented a Python code that automates the transmission of transaction data, ensuring the creation of financial statements that comply with accounting standards. This code was created while also considering taxing issues. In fact, we addressed both theoretically and practically tax requirements, including VAT and taxation of cryptocurrency income in Switzerland. Given the challenges that VAT poses for Swiss blockchain companies, we proposed an automated solution in our Python code, which calculates VAT using the *Saldosteuersatz* method at a rate of 5.9 percent. This approach simplifies VAT management and helps companies comply with tax regulations.

This research contributes to managing blockchain transactions, simplifying the accounting processes, and ensuring compliance with accounting standards. The practical approach to linking blockchain transactions with an accounting system offers new possibilities for Swiss companies in the blockchain industry that wish to integrate blockchain technology into their accounting. In conclusion, we created a bridge between the blockchain world and the off-chain world of accounting systems.

8. Limitations and future research

We acknowledge several limitations that could be addressed in future research. Specifically, two limitations stand out: the specificity of our code and the potential to associate different wallet addresses with the names of senders and receivers.

The first limitation relates to the inherent nature of smart contracts. Each smart contract is coded differently and relies on distinct functions. Our code is designed to work specifically with the functions present in the SCL smart contract, which served as our reference. To extend the usability of our solution to other companies employing different smart contracts, the code would need to be adapted accordingly to align with the functions utilized in those specific smart contracts.

Furthermore, for accounting purposes, it would be beneficial to have a comprehensive overview of the accounts and their corresponding total inflow and outflow of funds. However, our current solution does not have the capability to assign names to individual wallet addresses. It is important to note that this limitation stems from the anonymity inherent in blockchain technology. Nevertheless, it would be feasible to develop a tool that could establish a connection between each wallet address and the username used on the SCL platform. In addition, as mentioned before, other implementations that automize other accounting calculations such as a tool capable of calculating the Corporate Income Tax would be highly beneficial. However, we have chosen to leave this potential implementation as an avenue for future research and focus our efforts on the primary advantage of automated accounting.

Despite these limitations, we firmly believe that we have laid the groundwork for an automated accounting tool that mitigates the risk of errors and enhances the speed of accounting processes.

9. References

ACCA. (2022). *Blockchain; is it still the great accountancy disruptor?* ACCA Global. https://www.accaglobal.com/gb/en/student/sa/features/blockchain.html

AccountingTools. (2022). *Double entry system definition*. AccountingTools. https://www.accountingtools.com/articles/the-double-entry-system

Adams, K. (2020). *Data Science Central: How Blockchain is changing the Accounting Profession*. Data Science Central. https://www.datasciencecentral.com/how-blockchain-is-changing-the-accounting-profession

Alawida, M., Omolara, A. E., Abiodun, O. I., & Al-Rajab, M. (2022). A deeper look into cybersecurity issues in the wake of Covid-19: A survey. *Journal of King Saud University - Computer and Information Sciences*, *34*(10), 8176–8206. https://doi.org/10.1016/j.jksuci.2022.08.003

alexandersloan. (2023, March 1). *Digital Accounting systems are more important than ever*. https://www.alexandersloan.co.uk/news/firm-news/digital-accounting-systems-are-more-important-than-ever/#:~:text=These%20systems%20offer%20a%20wide,as%20well%20as%20better%20collaboration

Atlam, H. F., Alenezi, A., Alassafi, M. O., & Wills, G. B. (2018). Blockchain with Internet of Things: Benefits, Challenges, and Future Directions. *International Journal of Intelligent Systems and Applications*, *10*(6), 40–48. https://doi.org/10.5815/ijisa.2018.06.05

Banana.ch SA. (n.d.-a). *About us | Banana Accounting*. Retrieved May 25, 2023, from https://www.banana.ch/en/company

Banana.ch SA. (n.d.-b). *Description VAT Codes*. Retrieved June 3, 2023, from https://www.banana.ch/en/node/10873

Banana.ch SA. (n.d.-c). *Import Extensions | Banana Accounting*. Retrieved June 1, 2023, from https://www.banana.ch/apps/en/node/9561

Banana.ch SA. (n.d.-d). *The Blockchain in accounting | Banana Accounting*. Retrieved June 1, 2023, from https://www.banana.ch/en/blockchain-accounting

Blockchain Presence AG. (n.d.). *Blockchain Presence*. Retrieved May 26, 2023, from https://blockchainpresence.net/

Botzem, S., & Quack, S. (2009). (No) Limits to Anglo-American accounting? Reconstructing the history of the International Accounting Standards Committee: A review article. *Accounting, Organizations and Society*, *34*(8), 988–998. https://doi.org/10.1016/j.aos.2009.07.001

Britannica. (n.d.). *bookkeeping*. Retrieved April 13, 2023, from https://www.britannica.com/money/bookkeeping

Bundesrecht. (2023). *Bundesgesetz über Bucheffekten*. https://www.fedlex.admin.ch/eli/cc/2009/450/de

Coinbase. (n.d.). *What is a transaction hash/hash ID?* Retrieved June 1, 2023, from https://help.coinbase.com/en/coinbase/getting-started/crypto-education/what-is-a-transaction-hash-hash-id

CoinGecko. (2023). *CoinGecko*. https://www.coingecko.com/de

Coyne, J. G., & McMickle, P. L. (2017). Can Blockchains Serve an Accounting Purpose? *Journal of Emerging Technologies in Accounting*, *14*(2), 101–111. https://doi.org/10.2308/jeta-51910

CPA.com. (2019). *blockchain symposium: experts' insights indicate growing use cases and value for the technology*. https://www.cpa.com/sites/cpa/files/2019-12/2019-blockchain-symposium-report.pdf

Dai, J., & Vasarhelyi, M. A. (2017). Toward Blockchain-Based Accounting and Assurance. *Journal of Information Systems*, *31*(3), 5–21. https://doi.org/10.2308/isys-51804

Daneshpajooh, M. (2022, March 31). *Blockchain Bridges (Validators vs Light Client)*. Blog.Teleportdao.Xyz. https://blog.teleportdao.xyz/bridges-validator-vs-relay-b8836b4b4cf0

Deloitte. (2017). *Blockchain and its potential impact on the audit profession*. Deloitte. https://www2.deloitte.com/bd/en/pages/audit/articles/gx-impact-of-blockchain-in-accounting.html

Deloitte. (2022). *Cryptocurrencies and Your Swiss Tax Return: Reporting requirements and tax implications*. Blogs.Deloitte.Ch. https://blogs.deloitte.ch/tax/2022/04/cryptocurrencies-and-your-swiss-tax-return-reporting-requirements-and-tax-implications.html

Demirkan, S., Demirkan, I., & McKee, A. (2020). Blockchain technology in the future of business cyber security and accounting. *Journal of Management Analytics*, *7*(2), 189–208. https://doi.org/10.1080/23270012.2020.1731721

Eidgenössische Steuerverwaltung. (2018). *estv.admin.ch*. https://www.estv.admin.ch/estv/de/home/mehrwertsteuer/mwst-steuersaetze/mwst-saldosteuersatz-pauschalsteuersatz/saldosteuersaetze-20180101.html not sure please double check this citation.

