Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Edward Palm
Håkon Isaksen Frøland

Web3 Security in Reseller Markets with Proof of Concept Implementations

Master's thesis in Communication Technology
Supervisor: Danilo Gligoroski
Co-supervisor: Katina Kralevska
June 2023

NTNU
Norwegian University of Science and Technology

Edward Palm
Håkon Isaksen Frøland

Web3 Security in Reseller Markets with Proof of Concept Implementations

Master's thesis in Communication Technology
Supervisor: Danilo Gligoroski
Co-supervisor: Katina Kralevska
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Title: Web3 Security in Reseller Markets with Proof of Concept Implementations

Students: Frøland, Håkon I. and Palm, Edward

Problem description:

Reseller markets allow people to sell unused items and provide value for others, benefiting society and the environment. However, traditional reseller markets have issues, such as counterfeit products, mistrust between buyers and sellers, and lack of transparency in product value. Buyers can have a difficult time identifying fake items and may struggle to trust unfamiliar sellers. To address these challenges, our thesis proposes leveraging Web3 and blockchain technology to create a more secure, transparent, and fair reseller market that benefits all parties involved.

We will start our thesis by extending our project work by continuing our theoretical research on the implementations of Web3. The concept of Web3 is for many an abstract concept with a limited number of implementations having an actual use case. While Web3 technology is a promising solution for creating trust and transparency in transactions, we recognize that there have been instances of scams using blockchain that have undermined trust in the technology. Therefore, we seek to demonstrate that Web3 is not as intimidating as it may seem and can be used effectively in today's society. We believe that by showcasing a real-world use case, we can encourage wider adoption and facilitate the development of secure and transparent systems.

Further, we will create a proof-of-concept using Web3 and blockchain as a solution that solves the problems in the reseller markets. By combining the immutability of the blockchain, the properties of Non-Fungible Tokens (NFTs), and the incorporation of decentralization we want to show how Web3 can create a new way of trading used and/or new goods. Storing product information on the blockchain makes it possible for the buyers to verify the product's origin, history, and authenticity.

Approved on:	2023-2-23
Main supervisor:	Prof. Gligoroski Danilo, NTNU
Co-supervisor:	Assoc. Prof. Kralevska Katina, NTNU

Abstract

Traditional reseller markets face significant challenges, including inconsistent documentation, unethical practices, lack of transparency, trust issues, and concerns about authenticity. This thesis presents a Proof of Concept (PoC) solution that leverages Web3 technology, specifically blockchain and Non-Fungible Tokens (NFTs), to address these issues and create a secure, transparent, and fair reseller market without the need for a trusted third party.

The PoC demonstrates that NFTs can be used to authenticate items, build trust between parties and establish verifiable product ownership. By paralleling NFT transfers with the sale of items, the practicality of selling counterfeit goods is reduced. Furthermore, the blockchain-based storage of product information enables verification of origin, history, and authenticity by any involved party, fostering transparency and equal access to information for buyers and sellers.

This thesis demonstrates that Web3 technology can effectively address critical issues in traditional reseller markets, offering substantial benefits to all parties involved. However, the fluctuation and uncertainties associated with costs must be addressed to facilitate broader adoption. Further research and collaboration are essential to establish a predictable and affordable cost structure within the Web3 ecosystem, fully unlocking the potential of blockchain technology in modernizing reseller markets.

Sammendrag

I tradisjonelle bruktmarkeder er det utfordringer relatert til håndtering av kvitteringer, skjevfordelt informasjonstilgang, mangel på tillit og tvilsomt opphav. Denne avhandlingen presenterer et konseptbevis (POC) som utnytter Web3-teknologi, mer spesifikt blokkjedeteknolgi og NFTer til å adressere disse utfordringer. Formålet er å lage et sikkert, transparent og rettferdig bruktmarked uten å belage seg på en tredjepart.

Konseptbeviset demonstrerer at NFTer kan autentisere gjenstander, bygge tillit mellom parter og etablere verifiserbart eierskapsbevis. Ved å samtidig overføre NFTer ved salget av et produkt, reduseres legitimiteten til forfalskede produkter. Videre ved å lage produktinformasjon på blokkjeden tilrettelegges muligheten for å verifisere opphavet, historien og autentisiteten til ulike aktører. Dette skaper lik tilgang til informasjon og reduserer usikkerheten for de involverte partene.

Denne avhandlingen beviser at Web3-teknologi kan brukes for å bekjempe utfordringene i de tradisjonelle bruktmarkedene, og tilbyr betydelige fordeler for alle. Imidlertid må svingninger og usikkerheter knyttet til kostander reduseres for alminnelig anvendelse. Videre forskning er viktig for å skape en rimelig og forutsigbar kostnadsstruktur i Web3-økosystemet, og nødvendig for å kunne ta i bruk det fulle potensiale til blokkjedeteknologi i samfunnet.

Preface

This Master's thesis is the final delivery of a 5-year Communication Technology study program at Norwegian University of Science and Technology (NTNU). It was conducted in our last semester from January to June 2023.

We wish to express our profound gratitude to our supervising professor, Danilo Gligoroski, for his unwavering support and mentorship that took our initial idea to a fully realized master's thesis. We would also like to extend our sincere appreciation to our co-supervisor, Katina Kralevska for her invaluable assistance. Her expertise and insightful feedback helped shape the course and outcome of this master's thesis. Additionally, we like to extend our thanks to the Department of Information Security and Communication Technology at NTNU. Their persistent encouragement, stimulating challenges, and instructive guidance over the course of our five-year journey as students have been worth the student loan. They have played an instrumental role in our academic growth and development, for which we are immensely thankful.

Special thanks to our fellow classmates and friends, for the stimulating discussions, late-night study sessions, and all the fun we have had in the last five years. Your friendship made this journey enjoyable and memorable. We are heartily thankful to our family: our parents, our grandparents and our siblings, for their love and support. Your faith in us has always been a source of strength and inspiration when we have been feeling low.

We cannot end this without giving a shout-out to the unsung heroes of our academic journey - microwave dinners, energy drinks, and our trusted friend, coffee. They fueled countless late-night study sessions and last-minute paper writings. Their unwavering service, ready at the oddest hours, ensured our brains kept ticking and our eyelids resisted gravity. To them, we raise our glasses and promise to strive for a more balanced diet in our post-academic lives.

Håkon I. Frøland
Edward Palm
Trondheim 2023

Contents

List of Figures

List of Tables

List of Acronyms

ABI Application binary interface.

API Application programming interface.

B2C Business-to-Customer.

C2C Customer-to-Customer.

CDN Content Delivery Network.

DAO Decentralized Autonomous Organization.

dApp Decentralized application.

ECDSA Elliptic Curve Digital Signature Algorithm.

EdDSA Edwards-curve digital signature algorithm.

EVM Ethereum Virtual Machine.

IPFS InterPlanetary File System.

NFT Non-Fungible Token.

NIST National Institute of Standards and Technology.

NTNU Norwegian University of Science and Technology.

P2P Peer-to-Peer.

PoC Proof of Concept.

UML Unified Modeling Language.

Chapter 1
Introduction

1.1 Motivation

Reseller markets play a significant role in our daily lives, providing individuals with platforms to sell goods and services to one another. These markets offer a valuable arena for used items to find new owners, and people can sell items they no longer need, which in turn is great for the society and the environment. However, these markets have various issues. One of the most significant obstacles that traditional reseller markets face today is fake or counterfeit products. It can be difficult for buyers to identify these items and they are often left holding the short end of the stick. Establishing trust between buyers and sellers who have no affiliation is generally challenging, causing potential trades to fall through. The lack of certainty in the item's authenticity, the seller's morale, and the number of counterfeit products in circulation can discourage the buyer. Moreover, reseller markets often lack transparency, making it challenging for buyers to assess the true value of a product. Overall, traditional reseller markets suffer from multiple issues that harm both buyers and sellers.

Web3 [1] is an idea for a new iteration of the World Wide Web which incorporates concepts such as decentralization, blockchain technology, and token-based economics. Although it is often seen as an abstract concept with limited practical implementations, it holds great potential. The technology behind Web3 with decentralization and blockchain is often thought of as too technical and advanced, which heightens the threshold for widespread adoption in society. Unfortunately, there have been several cases of scams using blockchain today which fuel mistrust rather than trust in the blockchain [2], as is the core concept of the technology. However, Web3 has numerous use cases which can benefit society and help build bridges between people, and when utilized correctly has the potential to improve existing services significantly. By bridging the gap between the challenges faced by traditional reseller markets and the untapped potential of Web3 technology, we aim to create more secure, efficient, and reliable platforms for individuals to engage in the trade of secondhand goods.

1.2 Objectives

The primary objective of our thesis is to demonstrate that by taking advantage of modern ways of application development, it is possible to implement a decentralized application that leverages blockchain technology, to create a secure, transparent, and fair reseller market without the need for a trusted third party. This in turn would provide substantial benefits to all involved parties.

By creating a PoC using Web3 and blockchain as a solution that solves a real-life problem, we seek to show that the usage of Web3 is not as frightening as it might seem, as well as having a genuine usage in today's society. By combining the immutability of the blockchain, the properties of an NFT, and the incorporation of decentralization, we want to show how Web3 can create a new and better way of trading used goods. As our PoC, we plan on creating a network of NFTs that are transferred in parallel to the sale of an item. Each NFT is used to authenticate the item as well as to build trust between the two parties. In our solution, the item is useless without the NFT, which makes selling fake copies impractical as the NFT can only be transferred once. It will also authenticate the item's owner, as the item will be registered in their name. By storing product information on the blockchain, buyers can verify the product's origin, history, and authenticity. The data can be accessed by all involved parties, ensuring transparency as the buyers and sellers have equal access to relevant information about the product. The objectives can be summarized into the following points.

1: Identify the critical issues in the traditional reseller markets

To accurately assess the impact of our thesis, it is essential to thoroughly identify and define the key issues in the ecosystem of traditional reseller markets. Our aim here is to concretize these issues down to distinct points that can serve as a basis for evaluating the effectiveness of our proposed implementation.

2: Through the creation of a proof of concept solution, demonstrate that the traditional reseller market benefits from Web3 technology

Here we will implement the potential of the Web3 environment, incorporating blockchain technology and decentralized applications, in the context of the secondhand market industry. By creating a PoC solution, we aim to present the valuable features that Web3 can offer to the traditional reseller markets. Additionally, this research intends to demonstrate the potential of Web3 technology for enhancing existing services in various industries.

3: Clarify to which degree the implementation solves the defined issues

After implementation, we need to establish a connection between the developed solution and the identified issues in the traditional reseller markets. This connection will enable us to evaluate the extent to which the identified issues have been effectively addressed and resolved through the application of Web3 technology. In addition, this will reflect on the relevance of Web3 solutions in the present society.

1.3 Research methods

In this thesis we first get to know the state-of-the-art in the field by reviewing academic papers, reports, and articles, we then design, implement, and test a PoC application and validate the system's viability. The research will explore the relevant field and investigate how Web3 can be integrated into the targeted ecosystem. The resulting application can serve as a foundation for further research or provide assistance in the development of decentralized applications in the Web3 domain. This solution will be presented through the design of an artifact aimed at addressing the identified problems. Therefore, our approach is based on the principles of design science [3] and requirements engineering. Figure 1.1 illustrates how the domain theory initiates an iterative process that will be followed throughout this project.

The initial step in designing our solution involves identifying and formulating the system requirements that our application must comply with. We will follow best practices proposed by Wiegers and Beatty in their book "Software Requirements"[4] to ensure the quality of our requirements. This phase will translate general requirements to specific system requirements, which will serve as a basis for validating the implementation. Once the requirements are established, we will create an architecture that aligns with these requirements. This process will generate an architecture model for the system. In the third step, we will implement the solution based on the defined architecture model and test its functionalities. The tests will be designed based on the background research findings and the predetermined requirements. The outcome of this process will constitute our main findings, which will be used to evaluate the system and the overall project.

Our methodology's main output will be a system validation, which includes observations of the PoC along with the corresponding metrics collected from our conducted test scenarios. The observations will be used to validate if our system can fulfill the desired requirements and address the identified problems. The metrics, particularly the cost of use will provide objective results that can be used to assess our solution's feasibility.

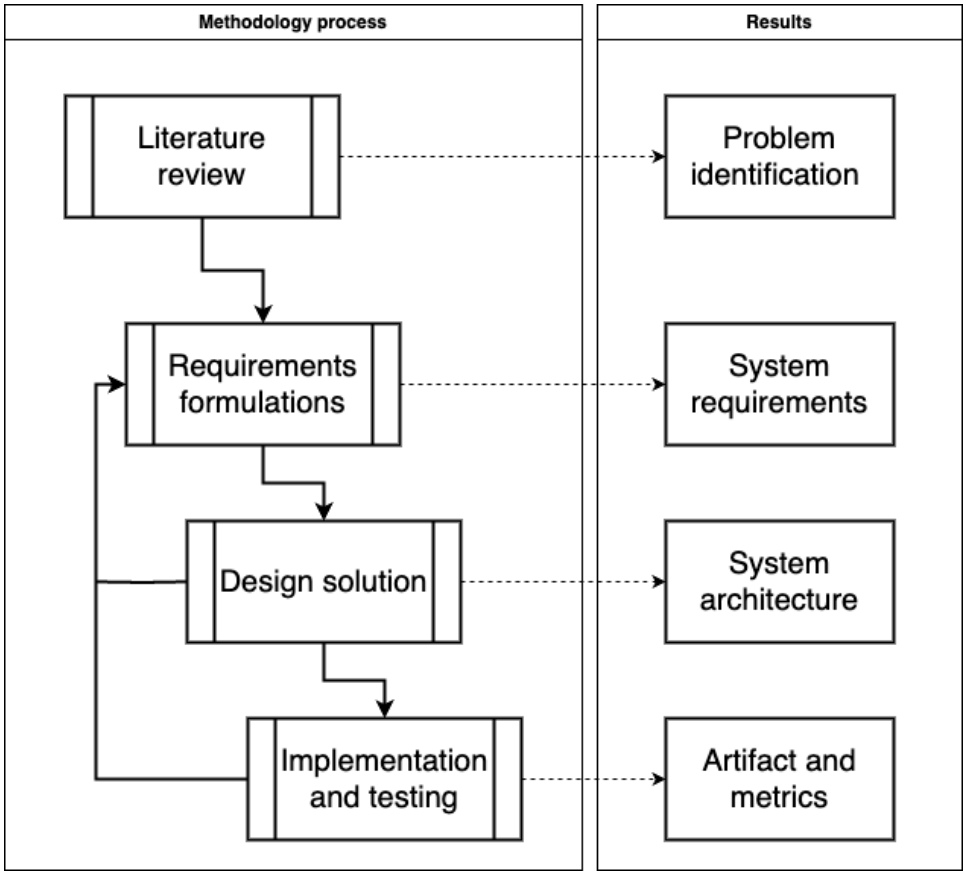


Figure 1.1: Flowchart representing the overall project progression based on the methodology.

1.4 Contribution

This thesis explores the potential of integrating Web3 technology into real-world scenarios, with a focus on retail and secondhand markets. Firstly, we conduct in-depth research and modeling of the traditional retail and secondhand markets, identifying the key challenges they face. Secondly, we propose a comprehensive design and architecture for integrating Web3 technology into these markets, addressing the identified issues and providing solutions. Finally, we demonstrate the effectiveness of our approach through the development and implementation of a PoC application. This practical demonstration showcases the real-world applicability of Web3 technology in improving transparency, trust, and authenticity in retail and secondhand markets. Our PoC, which is a fully functional prototype, is available as open-source software

on GitHub[1]. In addition to this thesis, the work has resulted in an academic paper "Web3 and Blockchain for Modernizing the Reseller Market" which will be submitted to a Blockchain conference and has been included in Appendix E.

1.5 Outline

This thesis is organized of eight chapters, the first of which is this introductory chapter. Chapter 2 provides the theoretical background, including an overview of different markets, cryptographic features, and blockchains. It also reviews related work in the research field. Chapter 3 presents the main findings, requirements and specifications derived from the background analysis. Chapter 4 offers an overview model of the system design and architecture, followed by a detailed description in Chapter 5 of the developed application, including deployment and implementation details. Chapter 6 presents the results of the implementation, while Chapter 7 discusses and analyzes the findings. Finally, Chapter 8 concludes the project.

[1] https://github.com/Autentisk

Chapter 2
Background

This chapter serves as a foundation to grasp the theory behind our implementation and to justify the significance and relevance of the PoC. Firstly, we will introduce meaningful background about different markets which exist today and their current status. Further, a section about cryptography features, blockchains, and Web3 is presented. This will provide an understanding of the necessary technological insight needed to understand the functionality behind our PoC. Lastly, relevant research will be introduced, while highlighting the differences and importance of this thesis.

The state-of-the-art was reviewed, and an identification of the relevant background material was carried out in the project preceding this thesis [5]. This is extended with other papers and sources.

2.1 Different markets

Retail markets and secondhand markets are distinct segments within the overall market landscape, differing in their nature and the types of products they offer.

Retail can be defined as "the activity of selling goods to the public, usually in shops [6]". In other words, retail markets offer a wide range of services or merchandise from various brands or manufacturers. Retailers are the final point of contact in the supply chain, interacting directly with customers and offering brand-new products that have not been previously owned or used. They typically operate physical stores, online platforms, or a combination of both. These companies focus on providing consumers with the latest products, ensuring quality and guaranteeing authenticity. The transactions in retail markets typically involve the purchase of products at their original or market price.

On the other hand, secondhand markets can be defined as markets selling products "owned or used in the past by someone else [7]". These markets revolve around the exchange of products that have been previously owned, used, or worn by other

individuals. Secondhand markets can encompass various categories, including clothing, furniture, electronics, vehicles, and more. These markets provide an avenue for individuals to sell their unwanted or unneeded items, while buyers can acquire products at lower prices compared to retail markets. Secondhand markets originally operated through channels like flea markets or garage sales, however, we are seeing a growing increase of secondhand products in reseller markets like Finn.no, Amazon, and eBay [8] [9].

The key distinction between retail markets and secondhand markets lies in the status of the products being traded. Retail markets focus on new goods, catering to consumers' desire for the latest products, while secondhand markets involve the resale of used items, emphasizing affordability, sustainability, and the potential for finding unique or vintage items. The latter utilizes popular reseller markets to connect with the customer.

2.1.1 Retail markets

The management of receipts and documentation in the context of retail purchases and secondhand sales presents various challenges and potential risks. When buying expensive items from retailers, consumers often receive receipts as proof of purchase and is important for potential returns or future resale. There exist digital solutions such as mobile applications, Bluetooth transfers, or the possibility of receipts received via email [10] [11] [12]. However, there is no defined standard and with all these different ways of storage, concerns can be raised regarding legitimacy and long-term accessibility. For example, digital storage of receipts via email can lead to issues, as they may be lost among numerous daily emails or subject to automatic deletion by email providers. Furthermore, additional paperwork may be acquired such as services, repairs, or insurance claims, further complicating the documentation process.

In Norway, instances have been reported where used products have been returned before the refund deadline and subsequently sold as brand new, highlighting deceptive practices within the retail industry [13] [14]. Moreover, news articles have shed light on the extensive disposal and destruction of returned products, highlighting the challenges retailers face in handling returned items [15] [16]. Such practices raise concerns about the transparency and integrity of retailers in their treatment of returned products, including potential deception of customers and the negative environmental impact of disposing of potential high-quality products.

2.1.2 Secondhand markets

The act of selling used items to others is referred to as secondhand trading. The market for secondhand goods has a rich history and significant importance. In European countries, the practice of buying and selling used garments dates back to

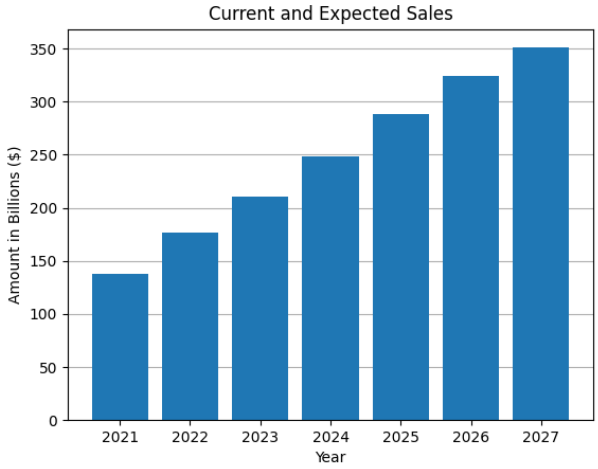


Figure 2.1: Trajectory of apparel sales in secondhand markets [19].

the 14th century, driven by economic constraints. The desire to own a variety of goods and the need to work within limited budgets fueled the growth of the trade in pre-owned items and continues to thrive today [17]. Secondhand markets make effective use of social resources and can even stretch global production networks [18]. The global secondhand markets are in a rapid growth trajectory, with as much as $177 billion sold in apparel alone in 2022 [19].

Typically, the value and quality of secondhand items may decrease due to their pre-owned status and the fact that their initial purchase or sale was not made by the final consumer. However, this market offers numerous benefits, including affordable pricing, the potential for profitable resale, and a wide selection of products from various brands that can rival their newly manufactured counterparts. The secondhand goods market includes various sectors, such as apparel, furniture, cars, books, machinery, electronics, and even properties with changed ownership.

To find the motivation of secondhand shoppers, Guiot and Roux [20] have introduced an 8-factor scale that demonstrates reliability and validity in capturing motivations related to both the desired products and the distribution channels that offer them. Through the development of this scale, the research uncovers the significance of key motives that drive consumers towards secondhand products and channels. Notably, the factors of "distance from the system" and "ethics and ecology" play a vital role in shaping consumer behavior in this context. These factors intertwine with more commonly discussed economic motives such as the "allocative role of price" and "seeking a fair price," as well as hedonic and recreational motives

like "treasure hunting," "originality," "social contact," and "social pleasure". They also note that in specific product categories that are perceived to carry a higher level of risk, such as household appliances, computers, or televisions and audio equipment, the potential of the secondhand market is less trusted.

In order to gain a deeper understanding of some of the challenges faced by the secondhand market, we have devised three important aspects: transparency, trust, and authenticity. Each of these plays a crucial role in shaping the dynamics of the secondhand market. We will address each aspect individually, exploring the specific issues and implications associated with them in the context of the secondhand market.

Transparency

In the realm of secondhand markets, the process of buying and selling is primarily facilitated through online platforms where sellers have the autonomy to provide descriptions and set prices for their commodities. This nature of the market allows buyers to engage in online discussions with sellers, enabling them to make informed decisions regarding their purchases. However, it is important to note that the lack of supervision in these platforms raises concerns about transparency and reliability.

One key issue lies in the subjective nature of product descriptions, which are predominantly written by the sellers themselves. This subjectivity introduces a level of ambiguity, particularly in relation to crucial aspects such as the condition of the items [21]. The accurate assessment of item condition plays a significant role in determining the true value of a product, yet the absence of robust evidence regarding this condition poses a challenge. Consequently, the probability of customer dissatisfaction is heightened, particularly when buyers encounter sellers with low or few reputation ratings [22].

Another key issue lies within the imbalance between the seller and the buyer in regard to knowledge about the product. Knowledge about previous owners, previous prices at purchase, and direct information about the product gives sellers a key edge. The buyers are often in a position where the only information they have access to is the information the seller is willing to give.

Given these circumstances, the evaluation of a seller's reputation becomes pivotal in mitigating risks for buyers in online secondhand markets. The provision of valuable information by sellers can alleviate buyers' perceived risk and increase the likelihood of satisfactory transactions [23]. When buyers have access to comprehensive and reliable details about the products they intend to purchase, they are more inclined to engage in successful and mutually beneficial exchanges [24].

Trust

As explained by Lee and Lee [25] in the realm of online used markets, customers harbor concerns regarding two distinct facets of initial trust. These two concerns can be divided into trust towards the seller and trust towards the product. Together these two dimensions encapsulate the fundamental factors that contribute to the establishment of trust in the online used market. Lee and Lee also observe that the establishment of initial trust in unknown stores and products can be facilitated by the presence of third-party assurance seals. This means that the existence of a third party that can vouch for the product or the seller can be a crucial impact on the trust and the buyer experience [26]. Additionally, trust is reciprocally transferred between the involved parties, thereby contributing to the maintenance of a cognitive balance structure.

To extend on these differences of trust, the trust towards the seller further extends to the factual information the seller states and the perception of transparency. There are more regulations for retail and resellers, while with secondhand goods the seller can often be vague and choose to not disclose important aspects. Terms for quality as "as good as new" [27] and "sold as is" are common, and place the customer at a disadvantage. Joo [28] discovered that with the presence of trust, the buyer is willing to pay a higher price. When customers buy more expensive products, their behavior and decision to buy are more sensitive to the trust in the seller than the price of the product.

Many e-commerce companies such as Finn.no, Amazon, and eBay create arenas where either Business-to-Customer (B2C) or Customer-to-Customer (C2C) can sell products. It is normal to implement the possibility for the buyer to rate the experience or the seller. This creates the possibility for sellers to build up ratings and thus trust through the validation of other buyers. Although this is a way to build trust on these platforms, there are studies showing that the rating system can be manipulated by sellers purchasing fake reviews [29] [30]. In addition, the requirements to create accounts on such marketplaces are often just e-mail and phone numbers which makes it possible for sellers to create new accounts if they get too many bad reviews. As one would expect, the quantity of good reviews builds up trust which has a positive impact on buyers' willingness to pay. However, it turns out that there is no impact on buyers' willingness to pay between new sellers with no reviews and new sellers with a couple of bad reviews [31]. This opens up the possibility for ill-intentioned sellers to create new accounts whenever the number of bad reviews starts infecting sales. To finish off in regarding rating sellers on digital marketplaces, in addition to the mentioned problems, there are no good implementations to track sellers through different arenas.

Authenticity

The purchase of goods from sources other than trusted retailers or brands raises concerns about the authenticity and quality of the product. Secondhand goods, in particular, may have gone through multiple transactions and could originate from unoriginal producers, raising doubts about their legitimacy and functionality. There is simply no way of knowing the origin of the product or the existence of previous owners. Further, the absence of authenticity compromises the overall quality, durability, and ethical standards of the product. Such compromises have led to severe injuries and, in some cases, even fatalities [32]. Counterfeit goods are often associated with the trade of fake luxury items in back alleys, but the issue extends far beyond that, encompassing a wide range of products that are currently in demand by consumers. Virtually every product that has a market demand has either been counterfeited in the past or is susceptible to counterfeiting in the future. While luxury items like purses, watches, and jewelry are commonly targeted, counterfeiting also affects industries such as automotive, electronics, defense, and healthcare [32].

Perpetrators of online counterfeiting employ various tactics to deceive consumers and evade detection. One common strategy involves the use of images featuring genuine brand logos, coupled with persuasive language emphasizing the value and affordability of their products. This makes it challenging for consumers to differentiate between genuine and counterfeit goods. Moreover, individuals who fall victim to counterfeit purchases on these sites often encounter difficulties when attempting to report their losses and seek compensation. Platforms like eBay do not directly compensate customers, although they may register complaints against sellers [21]. The lack of adequate regulation and enforcement on auction websites makes them highly susceptible to counterfeiting. This is exemplified by eBay's penalty for their inability to effectively combat counterfeit trading on their platform [33] or falsely listed certifications and scant information on Amazon [34].