Ellis, L. (2022, December 28). *Why so Many Accountants Are Quitting?* Wall Street Journal. https://www.wsj.com/articles/why-so-many-accountants-are-quitting-11672236016

Fernando, J. (2023, March 23). *Equity for Shareholders: How It Works and How to Calculate It*. Investopedia.Com. https://www.investopedia.com/terms/e/equity.asp#:~:text=Equity%2C%20typically%20referred%20to%20as,in%20the%20case%20of%20liquidation

Fraser, I. A. M. (1993). Triple-entry Bookkeeping: A Critique. *Accounting and Business Research*, *23*(90), 151–158. https://doi.org/10.1080/00014788.1993.9729872

FreshBooks. (2023, April 5). *What is Blockchain Accounting? A Primer for Small Businesses*. FreshBooks.Com. https://www.freshbooks.com/hub/accounting/blockchain-accounting

Gauthier, M. P., & Brender, N. (2021). How do the current auditing standards fit the emergent use of blockchain? *Managerial Auditing Journal*, *36*(3), 365–385. https://doi.org/10.1108/MAJ-12-2019-2513

Ghasemi, M., Shafeiepour, V., Aslani, M., & Barvayeh, E. (2011). The impact of Information Technology (IT) on modern accounting systems. *Procedia - Social and Behavioral Sciences*, *28*, 112–116. https://doi.org/10.1016/j.sbspro.2011.11.023

Giambonini, A. (2021). *GetFileForBanana_Polygon.py*. Blockchain Presence AG.

Hayes, A. (2021, March 20). *Double Entry: What It Means in Accounting and How It's Used*. Investopedia.Com. https://www.investopedia.com/terms/d/double-entry.asp

Imhoff, G. (2003). *Accounting Quality, Auditing and Corporate Governance*.

Koinly. (2023). *Crypto Tax Switzerland: 2023 Guide*. Koinly.Io. https://koinly.io/guides/switzerland-crypto-tax-guide/

Kunselman, K. (2021). *The Future of Blockchain In Accountancy*. Forbes. https://www.forbes.com/sites/forbesfinancecouncil/2021/01/29/the-future-of-blockchain-in-accountancy/?sh=a5b54041fd4a

LawyersSwitzerland.com. (2023). *VAT Registration Switzerland - 2023 Procedure*. https://www.lawyersswitzerland.com/vat-registration-in-

Originstamp. (2023). *Advantages and Disadvantages of Blockchain in Accounting*. Originstamp.Com. https://originstamp.com/blog/advantages-and-disadvantages-of-blockchain-in-accounting/

Pacioli, L. (1494). *Su[m]ma de arithmetica geometria proportioni [et] proportionalita*. Paganinus de Paganinis. https://doi.org/10.5479/sil.440357.39088007406663

Pawczuk, L., Massey, R., & Holdowsky, J. (2019, May 6). *Deloitte's 2019 global blockchain survey*. Deloitte. https://www2.deloitte.com/us/en/insights/topics/understanding-blockchain-potential/global-blockchain-survey-2019.html

Polygonscan. (n.d.). *Unit Converter | Polygonscan*. Retrieved June 1, 2023, from https://polygonscan.com/unitconverter

PWC. (2015). *The new Swiss financial reporting law*. www.pwc.ch

PWC. (2023). *Corporate - Taxes on corporate income*. https://taxsummaries.pwc.com/switzerland/corporate/taxes-on-corporate-

income#:~:text=Switzerland%20levies%20a%20direct%20federal,before%20tax%20of%20approximately%207.83%25

Rana, K. (2020, March 27). *Triple entry accounting system: A revolution with blockchain*. Medium. https://medium.com/dataseries/triple-entry-accounting-system-a-revolution-with-blockchain-768f4d8cabd8

Rue. (n.d.). *Switzerland Crypto Tax*. Rue.Ee. Retrieved May 17, 2023, from https://rue.ee/blog/switzerland-crypto-tax/#:~:text=Capital%20Gains%20Tax%20is%20imposed,are%20exempt%20from%20the%20tax

Sangster, A., & Scataglinibelghitar, G. (2010). Luca Pacioli: The Father of Accounting Education. *Accounting Education*, *19*(4), 423–438. https://doi.org/10.1080/09639284.2010.501955

Schmitz, J., & Leoni, G. (2019). Accounting and Auditing at the Time of Blockchain Technology: A Research Agenda. *Australian Accounting Review*, *29*(2), 331–342. https://doi.org/10.1111/auar.12286

Swiss Confederation. (2017, October 31). *Verordnung der ESTV über die Höhe der Saldosteuersätze nach Branchen und Tätigkeiten*. Admin.Ch. https://www.fedlex.admin.ch/eli/oc/2017/604/de

Swiss Confederation. (2023a). *Saldosteuersätze ab 1. Januar 2024*. Admin.Ch. https://www.estv.admin.ch/estv/de/home/mehrwertsteuer/mwst-steuersaetze/mwst-saldosteuersatz-pauschalsteuersatz/saldosteuersaetze-ab-2024.html

Swiss Confederation. (2023b). *Value-added tax (VAT)*. Ch.Ch. https://www.ch.ch/en/taxes-and-finances/types-of-taxation/value-added-tax--vat-/

swisstaxexpert.com. (n.d.). *How do Swiss VAT authorities treat cryptocurrencies?* Retrieved May 20, 2023, from https://www.swisstaxexpert.com/how-do-swiss-vat-authorities-treat-cryptocurrencies/

Tan, B. S., & Low, K. Y. (2019). Blockchain as the Database Engine in the Accounting System. *Australian Accounting Review*, *29*(2), 312–318. https://doi.org/10.1111/auar.12278

Testbook Edu Solutions Pvt. Ltd. (2023, March 18). *Difference between Manual Accounting and Computerized Accounting*. https://testbook.com/key-differences/difference-between-manual-accounting-and-computerized-accounting#:~:text=Manual%20accounting%20involves%20the%20use,in%20error%20checking%20and%20validation

Tuovila, A. (2022, June 17). *Reconciliation in Account Definition, Purpose, and Types*. Investopedia. https://www.investopedia.com/terms/r/reconciliation.asp

Visram, O. (2021, October 13). *Importance of Bookkeeping for Growing Small Businesses*. Enkel. https://www.enkel.ca/blog/bookkeeping/importance-of-bookkeeping-for-growing-small-businesses/

Vontobel. (2022). *Crypto tax in Switzerland: How do I correctly (not) pay tax on my cryptocurrencies?* Vontobel.Com. https://www.vontobel.com/en-ch/impact/crypto-tax-switzerland-what-matters-for-cryptocurrencies-46863/

Yang, P. (2021, September 29). *Curious Beginner's Guide to Crypto*. Creator Economy. https://creatoreconomy.so/p/curious-beginner-guide-to-crypto?utm_source=%2Fsearch%2Fproof%2520of%2520work&utm_medium=reader2

Yu, T., Lin, Z., & Tang, Q. (2018). Blockchain: The Introduction and Its Application in Financial Accounting. *Journal of Corporate Accounting & Finance*, *29*(4), 37–47. https://doi.org/10.1002/jcaf.22365

Zenko, J. (2022, March 2). *What's Account Reconciliation and Why Does It Matter?* Versapay. https://www.versapay.com/resources/account-reconciliation

Appendix A

The following lines are a function-wise translation of the Python interface:

import necessary libraries

```
 from web3 import Web3
 import pandas as pd
 import json
 import numpy as np
 from hexbytes import HexBytes
 import requests
 import asyncio
 import os
 import sqlite3
from web3._utils.events import get_event_data
import time
```

Creating to variables "url_polygon", which is an alchemy websocket to retrieve the data from the blockchain, and "SCL_ADDRESS", which is the address of the relevant smart coded by SCL.

```
url_polygon = 'wss://polygon-mumbai.g.alchemy.com/v2/DJtnV9hEqeQ-
2VuKFCAT2rlW7XbnMohV' (1)
SCL_ADDRESS ='0xD200F64cEcc8bBc1292c0187F5ee6cD7bDf1eeac' (2)
```

The code defines a function named get_logs that takes four arguments: SCL, from_block, to_block, and topics. The purpose of the function is to retrieve Ethereum logs from a specified block range.

The function uses a while loop to iterate through the block range in increments of max_block_range, which is set to 10000. Within the loop, the get_logs method of the SCL.web3.eth object is called with the specified block range and topics. The logs are appended to the logs_list array.

After the loop completes, the function retrieves the logs for the remaining block range and appends them to the logs_list. Finally, the function returns a flattened list of all the logs in logs_list.

Overall, the function is designed to fetch Ethereum logs from a specified block range by making multiple requests and aggregating the results into a single list.

```
def get_logs(SCL, from_block, to_block, topics):
    max_block_range = 10000
    logs_list = []
    while to_block>from_block:
```

```
        step_block = from_block+max_block_range
        print(f"Checking blocks from {from_block} to {step_block}")
        logs_list.append(SCL.w3.eth.get_logs({'fromBlock': from_block,
                                              'toBlock': step_block,
                                              'topics': [topics]}))
        from_block = step_block

    from_block = from_block - max_block_range
    logs_list.append(SCL.w3.eth.get_logs({'fromBlock': from_block,
                                          'toBlock': to_block,
                                          'topics': [topics]}))
    return [log for logs in logs_list for log in logs]
```

The code defines a function named getnewOrderEvents that takes four arguments: SCL, w3, minBlock, and maxBlock. The function is designed to retrieve new order events from a specified block range.

First, the function creates a filter for the newOrder event using the SCL.events.newOrder object. Then, it calculates the event signature hash for the newOrder event using the w3.keccak method.

Next, the function calls the get_logs function with the SCL, minBlock, maxBlock, and event_signature_hash parameters to retrieve the logs for the newOrder event.

The function then iterates through each log in the list of events, decodes the event data using the get_event_data method and the event ABI from the new_order_event object. The decoded events are stored in the order_events array.

Finally, the function returns a filtered list of events where the address field matches the SCL_ADDRESS constant. This filtered list of events represents the new order events that occurred within the specified block range on the SCL Ethereum network.

```
def getnewOrderEvents(SCL, w3, minBlock, maxBlock):
    # create filter
    new_order_event = SCL.events.newOrder
    event_signature_hash =
w3.keccak(text="newOrder(address,uint32,int64,string,uint32,uint40,uint
64,address)").hex()
    list_orders_events = get_logs(SCL, minBlock, maxBlock,
event_signature_hash)
    order_events = [
                            get_event_data(
                                new_order_event.w3.codec,
                                new_order_event._get_event_abi(),
                                event
```

```
                                )
                            for event in list_orders_events
                            ]
    return [event for event in order_events if
event['address']==SCL_ADDRESS]
```

The code defines a function named getDataDeliveredEvents that takes four arguments: SCL, w3, minBlock, and maxBlock. The function is designed to retrieve data delivered events from a specified block range.

First, the function creates a filter for the dataDelivered event using the SCL.events.dataDelivered object. Then, it calculates the event signature hash for the dataDelivered event using the w3.keccak method.

Next, the function calls the get_logs function with the SCL, minBlock, maxBlock, and event_signature_hash parameters to retrieve the logs for the dataDelivered event.

The function then iterates through each log in the list of events, decodes the event data using the get_event_data method and the event ABI from the data_delivered_event object. The decoded events are stored in the data_delivered_events array.

Finally, the function returns a filtered list of events where the address field matches the SCL_ADDRESS constant. This filtered list of events represents the data delivered events that occurred within the specified block range on the SCL Ethereum network.

```
def getDataDeliveredEvents(SCL,w3, minBlock,maxBlock):
    # create filter
    data_delivered_event = SCL.events.dataDelivered
    event_signature_hash =
w3.keccak(text="dataDelivered(uint32,bool,bool)").hex()
    list_data_delivered_events = get_logs(SCL, minBlock, maxBlock,
event_signature_hash)
    data_delivered_events = [
                            get_event_data(
                                data_delivered_event.w3.codec,
                                data_delivered_event._get_event_abi(),
                                event
                                )
                            for event in list_data_delivered_events
                            ]
    return [event for event in data_delivered_events if
event['address']==SCL_ADDRESS]
```

The code defines two functions: get_SCL_informations and getexchangeRate.

The first one takes in data and tx_hash as inputs. The function does the following:

Initializes a Web3 instance with a websocket provider. Loads the ABI (Application Binary Interface) of the SCL contract from a local JSON file and creates a contract instance with the given address and ABI. Gets the minimum and maximum block numbers from the data input. Calls two other functions getnewOrderEvents and getDataDeliveredEvents to get all the relevant events for the SCL contract within the block range. Adds order IDs, commitment IDs, Relay status flags, and SCL commitment information to the data input. Calls the coingecko API to get historical exchange rates for MATIC and CHF. Returns the modified data input.