2.2 Cryptographic features

In this section, we will provide a background into the cryptographic features utilized in this master's thesis. Cryptography plays a crucial role in safeguarding sensitive data and ensuring secure communication in various applications. The selection and implementation of appropriate cryptographic techniques are essential in achieving the desired level of confidentiality, integrity, and authenticity needed. This section aims to provide a comprehensive overview of the cryptographic features employed in this master's thesis, discussing their significance, functionality, and potential implications for our implementation. By exploring the cryptographic aspects incorporated within blockchain technology, we aim to shed light on the robustness and reliability of

our Web3 solution. An extensive overview of the use of modern cryptography in blockchain systems is given in [35].

2.2.1 Hash function

A hash function is a mathematical algorithm that takes a message as input and produces an output of fixed length which is unique for that particular input data [36]. We often refer to the output as a "hash" or "hash value". Hash functions are core components in modern cryptography and in this thesis, we utilize their properties and their purpose in blockchain such as address generation and tamper protection. More formally we can define a hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ as a function that maps an arbitrary finite length input to a fixed size output. To adopt a hash function into a blockchain system, it should include the following properties:

1. The hash function is applicable to blocks of data with varying lengths (In practice, the term "any length" may be limited by a large constant value, which is generally larger than any message we would want to hash [36].)
2. The hash function produces a fixed-length output [36].
3. Given a hash function H and an input x, computing the message digest H(x) is a computationally easy process [36].
4. *Pre-image resistance* - The hash is a one-way function, meaning that it is computationally infeasible to find the input from the hash output [37].
5. *Second pre-image resistance* - Even when given the input message, it is computationally infeasible to find any other input which leads to the same hash output [37].
6. *Collision resistance* - It is infeasible to find two inputs that lead to the same hash output [37].

Hash functions having properties 1 through 5 are called one-way functions. Meanwhile, hash functions satisfying all the properties are called collision-resistant hash functions [38].

2.2.2 Hash standards

The current approved hash algorithms issued by the National Institute of Standards and Technology (NIST) by the time of writing are FIPS 180-4 and FIPS 202 [39]. FIPS 180-4 specifies seven hash algorithms: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 [40]. It does have to be noted that SHA-1

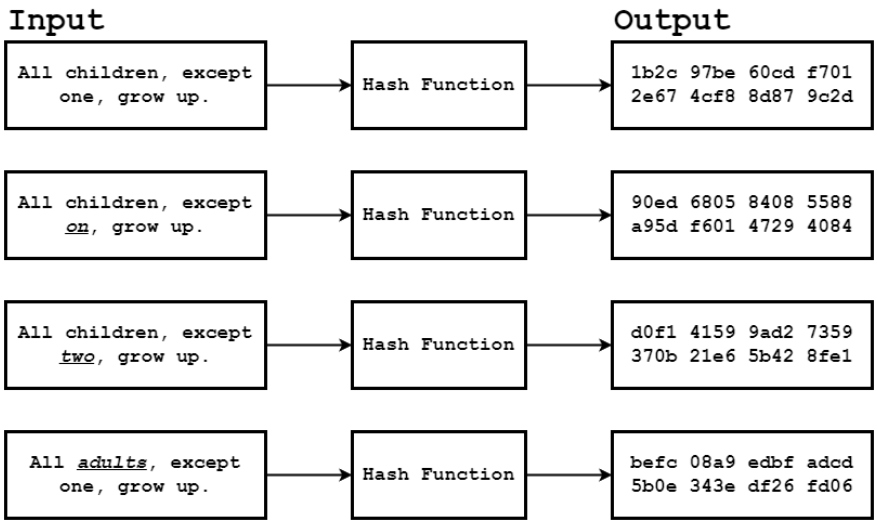


Figure 2.2: Example of a hash function.

has been revised from FIPS 180-4 as of March 7th 2023 [41], leaving only hash algorithms belonging to the SHA-2 family. Here the SHA-256 is the most known hashing algorithm as it is used by Bitcoin when mining for new hashes.

In regards to approved hash algorithms, FIPS 202 specifies four: SHA3-224, SHA3-256, SHA3-384, and SHA3-512 [42] which all belong to the SHA-3 family. NIST chose an instance of the KECCAK algorithm as the winner of the SHA-3 Cryptographic Hash Algorithm Competition, and each of the SHA-3 functions is based on this selected instance. We will not be diving into the technical aspect of these hash algorithms, but it is worth mentioning as Ethereum uses the KECCAK-256 hash algorithm [43]. This algorithm is not a part of NIST list of approved hash algorithms, but uses the same underlying computation as the SHA-3 family. The only difference is in the padding of the data, but the security level is the same [44].

2.2.3 Key-pairs

In most blockchains, public key cryptography (asymmetrical cryptography) is used to authenticate the different users. Here each user has a key pair, where a key pair consists of a public key and a private key. As indicated by their names, the private key is to be kept hidden by the owner while the public key is open to the public. They are considered a "pair" because the public key is derived from the private key using advanced mathematical operations such as elliptic curve cryptography [45]. This operation makes it easy for a user to produce the corresponding public key(s) to

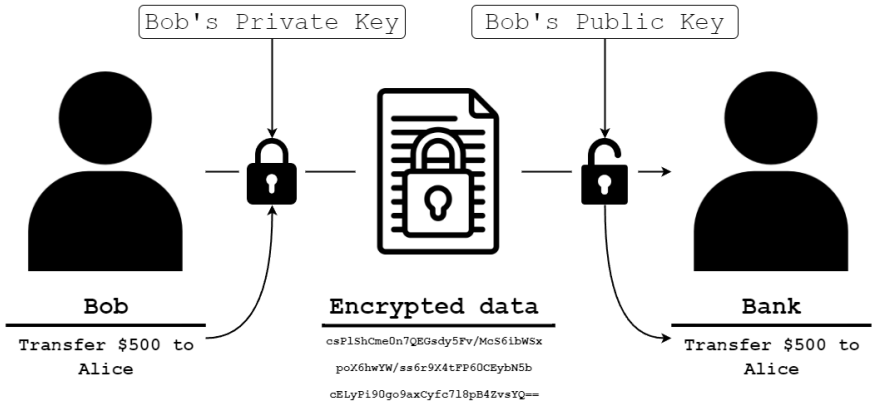


Figure 2.3: An example of digital signature in action. On the left side, Bob uses his private key to sign a message. The bank, on the right side, verifies Bobs signature with his public key[47].

its private one while making it difficult to calculate the private key from the public key.

We can use the key pair to sign messages, such that anyone can securely verify who sent a message. This is called a digital signature.

Digital signatures

Based on public-key cryptography we can generate digital signatures using the private key of the user and we can verify the signature with the corresponding public key. In the blockchain, digital signatures are used to authenticate transactions in order to verify honest transactions. In other words, a user proves his authenticity to spend owned assets in a transaction while preventing others from spending those assets.

The Elliptic Curve Digital Signature Algorithm (ECDSA) and the Edwards-curve digital signature algorithm (EdDSA) are the most widely used signature schemes in the blockchain. These signature schemes are considered secure due to their dependence on the elliptic curve version of the discrete logarithm problem's hardness assumption [46].

2.2.4 Wallets

One of the simplest methods to store private keys today is through hosted crypto wallets. These wallets, such as Coinbase, MetaMask, and TrustWallet, allow you to create an account with a third party that manages your keys. The process is similar to logging into other applications, using a password and two-factor authentication.

Since it is not practical to remember numerous characters, this method is likely to be the norm for key management if Web3 services become widely used. However, it places significant pressure on the hosted crypto wallets to safeguard your private keys, as well as on the user to properly establish their passwords and two-factor authentication. One disadvantage of decentralized assets and wallets is that if your wallet is breached, your assets are most likely lost forever [5].

2.3 Blockchain

In simple terms, a blockchain is a publicly accessible ledger or distributed database that records all transactions or digital events that have occurred and been shared among participants. Each transaction is verified by the majority consensus of the system's participants, and once recorded, the information cannot be erased. The blockchain provides an accurate and verifiable record of every single transaction ever made. The most well-known example of blockchain-based technology is Bitcoin, which is a decentralized, peer-to-peer digital currency. While Bitcoin itself is highly debated, the underlying blockchain technology has functioned nearly faultless and has been applied in a wide range of financial and non-financial domains [48].

To give a more visual picture, a blockchain can be thought of a series of blocks after each other. A new block contains either one or many new transactions which is then added to the end of the other blocks. These blocks are chained together by using the previous block's hash, making it impossible for someone to go back and change it without having to change all the succeeding blocks, as this would change that block's hash. Without going too far into how the validation of blocks on the blockchain works, the essence is that once a block has been validated, it cannot be changed.

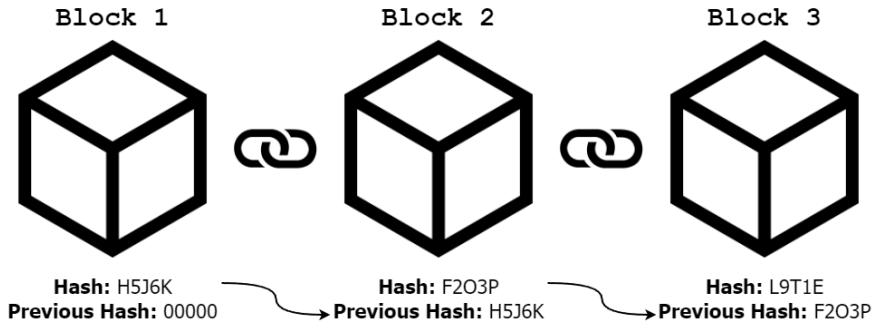


Figure 2.4: Visual representation of blockchain technology.

2.3.1 Transaction

A transaction on the blockchain is similar to any other transaction. When someone makes a physical transaction, there is an exchange of assets, e.g. coins. On the blockchain it is the same, the only difference is that the coins are electronic. An electronic coin is defined as a sequence of digital signatures. To transfer the coin to the next owner, the current owner digitally signs a hash of the preceding transaction and the public key of the succeeding owner and attaches it to the end of the chain. By examining the signatures, a payee can confirm the coin's chain of ownership [49].

Fungible tokens

These electronic coins can be either fungible or non-fungible, in the online community called tokens. Fungible tokens are uniform in which all units are identical and cannot be distinguished from one another [50]. Examples of this are Bitcoin and Ethereum, where one Bitcoin or one ether[1] holds the same value as others. This is similar to how money works in our society, one dollar bill has the same value as any other dollar bill.

Non-fungible tokens

NFTs however are different than fungible tokens, these are all unique. In NFTs, the value lies within the uniqueness and the value people put in them [50]. The most known use case for NFTs today are digital paintings, characters such as "Bored Ape" or in-game valuables. These are however not the only use case for NFTs, as digital copies of physical products create interesting applications [51]. Similar to paintings and antiques, the value of one does not equal another.

The reason why one NFT is different from others is that when they are minted[2], they get a unique identification number. Think that the first NFT is given the identification number 0, the second 1, etc. This makes it possible to differentiate the different tokens from one another. As these tokens also belong to a smart contract[3] with a unique contract address, one NFT with the identification number 0 will also be distinguishable from others with the same identification number [52].

2.3.2 Testnet

A testnet in blockchain technology refers to an instance of a blockchain that operates on the same or newer version of the underlying software. Its primary purpose is to provide a safe environment for testing and experimentation without exposing real funds or the main chain to potential risks [53]. Notably, testnet tokens are distinct

[1] Ether is the currency used on the Ethereum blockchain
[2] When a new token is created, it is called minting
[3] Smart contracts are described in the next section

from official (mainnet) tokens, hold no real value, and can be obtained freely [54]. By utilizing a testnet, developers can refine their blockchain-based applications with minimal risk, ensuring that they meet the required standards before deployment on the official main chain.

2.4 Smart contracts and the Ethereum blockchain

Ethereum is often referred to as "the world computer". Essentially, it is a deterministic state machine that is practically unbounded, comprising a globally accessible singleton state and a virtual machine that applies changes to that state. From a more practical viewpoint, Ethereum is a globally decentralized computing infrastructure that is open source and executes smart contracts programs. It leverages a blockchain to synchronize and store system state changes and utilizes a cryptocurrency named ether to measure and limit the costs of executing resources. By enabling developers to build potent decentralized applications with built-in economic functions, the Ethereum platform delivers high availability, auditability, transparency, and neutrality. It also minimizes or removes censorship and mitigates specific counterparty risks [45].

2.4.1 Decentralized application

A Decentralized application (dApp) is an application that operates autonomously on a blockchain or peer-to-peer network, typically through the use of smart contracts. Unlike centralized applications, dApps eliminate the need for a trusted centralized database to store the application state, removing the need for a specific centralized server to execute back-end logic. Instead, the program state runs on a virtual environment within the blockchain, with roughly half of all dApps hosted on the Ethereum blockchain [55] and running on the Ethereum Virtual Machine (EVM). This blockchain can be viewed as a state machine with strict rules for transitions, established by the smart contracts when the dApp is deployed, and for all practical purposes, unalterable. This creates a level of trust in the application that is not achievable in the current internet architecture, as the application owner cannot change their code.

Smart contracts play a significant role in dApp functionality. These contracts are computer code designed to execute automatically at significant events and represent a set of rules governing how the application operates. The logic in a smart contract primarily involves reading and writing data to the blockchain through actions such as buying or selling a token [5]. When smart contracts are deployed, they are deployed on an address similar to the address people have with their Ethereum accounts[4]. The address of this contract is deterministic and is the rightmost 160 bits of the Keccak-

[4]Their public key

256 hash of the nonce [5] and the address that deploys it, using RLP encoding [43]. If a smart contract implements NFTs, ERC-721 is the standard interface used [52]. This is an API for tokens within a smart contract and provides the functionalities needed for NFTs to have the properties we mention earlier.

2.4.2 Gas

An important aspect in the launching of dApps, execution of transactions, and movement of assets on the blockchain is gas. Gas is a quantifiable unit that measures the amount of computational effort required to execute particular operations within the Ethereum network [56]. There is then also a fee needed to be paid to have someone, e.g. a validator, to incentivize them to carry out these computations on your behalf, often referred to as gas fees. These fees are, as mentioned earlier, paid with ether. However, ether is further divided into a smaller unit named wei. One ether consist of 10^{18} wei [43]. As hundreds of millions of wei are used when carrying out these actions, they have again been grouped together to gwei, incorporating the unit prefix giga. One gwei equals 10^9 wei, which is then used denotation for gas prices. As of today, one gas equals 22 gwei[6] and one gwei equals \$0.00000173[7]. These has however fluctuated since the beginning of the Ethereum blockchain, which can be seen in Figure 2.5 [57] [58].

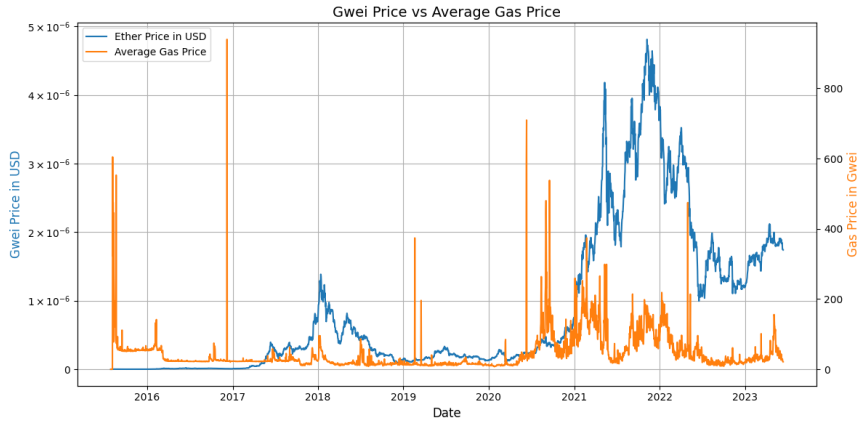


Figure 2.5: Historical representation of the gwei price and average gas price.

[5] Nonce is a scalar value equal to the number of transactions sent by the sender [43]
[6] As of 14.06.2023 21:00 GMT+2 on https://etherscan.io/chart/gasprice
[7] As of 14.06.2023 21:00 GMT+2 using https://coinmarketcap.com/currencies/ethereum/

2.5 Web3

Online services with the integration of blockchain technology and cryptocurrencies have begun to gain widespread acceptance. This movement is often referred to as Web3. The concept was introduced by Gavin Wood, a co-founder of Ethereum, where the idea is to facilitate the development of new online services without relying on trusted third parties, thus providing users with greater control over their data. A fundamental principle of Web3 applications is to leverage blockchain technology and infrastructure so users have complete control over user data instead of relying on big tech organizations which is the typical practice today.

Web3 seeks to be an improvement in privacy and security compared to its predecessors. According to Ethereum [1], one of the leading organizations in the field, there have been three iterations of the Internet. The first era of the World Wide Web is known as Web 1.0 which consisted mostly of static pages and minimal user interaction. We often call this period of the Internet "Read-only" [59]. Web 2.0 capitalized on improvements in Internet infrastructure, expanding its user base and transitioning from content provision to facilitating user-generated content sharing and enabling user-to-user interactions on platforms. It is therefore referred to as “read and write” [60] as users could easily read and contribute data to the Web. With more people coming online, a few leading companies started to dominate an inordinate share of traffic and value generated on the Internet. The era of Web 2.0 also saw the emergence of an advertising-based revenue model. Although users could generate content, they lacked ownership of it and didn’t gain from its monetization. Gavin put into words a resolution for an issue that many early cryptocurrency enthusiasts experienced: the Web demanded an unreasonable level of trust. The majority of today’s familiar and frequently used Web depends on placing trust in a handful group of private firms, hoping they would act in favor of public interests [1]. The solution is to move user data away from centralized servers and facilitating new applications and services that leverage blockchain technologies. The act of transitioning backend servers from centralized clouds to decentralized chains is a defining characteristic that sets Web3 apart from previous web paradigms. Web3 developers are, in fact, not obligated to create applications that depend on a single server for executing business logic or a centralized database for storing user data. Instead, Web3 applications deploy on decentralized networks such as blockchain platforms or distributed systems, which are hosted by multiple Peer-to-Peer (P2P) nodes.

It is hard to give Web3 a precise definition, but some core principles can explain its creation: *Trustless*, users have the ability to connect or exchange assets directly with unfamiliar users, eliminating the need for reliance on a trusted third party. *Permissionless*, user identities are not bound to any particular platform, and their activities are not linked to a central entity, such as a company. *Decentralization*, a

key advantage of decentralization in Web3 is increased availability, as it ensures that services are not reliant on a single server, thereby minimizing the risk of single points of failure[61].

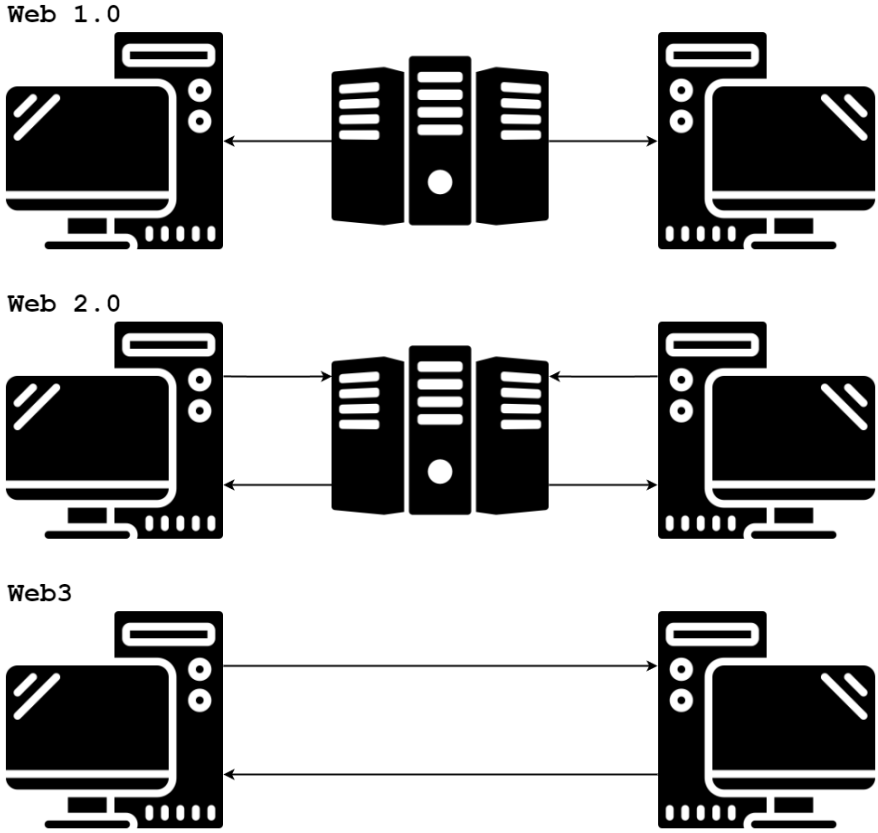


Figure 2.6: Simple representation of Web Iterations.

2.6 Related work

There are numerous papers investigating the benefits of blockchain technology and its implementation in various fields such as agriculture [62], imported foods [63], education [64] and healthcare [65]–[67] to mention some. Nevertheless, to the best of our knowledge, there does not exist any other PoC implementation utilizing Web3 as a solution to the problems in the reseller market in regard to secondhand goods. There are however some papers and companies that have done something in the same area.

2.6.1 Secondhand markets and blockchain technology

Shen et al. [68] wrote a paper where they researched the usage of blockchain in selling secondhand products. They examined the value of blockchain for disclosing secondhand product quality in a supply chain, with the usage of an online platform to resell the items. While they touch upon many similarities such as using the blockchain as a tool for identifying quality and item history, their research is more focused on the supply chain as a whole and on pricing and quality strategies, rather than the implementation itself. They also did not incorporate the cost of the usage of a blockchain, while this is an essential part of our potential solution.

In the study conducted by **Huang et al.** [69], the challenges within the secondhand market are addressed by introducing a framework incorporating a novel committee consensus mechanism. The authors illustrate the workflow of this framework, highlighting the utilization of smart contracts as a third-party agency on the permission-based Hyperledger Fabric ledger. Additionally, the proposed solution incorporates the InterPlanetary File System (IPFS) as a data storage component. However, it is important to note that the implementation does not involve any form of NFT or tokenization. The project aims to address issues such as transparency and traceability but does not target immutability or security concerns.

It is important to point out that we are not the first ones to utilize the Ethereum platform with Solidity and a library like Web3.js or Ethers.js to create a dApp . In 2021, **Panda and Satapathy** [70] wrote about investigating the possible usage of a solution like this. Numerous researchers are exploring the potential use of a Web3 solution in our society [71], but we have not found anyone with the same approach as ours. We are exploring potential areas where a Web3 solution could prove beneficial, while also developing a specific PoC tailored to address these identified needs.

There are papers looking into isolated parts of our paper, such as blockchain adaptation in combating counterfeits [72] or Web3 as an implementation in view of the blockchain [61]. However, we have yet not found anyone else who has investigated the combination of everything into one solution.

2.6.2 Existing solutions

There are some companies that have already launched platforms using NFTs to target the reseller market. **Tangible** [73] is a company that has built an ecosystem for tokenized real-world assets. They have essentially created a platform where you can buy an asset listed on their platform and receive a "TNFT", a Tangible NFT, which in turn is sent to your wallet. The item itself is sent to a warehouse for storage. While you are able to redeem the "TNFT" for the physical item at any time, the concept is more to buy luxury items that are stored and appreciated in value. To concretize,

while they do mint NFTs for their products, this is redeemed and disappears as soon as you want the physical item and does not help with reselling unless they keep the item in their vault.

Aura Blockchain Consortium [74] is an exclusive platform designed specifically for luxury brands aiming to combat counterfeit products through the use of NFT. This platform operates on the Quorum permissioned blockchain technology and is exclusively open to luxury brands across all sectors. Each brand must apply and supply its own product data to participate. On the contrary, our application is not restricted to luxury brands but accommodates a broader range of businesses and individuals. While Aura promotes the use of a single global blockchain for an improved user experience related to product authentication, our application operates on a public, permissionless Ethereum blockchain. This means that it is open to a broader range of businesses and individuals. Although both the Aura platform and our application use Ethereum-based technology and employ the ERC721 NFT standard for interoperability among Ethereum-based networks, our proposal offers a more open and decentralized approach. We aim to provide a permission based model for retailers, but which do not extend to buyers or other participants in the network, who can freely interact with the smart contracts. This is an important distinction and emphasizes that our system incorporates a degree of control to maintain integrity and trust, while it still maintains a higher level of openness and inclusivity compared to the Aura platform.

There are also existing solutions for the authentication of secondhand goods. **StockX** [75] focuses on reselling coveted items, such as sneakers, apparel, electronics, etc. where they are responsible for the authentication process of the item. While they do not utilize anything on the blockchain or have anything decentralized, they do offer a way for people to buy secondhand products and not be worried about counterfeits. However, this solution relies heavily on StockX as a third party and trust between the customer and them as a company and authenticator. A solution using this business model for all secondhand goods would require experts in every selling category as well as having a high quantity of these experts, to make sure there is not a backlog, thus, most likely infeasible as a global solution.

Work	**Retail/Secondhand**	**Blockchain**	**NFTs**	**Web3**
Inbook [70]	Neither	✓	X	X
Paper [68]	Secondhand	✓	X	X
Paper [69]	Secondhand	✓	X	X
StockX [75]	Secondhand	X	X	X
Tangible [73]	Retailer	✓	✓	X
Aura B.C. [74]	Both	✓	✓	X
This paper	Both	✓	✓	✓

Table 2.1: Summary of Related Work

Chapter 3
Requirements

In the context of our methodology, the requirements act as comprehensive set of specifications that outline the functionalities, features, and constraints that the application must adhere to. These serve as a blueprint for the development and help to ensure that the application meets the needs of its intended users. The requirements come from analysis of the state-of-the-art to point out participants' needs and expectations, as well as an assessment of the limitations and capabilities of the underlying technology, such as blockchain infrastructure.

3.1 Scope

In our objective to leverage Web3 technology in traditional reseller markets, we have analyzed the issues highlighted in our background research, identified and concretized these difficulties into distinct, quantifiable statements. These statements cover the significant issues in both markets, from cumbersome management of documentation and unethical practices in retail to concerns of transparency, trust, and authenticity in secondhand markets. By breaking these problems down in a methodical manner, we create a foundation that gives us a comprehensive insight into the core issues within these markets, and subsequently a clear scope for our project. As we move forward, this provides us with a roadmap that enables us to approach each issue systematically and guides us in designing an implementation.

3.1.1 Retail markets

- **Inconsistent and Non-standardized Documentation:**
    - Retailers use different methods when providing receipts, which can be cumbersome and have issues with long-term accessibility.
    - There is a risk of losing digital receipts, as they may get lost in different implementations or automatically deleted by email providers.

- Additional paperwork from services, repairs, or insurance claims further complicate the process.

– **Unethical Practices:**

- Instances of used products being returned and sold as new, deceiving customers about the product's status.
- Discarding and destruction of returned products have environmental implications and question the integrity of retailers.

3.1.2 Secondhand markets

– **Lack of Transparency:**

- Product descriptions are written by sellers themselves, which introduces subjectivity and ambiguity regarding the product's condition.
- Imbalance of information between the seller and buyer - sellers have an edge with more information about the product.
- Evaluating a seller's reputation is vital but can be misleading due to factors such as fake reviews or the possibility of creating new accounts to evade poor reviews.

– **Trust Issues:**

- Consumers have no built-up trust towards an unknown seller.
- The trust level can be influenced by the presence of third-party assurance seals, but these can be inconvenient or expensive.
- Consumers' purchasing decisions are highly sensitive to their level of trust in the seller, especially for more expensive products. Consequently, transactions for valuable items have a higher likelihood of not being completed.

– **Authenticity Concerns:**

- Secondhand goods may originate from unoriginal producers, raising doubts about their legitimacy and quality.
- Counterfeit goods have a high presence in the secondhand market, affecting a wide range of products, including luxury items and electronics.
- In regards to counterfeits, e-commerce platforms have few regulations and poor enforcement to prevent such activities. Consumers often encounter difficulties when attempting to report losses and seek compensation.

3.1.3 Using Web3 to target the problems

From our background research, we have identified several key Web3 principles that can be integrated into the existing market infrastructure, effectively addressing the problems mentioned in the retail and secondhand markets.

In the retail market, Web3 introduces the use of blockchain technology, which provides an immutable and transparent record of transactions. By leveraging blockchains, the issue of inconsistent and non-standardized documentation can be tackled. Receipts and transaction details can be securely stored on-chain, ensuring long-term accessibility and preventing any tampering or manipulation of records. This increased transparency and integrity foster trust between buyers and sellers, mitigating the risks associated with unethical practices.

Further, we can develop smart contracts which function as a trust and reputation system built on a decentralized network, which can address the lack of transparency and trust issues in the secondhand market. Through the utilization of technologies such as decentralized identity and digital wallets, buyers and sellers can establish and verify their accounts, creating a more trustworthy environment. Additionally, the implementation of reputation systems allows for feedback and ratings, empowering buyers to make informed decisions based on the experiences of others.

One significant aspect of Web3 is the use of NFTs to represent ownership of items. By utilizing NFTs, ownership is no longer tied to a specific retailer or secondhand market platform. Users can securely claim ownership of their items through their digital wallets, as the proof of ownership resides on the blockchain rather than relying on a centralized authority. This freedom allows users to utilize their proof of ownership across different platforms, ensuring the portability and accessibility of their assets.

Lastly, Web3 leverages permissionless blockchains, enabling the system to operate without dependence on a single entity or service provider. By utilizing a permissionless blockchain network, the risk of service disruption is minimized, ensuring that users can always access and manage their assets independently. Even if a specific retailer or platform ceases operations, users can retain control over their assets and continue to participate in the market.

3.2 Use case analysis

We use Freeman's definition [76] of stakeholders as "any group or individual who is affected by or can affect the achievement of an organization's objectives". According to this definition, stakeholders can include both those directly influencing the markets, such as retailers, and external actors and entities who benefit from the system, such

as secondhand sellers, repair shops, and buyers. To simplify the analysis, we can categorize the stakeholders into two main groups: Trusted Seller and Users. Trusted Sellers include both physical and online retailers that are integrated into the system. Meanwhile, users refer to individuals who have an interest in purchasing goods, whether new or secondhand, as well as individuals interested in selling secondhand items.

Stakeholders

The stakeholders used in our requirements are specified as follows:

Trusted Seller When specifying the physical or online retailer, which has been included in the system.

Users Individual buyers and individual sellers will be referred to as Users.

3.3 Application requirements

We need to specify a set of functional and non-functional requirements to further guide our development process. These requirements serve as the criteria used to evaluate the successfulness of our implementation. Functional requirements represent the fundamental actions that a system must perform. They describe what a system is supposed to accomplish in terms of specific behaviours or functionalities. On the other hand, non-functional requirements define the quality attributes or criteria that the system must meet or adhere to. Together, they provide the necessary detail and context for the design and implementation stages, helping to ensure that our solution is robust, reliable, and capable of effectively addressing the identified key challenges.

3.3.1 Functional requirements

Functionality	*Description*
Purchase Item and Receive NFT	**Users** should be able to purchase items from a **Trusted Seller**, both physical and online, and in return receive both the actual item and the corresponding NFT. This NFT serves as a digital proof of purchase and ownership, storing relevant information such as name, price, category, brand and serial number.
Overview of NFTs	**Users** should have access to an overview of all their NFTs and the associated information for potential resale.
Access Product Information	Other **Users** should have the ability to look up product details simply by referencing the NFT. This includes data such as item description, original purchase date, history of ownership, and potential warranties or services linked to the item.
Execute Transaction and Transfer NFT	During a transaction, it should be possible for the **User** to safely transfer the NFT to another **User**. This NFT transfer validates the change of ownership, ensures that the digital record aligns with the exchange of the product, and updates with the new information from the transaction.
Rate a Transaction	After the completion of a transaction, the **User** who bought the item should be able to rate their experience. This review will be open for all to see and follow the **Trusted Seller** or the **User** who sold the item across different e-commerce platforms.
Remove a Broken Product	In the event of a product becoming worn out or broken, **Users** should have the option to remove its corresponding NFT from their ownership. This ensures that **Users** are not bothered by old NFTs.

Table 3.1: Functional requirements and their description.

3.3.2 Non-functional requirements

Scalability Given the volatile and potentially extensive nature of the reseller market, the system needs to be designed to handle an increasing number of transactions and users without significant performance degradation.

Robustness The smart contracts should be able to handle unexpected situations or inputs gracefully without causing unintended consequences which damage the plat-

form's integrity. The NFTs created and managed within the system must be designed to maintain their existence indefinitely. This implies that under no circumstance should an NFT spontaneously disappear or get destroyed unintentionally.

Security As the system deals with digital copies representing valuable goods, security measures need to be in place to prevent fraudulent activities and protect user data. Ensuring secure storage and identity verification is crucial for protecting the value and integrity of the digital assets represented by NFTs. These include:

- Only the trusted system administrator should have access to add retailers to the system.
- Only trusted retailers should be able to mint NFTs.
- Only the owner should have access to transfer ownership.

Reliability The system should be able to operate for all users whenever they need it. This means high availability and low downtime.

Performance The system must be designed to provide swift responses to user requests and manage transactions efficiently. This includes optimising the smart contracts regarding resource usage and gas cost, contributing to a prompt transaction with minimal cost for the users.

Usability To ensure widespread adoption, the user interface must be intuitive and easy to use, even for individuals unfamiliar with NFTs or digital transactions. Clear instructions and a simple workflow for buying, selling, and transferring NFTs should be implemented. It should be easy to get an overview of all owned NFTs.

Upgradability The smart contracts should have a mechanism for upgrading their logic, enabling the addition of new features or fixing bugs without requiring the redeployment of the whole system anew.

Privacy The smart contracts should respect user privacy as much as possible, minimizing the amount of personal information accessible to everyone.

Chapter 4
Design and Architecture

In Chapter 3, we established the framework conditions and identified the main stakeholders and their individual needs. This chapter aims to design a solution to solve the identified problems. First, we provide an overview to give an insight into how our solution will work. We then cover design choices and present the suggested system architecture with the 4+1 model view.

4.1 System Overview

Before diving into the technical part, we will present a system description. This section will explain how the system works in interaction with the stakeholders. We also describe the choice of blockchain and its impact on our design.

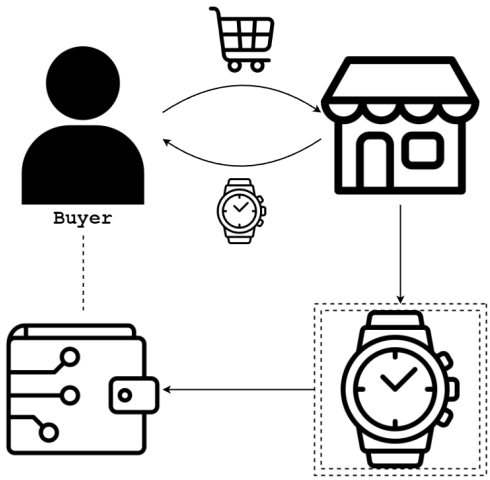


Figure 4.1: Modelling of how a user purchases an item, a watch, in the retail market. The retail seller produces an NFT, which transfers the NFT ownership to the buyer's crypto wallet.

Users typically receive proof of payment when they buy a product in a shop. In our solution, a purchase at a store should trigger the generation of an NFT that belongs to the effect that was just purchased. The user who has just paid will receive an NFT corresponding to the product in question. Figure 4.1 illustrates the user interaction during a purchase transaction. After the transaction, the retail store mints an NFT representing the purchased item and transfers the ownership to the user's digital wallet. This NFT is proof of ownership for the specific product. Further, on the secondhand market, a user sells an item to another user. In this process, the selling user also transfers the digital copy belonging to the article. Figure 4.2 illustrates the trade between two users involving a secondhand item. As the seller hands over the item to the new owner, they also transfer the corresponding NFT, thereby establishing proof of ownership for the new owner of the secondhand product.

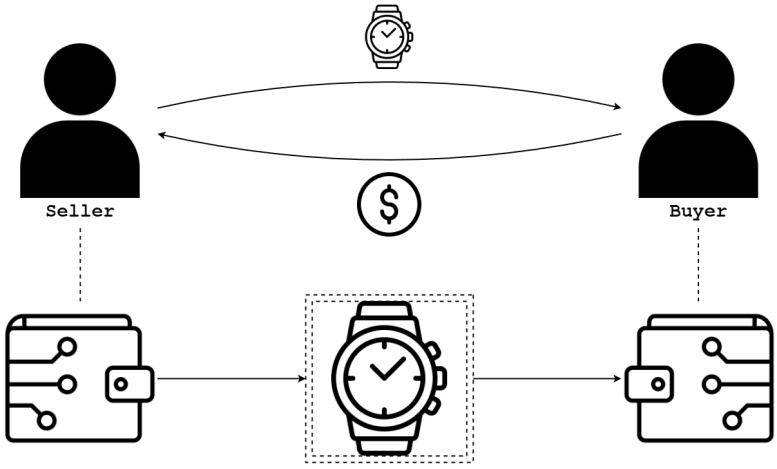


Figure 4.2: Modelling of how a seller sells an item, a watch, in the secondhand market. As a part of the transaction, the seller transfers the NFT ownership to the new owner's crypto wallet.

Owning an NFT within our system promotes security and trust between users and retail businesses. Figure 4.3 demonstrates how a user can verify the authenticity and ownership of a product before making a purchase. This verification process ensures that the item is genuine and not counterfeit or stolen. Users can confidently confirm that the secondhand seller possesses the relevant item, adding an additional layer of trust to the transaction.

While the system is primarily designed as an Application programming interface (API) integration into physical transaction systems and various reseller platforms online, it also provides a visual user interface accessible through a web browser. This interface simplifies the management of NFTs, allowing users to easily view, transfer,

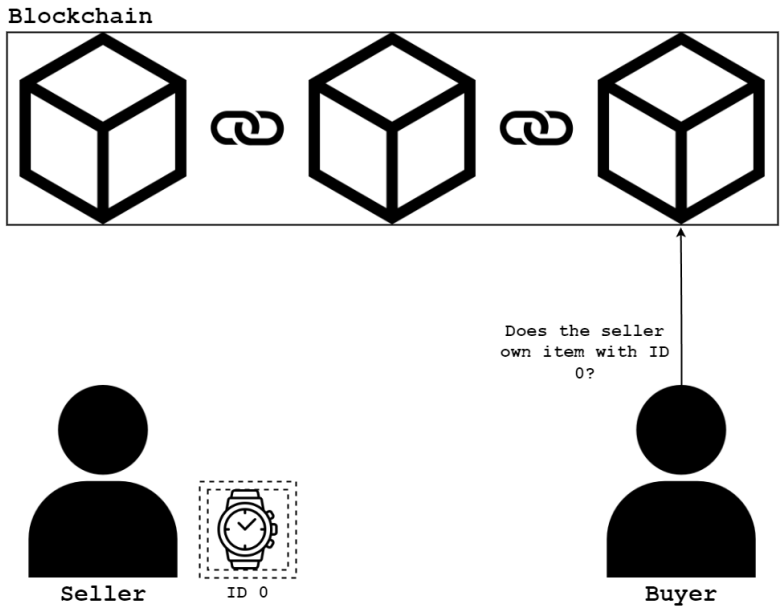


Figure 4.3: System modeling of a verifying process against the blockchain.

and interact with their digital assets.

A critical design choice is the selection of a blockchain network. First, the blockchain must allow smart contracts. We can implement complex system logic within smart contracts executed on the blockchain. Secondly, we must ensure our users with high availability and not risk any downtime or loss of data. To provide a high level of reliability and transparency to all users, we have chosen the Ethereum blockchain. Ethereum's open nature as a public transaction database aligns with our objectives. The blockchain imposes no access restrictions on data reading and writing, promoting transparency and inclusivity. Additionally, our smart contracts, which incorporate the core logic of the dApp, are publicly accessible, enhancing the transparency of the contract execution process. Considering the security requirements outlined in section 3.3, the system should ensure availability. The Ethereum blockchain is, by nature, resilient against faults and attacks and minimizes the risk of downtime or compromise of the backend system responsible for processing transactions. This feature adds an extra layer of reliability and stability to our application.

Ethereum is publicly available to anyone that has an internet connection. If we launch our NFT system without restrictions, it will result that anyone can create NFT, and the system will collapse. While the Ethereum blockchain allows anyone to mint

new tokens, our system restricts the minting process to a select group of authorized retailers. This measure ensures the authenticity and credibility of the platform, preventing the unauthorized creation of NFTs and maintaining the integrity of each token as proof of ownership. Additionally, Ethereum wallets establish pseudonymous identities that do not link directly to real-world users. To prevent the same individual from creating multiple accounts, meanwhile maintaining user privacy, we implement a user list that verifies the uniqueness of users while preserving their pseudonymity.

By utilizing a crypto wallet, individuals can establish their possession of a specific item, thereby facilitating the verification of ownership. This feature offers convenience for potential buyers in the secondhand market or repair shops, as they can confidently verify the legitimacy of the item in question. Moreover, the NFT serves as a digital twin for the physical product, representing the asset's unique qualities and history on the blockchain. This digital copy, once minted, is an immutable proof of ownership that cannot be modified or destroyed, providing a reliable and transparent track record for the product. This transparency is particularly beneficial for high-value or collectable items, where provenance and authenticity are paramount.

4.2 System architecture

In this section, we will explore the system architecture to illustrate how the system is designed. We employ the 4+1 model view[77] to analyze the system, which provides a comprehensive perspective on its various aspects and functionalities. The 4+1 view model constructs the architecture into five distinct perspectives. Each focused on a specific issue. "4+1" is used because it consists of four views around one central idea. Here's a brief overview of each view: Logical view, development view, process view, physical view and scenario view.

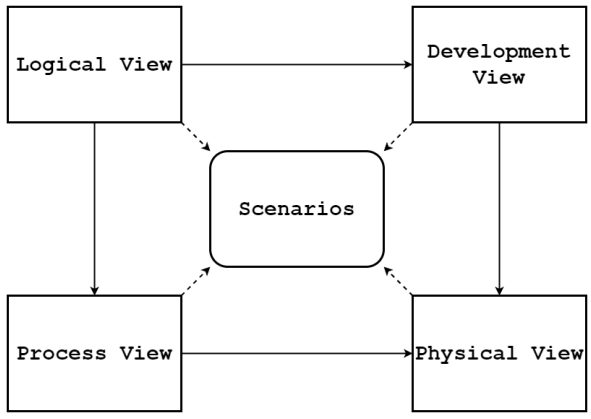


Figure 4.4: 4+1 architecture model.

4.2.1 Logical view

The logical view of our smart contracts demonstrates how the system fulfils its requirements by outlining the interactions between stakeholders and the corresponding tasks they perform. Figure 4.5 presents a class diagram illustrating these interactions, which we will further describe below.

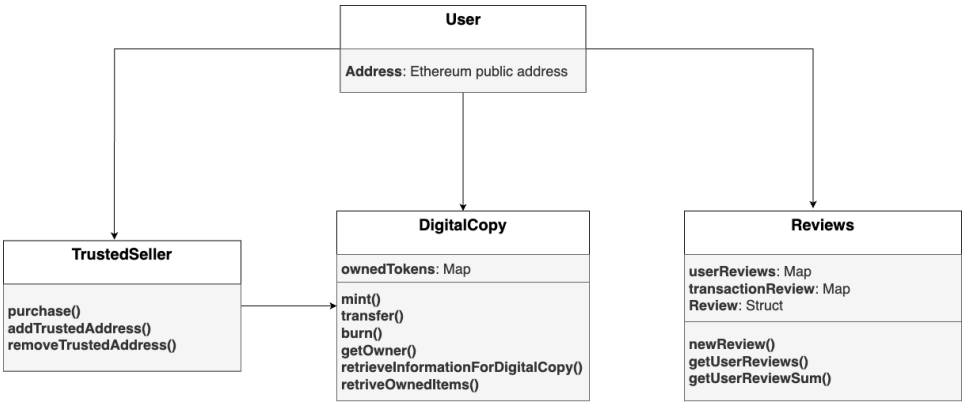


Figure 4.5: Logical view of the interactions between stakeholders and the corresponding tasks they perform in our proposed architecture.

User: To interact with the system, it is required to have an identity in a blockchain system. On the Ethereum blockchain, we handle this by assigning a unique address to every user. The user class has only one property which is an Ethereum address.

Digital copy: This is a digital representation of a physical item. This entity is responsible for creating and managing digital copies and can also be used to access product information stored on the blockchain. The creation of a digital copy is done through the mint function. The mint function takes in relevant information about the product that will be baked into the NFT. The transfer function handles transferring of ownership of a digital copy from one user to another. When a product is broken or destroyed and we need to delete the corresponding digital copy, the burn function permanently removes the digital copy from circulation. This entity can also provide users with their NFT. The retrieveInforamtionForDigitalCopy function returns information about a specific digital copy.

Trusted Seller: The system is also used by retailers, which we call trusted sellers. Each trusted seller is responsible for generating the digital copy NFT and transferring ownership to the buyer after completing a purchase transaction. This is done with the

purchase function, which inputs the user's address. The purchase function allows authorized retailers to mint new digital copies and transfer them to a buyer immediately.

Reviews: Designed to manage user reviews on the blockchain. Allows users to request reviews and scores of other users and retail businesses. The entity will review previous trades and enable buyers to rate transactions after a trade. The user can use the newReview() function to add reviews for trades it has done.

4.2.2 Development view

The development view offers a depiction of the system architecture from a developer's viewpoint.. It contains internal conditions tied to the simplicity of development and limitations set by the programming language or toolkit. [77]. We have included a component diagram in Figure 4.6 to present our proposed system architecture from the development view. A textual description has also offered a more comprehensive understanding of the solution.

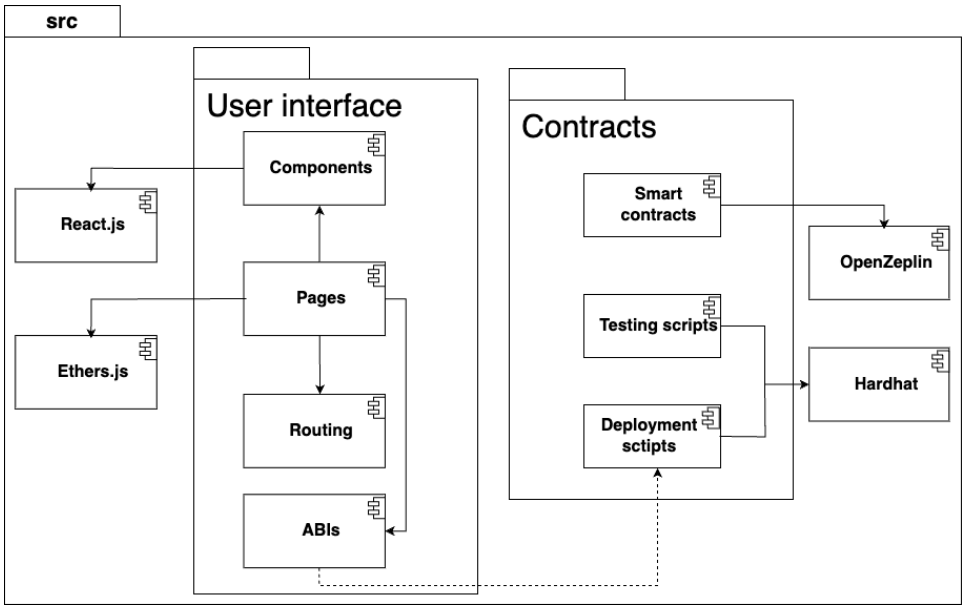


Figure 4.6: Repository representation of the proposed architecture from the developers view.

Repository

Our application will be organized as a mono-repo, consisting of two distinct modules: one for the front-end code and another for the smart contracts. Each module will maintain its own set of dependencies to avoid any potential conflicts or unnecessary dependency issues between the modules. This approach ensures a clean and isolated development environment for the front-end and smart contract components, allowing for efficient dependency management and minimizing potential compatibility issues.

User interface

All interfaces within the smart contracts will be accessible through `HTTP/RPC` calls. However, a dedicated user interface will be developed to enhance the user experience and interaction with the system. This user interface will facilitate the request and reception of API calls to other components.

Ethereum blockchain

Ethereum consists of a large and active developer community that can be a resource for finding help, support, and collaboration opportunities as we build this Web3 solution. Ethereum is also one of the oldest and most mature blockchain platforms. This maturity shows that the Ethereum project ensures stability and longevity.

Ethereum has well-established token standards (e.g., ERC-20 for fungible tokens and ERC-721 for non-fungible tokens) that have been widely adopted by the Web3 community. By leveraging these standards, we can provide compatibility with a wide range of other projects, services, and platforms. This standard defines asset ownership and includes methods for transferring ownership, enabling the verification, trading, and selling of items on marketplaces. As a result, the ERC721 interface provides a powerful tool for facilitating secure and efficient transactions of unique assets on the Ethereum blockchain.

Data storage

Data storage is necessary for storing information about each item. The chosen data storage solution must be accessible, cost-effective, and resistant to downtime. We mainly considered two options to store data: *Off-chain* and *on-chain storage*.

Off-chain storage involves storing data outside of the Ethereum blockchain. This approach is more cost-effective and scalable compared to on-chain storage. However, using centralized databases contradicts the decentralized nature of Web3 and may introduce single points of failure. Another storage solution is *Decentralized storage* like IPFS or Filecoin which can be used for storing data in a decentralized manner.

These will introduce unnecessary complications to our PoC and will therefore not be considered.

On-chain storage means that data can be stored on the Ethereum blockchain itself. This data can be accessed by anyone who is connected to the Internet. The data is stored in a decentralized manner, meaning it is not controlled by any single entity. This makes it difficult for anyone to tamper with the data or manipulate it in any way. It also provides a very good reliability feature since it does not risk any downtime. The downside of such a solution is the high cost of writing new data and is not considered an effective way of storing data. However, since the system only needs to store small chunks of data and to simplify our implementation, we will use the blockchain itself as data storage. In addition, there exist optimization solutions like Layer 2 protocols, such as rollups or sidechains, that store and process data off the Ethereum main chain while maintaining security and decentralization.

Deployment strategy

The software is built and deployed in two stages. First, we compile smart contracts and generate associatedApplication binary interface s (ABIs). The ABIs are then moved to the front-end module. Secondly, we build the front-end application and deploy it to a website service, and we deploy all smart contracts to the Ethereum network with a deployment script.

4.2.3 Process view

The process view in the 4+1 model provides a detailed understanding of the system's interactions between actors and components. These interactions can vary based on the specific actions performed by users, such as buying or selling an item. To showcase these different scenarios, we have selected two distinct situations to illustrate the process flow and interactions within the system.

Figure 4.7 provides an insight into the underlying process when a user purchases at the retail market. In this scenario, the retailer is recognized as a trusted seller, granting them the authority to mint an NFT and transfer it to the customer. Let us break down the steps involved: In step 1, the user takes the initiative and purchases an arbitrary product. The retailer must perform a lookup to validate whether the customer is a registered user. This verification process facilitates the User contract, which ensures the user's existence at a specific Ethereum address.

Next, the retailer initiates a function call to the DigitalCopy contract, requesting the creation of a new token. To confirm the seller's authority, the DigitalCopy contract reviews the SystemManager contract, verifying whether the retailer is an

approved entity. This ensures the creation of NFTs to only authorized individuals or entities, preventing unauthorized parties from creating NFT.

Once the retailer is accepted, the DigitalCopy contract generates a unique token, represented as an NFT, and designates the seller as its owner. This token will serve as a representation of the purchased product. To complete the transaction, the retailer transfers the ownership of the token to the customer's address. As a result, the customer now possesses an NFT corresponding to the purchased product.

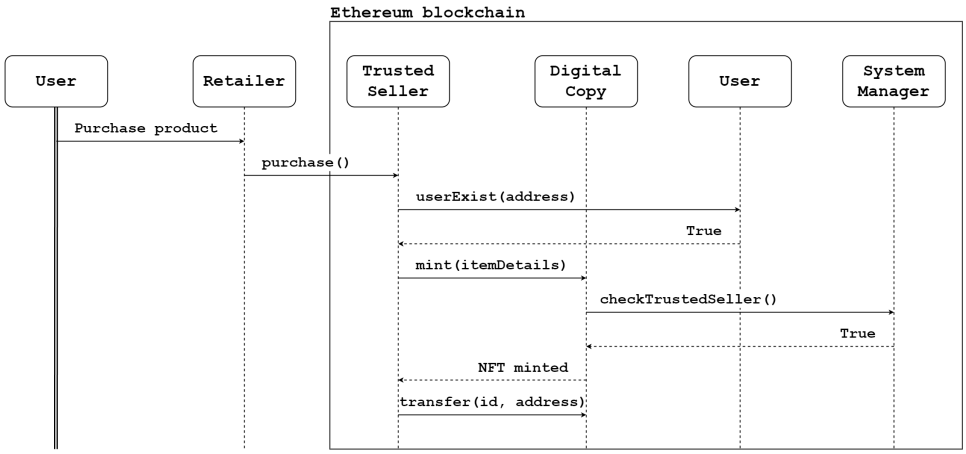


Figure 4.7: Visualization of the sequential steps involved in a user's purchase from a retailer, with user existence, verification and token creation.

In Figure 4.8, we observe the process of transferring an already minted NFT to a new owner, typically occurring during a sale in the secondhand market. The system facilitates the verification of the seller's possession of the relevant product before the trade takes place.

The process begins with the secondhand seller initiating the user interface to display all the items they possess. The user interface within a web browser communicates with the DigitalCopy contract on the blockchain, requesting a return of all tokens associated with the given user address. The response from the DigitalCopy contract provides a comprehensive list of all the tokens owned by the user within the system. It is necessary to note that this system exclusively showcases products from trusted retail businesses, as only these businesses can mint NFTs on the network.

Next, the seller places the desired item for sale, triggering the corresponding digital copy token to transition into "for-sale" mode. Now, all users can access product details and verify the ownership of the token.

In the following step, a potential buyer enters the scene. The buyer interacts with the user interface, specifically requesting information about the owner of a particular token. To fulfill this request, the user interface communicates with the Digital contract on the blockchain, which promptly retrieves and provides the current owner's address.