The second one getexchangeRate takes in data as an input. The function does the following:

Extracts unique dates from the DateTime column of data. Calls the coingecko API to get the MATIC/CHF exchange rate for each date and adds it to the ExchangeRate column of the corresponding row in data. Rounds the values in the ExchangeRate column to 12 decimal places. Returns the modified data input. The get_SCL_informations function mainly extracts information from the events emitted by the SCL smart contract and adds it to the input data. The getexchangeRate function fetches the MATIC/CHF exchange rate for each date in data['DateTime'] column and adds it to the ExchangeRate column of the corresponding row in data.

```
def get_SCL_informations(data, tx_hash):
    #define SCL contract
    url = url_polygon
    w3 = Web3(Web3.WebsocketProvider(url))
    with open('contract_abi.json') as f:
        abi = json.load(f)
    SCL = w3.eth.contract(address=SCL_ADDRESS, abi=abi)
    #get blockrange
    minBlock = int(np.min(data['blockNumber']))
    maxBlock = int(np.max(data['blockNumber']))
    #get all relevant SCL Orders in blockrange
    print("get new Order Events")
    Orders = getnewOrderEvents(SCL, w3, minBlock, maxBlock)
    print("get Data Delivered Events")
    Relay = getDataDeliveredEvents(SCL, w3,  minBlock, maxBlock)
    events = Orders + Relay
    print("adding orderIDs")
    for event in events:
        orderID = event['args']['orderID']
        transactionHash=event['transactionHash']
        data.loc[data['hash']==transactionHash,'orderID']=orderID
```

```
    print("adding commitmentIDs")
    for Order in Orders:
        orderID = Order['args']['orderID']
        commitmentID=Order['args']['commitmentID']
        receiverAddress = Order['args']['receiverAddress']
        sender_PIN = Order['args']['_PIN']
        gasForDelivery = Order['args']['_gasForDelivery']
        gasPrice = Order['args']['_gasPrice']
        gasCost = gasForDelivery*gasPrice
        data.loc[data['orderID']==orderID,'commitmentID']=commitmentID

data.loc[data['orderID']==orderID,'receiverAddress']=receiverAddress
        data.loc[data['orderID']==orderID,'sender_PIN']=sender_PIN
        data.loc[data['orderID']==orderID,'gasCostForDelivery
(Wei)']=gasCost
    print("adding Relay_StatusFlag")
    for event in Relay:
        transactionHash=event['transactionHash']
        statusFlag = event['args']['_statusFlag']

data.loc[data['hash']==transactionHash,'Relay_StatusFlag']=statusFlag
    commitment_list = list(set(data['commitmentID'].tolist()))
    commitment_list = [commitment for commitment in commitment_list if
commitment>=0]
    for commitment in commitment_list:
        try:
            commitment_infos =
SCL.functions.commitments(commitment).call()
            senderID = commitment_infos[0]
            commitment_fee = commitment_infos[2]
            data.loc[data['commitmentID']==commitment,'senderID'] =
senderID
            data.loc[data['commitmentID']==commitment,'commitment_fee']
= commitment_fee
        except:
            print(f'Commitment {commitment} not found')

    for i in range(len(data)):
        data.at[i,'hash'] = data.at[i,"hash"].hex()
    data = data.loc[~ data['hash'].isin(tx_hash), ]
    data = data.drop(data[data.commitmentID <0].index)
    return data

async def getexchangeRate(data):
```

```
    ExchangeRateList = data['DateTime'].tolist()
    ExchangeRateList = sorted(set(ExchangeRateList), key =
ExchangeRateList.index)
    with requests.Session() as s:
        print("adding exchange rates")
        url = 'https://api.coingecko.com/api/v3/coins/matic-
network/history'
        for i in ExchangeRateList:
            print(i)
            payload = {'date': i}
            api_response = s.get(url, params=payload)
            if api_response.status_code == 429:
                time.sleep(120)
                api_response = s.get(url, params=payload)
                if api_response.status_code == 429:
                    raise NotImplementedError("coingecko rate limit
needs additional work")
            data_response = api_response.json()
            data.loc[data['DateTime'] == i, 'ExchangeRate'] =
1/data_response['market_data']['current_price']['chf'] (3)
            await asyncio.sleep(0.3)
    data["ExchangeRate"] = data["ExchangeRate"].round(decimals=12)
    return data
```

The function TransformInternalTransaction takes in a Pandas DataFrame internalTx containing information on internal transactions on the blockchain. The function performs the following operations on the DataFrame:

Converts the 'isError' and 'value' columns to numeric data types using the pd.to_numeric() function. Multiplies the 'value' column by -1 for any row where the 'from' column matches the SCL address (i.e., negative values represent outgoing transactions from the SCL contract). Calculates the sum of 'value' for each transaction hash and adds the result as a new column 'value_internal'. Drops duplicate rows with the same transaction hash using drop_duplicates(). Filters the DataFrame to include only rows with 'isError' equal to 0. Selects the columns 'hash', 'blockNumber', 'timeStamp', and 'value_internal' to return the modified DataFrame. Overall, the function appears to be transforming the DataFrame to include only relevant information on internal transactions with non-zero value and without errors.

```
def TransformInternalTransaction(internalTx):
    internalTx[['isError', 'value']] = internalTx[['isError',
'value']].apply(pd.to_numeric)
    internalTx.loc[(internalTx['from'] ==SCL_ADDRESS.lower()), 'value']
= internalTx['value'] * -1
```

```
    internalTx['value_internal'] =
internalTx.groupby('hash')['value'].transform(sum)
    internalTx = internalTx.drop_duplicates(subset=['hash'])
    internalTx = internalTx[internalTx['isError'] == 0]
    columns = ['hash', 'blockNumber', 'timeStamp', 'value_internal']
    internalTx = internalTx[columns]
    return internalTx
```

The code defines a function TransformParentTransaction that takes a Pandas DataFrame parentTx as input. The function converts certain columns of the DataFrame to numeric type using the pd.to_numeric function. It then calculates the transaction fee in Gwei and adds a new column to the DataFrame for this. The function filters out rows where the transaction produced an error or where the recipient address is not the SCL_ADDRESS, as specified by a global variable. It then renames the value column to value_parent, selects only certain columns, and returns the resulting DataFrame. In summary, the function transforms and filters a DataFrame of parent transactions, calculates the transaction fee, and returns a DataFrame containing only the relevant columns.

```
def TransformParentTransaction(parentTx):
    parentTx[['isError','value','gasPrice',
'gasUsed']]=parentTx[['isError','value','gasPrice',
'gasUsed']].apply(pd.to_numeric)
    parentTx['Transaction_Fee (Gwei)']=parentTx['gasPrice']*10**(-
9)*parentTx['gasUsed']
    parentTx = parentTx.loc[(parentTx['isError'] == 0) &
(parentTx['to']==SCL_ADDRESS.lower()), ]
    parentTx = parentTx.rename(columns={'value': 'value_parent'})
    columns = ['hash', 'blockNumber', 'timeStamp', 'value_parent',
'Transaction_Fee (Gwei)']
    parentTx = parentTx[columns]
    return parentTx
```

The code defines a function MergeTransactions that takes two data frames parentTx and internalTx as inputs. The function merges the data frames using an outer join based on the hash column.

Then the function fills in missing values for the blockNumber_x, timeStamp_x, value_parent, and value_internal columns with 0. The column names are then renamed, and the blockNumber column is converted to a numeric data type.

Next, the hash column is converted from a string to a HexBytes object. The DateTime column is converted from Unix timestamp to a human-readable date format. Finally, the function selects the desired columns (hash, blockNumber, DateTime, value_internal, value_parent, and Transaction_Fee (Gwei)), and returns the resulting data frame.

```
def MergeTransactions(parentTx, internalTx):
    data = internalTx.merge(parentTx, how="outer", on="hash")
    data.loc[pd.isnull(data['blockNumber_x']), 'blockNumber_x'] =
data['blockNumber_y']
    data.loc[pd.isnull(data['timeStamp_x']), 'timeStamp_x'] =
data['timeStamp_y']
    data.loc[pd.isnull(data['value_parent']), 'value_parent'] = 0
    data.loc[pd.isnull(data['value_internal']), 'value_internal'] = 0
    data = data.rename(columns={'blockNumber_x': 'blockNumber',
'timeStamp_x': 'DateTime'})
    data['blockNumber'] = pd.to_numeric(data['blockNumber'])
    for i in range(len(data)):
        data.at[i,'hash'] = HexBytes(data.at[i,'hash'])
    data['DateTime'] = pd.to_datetime(data['DateTime'],
unit='s').dt.strftime('%d-%m-%Y')
    columns = ['hash', 'blockNumber', 'DateTime', 'value_internal',
'value_parent', 'Transaction_Fee (Gwei)']
    data = data[columns]
    return data
```

The function ask_for_VAT() is asking the user whether they want to take VAT (Value Added Tax) into account. It uses the input() function to prompt the user to enter 'y' for yes or 'n' for no. The function then enters a loop that continues to prompt the user until they enter a valid response. Once the user has entered a valid response, the function returns True if the user entered 'y' and False if the user entered 'n'.

```
def ask_for_VAT():
    VAT = input('Do you want to take VAT into account? (y/n): \n')
    while VAT not in ['y','n']:
        VAT = input("Please enter 'y' (yes) or 'n' (no): ")
    if VAT=='y':
        return True
    else:
        return False
```

The code defines a function ask_for_separator() that prompts the user to input a delimiter character for a CSV file. The function then checks if the input is either ',' or ';' and returns the input as the separator character. If the input is not a valid separator character, the function will keep prompting the user until a valid input is received.

```
def ask_for_separator():
    sep = input('Which delimiter do you want to use for the csv file?
(,/;): \n')
    while sep not in [',',';']:
```

```
        sep = input("Please enter ',' or ';': ")
    return sep
```

The code defines a function called Create_SCL_Revenue_file that takes two arguments, data and VAT_bool. data is a Pandas DataFrame containing transaction data. VAT_bool is a boolean that indicates whether or not to include VAT (Value Added Tax) in the revenue calculations.

The function creates a copy of the data DataFrame and calculates the revenue for each transaction by adding the value_parent and value_internal columns and subtracting the commitment_fee column. If the Relay_StatusFlag column is True, the commitment_fee is divided by two. Transactions with zero revenue are dropped from the DataFrame.

If VAT_bool is True, the function calculates VAT and adds a new row to the DataFrame for each transaction that includes VAT. The AccountCredit column for these rows is set to 2330. The revenue is multiplied by 0.941 to calculate the amount without VAT and by 0.059 to calculate the VAT amount. The Amount column is calculated as the revenue divided by the ExchangeRate column.

The function renames and reorders the columns in the DataFrame, sorts the DataFrame by blockNumber, and returns the DataFrame.

```
def Create_SCL_Revenue_file(data, VAT_bool):
    revenue_data = data.copy()
    revenue_data['Value (Wei)'] =
revenue_data['value_parent']+revenue_data['value_internal']-
revenue_data['commitment_fee']
    revenue_data.loc[revenue_data['Relay_StatusFlag']==True,'Value
(Wei)'] = revenue_data['commitment_fee']/2
    revenue_data.drop(revenue_data[revenue_data['Value
(Wei)']==0].index, inplace = True)
    revenue_data['Value (Gwei)']=revenue_data['Value (Wei)']*10**(-9)
    revenue_data['Doc']=''
    revenue_data['ExchangeCurrency']='gwei'
    revenue_data["AccountDebit"] = 1027
    revenue_data["AccountCredit"] = 3001
    if VAT_bool:
        VAT = revenue_data.copy()
        VAT["AccountCredit"] = 2330
        revenue_data['Value (Gwei)'] = revenue_data['Value
(Gwei)']*0.941
        revenue_data['Value (Gwei)'] = revenue_data['Value
(Gwei)'].round(decimals=18)
        VAT['Value (Gwei)'] = VAT['Value (Gwei)']*0.059
        VAT['Value (Gwei)'] = VAT['Value (Gwei)'].round(decimals=18)
        revenue_data = pd.concat([revenue_data,VAT])
```

```
    revenue_data["Amount"] = revenue_data["Value
(Gwei)"]/(revenue_data["ExchangeRate"]*10**9)
    revenue_data["Amount"] = revenue_data["Amount"].round(decimals=5)
    revenue_data['DateTime'] =
pd.to_datetime(revenue_data['DateTime'],format="%d-%m-%Y")
    revenue_data["VatCode"]='F3'

revenue_data.rename(columns={'DateTime':'Date','hash':'Description',
'Value (Gwei)': 'AmountCurrency'}, inplace = True)
    revenue_data.sort_values(by='blockNumber', inplace=True)
    revenue_columns = ['Date', 'Doc', 'Description','AccountDebit',
'AccountCredit','AmountCurrency','ExchangeCurrency','VatCode','Exchange
Rate','Amount']
    revenue_data =
revenue_data.loc[revenue_data['Relay_StatusFlag']!=False, ]
    revenue_data = revenue_data[revenue_columns]
    return revenue_data
```

This code defines a function called Database_MIS that takes a pandas dataframe (data) as input. The function then performs several operations on the data.