This process serves a crucial purpose. It empowers users who wish to purchase secondhand goods the ability to ensure the authenticity of the products and mitigate the risk of purchasing stolen items. Consequently, this verification mechanism greatly benefits buyers, particularly when engaging in transactions through online platforms. By establishing trust and transparency, verifying token ownership creates a safer and more secure trading environment, fostering buyers' and sellers' confidence and peace of mind.

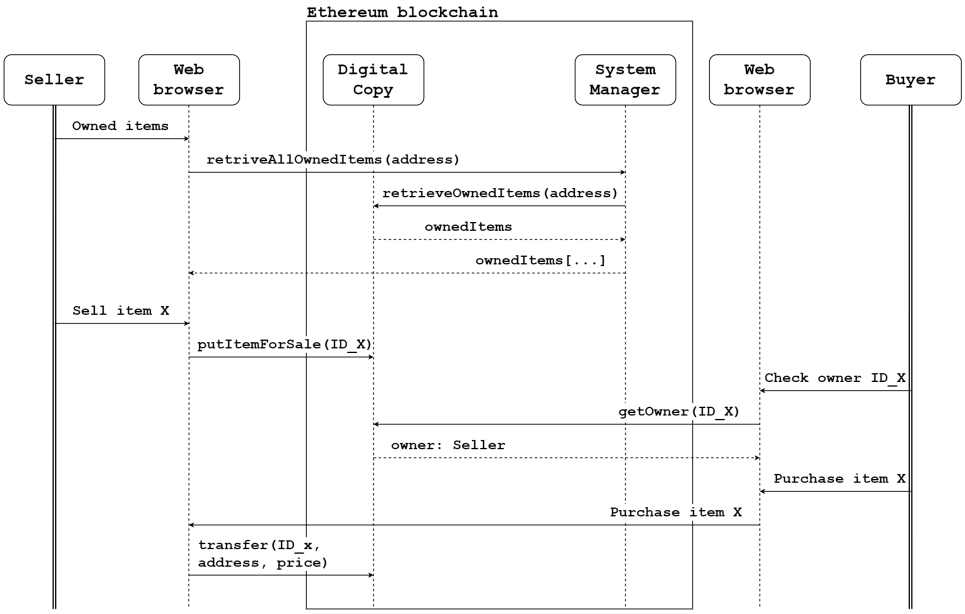


Figure 4.8: Visualization of the sequential steps involved in secondhand trade, where one user sells an item to another user.

4.2.4 Physical view

The physical view describes the interconnection between the software and hardware components of the application, as well as the operational aspects of the system upon deployment. Figure 4.9 represents the Unified Modeling Language (UML) deployment diagram, illustrating the application's architecture.

CDN network: We will utilize a CDN network to host our static website files. This network will efficiently deliver the required HTML, CSS, and JavaScript files to the user's web browser, enabling interaction with our distributed system through a front-end application.

Ethereum Node: To store smart contracts and maintain an accurate system state, we employ the public blockchain of Ethereum. It was a requirement to restrict access to specific acts., which led us to deploy in a hierarchical tree network. Within this network, the "SystemManager" contract operates independently, while the "TrustedSeller", "DigitalCopy", "Review", and "Users" contracts are dependent on it.

1. **SystemManager Contract**: This contract maintains the list of trusted sellers that are authorized to mint and trade digital copies and deploys the **Users** and **Reviews** contracts. It includes functionalities to add or remove an address from the trusted seller's list. Additionally, this contract facilitates the deployment of new DigitalCopy contracts, keeping track of all deployed instances. Further, it also provides functions to show all DigitalCopy contracts it has deployed and to check whether a certain address exists as a Digital-Copy contract. Lastly, there is also a method to retrieve all owned items for a specific owner and a method to stop specific DigitalCopy contracts from minting.

2. **TrustedSeller Contract:** This contract defines a trusted retailer's operations in the system. It allows a trusted retailer to sell an item, which entails minting an NFT and transferring it to a buyer. If any items are returned, the contract maintains a list of items it has sold to ensure that no items have multiple digital copies. Moreover, trusted sellers can add or remove addresses from their trusted address list, which allows for managing their internal operations (like multiple wallets for a single seller). It also allows a trusted seller to change the associated DigitalCopy contract and show the currently associated DigitalCopy contract.

3. **DigitalCopy Contract:** This contract deals with the creation, management, and ownership tracking of the digital representation of a physical object. Its essential functions include minting new digital copies, transferring ownership of digital copies, querying the owner of a specific digital copy, burning digital copies, and retrieving all digital copies owned by one particular address. The minting process is restricted to trusted sellers only, thereby enforcing the trust and authenticity of the NFTs in the system.

4. **Users Contract:** This contract manages user information and maintains a list of all the users in the system. The main tasks of this contract are to create new users and check for existing users.

5. **Reviews Contract:** This contract offers the opportunity for buyers to review transactions and, by extension, rate sellers. Although the contract is integrated into our implementation and is deployed alongside the SystemManager contract, it holds a somewhat secondary role. It is not crucial for the functionality of the other contracts, except when the SystemManager contract needs to access the reviews. Since leaving a review is optional, this contract does not play a critical role in the system's day-to-day operations. Nevertheless, it is vital as it archives all transactions that have been reviewed and the reviews themselves.

Each contract performs a distinct function in the application, and they work together to provide security and trust in the marketplace for trading digital representations for items. The DigitalCopy contract defines the core functionality of creating and managing NFTs, the SystemManager contract secures the integrity of the marketplace by controlling who can mint NFTs, and the TrustedSeller contract handles the interactions of trusted sellers within the system. The Users contract manages the user accounts and the Review contract does the same for reviews.

Web-browser unit: The web browser interacts with the Ethereum blockchain using the ethers.js API, enabling smooth communication and data exchange.

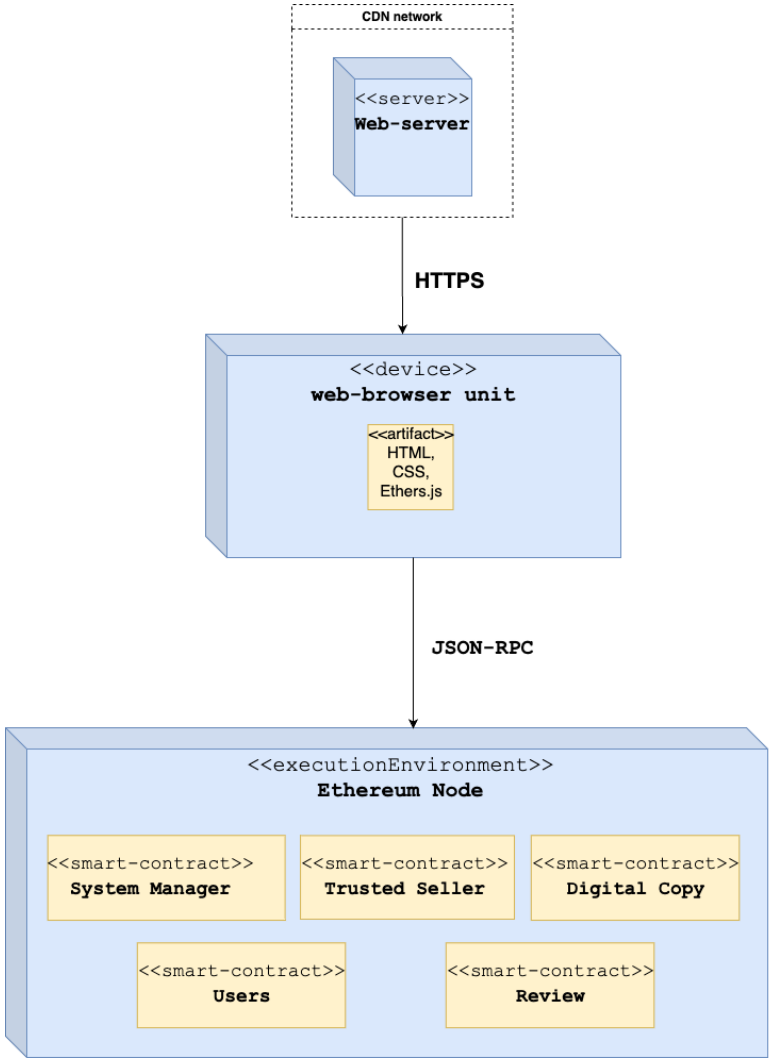


Figure 4.9: UML deployment diagram of the proposed system.

4.2.5 Scenarios

The scenarios presented in the 4+1 model view comprehensively explore the system's behavior and functionality from different perspectives. These scenarios capture fundamental user interactions and system responses, providing a deeper understanding of how the system operates in various real-world situations. By examining these scenarios, we can gain valuable insights into the system's performance, reliability, and adherence to requirements. Each scenario sheds light on specific aspects of the system's behavior, enabling us to evaluate its effectiveness in meeting the needs of stakeholders and achieving the desired outcomes. Through these scenarios, we aim to illustrate the system's functionality, highlight potential challenges, and identify opportunities for improvement.

Scenario 1: NFT Minting and Duplicate Prevention

In this scenario, Kari Olsen purchases two watches from the retailer Trusted Watches, established during deployment. Trusted Watches then attempts to sell a watch for which the NFT has already been minted. We will examine how Trusted Watches handles the attempted resale of a minted NFT. This scenario aims to prevent duplicate NFT creation and maintain the integrity of the NFT system.

Scenario 2: Listing, Selling, and Ownership Transfer

Initially, Ola attempts to retrieve information about an item that is not currently available for sale. Following this, Kari lists the item for sale, and now Ola can gather information about it. Kari successfully sells the watch to Ola. Subsequently, Kari attempts to purchase the watch back at a cheaper price, but this attempt fails as Kari no longer holds ownership of the corresponding NFT. We will observe the smooth process of listing, selling, and transferring ownership of the item.

Scenario 3: Review and Rating System

The scenario begins with Ola attempting to write a review for a product different from the one he purchased. He then proceeds to leave a review for the actual product he bought. As Kari seeks to sell another item subsequently, it is evident that her rating has been influenced by Ola's review. Later on, when Kari decides to put another item up for sale, we can see that her rating has been influenced by Ola's review. We will analyze how this review impacts Kari's rating and observe the overall functioning of the review system.

Scenario 4: NFT Deletion

In this scenario, Kari makes an unsuccessful attempt to destroy Ola's NFT. Later on, due to damage beyond repair, she successfully carries out the burn operation of

her NFT. This scenario allows us to evaluate the effectiveness of the NFT deletion process and its role in maintaining the system's integrity.

Chapter 5
Implementation

After setting up our system requirements, designing our system architecture, we now move into the implementation phase. We'll first discuss the smart contracts, detailing the technologies we've used and the implementation of each contract along with their functionalities. Next, we'll explore the implementation of the web application, focusing on the technology choices for the front-end, the development process, and its practical use. Lastly, we'll explain the scripts we've developed to deploy the smart contracts locally and on the Sepolia testnet.

5.1 Smart contract implementation

5.1.1 Technology

Solidity

Solidity is a procedural, statically-typed programming language and the desired language for writing smart contracts on the Ethereum blockchain [45]. It was designed to leverage the EVM and facilitates the creation of dApps.

Functions

Functions in Solidity are the contracts' primary components and represent the contract's overall functionality. They contain the executable code within a contract and can be used to retrieve information, change the state of a contract, perform computations, change stored data and interact with other contracts. Functions may take parameters as input, process that data and then can return an output.

Input parameters are the values that a function takes when it is called. They are passed to the function when it is executed and are necessary for the function to have the information needed to perform the operation(s).

In Solidity, functions need to have a visibility type that determines who can call the function. These visibility types are the same for variables in the contract and act

as a barrier to visibility. The visibility types in Solidity are:

- **Public** If a function is marked as public, it can be accessed from outside the contract, from within the contract, and from derived contracts.
- **Private** If a function is marked as private, it can only be accessed within the current contract. No other contracts can access the function.
- **Internal** If a function is marked as internal, it can be called from within the current contract and any contracts inherited from it. No external contract can access it.
- **External** If a function is marked as external, it can only be called from outside of the contract. For all intents and purposes, it can not be called from inside the current contract.

As the visibility types public and external have the same gas cost, and for our purpose, there is no need for the extra restrictions from external, we have chosen to only use public. Internal is also not used as we have no inheritance in our code.

Functions may be of a view type, meaning they are read-only and cannot change state variables. Such functions do not have any gas cost when executed. Functions can also return values, and the caller can then use these values.

Events In Solidity, events are inheritable members of contracts. Events provide a way for smart contracts to produce a custom output that other code can interact with. When implemented inside functions, when the function is called, this output will be stored on the Ethereum blockchain. These can then be used to confirm transactions, store data or trigger actions outside of the smart contracts.

Modifiers and Requires Modifiers and requirements in Solidity provide a way to check input conditions and the smart contract's state. They are requirements that will be inspected and verified before a function is allowed to execute. In the case where a condition inside the requirement is not met, an error is thrown and the execution of the function is stopped.

Hardhat

Hardhat is an Ethereum development environment that has allowed us to deploy contracts, run tests and debug the Solidity code. We have used Hardhat to compile our smart contracts, deploy them on a local network and then interact with them as if they were deployed on the actual Ethereum network. This has allowed us to write code, test it out and make necessary changes without having to spend any ether.

5.1.2 Smart Contracts

As mentioned in Section 4.2.4, our implementation has five smart contracts which make up our system. These are:

- **SystemManager**
- **TrustedSeller**
- **DigitalCopy**
- **Reviews**
- **Users**

As it would be too much to include all the code in all the smart contracts, we have collapsed all the functions to only show their name, input parameters, visibility type, modifiers and return parameters. This is done since the code is several hundred lines long and would be impractical to include here. We will be explaining all the important parts of what happens in the functions, and the whole code can be found in an open repository on Github[1].

SystemManager

The initial smart contract, **SystemManager**, in Listing 5.1 starts by establishing two private mappings[2]. The first mapping essentially forms a list of trusted sellers, while the second keeps track of all the deployed **DigitalCopy** contracts. The latter ensures an overview of the number of such contracts currently operational. These are made private so only this contract can directly access or modify them.

Next, we focus on the contract's constructor[3] on line 23. The constructor sets the address of the deployer as the 'ownerAddress'. Furthermore, the constructor triggers the deployment of the **Users** and **Reviews** contracts. Importantly, the addresses of these contracts are emitted as events, which is useful for future interactions.

Moving on, the contract features 'add' and 'remove' functions that handles the inclusion or exclusion of **TrustedSeller** contract addresses in the system. The change in the state of approval are then announced through the events 'AddedTrustedSeller' and 'RemovedTrustedSeller' respectively. To ensure security and control, these functions are tagged with the 'Authorized' modifier, which ensures that only the owner of the **SystemManager** contract can execute them.

[1] https://github.com/Autentisk/
[2] In simple terms, a mapping is like a dictionary; you store and retrieve data using a specific key.
[3] A constructor is a unique function that initiates when the contract is deployed

On line 26, the function performs the gas-free operation of verifying if a retailer exists on the list of trusted retailers. On the next line, there is the 'deployDigitalCopy' function, which, restricted by the 'Authorized' modifier, only permits the deployment of a new **DigitalCopy** contract by the contract owner. Each deployment is announced via the 'DeployedDigitalCopyContract' event. In addition, the next function has the ability to change if a specific **DigitalCopy** contract has the ability to mint more NFTs. This provides the ability to stop the production of NFTs in case of a compromised retailer and assess the situation and is understandably restricted by the 'Authorized' modifier.

The subsequent view function returns a list of existing **DigitalCopy** contract addresses and provides an overview of the existing **DigitalCopy** contracts. Another gas-free function follows, confirming the existence of a **DigitalCopy** contract address within the system. This makes it possible for a user or a system to verify that a smart contract is legit before making a transaction.

The penultimate function, 'retrieveAllOwnedItems', is a view function that generates a two-dimensional list of all items owned by a user. This provides a summary of all owned items across all possible **DigitalCopy** contracts and is only accessible to the owner of the items. Lastly, the concluding function is gas-free and gives a user all relevant information required to facilitate the sale of an owned product, thereby simplifying the process when planning to list an item for sale. In addition to information about the product, it also returns the rating of the seller. This function is only possible for the owner of the product to call, making it impossible for anyone to figure out the rating of a user outside of the sale of a product.

```
1  pragma solidity ^0.8.17;
2
3  import './DigitalCopy.sol';
4  import './Reviews.sol';
5  import './Users.sol';
6  import "hardhat/console.sol";
7
8
9  contract SystemManager{
10
11     // Variables:
12     mapping (address => bool) private SystemManager;
13     mapping (uint256 => address) private mintedDigitalCopiesMapping;
14     uint256 private mintedDigitalCopyCount;
15     address private ownerAddress;
16     Users private users;
17     Reviews private reviews;
18
19     // Events:
20     event DeployedUsersContract(address indexed contractAddress);
21     event DeployedReviewsContract(address indexed contractAddress);
```

```
22      event DeployedDigitalCopyContract(address indexed contractAddress);
23      event StatusMintable(address indexed contractAddress, bool indexed
            mintability);
24      event AddedTrustedSeller(address indexed addedAddress);
25      event RemovedTrustedSeller(address indexed removedAddress);
26
27      // Modifiers
28      modifier Authorized() {...}
29      modifier ExistingDigitalCopies() {...}
30
31      // Functions
32      constructor() {...}
33      function add(address _address) public Authorized {...}
34      function remove(address _address) public Authorized {...}
35      function checkSystemManager(address _address) public view returns (
            bool) {...}
36      function deployDigitalCopy() public Authorized {...}
37      function showDigitalCopies() public view ExistingDigitalCopies
            returns(address[] memory) {...}
38      function statusMintable(address _address, bool _status) public
            Authorized{...}
39      function digitalCopyAddressExist(address _address) public view
            ExistingDigitalCopies returns(bool) {...}
40      function retrieveAllOwnedItems (address _address) public view
            ExistingDigitalCopies returns(uint256[][] memory) {...}
41      function retrieveListingInformation (address _address, uint256
            _digitalCopyID) public view returns(DigitalCopy.
            digitalCopyInformationRetrival memory, uint256, uint256) {...}
42  }
```

Listing 5.1: Collapsed SystemManager contract

TrustedSeller

Listing 5.2 shows the collapsed code of the smart contract **TrustedSeller**, and begins with the implementation of the IERC721Receiver interface on line 7. This is important for it to be able to own NFTs as a smart contract.

On line 10-11 private mappings, 'trustedSellerAddressMapping' and 'mintedDigitalCopiesMapping', are declared. The former keeps a record of addresses that are allowed to execute functions acting on the retailers' behalf, while the latter keeps a record of minted NFTs by the retailer. Then on lines 12 and 13, a reference to the **DigitalCopy** contract and the **SystemManager** contract is declared respectively. These private variables ensure that only functions within this contract have the authority to modify them.

Moving on to functions, the contract's constructor is found on line 24. It sets the name of the trusted seller and the address of the **SystemManager** contract. The address which initiates the contract is added to the 'trustedSellerAddressMapping'

and thus the only address allowed to act on its' behalf initially. It is important to point out here that a **TrustedSeller** contract is impossible to deploy without mentioning the **SystemManager** contract it will interact with, and is not possible to change after deployment.

In the next line, we find the gas-free 'retrieveName' function which does exactly what one would expect, and returns the name of the retailer. Following this, there is a function for purchasing items. Tagged with the Authorized modifier, it ensures that only authorized accounts can execute a purchase on behalf of the retailer. The function takes in details of the product being sold, here we have chosen name, price, category, brand and serial number, and the address of the buyer. It makes sure that the buyer exists in our **User** contract, and then creates a hash from the input variables name, brand and serial number. This is then checked against the mapping 'mintedDigitalCopiesMapping' to make sure that they are not trying to mint a product with an already existing NFT. After completion, it then emits a 'Purchase' event.

The next function. 'onERC721Received', is not important in the terms of our functionality, but is there to show that we have implemented a way for the contract to receive NFTs and handle them so that they are not "trapped" when sent to the contract.

Two more functions, addTrustedAddress and removeTrustedAddress, defined on lines 28 and 29, allow for the inclusion or exclusion of addresses in the 'SystemManagerMapping'. Protected by the 'Authorized' modifier, only approved addresses can add or remove respectively which addresses are allowed to act on the retailers' behalf and thus allowed to execute functions with the 'Authorized' modifier. When these functions are executed, there are also events emitted, broadcasting to the network what changes have occurred.

Lastly, there are two more functions: 'changeDigitalCopyContract' and 'showDigitalCopyContract'. The former allows the contract to change the address of the associated DigitalCopy contract, while the latter reveals the current DigitalCopy contract's address. It is important to specify that a **DigitalCopy** contract is not established when the contract is deployed, making it necessary for this function to be executed before any purchases can be made. They are also both controlled by the 'Authorized' modifier, making sure that only trusted addresses can change or show which **DigitalCopy** contract the retailer is interacting with.

```
1 pragma solidity ^0.8.17;
2
3 import './DigitalCopy.sol';
4 import './SystemManager.sol';
5 import '@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol';
```

```
6
7  contract TrustedSeller is IERC721Receiver {
8
9      // Variables
10     mapping (address => bool) private trustedSellerAddressMapping;
11     mapping (bytes32 => bool) private mintedDigitalCopiesMapping;
12     string private trustedSellerName;
13     DigitalCopy private digitalCopy;
14     SystemManager private SystemManager;
15
16     // Events
17     event Purchase(address indexed buyer, address indexed seller);
18     event AddedTrustedAddress(address indexed addedAddress);
19     event RemovedTrustedAddress(address indexed removedAddress);
20
21     // Modifiers
22     modifier Authorized() {...}
23
24     // Functions
25     constructor(string memory _trustedSellerName, address
           _SystemManager) {...}
26     function retrieveName() public view returns(string memory){...}
27     function purchase(string calldata _name, string calldata _price,
           string calldata _category, string calldata _brand, string
           calldata _serialnumber, address _buyersAddress) public
           Authorized {...}
28     function onERC721Received(address, address, uint256, bytes calldata
           ) public pure returns (bytes4) {...}
29     function addTrustedAddress(address _address) public Authorized
           {...}
30     function removeTrustedAddress(address _address) public Authorized
           {...}
31     function changeDigitalCopyContract(address _address) public
           Authorized {...}
32     function showDigitalCopyContract() external view Authorized returns
           (address) {...}
```

Listing 5.2: Collapsed TrustedSeller contract

DigitalCopy

The next contract we implement is the **Digital Copy** contract, seen in Listing 5.3. This smart contract is defined as an ERC721 contract, which is an OpenZeppelin contract implementation of the ERC721 standard for NFTs.

The contract begins by defining the private variables on lines 11-16. The 'SystemManager' variable links the contract to a specific instance of a SystemManager contract. This is important to highlight as this has to be defined before deploying any of the contracts, and the address of the deployed **SystemManager** has to be calculated beforehand. The reasoning behind this will be explained in subsequent

paragraphs. The 'ownedTokensMapping' maintains a record of the digital copies owned by each address, while the 'digitalCopyInformationMapping' is a mapping storing information for each specific NFT. Finally, the mintable boolean variable controls whether new digital copies can be minted.

On line 19, the contract defines a structure[4]. The 'digitalCopyInformation' structure captures all relevant details about a digital copy, including its name, price, owner, category, brand, serial number, time, lists with the history of time, prices and owners, number of transfers, and a boolean indicating whether it is for sale. This is then stored in the mapping mentioned earlier, 'digitalCopyInformation'.

The contract then defines several events on lines 32-34 and modifiers on lines 37-41, before we move on to the functions. The first, the constructor, sets the name and symbol of the ERC721 token and assigns the address of the **Users** contract. Here the 'onlySystemManager' modifier explains why the address of the **SystemManager** contract had to be calculated beforehand, as the address of that contract is the only one allowed to deploy this contract.

The 'mint' function, only accessible by a trusted retailer and when minting is allowed, creates a new digital copy. Making this the only function that can be halted entails that the rest of the contract will be operational, even when no new NFTs can be created. This provides a safeguard so that even if a contract is compromised, the owners of the NFTs still have the ability to use them. In addition, the input parameters are used to create the struct, 'digitalCopyInformation', and specifying the "forSale" variable in the corresponding structure as "True". As the function is finished, it emits the 'MintedDigitalCopy' event.

Next is the 'transfer' function which transfer ownership of a digital copy from one address to another, updating its' time, price, owner and their histories. There are measures that insures that only the owner can transfer the asset, and that the buyer is an existing user of the system. After the transaction is successful, the event 'TransferredDigitalCopy' is emitted and the "forSale" variable is set as "False".

The getOwner view function on line 51 returns the owner of a specific digital copy. The burn function, accessible only by the owner of a digital copy, provides the opportunity to destroy the specified digital copy and emits this as an event. The gas-free function 'retrieveInformationForDigitalCopy' returns the data stored about the product as specified in the struct when it was minted. This function is only executable by the owner unless the product is put for sale, ("forSale" variable set as "True), making the information accessible for everyone.

[4]A structure is a data structure format where you can store multiple data of various types in one variable

Following this is the 'retrieveOwnedItems' function which is gas-free and only callable for the **SystemManager** contract by the 'onlySystemManager' modifier, and returns a list of digital copies owned by a specific address.

The 'existingUser' view function, found on line 55, checks if an address corresponds to an existing user in the **Users** contract. The gas-free 'isItemForSale' function checks if a specific digital copy is currently for sale, while the 'putItemForSale' and 'putItemNotForSale' functions allow the owner of a digital copy to change its sale status.

Lastly, the 'changeMintability' function allows the **SystemManager** contract to change the mintability status of this contract. This provides the possibility to both start and stop the creation of future NFTs.

```
1  pragma solidity ^0.8.17;
2
3  import './SystemManager.sol';
4  import './TrustedSeller.sol';
5  import './Users.sol';
6  import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
7
8  contract DigitalCopy is ERC721 {
9
10     // Variables
11     SystemManager private SystemManager = SystemManager (...);
12     mapping (address => uint256[]) private ownedTokensMapping;
13     mapping (uint256 => digitalCopyInformation) private
           digitalCopyInformationMapping;
14     Users private users;
15     uint256 private digitalCopyID;
16     bool private mintable = true;
17
18     // Structs
19     struct digitalCopyInformation {
20         string name;
21         string price;
22         address owner;
23         string category;
24         string brand;
25         string serialNumber;
26         mapping(uint256 => string) priceHistory;
27         mapping(uint256 => address) ownerHistory;
28         uint256 transfers;
29         bool forSale;
30     }
31     struct digitalCopyInformationRetrival {...}
32
33     // Events
34     event MintedDigitalCopy(uint256 indexed digitalCopyID);
```

```
35      event TransferredDigitalCopy(address indexed seller, address
            indexed buyer, uint256 indexed digitalCopyID);
36      event BurnedDigitalCopy(uint256 indexed digitalCopyID);
37
38      // Modifiers
39      modifier onlySystemManager() {...}
40      modifier onlyTrustedSeller() {...}
41      modifier onlyOwner(uint256 _digitalCopyID) {...}
42      modifier forSale(uint256 _digitalCopyID) {...}
43      modifier Mintable() {...}
44
45      // Functions
46      constructor(address _users) ERC721("DigitalCopyAutentisk", "DCA")
            onlySystemManager {...}
47      function mint(string calldata _name, string calldata _price, string
             calldata _category,
48      string calldata _brand, string calldata _serialnumber)
49      public onlyTrustedSeller Mintable returns(uint256
            mintedDigitalCopyID) {...}
50      function transfer(uint256 _digitalCopyID, address _buyersAddress,
            string memory _price) public forSale(_digitalCopyID) {...}
51      function getOwner(uint256 _digitalCopyID) public view returns(
            address) {...}
52      function burn(uint256 _digitalCopyID) public onlyOwner(
            _digitalCopyID) {...}
53      function retrieveInformationForDigitalCopy(uint256 _digitalCopyID)
            public view returns (digitalCopyInformationRetrival memory)
            {...}
54      function retriveOwnedItems(address _address) public view
            onlySystemManager returns(uint256[] memory) {...}
55      function existingUser(address _address) public view returns(bool)
            {...}
56      function isItemForSale(uint256 _digitalCopyID) public view returns(
            bool) {...}
57      function putItemForSale (uint256 _digitalCopyID) public onlyOwner(
            _digitalCopyID) {...}
58      function putItemNotForSale (uint256 _digitalCopyID) public
            onlyOwner(_digitalCopyID) {...}
59      function changeMintability (bool _boolean) public onlySystemManager
             {...}
60  }
```

Listing 5.3: Collapsed DigitalCopy contract

Users

The fourth contract, **Users**, starts, as shown in Listing 5.4, with the declaration of two private mappings: 'existingUsersMapping' and 'userInformationMapping'. The former maintains a list of existing users in the system, while the latter stores user-specific data. The User struct then is declared, containing relevant data for a user, here only name and wallet address.

The functions starts on line 22, first just with a basic constructor. Following is the 'createUser' function with the 'userDoesNotExist' modifier, that can be called by any address to create a new user, provided that user doesn't already exist. The input parameters "name" and "personalIdentifier" are hashed together, to ensure that no confidential data are stored, while still generating a unique hash to be able to store the 'User' struct in the 'userInformationMapping' mapping. This user address is then added to the 'existingUsersMapping' mapping before it emits the 'AddedUser' event.

Lastly, the userExist function is a view function that takes an address as a parameter. It returns a boolean indicating whether the given address corresponds to an existing user in the system.

```
1  pragma solidity ^0.8.17;
2
3  contract Users {
4
5      // Variables
6      mapping(address => bool) private existingUsersMapping;
7      mapping(bytes32 => User) private userInformationMapping;
8
9      // Structs
10     struct User {
11         string name;
12         address walletAddress;
13     }
14
15     // Events
16     event AddedUser(address indexed userAddress);
17
18     // Modifers
19     modifier userDoesNotExist() {...}
20
21     // Functions
22     constructor() {...}
23     function createUser(string memory _name, string memory
           _personalIdentifier) public userDoesNotExist {...}
24     function userExist(address _address) public view returns(bool)
           {...}
25 }
```

Listing 5.4: Collapsed Users contract

Reviews

The final contract **Reviews** can be seen in Listing 5.5. It begins with the two variables 'userReviewsMapping' and 'transactionReviewedMapping', the first mapping maintains a list of all reviews made for a seller, while the second keeps track of which

transactions have already been reviewed. The last variable on line 10 makes sure that only the **SystemManager** has direct access to the reviews.

The 'Review' struct is defined with several fields: "seller" and "buyer" are Ethereum addresses representing the seller and buyer involved in the transaction, "rating" is positive number between 0 and 5 representing the rating given in the review, "digitalCopy" is the address of the **DigitalCopy** contract that was involved in the transaction, 'digitalCopyID' is the identifier for the NFT involved, 'transactionID' is the transaction which was reviewed, and 'text' contains possible comments for the review.

On line 31, the 'newReview' function accepts various parameters related to a review which are used to create the structure defined in the previous section. It is protected by the 'ownerOfNFT' and 'nonExistingReview' modifiers, making sure that only the new owner from the transaction can rate the seller and that they have not already submitted a review of the transaction. After the successful creation of a new review, the 'AddedReview' event is emitted.

Next, the 'getUserReviews' function returns all reviews for a given user. This function is gas-free and uses the 'OnlySystemManager' modifier, meaning that it can only be called by the **SystemManager** contract to fetch reviews for a specific user.

Finally, the 'getUserReviewSum' function calculates the sum of all review ratings and the amount of reviews, before returning them to the function caller. Since Solidity does not support decimal numbers, these numbers can then later be used to calculate the average score of a user. This function is gas-free and only executable by the **SystemManager** contract through the 'OnlySystemManager' modifier.

```
1  pragma solidity ^0.8.17;
2
3  import './DigitalCopy.sol';
4
5  contract Reviews {
6
7      // Variables
8      mapping(address => Review[]) private userReviewsMapping;
9      mapping(bytes32 => bool) private transactionReviewedMapping;
10     address private SystemManagerAddress = ...;
11
12     // Structs
13     struct Review {
14         address seller;
15         address buyer;
16         uint8 rating;
17         address digitalCopy;
18         uint256 digitalCopyID;
19         bytes32 transactionID;
```

```
20          string text;
21      }
22
23      //Events
24      event AddedReview(bytes32 indexed transactionID);
25
26      // Modifiers
27      modifier nonExistingReview(bytes32 transactionID) {...}
28      modifier OnlySystemManager() {...}
29      modifier ownerOfNFT(address _address, uint256 _digitalCopyID) {...}
30
31      // Functions
32      constructor() {...}
33      function newReview(address _seller, uint8 rating, address
            _digitalCopy, uint256 _digitalCopyID, bytes32 _transactionID,
            string memory _text) public
34      ownerOfNFT(_digitalCopy, _digitalCopyID) nonExistingReview(
            _transactionID) {...}
35      function getUserReviews(address _address) public view
            OnlySystemManager returns(Review[] memory) {...}
36      function getUserReviewSum(address _address) public view
            OnlySystemManager returns(uint256, uint256) {...}
37  }
```

Listing 5.5: Collapsed Reviews contract

5.2 Front-end web application implementation

The front-end application for our decentralized system is built using React and Typescript. React is chosen for its simplicity and it allows us to quickly set up a user interface without using too much time on design. The use of typescript brings in strong typing, which results in a more robust running application. Compared to using javascript, Typescript helps with catching errors during development instead of in run-time that typically occurs when interacting with smart contracts.

To make the front-end interact with the Ethereum blockchain, we use the Ethers.js library. Ethers.js is a lightweight and optimized library with tools for working with Ethereum, enabling us to work with smart contracts, make transactions and interact with wallets from the client application.

For a more productive development process and cleaner code, we use eth-hooks. Eth-hooks is a custom hooks library for Ethers.js that follows the best practices of the hooks pattern in React, encapsulating complex blockchain logic into reusable hooks. This not only simplifies the code but also abstracts away much of the complexity involved in interacting with the Ethereum blockchain.

The front-end application is deployed using Surge, a platform for deploying static web pages to a Content Delivery Network (CDN). Surge has been chosen due to its simplicity and efficiency in serving static websites. We especially facilitate two advantages of Surge: Speed: Surge deploys the content to a CDN, which makes the site load quickly regardless of the user's location. Decentralization: By serving the application through a decentralized platform like Surge, we align with the principles of decentralization that underpin our blockchain-based system.

5.3 Deployment process

Before deployment we need to calculate the address of the future SystemManager contract, this can be done using the python script found in Appendix D. By using our private key, which by security reasons will not be written here, and the nonce of 0, as this is our first interaction with this network using this key, we get the address "0x0CEC837a835A4b4356a811cdB2CC3Fa66Ac30dE2". This will then be placed inside the DigitalCopy contract as the SystemManager variable.

5.3.1 Deployment script

In Listing 5.6 we deploy the SystemManager contract, which in its' construction concurrently deploys the Users and Reviews contracts. Proceeding with the deployment, we introduce the DigitalCopy contract via the SystemManager and initiate a credible retailer named Trusted Watches, which we subsequently incorporate into our system. Trusted Watches also adds the deployed DigitalCopy to its' system. Following the deployment of all the contracts, we document their addresses detected by the events into a file, thereby ensuring their accessibility for the next scripts.

The deployment process on the Sepolia testnet has been written to be quite similar to that of the local deployment, however with some necessary initial changes. These changes are minimal to the code in Listing 5.6 and if interested can be found in the repository on Github [5].

```
1    ...
2
3    // Deploy SystemManager with Users and Reviews
4    ...
5    SystemManager = await SystemManagerFactory.deploy();
6    let UsersEvent: Promise <void> = new Promise((resolve, reject) =>
         {...});
7    let ReviewsEvent: Promise <void> = new Promise((resolve, reject) =>
          {...});
8
9    await UsersEvent;
10   await ReviewsEvent;
```

[5] https://github.com/Autentisk

```
11      await SystemManager.deployed();
12
13      console.log("SystemManager.sol contract deployed to address: ",
            SystemManager.address);
14
15    // Deploy DigitalCopy.sol
16      const DigitalCopyFactory: ContractFactory = await hre.ethers.
            getContractFactory("DigitalCopy");
17
18      let digitalCopyAddress: string | null = "";
19      await SystemManager.deployDigitalCopy();
20      let DigialCopyEvent: Promise <void> = new Promise((resolve, reject)
             => {...});
21
22      await DigialCopyEvent;
23      digitalCopy = await DigitalCopyFactory.attach(digitalCopyAddress);
24
25      // Deploy TrustedSeller.sol
26      const TrustedSellerFactory: ContractFactory = await hre.ethers.
            getContractFactory("TrustedSeller");
27      trustedSeller = await TrustedSellerFactory.deploy("TrustedWatches",
             SystemManager.address);
28      await trustedSeller.deployed();
29      console.log("TrustedSeller.sol contract deployed to address: ",
            trustedSeller.address);
30
31
32      // Add Trusted Seller to the approved list
33      await SystemManager.add(trustedSeller.address);
34
35      // Set Trusted Sellers' DigitalCopy contract to interact with
36      await trustedSeller.changeDigitalCopyContract(digitalCopyAddress);
37
38      // Write variables to file
39      let contractAddresses = {...};
40      fs.writeFileSync('scripts/contractAddresses.json', JSON.stringify(
            contractAddresses, null, 2));
```

Listing 5.6: Selected parts of script showing the deployment of the contracts

Chapter 6
Results

In this chapter, we present results from the practical deployment and operation of our system. This includes setting up our smart contracts on the Sepolia testnet and running the scripts from Appendix C based on the scenarios discussed in Section 4.2.5. Our front-end interface, which communicates directly with the blockchain[1], will be used to highlight these scenarios in action. We present the various transactions that have been successfully completed on the blockchain and the instances where transactions have been reverted.

Next, we move to the economic aspect of the system, discussing the 'gas costs' associated with the different scenarios and their transactions. We examine how factors like gas price, gwei value, and the number of owned NFTs can influence these costs. This chapter aims to bridge our system's theoretical design with its practical application, providing a comprehensive understanding of its functionality and cost implications.

6.1 Deployment

As we deploy our system onto the Sepolia testnet, we utilize the block explorer Etherscan[2] on the Sepolia testnet to be able to find and easily display our transactions. By inputting the different contract addresses acquired when we run the deploy script, we are able to observe the transactions as they appear on the blockchain network. For readers who wish to explore further, the corresponding information is public and easily accessible on any block explorer.

After the completion of the deploy script, these are the addresses our smart contracts are deployed to:

[1] Currently only on localhost
[2] https://sepolia.etherscan.io/

- **SystemManager:** 0x0CEC837a835A4b4356a811cdB2CC3Fa66Ac30dE2
- **TrustedSeller:** 0x74aCBE18ADEBD759e2944f34d2719B674A923718
- **DigitalCopy:** 0xE446bb298E67f0C27ADb8DE9BF20cc203bb6e9B2
- **Users:** 0xb15c0616E6CeE8a644F69a72eBDceBf6320a2457
- **Reviews:** 0x656aEc7fa90d169DC95b90D912Fc4513841e6248

Further in Figure 6.1, we can see the deployment of the SystemManager contract. We have highlighted five areas that we would like to explain further. First is the transaction hash, which is the unique identifier for this change of state in the smart contract, and can be used to search for a specific change of state. Next, as highlighted by the number two in the figure, we can see the "From" and "To" fields. The from address is the public address of the private key we used to deploy the contract and the to address is the same address we calculated the smart contract to have beforehand in 5.3. Then we have the highlighted part specified by the number three, and this part shows the price of gas when we deployed the contract, around 1.5 gwei. Penultimately we have the gas usage of the contract, by focusing on the last section of the right hand side of the higlighted part, referenced by the number four, we can see that the gas cost of this deployment was 5,155,282 gas. Lastly, the part highlighted by the number five is not as important, but we wanted to mentioned that the nonce is also visible and is set to zero as this is this account's first interaction with this network.

6.2 Scenario results

We used the implemented PoC with the test scenarios introduced in 4.2.5 to validate that we have met the functional requirements presented in Section 3.3. We validate this by utilizing the functionality in our application by executing these 4 different scenarios where each of these covers one or several requirements. We have also used Alchemy, a Web3 development platform, to deploy our system through and to observe successful and reverted functions. We refer to Appendix B to reproduce the execution.

6.2.1 Scenario 1: NFT Minting and Duplicate Prevention

Scenario 1 (4.2.5) focuses on testing the functionality related to a store selling an item to a user and generating a corresponding NFT for the transaction. The user interface of the store during this process is shown in Figure 6.2.

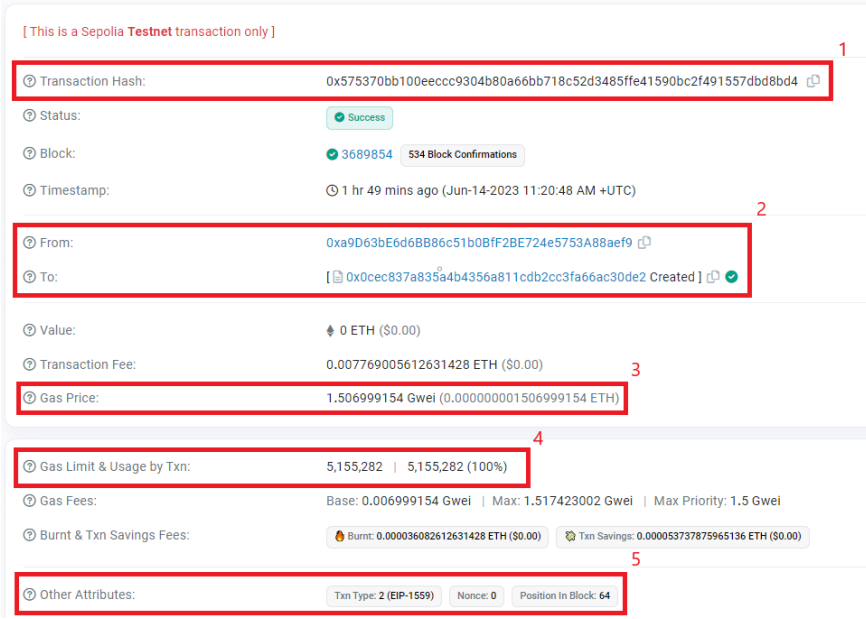


Figure 6.1: Screenshot of SystemManger deployment on Etherscan.

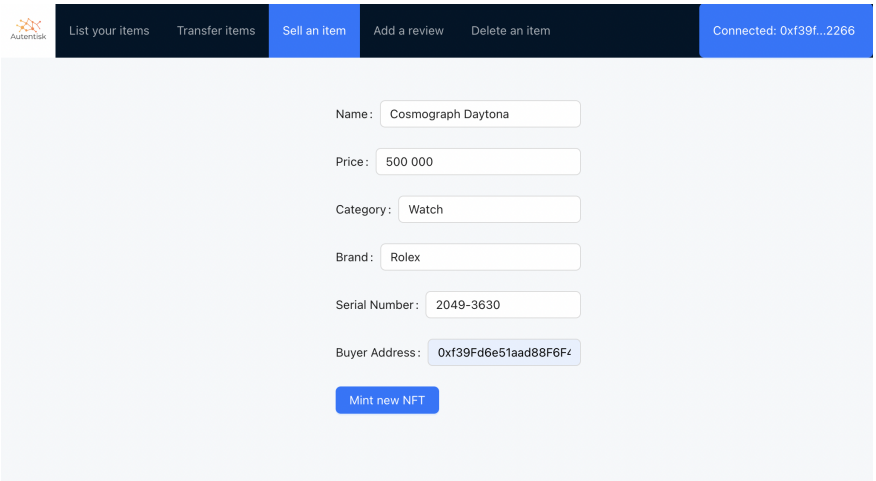


Figure 6.2: Front-end interface that shows the purchasing process of an item from a retailer's perspective.

Once the trusted seller initiates the mint button in Figure 6.2, the NFT is minted and subsequently transferred to the new owner. Confirmation of a successful purchase

is shown in Figure 6.3, which presents a capture from Etherscan[3].

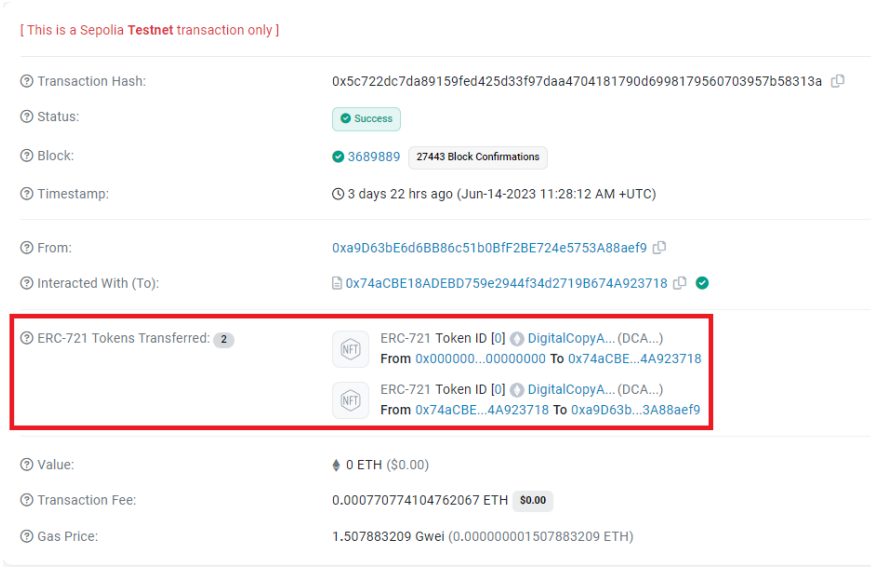


Figure 6.3: Screenshot of the completed purchase on the Sepolia testnet.

To further examine the system's behavior, we explore the scenario where the store attempts to generate a new NFT for the same product. Figure 6.4 demonstrates that the generation of the NFT is not successful since it already exists on the blockchain. This behavior ensures that duplicate NFTs for the same product is not created, maintaining the integrity and uniqueness of the NFT system.

Figure 6.4: Screenshot of the reverted transaction where the retailer tried to mint an item for the second time.

During our test scenario, the store successfully generates an NFT for the customer and transfers it to them. The completion of the transfer can be observed in Figure 6.5, where the user is shown as the new owner of the NFT.

[3]Can be found at https://tinyurl.com/mw547wd5

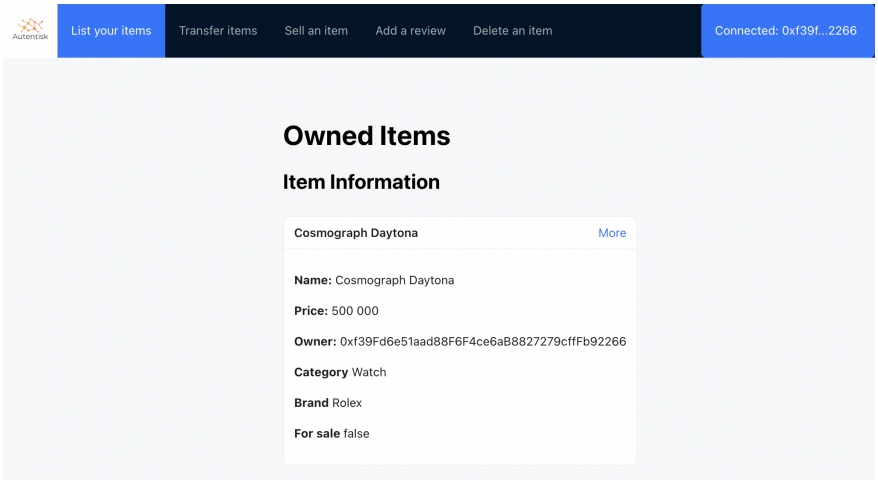


Figure 6.5: Front-end interface that shows the owned items for a user.

By conducting these tests and analyzing the corresponding figures, we validate that the system effectively generates and transfers NFTs for transactions, preventing the creation of duplicate NFTs and ensuring the accuracy and reliability of the process.

6.2.2 Scenario 2: Listing, Selling, and Ownership Transfer

Scenario 2 (4.2.5) involves testing the process of a secondhand trade between two users. The verification of the seller's ownership of the Digital Copy is conducted by the first user, as illustrated in the user interface shown in Figure 6.6.

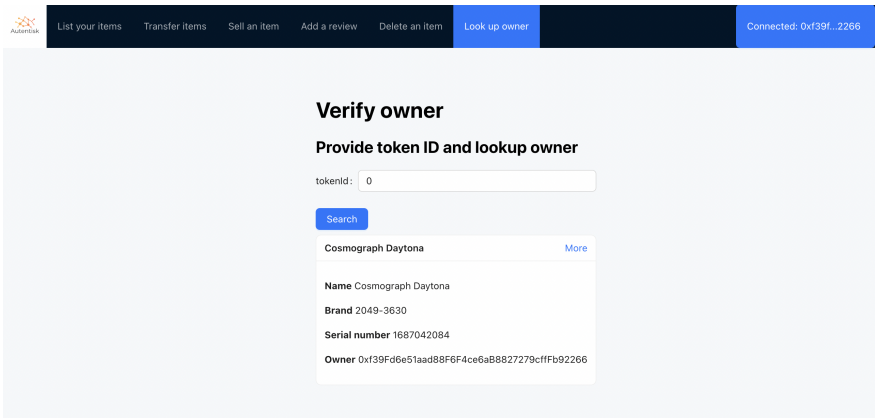


Figure 6.6: Front-end interface that helps one user to verify the ownership of other user's NFT.

Next, we proceed to test the transfer of NFT ownership from one user to another. Once the trade is executed, the current NFT owner transfers digital ownership. This can be accomplished by invoking the transfer function or by utilizing the user interface seen in Figure 6.7.

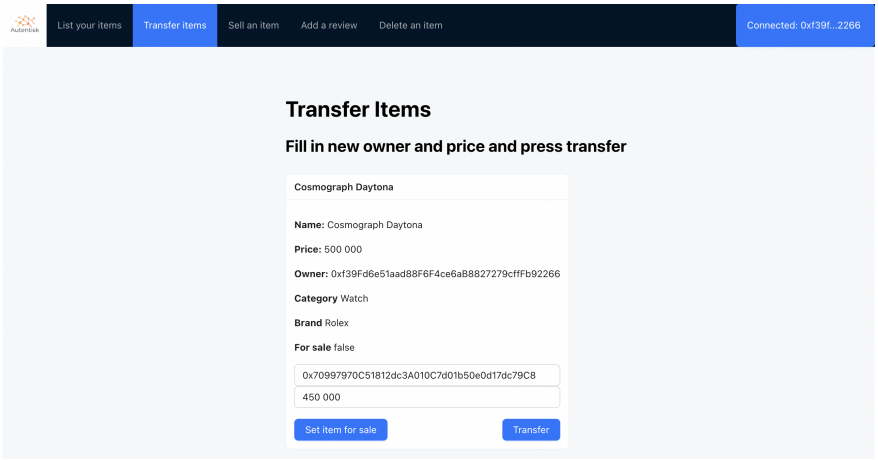


Figure 6.7: Front-end interface that shows the transfer of an NFT to another user from a seller's perspective.

Figure 6.8 illustrates a successful transfer of ownership[4], confirming that the NFT has been securely transferred from the seller to the buyer.

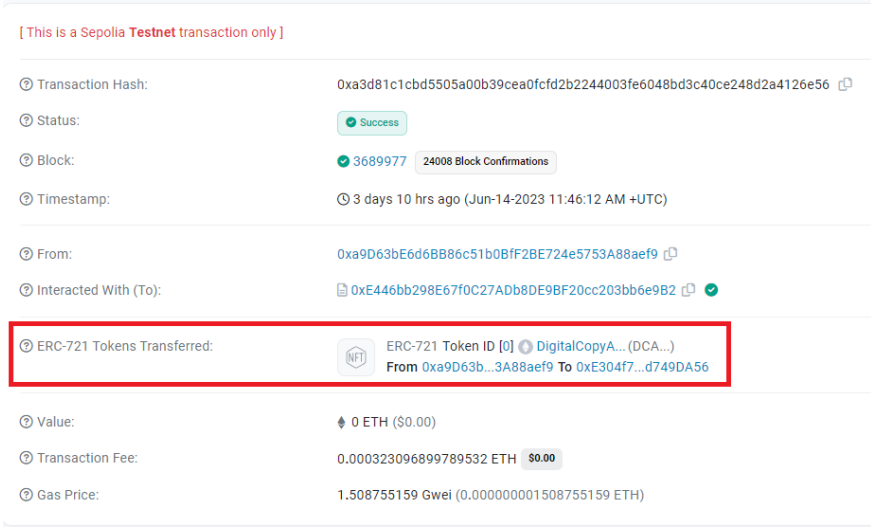


Figure 6.8: Screenshot of the completed transfer on the Sepolia testnet.

To ensure the integrity of the transaction, we also examine the behavior of the

[4]Can be found at https://tinyurl.com/4ye3hs5x

previous owner after the transfer. Figure 6.9 demonstrates that the previous owner, having completed the transfer, no longer has visibility or the ability to perform actions on the NFT.

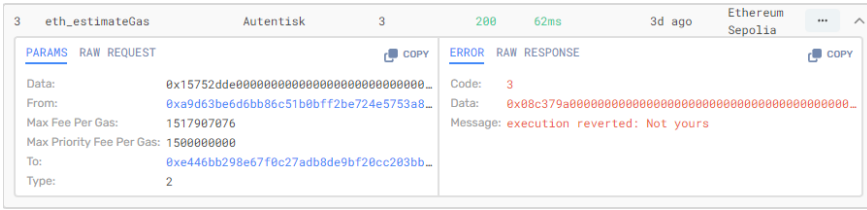


Figure 6.9: Screenshot of the reverted transaction where the seller tries to buy back the item sold cheaper.

By conducting these tests and analyzing the corresponding figures, we verify that the system accurately facilitates secondhand trades by verifying ownership and enabling secure transfers of NFTs between users.

6.2.3 Scenario 3: Review and Rating System

In Scenario 3 (4.2.5), we focus on the process of users reviewing each other. We conducted tests to verify two specific cases: whether a user can review a transaction they were not involved in, and whether they can submit a review for a transaction they were part of.

Figure 6.10 shows the implemented user interface for sending a review for an item they have received. By submitting a review, the previous owner of the item will receive a rating that is stored on the blockchain and contributes to their overall rating.