First, the function converts the commitment_fee column from wei to Gwei by multiplying it by 10^(-9) and adds a new column called commitment_fee (GWei) to the dataframe.

Next, the function creates a new column called transactionType that is either "Relay" or "Order" depending on the value in the Relay_StatusFlag column. The function then creates two new dataframes (orders and relay) that contain all rows where transactionType is "Order" or "Relay", respectively.

The function then creates a list of unique orderIDs from the orders dataframe and loops over each order, updating the Relay_StatusFlag in the orders dataframe based on the corresponding value in the relay dataframe.

The function then converts the gasCostForDelivery (Wei) column to Gwei by multiplying it by 10^(-9) and fills any missing values in the Transaction_Fee (Gwei) column with 0. The function then calculates a new column called OrderCost (GWei) as the sum of gasCostForDelivery (GWei), commitment_fee (GWei), and Transaction_Fee (Gwei) for each order. If Relay_StatusFlag is False, the commitment_fee (GWei) component of the OrderCost is removed. The function also calculates a new column called OrderCost (CHF) by dividing OrderCost (GWei) by the ExchangeRate column.

Finally, the function creates a list of unique orderIDs from the relay dataframe and loops over each order, updating the GasObtainedForDelivery (GWei) column in the relay dataframe based on the corresponding value in the orders dataframe. The function then calculates a new column called Unused_GasForDelivery (GWei) as the difference between GasObtainedForDelivery (GWei) and

Transaction_Fee (Gwei) for each order. The function also calculates a new column called Sender_Profit (Gwei) as the sum of Unused_GasForDelivery (GWei) and half of commitment_fee (GWei) for each order. If Relay_StatusFlag is not True, the commitment_fee (GWei) component of the Sender_Profit is removed. The function also calculates a new column called Sender_Profit (CHF) by dividing Sender_Profit (Gwei) by the ExchangeRate column.

The function returns two dataframes: orders and relay.

```
def Database_MIS(data):
    data['commitment_fee (GWei)'] = data['commitment_fee']*10**(-9)

data.loc[data['Relay_StatusFlag']!='pending','transactionType']='Relay'

data.loc[data['Relay_StatusFlag']=='pending','transactionType']='Order'
    orders = data.loc[data['transactionType']=='Order', ].copy()
    relay = data.loc[data['transactionType']=='Relay', ].copy()
    order_list = list(set(orders['orderID'].tolist()))
    for order in order_list:
        try:
            orders.loc[orders['orderID']==order,'Relay_StatusFlag']=
relay.loc[relay['orderID']==order,'Relay_StatusFlag'].tolist()[0]
        except (KeyError, IndexError):
            print(f'No Relay with orderID {order}')
            pass

    orders['gasCostForDelivery (GWei)']= orders['gasCostForDelivery
(Wei)']*10**(-9)
    orders['Transaction_Fee (Gwei)'] = orders['Transaction_Fee
(Gwei)'].fillna(0)
    orders['OrderCost (GWei)'] = orders['gasCostForDelivery
(GWei)']+orders['commitment_fee (GWei)'] +orders['Transaction_Fee
(Gwei)']
    orders.loc[orders['Relay_StatusFlag']==False, 'OrderCost (GWei)'] =
orders['gasCostForDelivery (GWei)'] + orders['Transaction_Fee (Gwei)']
    orders["OrderCost (CHF)"] = orders["OrderCost
(GWei)"]/(orders["ExchangeRate"]*10**9)
    orders_columns = ['hash', 'orderID', 'receiverAddress', 'DateTime',
'sender_PIN', 'senderID',
                      'commitmentID','Relay_StatusFlag', 'Transaction_Fee
(Gwei)', 'gasCostForDelivery (GWei)', 'commitment_fee (GWei)',
                      'OrderCost (GWei)','ExchangeRate','OrderCost
(CHF)']
    orders = orders[orders_columns]
    relay_list = list(set(relay['orderID'].tolist()))
```

```
    if len(relay_list)==0:
        print('no complete Order&Delivery transaction since last
scanned block')
        exit()
    else:
        for order in relay_list:
            try:

relay.loc[relay['orderID']==order,'GasObtainedForDelivery (GWei)']=
orders.loc[orders['orderID']==order,'gasCostForDelivery
(GWei)'].tolist()[0]
            except (KeyError, IndexError):
                print(f'Not order for for orderID {order}?')

relay.loc[relay['orderID']==order,'GasObtainedForDelivery (GWei)']=0
                pass
        relay['Unused_GasForDelivery (GWei)'] =
relay['GasObtainedForDelivery (GWei)'] - relay['Transaction_Fee
(Gwei)']
        relay['Sender_Profit (Gwei)'] = relay['Unused_GasForDelivery
(GWei)'] + relay['commitment_fee (GWei)']/2
        relay.loc[relay['Relay_StatusFlag']!=True, 'Sender_Profit
(Gwei)'] = relay['Unused_GasForDelivery (GWei)']
        relay['Sender_Profit (CHF)'] =  relay['Sender_Profit
(Gwei)']/(relay["ExchangeRate"]*10**9)
        relay_columns = ['hash', 'orderID', 'receiverAddress',
'DateTime', 'sender_PIN', 'senderID',
                    'commitmentID','Relay_StatusFlag',
'GasObtainedForDelivery (GWei)', 'Unused_GasForDelivery (GWei)',
'commitment_fee (GWei)',
                    'Sender_Profit
(Gwei)','ExchangeRate','Sender_Profit (CHF)']
        relay = relay[relay_columns]
        return orders, relay
```