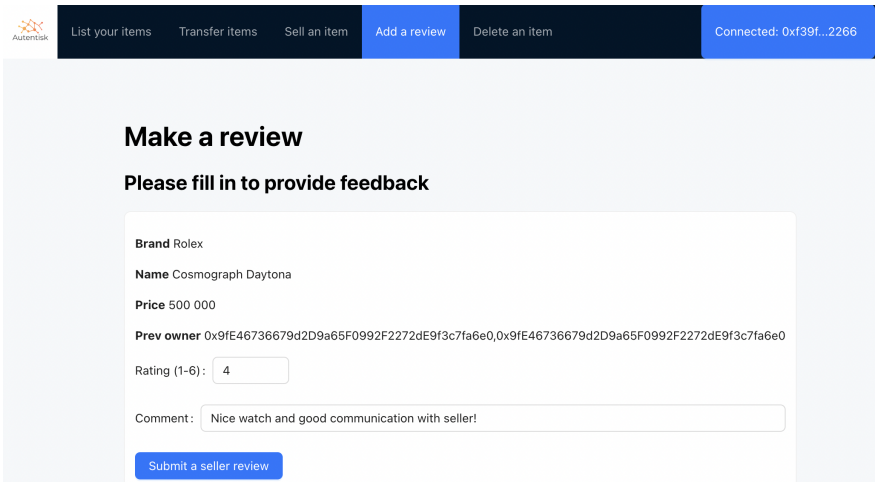


Figure 6.10: Front-end interface that shows the review of a transaction from the buyer's perspective.

It is crucial to note users are only permitted to submit reviews on items they have bought. Figure 6.11 demonstrates a situation where a dishonest user attempts to submit a review for a product they do not possess. In such cases, the system rejects the review, ensuring that only legitimate reviewers can provide feedback.

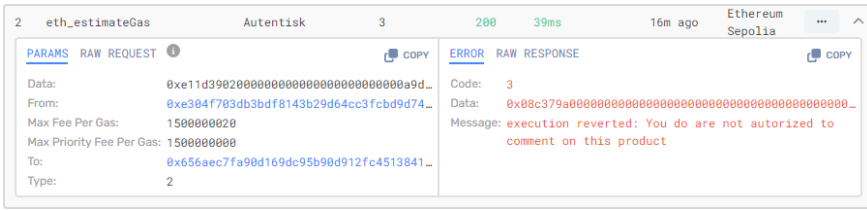


Figure 6.11: Screenshot of the reverted transaction where the buyer tries to review the wrong item.

Conversely, Figure 6.12 showcases a successful transaction where the reviewer was indeed involved in the trade, and the system accepts their review[5].

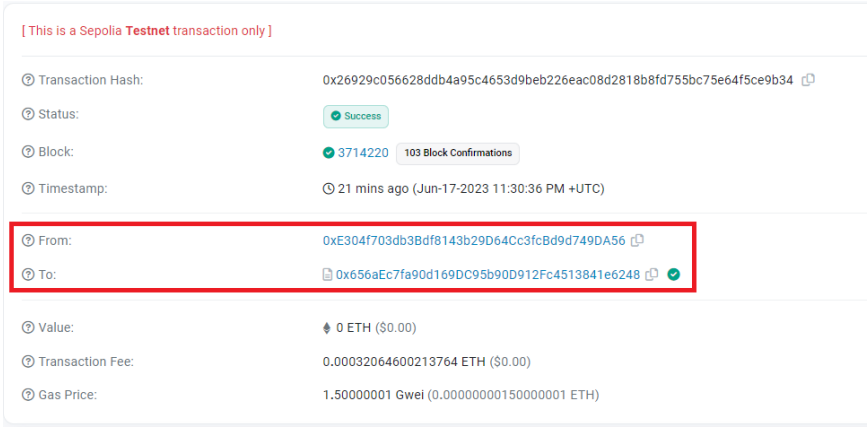


Figure 6.12: Screenshot of the completed review on the Sepolia testnet.

Through these tests and the corresponding figures, we validate the functionality of the review feature within our system, ensuring that it operates correctly and safeguards against fraudulent or unauthorized reviews.

[5]Can be found at https://tinyurl.com/42389e7k

6.2.4 Scenario 4: NFT Deletion

Scenario 4 (4.2.5) focuses on the process of a user deleting an NFT. Figure 6.13 presents the user interface that allows users to initiate the deletion of an NFT they possess.

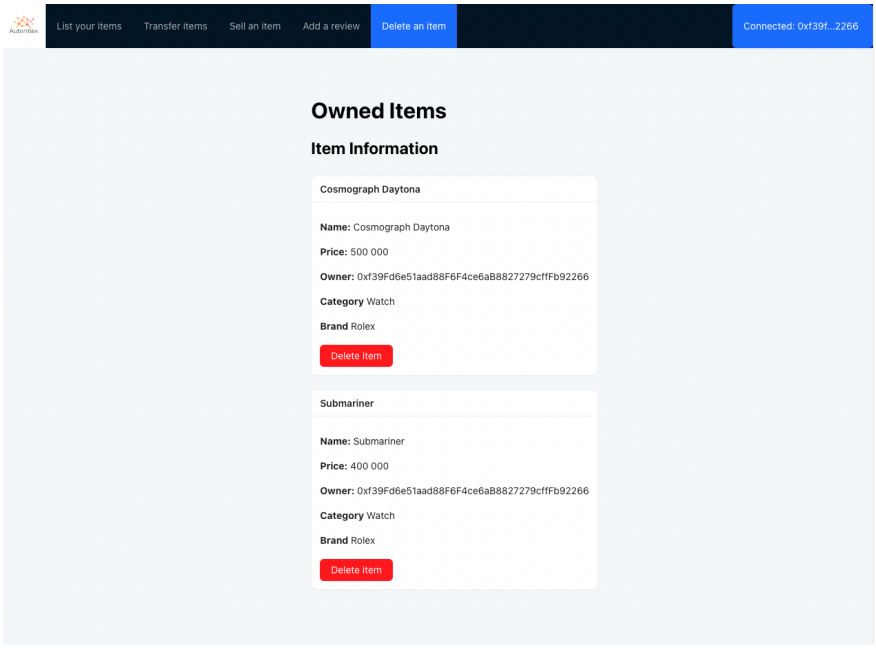


Figure 6.13: Front-end interface that shows the deletion of an NFT.

Figure 6.14 illustrates a successful operation where the digital copy associated with the NFT is successfully removed from the blockchain[6]. Within the highlighted red region, it is evident that the ownership has been transferred to the zero address, which is effectively equivalent to erasing the token.

[6]Can be found at https://tinyurl.com/yszctxf8

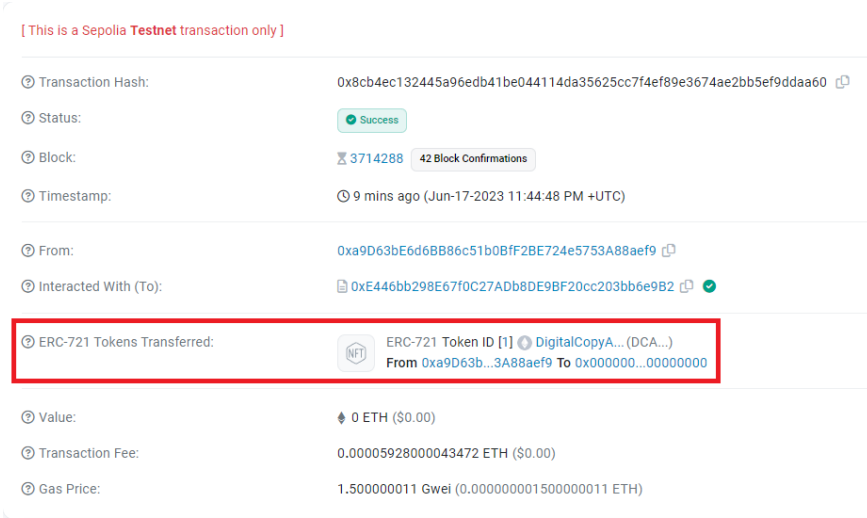


Figure 6.14: The provided image displays a captured transaction of the completed burn process on the Sepolia testnet.

In this scenario, we also conducted tests to examine the system's response when an unauthorized user attempts to delete another user's NFT. Figure 6.15 demonstrates that such transactions are not processed by the system, effectively protecting against unwanted deletion of ownership.

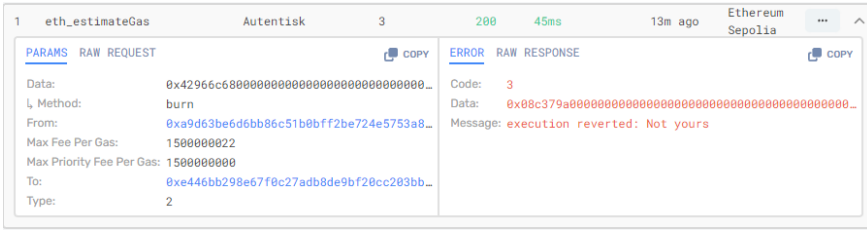


Figure 6.15: Screenshot of the reverted transaction where a user tries to burn an item they do not own.

By conducting these tests and analyzing the corresponding figures, we ensure that the system functions as intended, allowing users to delete their own NFTs while preventing unauthorized individuals from tampering with others' ownership rights.

6.3 Gas cost

6.3.1 General costs

In Table 6.1 we have the gas cost and the cost in USD, which were paid during the execution of the multiple scenarios. The costs depend on the gas price and the value of gwei, but we have, unless otherwise specified, used the values of 22 gwei and $0.0000018, respectively. As can be seen from the screenshots from the block explorer when we executed our deployment and scenarios, the price at the time was 0.0000015 USD/gwei, but then again this varies over time.

An essential observation from the table is the disparity in gas costs associated with different operations. This variation is due to the varying computational requirements of the operations. For instance, the 'Deploy SystemManager' operation is the most computationally intensive task and hence incurs the highest gas cost (5,155,282), translating into $204. This operation is essentially a one-time cost, which, although high, is crucial for the initial setup of the system and expected to be the highest as it implements the management of the system. Following, we see that the deployment of the other contracts, DigitalCopy and TrustedSeller, are also at the top of the list in reference to cost. However, these operations are generally part of the initial system setup and hence do not contribute to the recurring costs of system maintenance and usage. Once a DigitalCopy or a TrustedSeller contract has been set up, this is most often a one-time cost for the deploying stakeholder.

Operations like 'Purchase 1 in Trusted Seller' and 'Purchase 2 in Trusted Seller' represent the two watch purchases in Scenario 1 (4.2.5) and are typical user interactions with the system. Their individual costs are relatively high ($20.2 and $19, respectively), and it is important to note that such transactions are expected to occur frequently. However, only once per item and not frequently for the individual user, making it a more affordable price for the user. We can observe that the first purchase cost is more costly than the second by around 31 000 gas. This change in gas usage is caused by the difference in the names of the two watches. "Cosmograph Daytona" is longer than "Submariner", making it more computationally expensive to work with. As we wanted to make sure that this increased cost did not turn exorbitant, we made a purchase where each of the affected variables, namely name, price, category, brand and serial number, were made to consist of 69 words each. This resulted in a cost of 2,723,742 gas, translating into $108. As this is an unreasonable length for even just one of the variables, we can safely say that a purchase would never achieve this gas cost, even in extreme situations.

Meanwhile, operations like 'Create user in Users (1st time)' and 'Create user in Users (2nd time)' exhibit much lower gas costs. Worth commentating is that their costs are almost identical, suggesting that the computational effort to add a

new user to the system remains almost the same. This can be further confirmed as the computational cost only depends on the name's length and the user's personal identifier. This is also a one-time fee paid by the users of the system when they first register. We also wanted to ensure that this cost did not turn exorbitant with longer names and created a user with 69 words in both name and personal identifier. This resulted in a gas cost of 418,645, which in turn equals $16.6, which in reality would be highly unlikely.

Operation	**Gas Cost**	**Cost in USD**
Deploy SystemManager	5,155,282	$204
Deploy DigitalCopy in SystemManager	2,636,386	$104
Deploy TrustedSeller	847,738	$33.6
Purchase 1 in Trusted Seller	511,163	$20.2
Purchase 2 in Trusted Seller	479,667	$19
Transfer item in DigitalCopy	214,148	$8.5
New review in Reviews	213,752	$8.5
Create user in Users (2nd time)	92,019	$3.6
Create user in Users (1st time)	91,983	$3.6
Change DigitalCopy in TrustedSeller	56,044	$2.2
Add TrustedSeller in SystemManager	47,247	$1.9
Put item for sale in DigitalCopy (2nd time)	46,012	$1.8
Put item for sale in DigitalCopy (1st time)	46,000	$1.8
Burn item in DigitalCopy	39,520	$1.6

Table 6.1: Gas costs for different operations and their cost in USD (sorted by gas cost) using the gas cost of 22 gwei and gwei value of $0.00000180

6.3.2 Transaction cost

We further experimented with transaction cost differences, as seen in Figure 6.16. The cost of transfers is subject to one major influencing factor, the quantity of items owned by a given user, in this case, Kari.

Kari owns 50 identical items, which are then sequentially transferred to a second user at the same cost, starting from the 50th item. The computational effort required to iterate through a list of items is higher depending on the position of the item in the list. This necessitates a larger amount of gas to transfer items further back in the list, as with our initial transfers. As the list shortens with each transaction, the associated cost declines progressively.

Interestingly, the first transfer also experiences a notable spike in cost. This is due to the additional computational resources needed to set up the list for the receiving user, who, prior to this transaction, did not own any items. This initial setup of the list costs approximately an additional 36,500 gas. Following this setup, the cost per transfer demonstrates a consistent decrease by 2,339 gas as the list of items decreases with each successive transaction.

The final transfer, involving the last item on the list, requires a slightly lower amount of gas, costing 168,195 gas which is equivalent to $6.7. This is attributed to the absence of the need for iteration through the list, thus requiring less computational effort and, consequently, less gas.

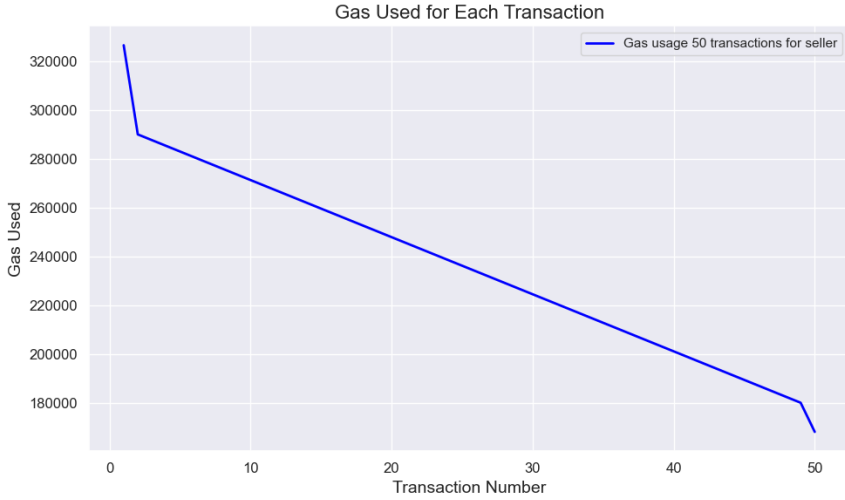


Figure 6.16: Gas cost in gwei of 50 worst-case transactions.

This makes it possible to calculate the worst-case expected cost of a transfer depending on the amount of NFTs the seller has. Disregarding the cost of the initial setup if the receiver has yet to receive an NFT as well as the decrease in the cost of transferring the last item, we get this equation:

$$\widehat{GasCost} = NumberOfOwnedItems \times 2400 + 177795$$

Which again can be further expanded to estimate the cost in USD:

$$\widehat{Price} = (NumberOfOwnedItems \times 2400 + 177795) \times GasPrice \times \frac{EtherValue}{10^9}$$

Using a gas price of 22 gwei, an ether value of $1,800 and having 100 items:

$$\widehat{Price} = (100 \times 2400 + 177795) \times 22 \times \frac{1800}{10^9} = 16.54$$

This results in a worst-case cost of $16.5 for this transfer. It is worth mentioning that these are estimators, and other factors, such as the item's price, also affect the gas cost.

6.3.3 Historical prices

As the gas cost is static, they will never change as the smart contracts are immutable, a purchase or a transfer with the same variables will always have the same gas cost. However, the gwei price for the execution of this gas and the gwei value will fluctuate. Therefore, we have plotted the most used functions as well as the contracts deployment cost both in gwei and USD from 2020 onwards. In addition, we calculated a table with the average cost from the same timeline.

Deployment

Transaction Type	Gas Cost	Price (gwei)	Price (USD)
SystemManager	5,155,282	347,413,189	$693
DigitalCopy	2,636,386	177,665,406	$354
TrustedSeller	847,738	57,128,855	$114

Table 6.2: Average historical deployment prices in gwei and USD

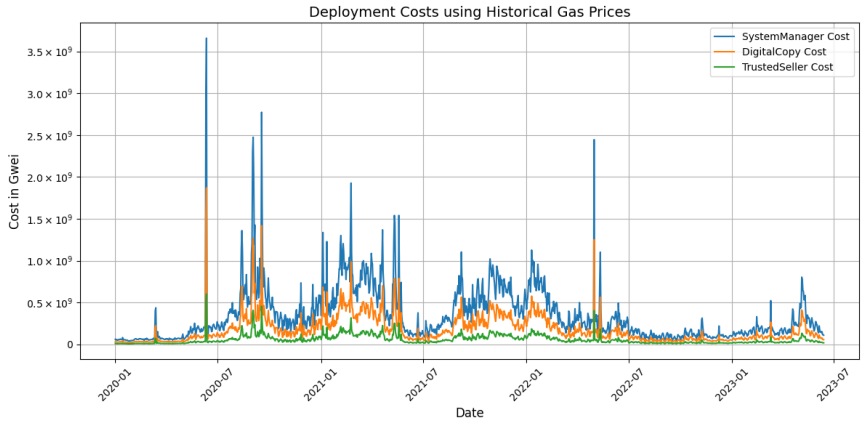


Figure 6.17: Simulation charts of deployment cost based on historical prices in gwei.

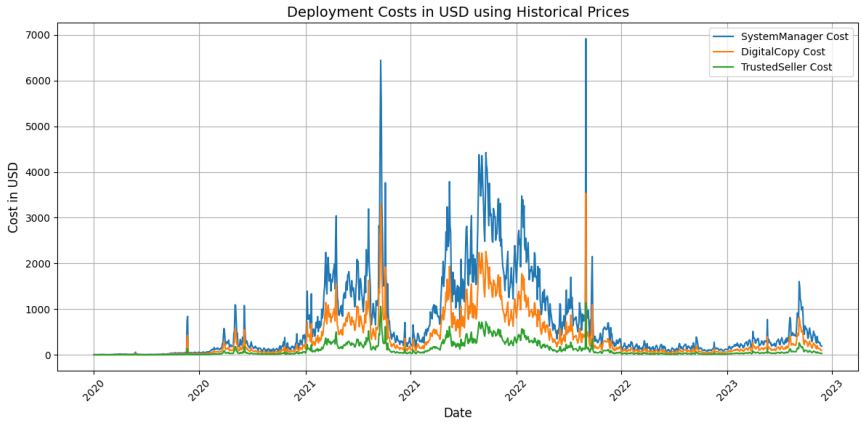


Figure 6.18: Simulation charts of deployment cost based on historical prices in USD.

Functions

Transaction Type	Gas Cost	Price (gwei)	Price (USD)
Purchase	511,000	34,436,164	$68
Transfer	214,000	14,421,407	$28
Create User	92,000	6,199,857	$12
Burn	39,500	2,661,895	$5

Table 6.3: Average historical function prices in gwei and USD.

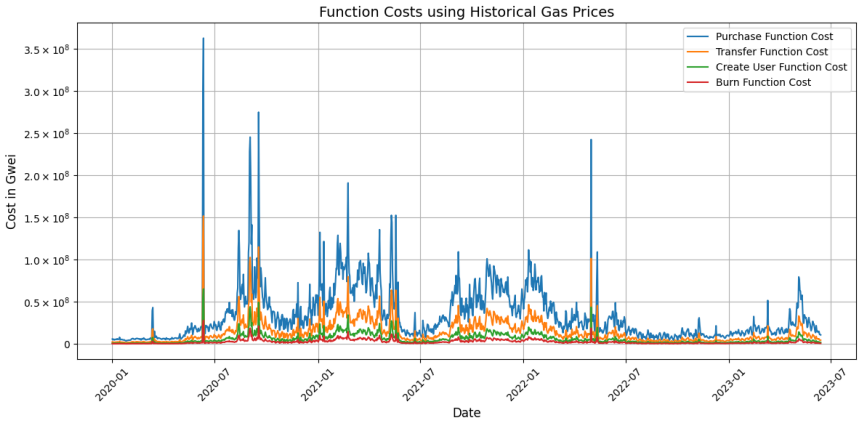


Figure 6.19: Simulation charts of functions cost based on historical prices in gwei.

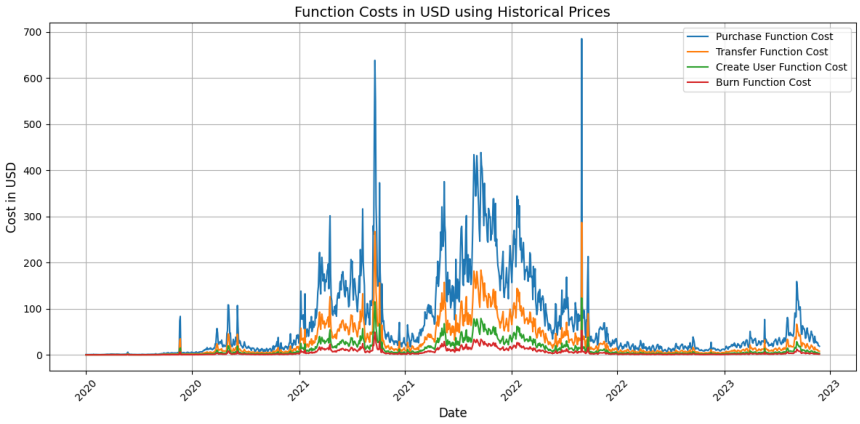


Figure 6.20: Simulation charts of functions cost based on historical prices in USD.

Chapter 7
Discussion

7.1 Analysis and comparison with requirements

In this section, we present the evaluation of the system based on how the developed architecture fulfills the stakeholder requirements, followed by the evaluation based on the developed prototype. The test scenarios from Section 4.2.5 were used for the evaluation.

1. **Purchase Item and Receive NFT**: This requirement is satisfied with the `purchase()` function in the TrustedSeller contract. We see the requirement is fulfilled in Scenario 1 in the result chapter.

2. **Overview and Listings**: An overview of all owned NFTs can be seen at the end of Scenario 1, as well as information about each of the products, accessible to the owner.

3. **Access Product Information**: We can see in Scenario 2 that a user does not have access to information about products which are not for sale, but as soon as it is marked for sale, the buyer acquires access. While we did not implement information regarding warranties, this information is often accessible through their serial number. We did not implement functionality for showing services and repairs to items, as we evaluated this to add unnecessary complexity to our PoC. However, one could transfer the NFTs back and forth between such a provider, to document a proof of the service rendered.

4. **Execute Transaction and Transfer NFT**: In Scenario 2 we can see that this functionality is successfully implemented with the `transfer()` function in the DigitalCopy contract. As the seller is not able to purchase it back, we also prove that the NFT's information is updated.

5. **Rate a Transaction**: In Scenario 3, we see that the user is able to review the completed purchase, and only the transaction it is involved in, and that this

review will follow the seller in the future. As the review is connected to the seller's public wallet, this will also follow them through different platforms.

6. **Remove a Broken Product**: Scenario 4 shows that this functional requirement is successfully incorporated, as well as highlighting that this option is only possible for the owner of the NFT.

7.2 The system's ability to cope with non-functional requirements

Scalability: Our implementation is on the Ethereum blockchain which inherently supports scalability. Any number of users can interact with the smart contracts simultaneously, and the platform can handle an increasing number of transactions without any major performance degradation. However, as it is a public blockchain, the transaction throughput depends on the Ethereum network congestion and the gas price that the users are willing to pay.

Robustness: Smart contracts are designed to be immutable and irreversible, adding to their robustness. Our smart contracts are designed to handle unexpected situations gracefully with the help of modifiers and require statements. No change to the system is possible without passing these requirements, as has been demonstrated in the scenarios, and we believe we have covered the most important edge cases. For example, the NFT cannot be destroyed unintentionally or that retailers cannot mint multiple NFTs for the same product due to the internal requirements of the code.

Security: Our smart contracts are designed with security as a primary concern. The contracts have been meticulously designed with mechanisms, like permissions and modifiers, to ensure that only authorized users have access to certain operations. While the SystemManager contract has a lot of power when it comes to the inclusion of trusted retailers and control over the deployment and functionality of the DigitalCopy contract, it has no control over the NFTs themselves. Only the trusted retailers have the power to add accounts which can act on their behalf and only these accounts are allowed to mint digital copies of the product they sell. Following, the ownership of an NFT can only be transferred by the current owner. While the SystemManager can remove trusted sellers and stop the "mintability" of a DigitalCopy contract, the existing NFTs still have full functionality, which ensures the protection of the digital assets. However, if the wallet of the owner of the SystemManager contract is compromised this would compromise the entire system. Thus highlighting the importance of security knowledge and practices for this entity.

Reliability: As the system operates on the Ethereum blockchain, it inherits the reliability of the Ethereum network. This means that as long as Ethereum is operating,

our system will also be operational with high availability and minimal downtime. As Ethereum is coming up on 10 years of being operational and is one if the most used blockchain as of today, we find this blockchain as sufficient in its reliability.

Performance: The smart contracts are probably not as optimized as they could be, but we have had gas cost on our mind throughout the implementation of them. For example, by using mappings as often as possible instead of arrays, this has lowered the cost of computational operations in exchange for a more complicated system. Our implementation has, from our testing, swift responses and efficient handling of transactions, thereby providing a satisfying user experience. However, as can be seen on the increased cost of gas with transfers for owners with multiple items, this could have been made cheaper by adding more complexity to the code. As our solution is just a PoC and the transaction cost still remains on a reasonable level, we still find this requirement to be fulfilled.

Usability: Although the user interface is not a fully developed solution, we have created a functional interface which presents the core functionality of the system. Through this interface, we show how a large-scale system could be made intuitive and easy to use, even for individuals not familiar with blockchain technology. The design of the system makes it possible to develop a front-end which is user-friendly and makes it easy to buy, sell and transfer NFTs as well as providing an overview of owned goods. The system can be accessed through any Ethereum-compatible wallet, such as MetaMask, and be incorporated into already existing e-commerce platforms as an API such as Amazon, Ebay or Finn.no. Even though this thesis highlights improvements Web3 can provide to the reseller markets, it still faces difficulties in convincing people who do not know much about blockchain technology to trust it as well as being heavily dependent on functional wallet systems such as MetaMask.

Upgradability: As of now, our smart contracts are not upgradable. Once deployed, the logic inside them cannot be changed. This is a trade-off between upgradability and immutability, which is a fundamental property of smart contracts. However, there are design patterns like the proxy pattern which can be used to add upgradability to smart contracts, which can be considered for future work. However, this will impact the level of security, and could provide too much power to the owner of the system and creates a degree of centralization. One the other hand, there are other solutions, such as implementing a DAO[1], which would remove the risk of centralization.

Privacy: User privacy is ensured as much as possible in our smart contracts. We chose to not store any personal user information on the blockchain other than name, as this would require functionality which adhers to regulations such as GDPR, which

[1]Decentralized Autonomous Organization

we found to be outside of the scope of this thesis. We did however choose to use a personal-identifier metric when creating a user to show a possibility which is adaptable to the different ways countries have to identify its' citizens. Additionally, we implemented a feature making it only possible for users other than the owner to retrieve information about items when they are marked as for sale, to protect privacy to some degree. However, transactions and interactions on public blockchains are transparent and can be viewed by others, which is a fundamental property of the technology. As we have several events emitted when changes occur to the state of the smart contracts in order to provide functionality, this also makes it possible for anyone to capture, store and analyze this data. As this is just a PoC we did not find it necessary to address this issue, but solutions could be to encryption the data emitted or deploy the system on a private or permissioned blockchain.

7.3 Cost of usage

System Manager

The cost of operation for the system manager, the one who deploys the SystemManager contract and governs the overall system, is dependent on the size of the system. First of, there is a one-time cost for deployment of around $204 plus $104 for each needed DigitalCopy contract, together coming to $308[2]. Then there is a constant cost of $2 for each time there is a need of adding or removing a retailer to or from the system. However, as mentioned these prices fluctuate on a daily basis, thus we find it more reasonable to look at the average price from 2020 onwards. With a cost of $693 plus $354 for each DigitalCopy Contract, for a total of $1047[2]. In addition, there is the cost of $6 for each time a retailer is added or removed from the system. This means that a minimal system with 10 retailers included would cost roughly $1100 to set up, while it is not a lot for most companies, could be a threshold especially without certainty of usage from customers.

Retailers

For retailers to use the system, they will have a deployment cost of $33.6 and a $2.2 cost for changing to the correct DigitalCopy contract, total of $35.8. This would be a one-time cost for the store, and is relatively low compared to the overall operational cost of most companies. Even if a multi-store retailer were to establish a new contract at each of its stores, the cost would still be minimal. However, if we are to look at this price in an average historical aspect, the total price would be $122. While this price is more expensive, it is still not a big expense for most retailers, especially as the items this solution addresses are often of the more expensive type. However,

[2]For a minimal system with one DigitalCopy

more incentive might be needed with data supporting that with the implementation is sought-after and would be widely adapted.

Individual users

As can be seen from the results, the prices for the most important day-to-day functions such as purchase and transfer are around \$20 and \$10[3]. As this concept of digital copies are meant for more valuable items, the cost is minimal in comparison to the price of the item itself. We would argue that the advantages of our system is worth the extra cost. If the purchase and ownership of the NFT makes the seller more trustworthy and desirable when they are interested in selling the item, this extra investment would be cost-effective. However, costumers could also choose not to purchase the digital copy for items where they feel that the pros does not outweigh the cons. Another disadvantage in relevance to cost is that submitting a review cost \$8.5 and burning a NFT cost \$1.6, and as these actions does not provide the user with a personal gain other than contributing to the community and a more tidy overview of owned items, this cost might make these functions seldom used. On the other hand, since the transaction cost increases with the number of owned items, it might be cost-effective to burn old NFTs to reduce this cost. To be able to use the system, the user would also need to create a user, paying around \$4. As this is a one-time fee, we interpret it do be minimal in comparison to the other costs payed when using the system over a long time.

However, we also have to look at the average historical prices of these functions. Based on the table of historical prices in our results, the average cost of a purchase, transfer, creation of a user and burning of a NFT are \$68, \$28, \$12 and \$5 respectively. These prices are quite high and in comparison to the prices mentioned earlier, and could be an important threshold for the adaptation of our suggested system.

Even though the prices of our system are somewhat expensive, we are to argue that the prices reported in Table 6.1 could be acceptable for this kind of system. Although the historical prices indicates a probability that there could be an increase in cost of usage of the system, and that this provides an uncertainty for our proposed system, neither us or anyone else can predict with a hundred percent certainty the future trajectory of these costs.

7.4 Web3 as a solution to issues in the reseller markets

Retail markets

Our solution replaces traditional paper receipts with digital receipts stored on the blockchain. By leveraging NFTs, each product purchase is associated with a unique

[3]Including marking them for sale

digital token that represents the ownership and transaction history. This standardized and immutable record eliminates the issues of inconsistent documentation and provides long-term accessibility. Digital receipts stored on the blockchain are not subject to deletion or loss, ensuring their availability for future reference.

The transparency and verifiability provided by our solution help combat unethical practices in retail, such as selling used products as new. With the immutable transaction history recorded on the blockchain, it becomes evident if a product has been returned and resold. This transparency holds retailers accountable for their actions and safeguards customers against deceitful practices, as the store is unable to mint a new NFT for the same product. Furthermore, the ability to track returned items and maintain an unbroken chain of custody discourages retailers from engaging in such unethical behavior.

Lastly, even though our solution does not offer a direct way to end the discarding and destruction of returned products, the events emitted provide an indirect way of tracking these actions. As all returned products emit a transfer event, it is possible to log all products returned to the retailers. This log can then be used to keep track of the inventory each store is supposed to have. If the store were to burn a lot of NFTs for returned products, this would emit events which could be logged as well.

Secondhand markets

Our solution enhances transparency in the secondhand market by providing a standardized platform where product information and ownership history are recorded on the blockchain. Buyers can access accurate and objective information about a product's condition, including detailed descriptions, repair history, and service records[4]. This reduces subjectivity and ambiguity, enabling buyers to make informed decisions on equal grounds as the sellers.

By leveraging blockchain technology, our solution instills trust in the secondhand market. The decentralized nature of the blockchain eliminates the need to trust individual sellers, as the transaction history and ownership records are independently verified by the network. Additionally, our system allows for the integration of third-party assurance mechanisms, such as seller ratings and reviews, which further enhance trust and provide buyers with valuable information when evaluating sellers. As these third-party assurance mechanisms would follow the seller across platforms, as well as binding their address to their personal identifier, the buyer can be guaranteed a comprehensive view of the seller.

Our solution tackles authenticity concerns by providing the products to be resold in the secondhand market the possibility to be associated with a unique NFT. These

[4]Provided that the item is transferred back and forth with the repair or service provider

NFTs represent the authenticity and origin of the product. Buyers can independently verify the legitimacy of the item by checking the ownership history recorded on the blockchain. This significantly reduces the risk of purchasing counterfeit goods and provides consumers with a higher level of confidence when buying secondhand items.

7.5 Future work

Implementing DAO

Exploring the possibility of in integrating of a Decentralized Autonomous Organization (DAO) into our solution instead of a single system manager can enhance the governance and decision-making processes. By enabling community participation and decentralized decision-making, DAO implementation can further empower users and promote a sense of ownership within the reseller market ecosystem.

User identification

In future work, a valuable addition to our reseller market solution would be the incorporation of a decentralized identification system. By leveraging cryptographic techniques and blockchain technology, a decentralized identification system can provide enhanced privacy, user control over personal information, and improved interoperability across platforms. Users would have ownership and control of their digital identities, selectively sharing relevant information with trusted parties. This approach not only minimizes reliance on centralized authorities but also reduces the risk of data breaches and unauthorized access. By integrating a decentralized identification system, our solution would offer increased privacy, user empowerment, and seamless interoperability, contributing to a more secure and user-centric reseller market ecosystem.

Creating a proxy for upgradability

To facilitate system updates and enhancements without disrupting the functionality and integrity of the existing contracts, implementing a proxy contract architecture can provide a mechanism for seamless upgradability. This approach allows for the introduction of new features, bug fixes, and improvements while preserving the core functionality and user experience. However, careful consideration of how this might breach the aspect of immutability is needed.

Interface for different types of products

Expanding the solution's capabilities to accommodate various types of products beyond the initial basic focus can broaden its applicability. Building an adaptable and intuitive interface that caters to different product categories, such as electronics,

vehicles, fashion, or collectibles, can enhance the user experience and attract a wider user base. These categories can then have different requirements, e.g. VIN number for cars or RFID tags in clothing.

Implementing limitations on DigitalCopy contracts

To ensure the integrity of the reseller market, setting limitations on the number of TrustedSellers per DigitalCopy contract and registering them within the contract can enhance trust and accountability. This approach can prevent malicious actors from abusing the system and maintain a higher level of quality control within the marketplace.

Addressing data emitted in events

Another important consideration for future work is the implementation of data protection measures when emitting events in our reseller market solution. To ensure user privacy and data security, it is crucial to carefully manage the information shared through event emissions. Encryption techniques can be employed to safeguard sensitive data, preventing unauthorized access and maintaining confidentiality. Additionally, limiting the data emitted in events to only essential and non-sensitive information can further mitigate privacy risks.

Advantages of private/permissioned network

While our solution currently operates on a public blockchain, exploring the implementation of a private or permissioned network can provide additional benefits in terms of scalability, privacy, cost, and control. Future work can involve the evaluation and development of a private or permissioned network infrastructure tailored to the specific needs of the reseller market. However, there might be conflicting interest as this would entail more centralization.

Achieving affordable costs

To encourage widespread adoption and ensure affordability for individuals, efforts should be made to optimize costs associated with gas fees and transaction processing. This can involve exploring layer 2 solutions, gas optimization techniques, or collaborations with network validators and infrastructure providers to reduce transaction costs and make the system more economically viable for users. This could also include an empirical study of expected and reasonable cost for such a system from a users point of view.

Chapter 8
Conclusion

In conclusion, our thesis successfully fulfills the outlined objectives through the creation of a PoC solution that effectively addresses key issues prevalent in traditional reseller markets. These issues, including inconsistent documentation, unethical practices, lack of transparency, trust issues, and concerns about authenticity, have been methodically identified and defined. Our solution employs Web3 technology, including the immutability of blockchain and the unique properties NFTs, to create a secure, transparent, and fair reseller market absent of a need for a trusted third party.

By paralleling the transfer of NFTs with the sale of items, each NFT functions as an authenticator of the item, builds trust between parties, and establishes product ownership in a verifiable manner. This reduces the practicality of selling counterfeit items, thereby addressing the aforementioned issues. The blockchain-based storage of product information also enables verification of a product's origin, history, and authenticity by any involved party. Thus, we create a level of transparency that ensures equal access to relevant information about products for both buyers and sellers.

Our PoC is not only representative of the benefits of Web3 technology for the traditional reseller market but also highlights its potential for enhancing services across various industries. The PoC showcases that Web3 technology's usage need not to be intimidating but can have practical and beneficial applications in modern society.

However, despite these significant advantages, the broader adoption and acceptance of Web3 technology are challenged by the volatility and uncertainties associated with gas costs and ether value. The variability of transaction costs, impacted by network congestion and computational complexity, can deter users from engaging with Web3 applications due to perceived unpredictability and prohibitive costs. Moreover, the fluctuating value of ether can affect the affordability and feasibility of employing

Web3 applications for everyday transactions.

To ensure the successful integration of Web3 technology into society, it is necessary to strive for cost predictability and affordability. This could involve developing mechanisms that provide users with more visibility and control over gas costs, such as optimizing transaction efficiency and exploring alternative scaling solutions. Furthermore, establishing stability and confidence in the value of cryptocurrencies like ether can help ease concerns related to cost fluctuations. Collaboration among blockchain developers, network validators, and policymakers will be essential in achieving a sustainable, cost-effective Web3 environment.

As a summary, our thesis demonstrates that modern application development techniques can effectively address the critical issues in traditional reseller markets. By creating a decentralized, transparent, and fair trading platform, substantial benefits can be provided to all parties involved. However, for the broader adoption of these solutions, the fluctuation and uncertainties associated with the costs must be addressed. This thesis hence underlines the importance of further research and cooperation to establish a more predictable and affordable cost structure within the Web3 ecosystem, to fully unleash the potential of blockchain technology in modernizing reseller markets.

References

[1] «Introduction to web3». (2023), [Online]. Available: https://ethereum.org/en/web3/ (last visited: May 12, 2023).

[2] M. White. «Web3 is going great». (2022), [Online]. Available: https://web3isgoinggreat.com/ (last visited: May 9, 2023).

[3] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.

[4] K. Wiegers and J. Beatty, *Software Requirements* (Best practices). Microsoft Press, 2013. [Online]. Available: https://books.google.no/books?id=40lDmAEACAAJ.

[5] H. Frøland and E. Palm, «Security issues in web3», Department of Information Security, Communication Technology, NTNU – Norwegian University of Science, and Technology, Project report in TTM4502, Dec. 2022.

[6] C. A. L. Dictionary and Thesaurus. «Retail». (2023), [Online]. Available: https://dictionary.cambridge.org/dictionary/english/retail (last visited: May 12, 2023).

[7] C. A. L. Dictionary and Thesaurus. «Secondhand». (2023), [Online]. Available: https://dictionary.cambridge.org/dictionary/english/secondhand (last visited: May 12, 2023).

[8] J. A. Yeap, S. K. Ooi, *et al.*, «Preloved is reloved: Investigating predispositions of second-hand clothing purchase on c2c platforms», *The Service Industries Journal*, pp. 1–25, 2022.

[9] Y. Jang and S. Kim, «The factors influencing users' trust in and loyalty to consumer-to-consumer secondhand marketplace platform», *Behavioral Sciences*, vol. 13, no. 3, p. 242, 2023.

[10] K. Goel and N. R. Patel, «Digital receipts: A viable replacement for the printed receipts on thermal papers», *International Journal Of Innovative Research And Development, Guna*, vol. 2, no. 12, pp. 38–41, 2013.

[11] V. Vadde, C. Nithya, and A. P. Surhonne, «An nfc based innovation for paperless retail transactions and digital receipts management», in *2015 Annual IEEE India Conference (INDICON)*, IEEE, 2015, pp. 1–6.

[12] D. L. Nguyen, «Digital receipt system using mobile device technologies», 2008.

[13] V. M. Horvei. «Forbruker-program lurte Elkjøp og kunde med ødelagt TV». (2023), [Online]. Available: https://www.tek.no/nyheter/nyhet/i/nQvx2o/kristian-trodde-tv-en-var-ny-men-paa-innsiden-laa-det-en-skjult-sporingsbrikke (last visited: May 10, 2023).

[14] T. Risberg. «Lefdal solgte "ny" PC med sensitivt innhold». (2016), [Online]. Available: https://www.nrk.no/livsstil/xl/lefdal-solgte-_ny_-pc-med-sensitivt-innhold-1.12899149 (last visited: May 10, 2023).

[15] K. Gibson. «Amazon warehouses trash millions of unsold products, media reports say». (2019), [Online]. Available: https://www.cbsnews.com/news/amazon-warehouses-trash-millions-of-unsold-products-say-media-reports/ (last visited: May 10, 2023).

[16] I. A. Hamilton. «One Amazon warehouse reportedly throws out 130,000 products a week, including some that are brand new. An expert blames its giant third-party retail business.» (2021), [Online]. Available: https://www.businessinsider.com/amazon-throws-away-new-products-waste-third-party-sellers-profitable-2021-6?r=US&IR=T (last visited: May 10, 2023).

[17] Y. Hristova, «The second-hand goods market: Trends and challenges», *Izvestia Journal of the Union of Scientists - Varna. Economic Sciences Series*, vol. 8, pp. 62–71, Jan. 2019.

[18] A. Brooks, «Stretching global production networks: The international second-hand clothing trade», *Geoforum*, vol. 44, pp. 10–22, 2013, Global Production Networks, Labour and Development. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0016718512001091.

[19] Thredup. «2023 resale report». (2023), [Online]. Available: https://cf-assets-tup.thredup.com/resale_report/2023/thredUP_2023_Resale_Report_FINAL.pdf (last visited: May 9, 2023).

[20] D. Guiot and D. Roux, «A second-hand shoppers' motivation scale: Antecedents, consequences, and implications for retailers», *Journal of retailing*, vol. 86, no. 4, pp. 355–371, 2010.

[21] J. A. Heinonen, T. J. Holt, and J. M. Wilson, «Product counterfeits in the online environment: An empirical assessment of victimization and reporting characteristics», *International Criminal Justice Review*, vol. 22, no. 4, pp. 353–371, 2012.

[22] R. Chen, Y. Zheng, *et al.*, «Secondhand seller reputation in online markets: A text analytics framework», *Decision Support Systems*, vol. 108, pp. 96–106, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016792361830037X (last visited: May 9, 2023).

[23] C. Forman, A. Ghose, and B. Wiesenfeld, «Examining the relationship between reviews and sales: The role of reviewer identity disclosure in electronic markets», *Information systems research*, vol. 19, no. 3, pp. 291–313, 2008.

[24] M. Li, K.-K. Wei, *et al.*, «The moderating role of information load on online product presentation», *Information & Management*, vol. 53, no. 4, pp. 467–480, 2016.

[25] S. M. Lee and S. J. Lee, «Consumers' initial trust toward second-hand products in the electronic market», *Journal of computer information systems*, vol. 46, no. 2, pp. 85–98, 2005.

[26] L. N. Leonard and K. Jones, «Trust in c2c electronic commerce: Ten years later», *Journal of Computer Information Systems*, vol. 61, no. 3, pp. 240–246, 2021. [Online]. Available: https://doi.org/10.1080/08874417.2019.1598829.

[27] J. Sihvonen and L. L. M. Turunen, «As good as new–valuing fashion brands in the online second-hand markets», *Journal of Product & Brand Management*, 2016.

[28] J. Joo, «Roles of the buyer's trust in seller in posted-price model of consumer to consumer e-commerce», *Journal of theoretical and applied electronic commerce research*, vol. 10, no. 3, pp. 30–44, 2015.

[29] S. He, B. Hollenbeck, and D. Proserpio, «The market for fake reviews», *Marketing Science*, vol. 41, no. 5, pp. 896–921, 2022.

[30] F. Dini and G. Spagnolo, «Buying reputation on ebay: Do recent changes help?», *International Journal of Electronic Business*, vol. 7, no. 6, pp. 581–598, 2009.

[31] P. Resnick, R. Zeckhauser, *et al.*, «The value of reputation on ebay: A controlled experiment», *Experimental Economics*, vol. 9, Jul. 2003.

[32] J. P. Kennedy, «Counterfeit products online», *The Palgrave handbook of international cybercrime and cyberdeviance*, pp. 1001–1024, 2020.

[33] J. Treadwell, «From the car boot to booting it up? ebay, online counterfeit crime and the transformation of the criminal marketplace», *Criminology & Criminal Justice*, vol. 12, no. 2, pp. 175–191, 2012.

[34] A. Berzon, S. Shifflett, and J. Scheck. «Amazon has ceded control of its site. the result: Thousands of banned, unsafe or mislabeled products». (2019), [Online]. Available: https://www.wsj.com/articles/amazon-has-ceded-control-of-its-site-the-result-thousands-of-banned-unsafe-or-mislabeled-products-11566564990 (last visited: May 11, 2023).

[35] M. Raikwar, D. Gligoroski, and K. Kralevska, «SoK of used cryptography in blockchain», *IEEE Access*, vol. 7, pp. 148 550–148 575, 2019.

[36] R. Sobti and G. Geetha, «Cryptographic hash functions: A review», *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 2, p. 461, 2012.

[37] P. Rogaway and T. Shrimpton, «Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance», in *FSE*, vol. 3017, 2004, pp. 371–388.

[38] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *S. a. handbook of applied cryptography. 2018.*

[39] NIST. «Hash functions». (2017), [Online]. Available: https://csrc.nist.gov/Projects/hash-functions (last visited: May 9, 2023).

[40] P. FIPS, «180-4 secure hash standard (shs)», *US Department of Commerce, National Institute of Standards and Technology (NIST)*, 2015. [Online]. Available: http://dx.doi.org/10.6028/NIST.FIPS.180-4 (last visited: May 9, 2023).

[41] NIST. «Hash functions». (2023), [Online]. Available: https://csrc.nist.gov/news/2023/decision-to-revise-fips-180-4 (last visited: May 9, 2023).

[42] P. FIPS, «Sha-3 standard: Permutation-based hash and extendable-output functions», *US Department of Commerce, National Institute of Standards and Technology (NIST)*, 2015. [Online]. Available: http://dx.doi.org/10.6028/NIST.FIPS.202 (last visited: May 9, 2023).

[43] G. Wood *et al.*, «Ethereum: A secure decentralised generalised transaction ledger», *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[44] P. B. B. OBE. «Do You Know Your Keccak From Your SHA-3, And Your SHAKE From Your KMAC». (2022), [Online]. Available: https://medium.com/asecuritysite-when-bob-met-alice/do-you-know-your-keccak-from-your-sha-3-and-you-shake-from-your-kmac-acc3a9e9f1f2 (last visited: May 9, 2023).

[45] A. Antonopoulos, G. Wood, and G. Wood, *Mastering Ethereum: Building Smart Contracts and DApps*. O'Reilly Media, Incorporated, 2018. [Online]. Available: https://books.google.no/books?id=SedSMQAACAAJ.

[46] «Eddsa and ed25519». (2021), [Online]. Available: https://cryptobook.nakov.com/digital-signatures/eddsa-and-ed25519 (last visited: Jun. 16, 2023).

[47] «Icons from this site has been used for the design of some of the figures in this thesis». (2023), [Online]. Available: Flaticon.com (last visited: Jun. 19, 2023).

[48] M. Crosby, Nachiappan, *et al.*, «Blockchain technology: Beyond bitcoin», Sutardja Center for Entrepreneurship & Technology, Berkeley – University of California, Berkeley, Technical Report made in open classroom lead by Prof. Ikhlaq Sidhu, Aug. 2015.

[49] S. Nakamoto, «Bitcoin: A peer-to-peer electronic cash system», *Cryptography Mailing list at https://metzdowd.com*, Mar. 2009.

[50] Q. Wang, R. Li, *et al.*, «Non-fungible token (nft): Overview, evaluation, opportunities and challenges», *arXiv preprint arXiv:2105.07447*, 2021.

[51] P. Gonserkewitz, E. Karger, and M. Jagals, «Non-fungible tokens: Use cases of nfts and future research agenda», *Risk Governance and Control Financial Markets & Institutions*, vol. 12, pp. 8–18, Sep. 2022.

[52] «ERC-721 non-fungible token standard». (2023), [Online]. Available: https://ethereum.org/en/developers/docs/standards/tokens/erc-721/ (last visited: May 4, 2023).

[53] «Transactions», in *Understanding Bitcoin*. John Wiley & Sons, Ltd, 2014, ch. 6, pp. 77–93. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119019138.ch6.

[54] B. Badr, R. Horrocks, and X. Wu, *Blockchain By Example: A developer's guide to creating decentralized applications using Bitcoin, Ethereum, and Hyperledger*. Packt Publishing, 2018. [Online]. Available: https://books.google.no/books?id=ci59DwAAQBAJ.

[55] C. Staff. «Ethereum explained: A guide to the world supercomputer». (2022), [Online]. Available: https://www.gemini.com/cryptopedia/ethereum-blockchain-smart-contracts-dapps#section-ethereums-d-app-ecosystem (last visited: Jun. 16, 2023).

[56] «Gas and fees». (2023), [Online]. Available: https://ethereum.org/en/developers/docs/gas/ (last visited: May 12, 2023).

[57] «Ether daily price (usd) chart». (2023), [Online]. Available: https://etherscan.io/chart/etherprice (last visited: Jun. 15, 2023).

[58] «Ethereum average gas price chart». (2023), [Online]. Available: https://etherscan.io/chart/gasprice (last visited: Jun. 15, 2023).

[59] K. C. Tran. «What is Web3?» (2019), [Online]. Available: https://decrypt.co/resources/what-is-web-3 (last visited: May 5, 2023).

[60] J. Beck, *What is Web3?*, 2022. [Online]. Available: https://consensys.net/blog/blockchain-explained/what-is-web3-here-are-some-ways-to-explain-it-to-a-friend/ (last visited: May 5, 2023).

[61] Q. Wang, R. Li, *et al.*, «Exploring web3 from the view of blockchain», *arXiv preprint arXiv:2206.08821*, 2022.

[62] Y. Li, C. Tan, *et al.*, «Dynamic blockchain adoption for freshness-keeping in the fresh agricultural product supply chain», *Expert Systems with Applications*, p. 119 494, 2023.

[63] S. Liu, G. Hua, *et al.*, «What value does blockchain bring to the imported fresh food supply chain?», *Transportation Research Part E: Logistics and Transportation Review*, vol. 165, p. 102 859, 2022.

[64] A. Hasselgren, K. Kralevska, *et al.*, «Medical students' perceptions of a blockchain-based decentralized work history and credentials portfolio: Qualitative feasibility study», *JMIR Form Res*, vol. 5, no. 10, e33113, Oct. 2021. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/34677137.

[65] J.-A. H. Rensaa, D. Gligoroski, *et al.*, «Verifymed-a blockchain platform for transparent trust in virtualized healthcare: Proof-of-concept», in *Proceedings of the 2nd International Electronics Communication Conference*, 2020, pp. 73–80.

[66] A. Hasselgren, P. K. Wan, *et al.*, «Gdpr compliance for blockchain applications in healthcare», *arXiv preprint arXiv:2009.12913*, 2020.

[67] A. Hasselgren, K. Kralevska, *et al.*, «Blockchain in healthcare and health sciences—a scoping review», *International Journal of Medical Informatics*, vol. 134, p. 104 040, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1386505619305261X.

[68] B. Shen, X. Xu, and Q. Yuan, «Selling secondhand products through an online platform with blockchain», *Transportation Research Part E: Logistics and Transportation Review*, vol. 142, p. 102 066, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1366554520307171.

[69] S. Huang, D. Hou, *et al.*, «A trustable and traceable blockchain-based secondhand market with committee consensus», in *2023 IEEE 8th International Conference on Big Data Analytics (ICBDA)*, 2023, pp. 72–76.

[70] S. Panda and S. Satapathy, «An investigation into smart contract deployment on ethereum platform using web3.js and solidity using blockchain», in May 2021, pp. 549–561.

[71] A. Nedaković, «Analysis and improvements of VerifyMed — the blockchain solution for virtualized healthcare trust relations», English, Master's thesis, Aalto University. School of Science, 2022, pp. 104+22. [Online]. Available: http://urn.fi/URN:NBN:fi:aalto-202208285217.

[72] Z. Li, X. Xu, *et al.*, «The interplay between blockchain adoption and channel selection in combating counterfeits», *Transportation Research Part E: Logistics and Transportation Review*, vol. 155, p. 102 451, 2021.

[73] «Tangible». (2023), [Online]. Available: https://www.tangible.store/ (last visited: May 19, 2023).

[74] «Auralb». (2023), [Online]. Available: https://auraluxuryblockchain.com/ (last visited: May 19, 2023).

[75] «Stockx». (2023), [Online]. Available: https://stockx.com/ (last visited: May 19, 2023).

[76] R. E. Freeman, *Strategic Management: A Stakeholder Approach*. Cambridge University Press, 2010.

[77] P. B. Kruchten, «The 4+ 1 view model of architecture», *IEEE software*, vol. 12, no. 6, pp. 42–50, 1995.

Appendix A
Web application

This guide demonstrates how to use the implemented user interface for the PoC. We begin by presenting the overall functionality. We then provide step-by-step instructions for various workflows. This user interface is not a finished product but serves as an example of how the system works and simplifies interactions for our users. The instructions to set up the application for local testing can be found in Appendix B.