This code is a Python script that connects to an Ethereum blockchain and retrieves transactions for a given address. It uses the Etherscan API to retrieve normal and internal transactions and transforms them into a more readable format, merges them, and then adds additional information such as commitmentID, Relay_StatusFlag, receiverAddress, Sender_PIN, orderID, and commitment_fee. It also retrieves exchange rates and creates several CSV files containing the transaction data, the exchange rates, and MIS databases, which track orders and deliveries. The code then creates a checkpoint to keep track of the last block scanned and updates the SQLite database with the new transactions.

```
if __name__=='__main__':
    if os.path.isfile("checkpoint/startblock.txt"): (4)
        with open("checkpoint/startblock.txt", "r") as f1:
            b = f1.read()
            try:
                startblock = str(b)
            except:
                startblock = '0'
    else:
        startblock = '0'

    connection = sqlite3.connect('checkpoint/sqlite_tx.db')
    cursor = connection.cursor()
    # create table in db if not yet created before
    cursor.execute(
        "CREATE TABLE IF NOT EXISTS tx_hash (id INTEGER PRIMARY KEY,
tx_hash TEXT, blockNumber INTEGER)")

    # Get normal Transaction from Polygonscan API and create DataFrame
    REQUESTS_HEADERS = {"User-Agent": "BCP/accounting"}
    API_key = '9B9QU1IK31EYVMAV1CTTPFKQC7JP2WSJEH'
    URL = 'https://api-
testnet.polygonscan.com/api?module=account&action=txlist&address='+SCL_
ADDRESS+'&startblock='+startblock+'&endblock=latest&sort=asc&apikey=' +
API_key
    response = requests.get(URL, headers=REQUESTS_HEADERS)
    parentTx = pd.DataFrame.from_dict(response.json()['result'])

    #Get internal Transaction from Polygonscan API and create DataFrame
    REQUESTS_HEADERS = {"User-Agent": "BCP/accounting"}
    API_key = '9B9QU1IK31EYVMAV1CTTPFKQC7JP2WSJEH'
    URL = 'https://api-
testnet.polygonscan.com/api?module=account&action=txlistinternal&addres
s='+SCL_ADDRESS+'&startblock='+startblock+'&endblock=latest&sort=asc&ap
ikey=' + API_key
    response_internal = requests.get(URL, headers=REQUESTS_HEADERS)
    internalTx =
pd.DataFrame.from_dict(response_internal.json()['result'])
    if len(internalTx)== 0:
        print('No transaction since last scanned block')
        exit()

    # Tranform normal and internal Transaction
    print('normalTransaction: clean up')
```

```
    parentTx = TransformParentTransaction(parentTx)
    print('internalTransaction: clean up')
    internalTx = TransformInternalTransaction(internalTx)

    # Merge parent and internal Transactions
    print('merge transactions')
    data = MergeTransactions(parentTx, internalTx)

    #add col commitmentID, Relay_StatusFlag
    data['commitmentID'] = -1
    data['Relay_StatusFlag'] = 'pending'

    # create list of tx hashes already considered in minBlock
    minBlock = np.min(data['blockNumber'])
    cursor.execute(f"SELECT * FROM tx_hash where blockNumber
={minBlock}")
    rows = cursor.fetchall()
    tx_hash = []
    for row in rows:
        tx_hash.append(row[1])
    # add receiverAddress, Sender_PIN, orderID, CommitmentID,
statusFlag and commitment_fee
    data = get_SCL_informations(data, tx_hash)
    if len(data) == 0:
        print('No new transaction since last scanned block')
        exit()

    # add exchange rates
        data = asyncio.run(getexchangeRate(data))

    # get the last scanned block
    maxBlock = np.max(data['blockNumber'])
    # create .csv files
    MIS_database = Database_MIS(data)
    # ask if VAT should be taken into account
    VAT_bool = ask_for_VAT()
    csv_separator = ask_for_separator()
    Create_SCL_Revenue_file(data,
VAT_bool).to_csv(f'SCL_block_{startblock}_to_{maxBlock}.csv',sep=csv_se
parator, index=False)

    if startblock =='0':
```

```
MIS_database[0].to_csv(f'SCL_Orders_Database_up_to_block_{startblock}.c
sv',sep=csv_separator, index=False)

MIS_database[1].to_csv(f'SCL_Delivery_Database_up_to_block_{startblock}
.csv', sep=csv_separator, index=False)
    else:


MIS_database[0].to_csv(f'SCL_Orders_Database_up_to_block_{startblock}.c
sv', mode ='w+', sep=csv_separator, index=False, header= True)

MIS_database[1].to_csv(f'SCL_Delivery_Database_up_to_block_{startblock}
.csv', mode = 'w+', sep=csv_separator, index=False, header= True)

os.rename(f'SCL_Orders_Database_up_to_block_{startblock}.csv',f'SCL_Ord
ers_Database_up_to_block_{maxBlock}.csv')

os.rename(f'SCL_Delivery_Database_up_to_block_{startblock}.csv',f'SCL_D
elivery_Database_up_to_block_{maxBlock}.csv')

    # Create Checkpoints
    # add transaction on the last scanned block to the database
    LastBlockTransactions = data.loc[data['blockNumber'] == maxBlock, ]
    LastBlockTransactions =
list(set(LastBlockTransactions['hash'].tolist()))
    for hash in LastBlockTransactions:
        cursor.execute("INSERT INTO tx_hash(tx_hash, blockNumber)
VALUES (?,?)", (hash, int(maxBlock),))
        connection.commit()
    # store the number of the last scanned block
    with open("checkpoint/startblock.txt", "w") as f2:
        f2.write(str(maxBlock))
```

Appendix B

Output 1

	Date	Doc	Description	Accoun-tDebit	Accoun-tCredit	Amount-Currency	Exchan-geCurr-	Vat-Code	ExchangeRate	Amount
1	27.03.2023		0x18f78ff5024c16d4	1027	3001	47050000.0	gwei	F3	0.983356572882	0.04785
2	27.03.2023		0x18f78ff5024c16d4	1027	2330	2950000.0	gwei	F3	0.983356572882	0.00300
3	05.04.2023		0x76468049390e71c	1027	3001	47050000.0	gwei	F3	0.968804682255	0.04857
4	05.04.2023		0x76468049390e71c	1027	2330	2950000.0	gwei	F3	0.968804682255	0.00304
5	05.04.2023		0x393c41370f313bc	1027	3001	47050000.0	gwei	F3	0.968804682255	0.04857
6	05.04.2023		0x393c41370f313bc	1027	2330	2950000.0	gwei	F3	0.968804682255	0.00304
7	05.04.2023		0x8537fdcbb3f0b4e	1027	3001	47050000.0	gwei	F3	0.968804682255	0.04857
8	05.04.2023		0x8537fdcbb3f0b4e	1027	2330	2950000.0	gwei	F3	0.968804682255	0.00304
9	12.04.2023		0x20da34e5f27826	1027	3001	47050000.0	gwei	F3	0.993906842945	0.04734
10	12.04.2023		0x20da34e5f27826	1027	2330	2950000.0	gwei	F3	0.993906842945	0.00297
11	12.04.2023		0xf436c9354fa412b	1027	3001	47050000.0	gwei	F3	0.993906842945	0.04734
12	12.04.2023		0xf436c9354fa412b	1027	2330	2950000.0	gwei	F3	0.993906842945	0.00297
13	13.04.2023		0xc98dde9b03ea03	1027	3001	47050000.0	gwei	F3	1.011125454223	0.04853
14	13.04.2023		0xc98dde9b03ea03	1027	2330	2950000.0	gwei	F3	1.011125454223	0.00292
15	14.04.2023		0xd07237d7a8e54e	1027	2330	2950000.0	gwei	F3	0.990784716129	0.00298
16	14.04.2023		0xd07237d7a8e54e	1027	3001	47050000.0	gwei	F3	0.990784716129	0.04749
17	14.04.2023		0x108eaa113cd85f70	1027	3001	47050000.0	gwei	F3	0.990784716129	0.04749
18	14.04.2023		0x108eaa113cd85f70	1027	2330	2950000.0	gwei	F3	0.990784716129	0.00298
19	03.05.2023		0x273b886975d8430	1027	2330	2950000.0	gwei	F3	1.146897652725	0.00257
20	03.05.2023		0x273b886975d8430	1027	3001	47050000.0	gwei	F3	1.146897652725	0.04103
21	16.05.2023		0x262e088505e1ae	1027	2330	2950000.0	gwei	F3	1.295489432075	0.00228
22	16.05.2023		0x262e088505e1ae	1027	3001	47050000.0	gwei	F3	1.295489432075	0.03632
23	16.05.2023		0xc2ac4f9c0029114	1027	2330	2950000.0	gwei	F3	1.295489432075	0.00228
24	16.05.2023		0xc2ac4f9c0029114	1027	3001	47050000.0	gwei	F3	1.295489432075	0.03632
25	16.05.2023		0x4ee5d3cae800794	1027	2330	2950000.0	gwei	F3	1.295489432075	0.00228
26	16.05.2023		0x4ee5d3cae800794	1027	3001	47050000.0	gwei	F3	1.295489432075	0.03632
27	16.05.2023		0xc3a8d8703aa1c1	1027	3001	47050000.0	gwei	F3	1.295489432075	0.03632
28	16.05.2023		0xc3a8d8703aa1c1	1027	2330	2950000.0	gwei	F3	1.295489432075	0.00228

Output 2

	hash	orderID	receiverAddr-ess	DateTime	sender_PIN	senderID	commit-mentID	Relay_Status-Flag	GasObtainedForDeli-very (GWei)	Unused_GasForDelivery (GWei)	commitment_fee (GWei)	Sender_Profit (GWei)	ExchangeRate	Sender_Profit (CHF)
1	0x18f78ff5024	266.0	0x067504587E1	27.03.2023	0x81996D1(	1.0	1	True	14400000.0	10167000.00007055	100000000.0	60167000.00007055	0.983356572882	0.061185333641218690
2	0x764664939(	270.0	0x067504587E1	05.04.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	0.968804682255	0.062103622228607410
3	0x393c41370f	268.0	0x067504587E1	05.04.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	0.968804682255	0.062103622228607410
4	0x8537fdcbb3l	269.0	0x067504587E1	05.04.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	0.968804682255	0.062103622228607410
5	0x20da34e5f2	272.0	0x067504587E1	12.04.2023	0x81996D1(	1.0	1	True	14400000.0	10167000.00007055	100000000.0	60167000.00007055	0.993906842945	0.060535854468807625
6	0xf436c93544a	273.0	0x067504587E1	12.04.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	0.993906842945	0.060535130054839550
7	0xc98dde9b03	276.0	0x067504587E1	13.04.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	1.01112454223	0.059504267990469470
8	0xd07237d7a(	279.0	0x067504587E1	14.04.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	0.990784716129	0.060725886280463090
9	0x108eaa113(	280.0	0x067504587E1	14.04.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	0.990784716129	0.060725886280463090
10	0x273b86975(	284.0	0x067504587E1	03.05.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	1.14669765272	0.052469175163210690
11	0x262e08850f	287.0	0x067504587E1	16.05.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	1.295489432075	0.046442895256738270
12	0xc2ac4f9c00	289.0	0x067504587E1	16.05.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	1.295489432075	0.046442895256738270
13	0x4ee5d3cce6	290.0	0x067504587E1	16.05.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	1.295489432075	0.046442895256738270
14	0xc3a8d8703a	288.0	0x067504587E1	16.05.2023	0x81996D1(	1.0	1	True	14400000.0	10166280.00007056	100000000.0	60166280.00007056	1.295489432075	0.046442895256738270

Output 3

	hash	orderID	receiverAddress	DateTime	sender_PIN	senderID	commitmentID	Relay_StatusFlag	Transaction_Fee	gasCostForDelivery (GWei)	commitment_fee (GWei)	OrderCost (GWei)	ExchangeRate	OrderCost (CHF)
1	0x96720e0t	266.0	0x067504	27.03.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	0.983356572882	0.11633623362552911
2	0x1875695	268.0	0x067504	05.04.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	0.968804682255	0.11180836572070608
3	0xb19124b	269.0	0x067504	05.04.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	0.968804682255	0.11180836572070608
4	0x588c66f0	270.0	0x067504	05.04.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	0.968804682255	0.11180836572070608
5	0x86e474a	272.0	0x067504	12.04.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	0.993906842945	0.11510133048387773
6	0xe34e945	273.0	0x067504	12.04.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	0.993906842945	0.11510133048387773
7	0x24c1daa	276.0	0x067504	13.04.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	1.011125454223	0.11314125217815900
8	0xb2352a0	279.0	0x067504	14.04.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	0.990784716129	0.11546403384880752
9	0xee67f33e	280.0	0x067504	14.04.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	0.990784716129	0.11546403384880752
10	0x09020f22	284.0	0x067504	03.05.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	1.146697652725	0.09976474594514176
11	0x764d919	287.0	0x067504	16.05.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	1.295489432075	0.08830639383662454
12	0xe2ba883	288.0	0x067504	16.05.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	1.295489432075	0.08830639383662454
13	0xdf7333b6	289.0	0x067504	16.05.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	1.295489432075	0.08830639383662454
14	0x75a8ab5	290.0	0x067504	16.05.2023	0x81996D	1.0	1	True		14400000.0	100000000.0	114400000.0	1.295489432075	0.08830639383662454

Balance Sheet

Smart Contracts Lab
Accounting 2023
2023

BALANCE SHEET

ASSETS

	31.12.2023
Cryptocurrencies account	0.65469
Cash and cash equivalents	**0.65469**
Current assets	**0.65469**
Total Assets	**0.65469**

LIABILITIES

	31.12.2023
VAT	0.03863
Accruals and deferred income	**0.03863**
Short-term third party capital	**0.03863**
Third party capital	**0.03863**
Profit	0.61606
Profit / Loss from Balance Sheet	**0.61606**
Equity	**0.61606**
Total liabilities	**0.65469**