A.1 User interface

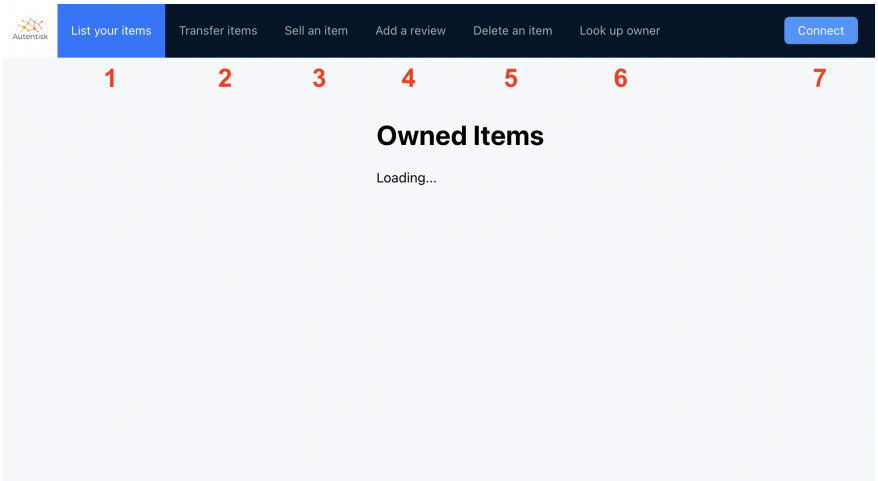


Figure A.1: An overview of the menubar in the proof of concept user interface.

Figure A.1 shows the overall user interface. On the top is a navigation bar with the following sections as numbered in the Figure.

1. Section for navigating for listing owned times.
2. Section for navigating for transferring of NFTs.
3. Section for navigating to panels relevant to Trusted Seller stakeholders.
4. Section for navigating to relevant reviews.
5. Section for navigating to destroying of NFTs.
6. Section for navigating for lookup owner of digital copies.
7. Connect button (Log in).

As we see, there are no listed owned items. If we use the button marked with 7 in Figure A.1, we will connect with our crypto wallet[1], and the website will then list our digital copies stored on the blockchain, which we see in Figure A.2.

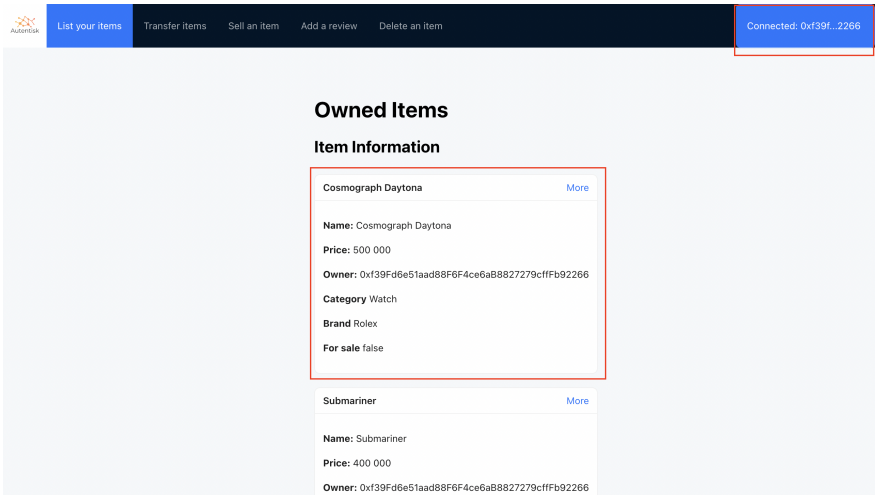


Figure A.2: When connected to a crypto wallet. The user interface will list up all owned NFTs.

[1]Metamask chrome extension is required

A.2 Transferring a digital copy

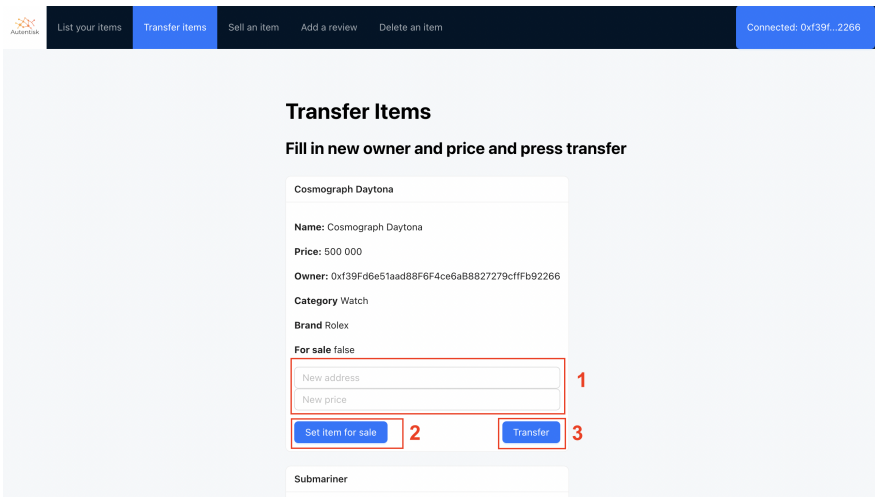


Figure A.3: User interface for transferring an NFT to another user. The current owner needs to fill in the address of the new owner and give a price.

When navigating to the transfer section, the user will get a list of all items they possess. With options for transferring are illustrated in Figure A.3. If we look at Figure A.3, we see that there is an input field, marked with the number 1, where the current owner fills in the new address and price before pressing the transfer button labeled "3" in Figure A.3 which will transfer the ownership to the new address. We also see the "Set item for sale" button, which is marked with label 2 in Figure A.3. This button triggers the NFT over in the "for sale" condition, allowing other users to look up the token, which is a practical secondhand trade for verification of ownership.

When the user presses the transfer button, a Metamask window will appear to conduct the transfer over the Ethereum network, illustrated in Figure A.4.

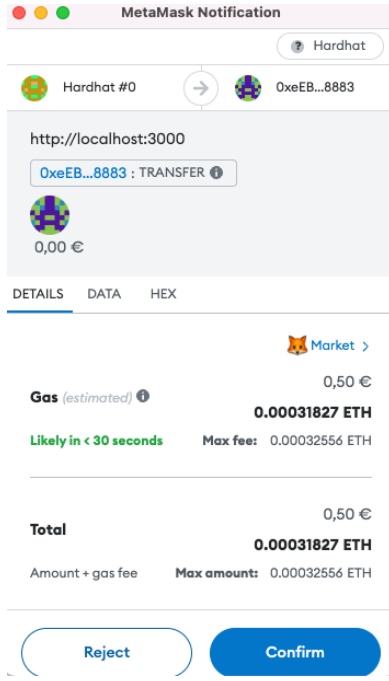


Figure A.4: Metamask panel which handles the signing of transactions done in the user interface.

A.3 Trusted seller

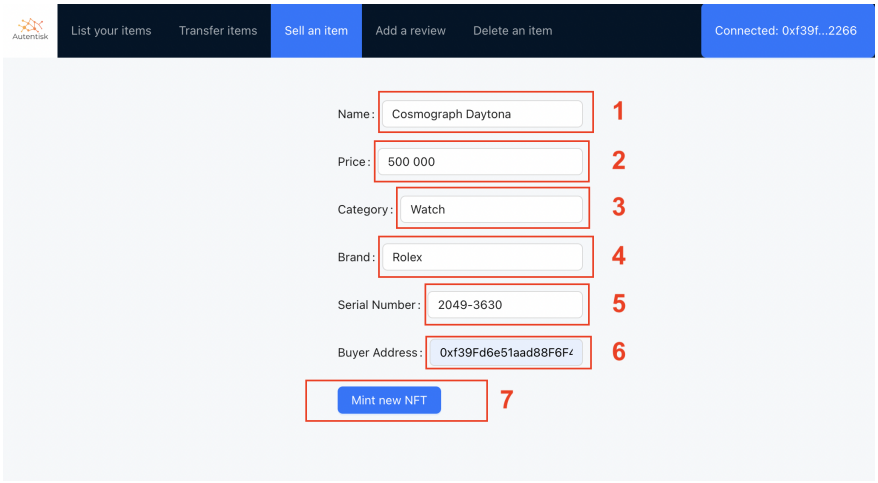


Figure A.5: Section for trusted sellers for creating new NFTs when a user has purchased a product.

From the trusted sellers' point of view, Figure A.5 shows how we can create new NFTs in our system. The system requires the trusted seller to fill in all numbered sections 1-6 before pressing button 7, which generates a new token with the provided information. We see this task as more automated in a well-developed product, but this is outside our scope.

A.4 Making reviews

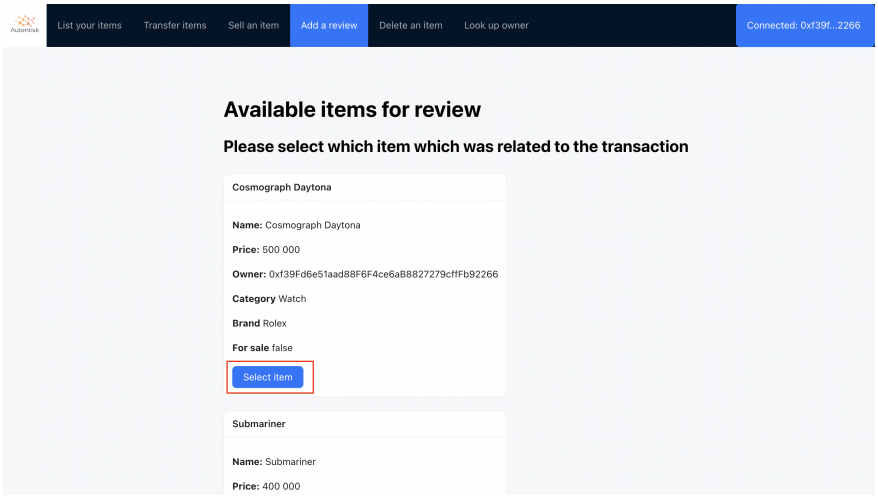


Figure A.6: User interface for giving a review for a transaction. The user will here need to choose which digital copy is related to the transaction.

If the user wants to add a review for a transaction, given they have been a part of it. They should navigate to the review page shown in Figure A.6. First, the user will pick which NFT is related to the trade. They will then press the select item button marked red in Figure A.6. They will then be routed to a review page specific to that transaction which shows the previous owner. The user should then fill in the input fields marked 1 and 2 in Figure A.7 and submit the review by pressing the button marked 3.

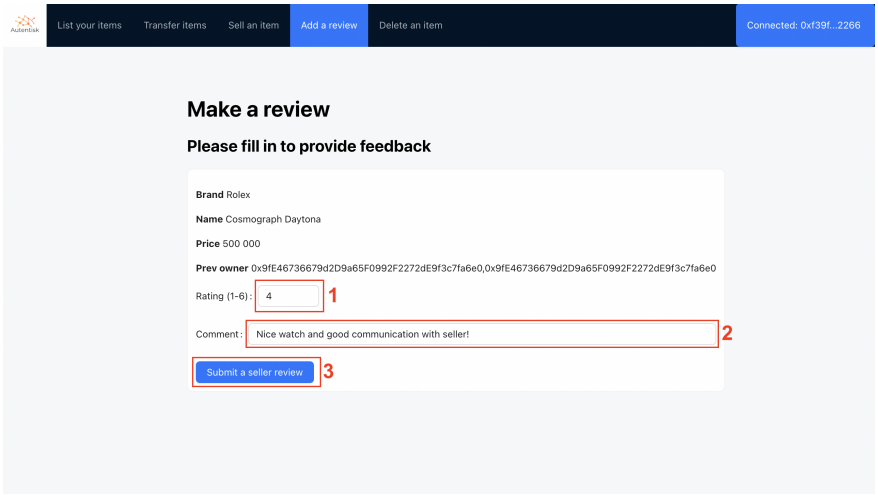


Figure A.7: A user interface for giving feedback on a specific transaction. The user provides feedback through the input fields marked 1 and 2 and submits with the button marked 3.

A.5 Verifying the owner of an item

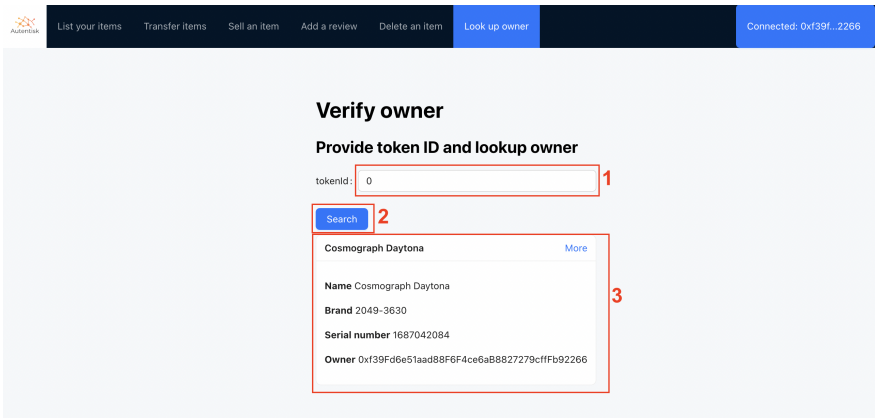


Figure A.8: User interface to obtain information about other users' NFTs. The user provides a tokenId and will in return see the product information.

To gain information about other NFTs, the user will use the Look up owner section in Figure A.8. In the red field marked 1, the user inputs the relevant token ID. Then, given that the targeted NFT is in the state "for sale", the information about it will appear.

Appendix B
Guide for local testing

This appendix instructs how to run and test the proposed solution on a local machine. We refer to Appendix C for a closer look at how the test scenarios are implemented.

B.1 Setup with yarn

To run this application on your local machine, you will need the following tools:

- Node (v18 LTS)
- Yarn ([v1] install with: `npm install -global yarn`)

1. Download the repository from: https://github.com/Autentisk/Localhost
2. cd into the directory `cd Localhost`
3. Install dependencies, run: `yarn install`
4. To run a local blockchain, run `yarn chain`
5. Open a new terminal and run `yarn deploy`

Now the system is deployed on a local running server. To run our test scenarios, you can run the following commands:

- To run scenario 1: `yarn scenario1`
- To run scenario 2: `yarn scenario2`
- To run scenario 3: `yarn scenario3`
- To run scenario 4: `yarn scenario4`

Outputs can be seen in the first terminal, where the `yarn chain` command was executed.

Appendix C
Guide for local testing

Here we show the produced code which represents test scenarios in our validation of the system. For more details, please visit the developed repository on Github: https://github.com/Autentisk/Localhost

Scenario 1

```
// Setting up the user
try {
    await users.createUser("Kari Olsen", "01011991-04200")
} catch (error) {...}

// Making purchases
await trustedSeller.purchase("Cosmograph Daytona", "500 000", "Watch",
    "Rolex", "2049-3630", signerAddress);
await trustedSeller.purchase("Submariner", "400 000", "Watch", "Rolex",
     "2050-3630", signerAddress);

// Trusted Watches tries to sell the same item to Kari again
try {
    await trustedSeller.purchase("Cosmograph Daytona", "500 000", "
        Watch", "Rolex", "2049-3630", signerAddress);
} catch (error) {...}

// Showing proof of of ownership
console.log(await digitalCopy.getOwner(0));
console.log(await digitalCopy.retrieveInformationForDigitalCopy(0));
console.log(await digitalCopy.getOwner(1));
console.log(await digitalCopy.retrieveInformationForDigitalCopy(1));
console.log(await systemManager.retrieveAllOwnedItems(signerAddress));
```

Listing C.1: Selected parts of purchase script

Scenario 2

```
// Buyer tries to get information about an item not for sale
console.log (await digitalCopy.getOwner(0));
```

```
3  console.log (await buyersDigitalCopy.isItemForSale(0));
4  try {
5  console.log (await buyersDigitalCopy.retrieveInformationForDigitalCopy
       (0));
6  } catch (error) {...}
7
8  // Seller puts the item for sale, now the buyer can access it
9  await digitalCopy.putItemForSale(0);
10 console.log(await buyersDigitalCopy.retrieveInformationForDigitalCopy
       (0));
11 console.log(await buyersDigitalCopy.isItemForSale(0));
12
13 // The seller makes the sale and transfers the NFT to the buyer
14 await digitalCopy.transfer(0, buyer.address, "450 000");
15 console.log(await digitalCopy.getOwner(0));
16
17 // The seller tries to sell item back the themself cheaper
18 await digitalCopy.putItemForSale(0);
19 await digitalCopy.transfer(0, sellerAddress, "350 000");
```

Listing C.2: Selected parts of sellSecondhand script

Scenario 3

```
1  // Trying to make a review for wrong item
2  try {
3      console.log(signerAddress, digitalCopy.address);
4      await buyersReviews.newReview(signerAddress, 4, digitalCopy.address
           , 1,
5          "0x...", "Nice watch and good communication with seller!");
6  } catch (error) {...}
7
8  // Making the review for the right item
9  await buyersReviews.newReview(signerAddress, 4, digitalCopy.address, 0,
10     "0x74657374537472696e6720746f20636f6e7665727420746f2062797465736932
           ", "Nice watch and good communication with seller!");
11
12 // Check that the review follows the seller for future sales
13 await digitalCopy.putItemForSale(1);
14 console.log(await systemManager.retrieveListingInformation(digitalCopy.
       address, 1));
```

Listing C.3: Selected parts of reviewTransaction script

Scenario 4

```
1  // Trying to burn the wrong NFT
2  try {
3      await digitalCopy.burn(0);
4  } catch (error) {...}
5
```

```
6 // Burning the right NFT
7 console.log( await systemManager.retrieveAllOwnedItems(signerAddress));
8 await digitalCopy.burn(1);
9 console.log( await systemManager.retrieveAllOwnedItems(signerAddress));
```

Listing C.4: Selected parts of burnNFT script

Appendix D
Address calculation

Sometimes it is necessary to know the public Ethereum address before deployment. This Python script inputs a private key and a "nonce" and calculates the expected public address.

```
from Crypto.Hash import keccak
from rlp import encode

def get_contract_address(address: str, nonce: int) -> str:
    """Get the contract address given an address and nonce."""
    address_bytes = bytes.fromhex(address[2:])  # Remove the 0x prefix

    # RLP encoding for the address and the nonce
    rlp_encoded = encode([address_bytes,nonce])

    # Get the Keccak-256 hash of the RLP encoded
    keccak_hash = keccak.new(digest_bits=256)
    keccak_hash.update(rlp_encoded)
    hashed = keccak_hash.digest()

    # The address without checksum
    contract_address_lower = hashed[-20:].hex()

    # Calculate the Keccak-256 hash of the lowercase address
    keccak_hash_address = keccak.new(digest_bits=256)
    keccak_hash_address.update(contract_address_lower.encode())
    keccak_hash_address_digest = keccak_hash_address.hexdigest()

    checksum_address = '0x'
    for i in range(len(contract_address_lower)):
        if int(keccak_hash_address_digest[i], 16) >= 8:
            checksum_address += contract_address_lower[i].upper()
        else:
            checksum_address += contract_address_lower[i]
    return checksum_address

if __name__ == '__main__':
    address = input("Enter the wallet address: ")
```

```
34      nonce = int(input("Enter the nonce: "))
35      contract_address = get_contract_address(address, nonce)
36      print(f'The future contract address will be: {contract_address}')
```

Listing D.1: Script to calculate Ethereum addresses in python

Appendix E
Academic paper

The work in this thesis resulted in a paper describing the proposed system. The paper was submitted right before the delivery of this thesis. The first draft of this paper complements the remainder of this appendix.

Web3 and Blockchain for Modernizing the Reseller Market

Håkon I. Frøland, Edward Palm, Katina Kralevska, and Danilo Gligoroski
Department of Information Security and Communication Technology (IIK)
Norwegian University of Science and Technology - NTNU, Trondheim, Norway
Email: {haakonif, edwardp, katinak, danilog}@ntnu.no

***Abstract*—This paper presents a Proof-of-Concept (POC) that leverages Web3 technology, specifically blockchain and Non-Fungible Tokens (NFTs), to create a secure, transparent, and fair reseller market without the need for a trusted third party. The POC demonstrates that NFTs can be used to authenticate items, build trust between parties and establish verifiable product ownership. By paralleling NFT transfers with the sale of items, the practicality of selling counterfeit goods is reduced. Furthermore, the blockchain-based storage of product information enables verification of origin, history, and authenticity by any involved party, fostering transparency and equal access to information for buyers and sellers. We demonstrate that Web3 technology can effectively address critical issues in traditional reseller markets, offering substantial benefits to all parties involved. To the best of our knowledge, this is the first POC designed to enhance trust in a virtualized secondhand market by utilizing Web3 technology.**

***Index Terms*—Web3, NFT, Blockchain, Reseller market, Secondhand market**

I. Introduction

Reseller markets play a significant role in our daily lives, providing individuals with platforms to sell goods and services to one another. These markets offer a valuable arena for used items to find new owners, and people can sell items they no longer need, which is beneficial for the society and the environment. However, these markets have various issues. One of the most significant obstacles traditional reseller markets face today is fake or counterfeit products. It can be difficult for buyers to identify these items, and they are often left in unfavorable position. Establishing trust between buyers and sellers who have no affiliation is generally challenging, causing potential trades to fall through. The lack of certainty in the item's authenticity, the seller's morale, and the number of counterfeit products in circulation can discourage the buyer. Moreover, reseller markets often lack transparency, making it challenging for buyers to assess the true value of a product. Overall, traditional reseller markets suffer from multiple issues that harm both buyers and sellers.

The technology behind Web3 [1] with decentralization and blockchain is often considered too technical and advanced, heightening the threshold for widespread adoption in society. Unfortunately, there have been several cases of scams using blockchain today, which fuel mistrust rather than trust in blockchain [2], as it is the core concept of the technology. However, Web3 has numerous use cases which can benefit society and help build bridges between people, and when utilized correctly has the potential to improve significantly existing services. By bridging the gap between the challenges faced by traditional reseller markets and the untapped potential of Web3 technology, we aim to create more secure, efficient, and reliable platforms for individuals to trade secondhand goods.

This paper demonstrates the benefits of leveraging modern application development techniques, implementing a decentralized application for a secure, transparent, and fair reseller market. By developing a proof-of-concept (POC) using Web3 and its features, we aim to dispel misconceptions about Web3's complexity and underline its practical applications in modern society. The contributions of this paper are as follows:

- We identify the critical issues in the traditional reseller market, and we model the data sharing and trust establishment in the virtualized reseller market.
- We implement a PoC based on Web3 technology, encompassing blockchain technology and decentralized applications. We make this PoC publicly available[1].
- We demonstrate the potential of Web3 technology in the context of the secondhand market.

To our knowledge, this is the first POC designed to enhance trust and authenticity in a virtualized secondhand market by utilizing Web3 technology.

This paper is organized as follows: Section II provides the theoretical background and related works. Section III presents the principles of the Web3 application model, followed by a detailed description of the design and implementation of the application in Section IV. Section V presents the finished POC and results from the system's deployment. Finally, Section VI concludes the paper and highlights future research directions.

II. Background and Related Work

We present the background of different markets and the related work. An extensive overview of the use of modern cryptography in blockchain systems is given in [3].

A. Market Types: Retail vs Secondhand

Retail and secondhand markets represent separate segments of the larger commercial ecosystem. Retail markets, or "the activity of selling goods to the public, usually in shops" [2], supply consumers with new products from a variety of

[1]https://github.com/Autentisk
[2]https://tinyurl.com/2yhj7pzt, Cambridge University Press

brands. Secondhand markets, on the other hand, focus on the resale of used or pre-owned goods, offering economic and sustainability benefits. The primary distinction between the two is the condition of the products for sale. Retailers supply new items while secondhand markets offer an affordable alternative and the opportunity to find unique or vintage items.

1) Retail Market Challenges: In retail markets, managing receipts and documentation can be a hassle accompanied with risks. Consumers receive receipts for expensive items as proof of purchase and for potential returns or future resale. Digital solutions exist [4], [5], but there is no defined standard and concerns around accessibility and legitimacy remain. For instance, receipts stored via email may be lost among daily emails or deleted by providers. Also, practices such as reselling returned items as new have been reported, as well as unethical product disposal of returned items, raising concerns about retailers' transparency and their impact on the environment [6].

2) Secondhand Market Significance and Challenges: The secondhand market, with its rich history and significant economic impact, continues to thrive [7], stretching global production networks [8]. Despite the depreciating value of pre-owned goods, the market offers affordability and profitable resale potential for all types of products. Guiot and Roux's 8-factor scale [9] reveals the motivations behind secondhand purchases, highlighting the importance of economic, ethical, and hedonic motives. However, these markets also present challenges related to transparency, trust, and authenticity.

a) Transparency: In secondhand markets, online platforms typically facilitate transactions, but the lack of oversight introduces concerns about transparency and reliability. Product descriptions, often subjective and ambiguous, pose a challenge for accurately determining an item's condition [10]. Sellers' extra product knowledge may disadvantage buyers. Hence, a seller's reputation becomes crucial for reducing buyers' risks [11].

b) Trust: In the online used market, trust is twofold-trust in the seller and trust in the product [12]. Establishing this trust can be helped by third-party assurance seals [13]. Moreover, buyer-seller rating systems play a role in fostering trust between the two parts but faces challenges of potential manipulation by the low entry barrier for new accounts as well as fake bought reviews [14], [15].

c) Authenticity: Purchasing secondhand goods raises concerns about their authenticity and quality. Counterfeit goods, which pose potential risks to safety [16], are difficult to identify due to deceptive online practices. Online platforms often lack adequate regulation, leaving consumers vulnerable to counterfeit purchases [10], [17].

B. Related Work

There are numerous papers investigating the benefits of blockchain technology and its implementation in various fields. Shen et al. [18] explored blockchain's role in secondhand sales, focusing on supply chain, pricing, and quality strategies. They did not consider the blockchain's costs, a crucial factor in our solution. Huang et al. [19] addressed secondhand market issues with a novel framework, incorporating smart contracts and IPFS. However, the solution does not consider NFTs, immutability, or security concerns. Panda and Satapathy [20] investigated Ethereum platform usage for dApp development in general. On the other hand, in our approach, we are exploring Web3's potential benefits in reseller markets while developing a POC to address identified needs. Tangible [21] uses NFTs for tokenizing real-world assets. They focus on luxury items stored in a vault. NFTs are redeemable for physical items but disappear upon redemption, not aiding resale. Aura Blockchain Consortium [22] is a luxury-only platform using NFTs to fight counterfeits. Operates on a permissioned blockchain, limiting inclusivity. For comparison, our solution operates on a public, permissionless Ethereum blockchain, offering more openness. StockX [23] offers a platform for resale of coveted items. They authenticate items but do not utilize blockchain or decentralization i.e. they heavily rely on StockX as a trusted third party. Table I summarizes the works mentioned in this section and compares them with our contribution.

TABLE I
SUMMARY OF RELATED WORK

Work	Retail/Secondhand	Blockchain	NFTs	Web3
Inbook [20]	Neither	✓	X	X
Paper [18]	Secondhand	✓	X	X
Paper [19]	Secondhand	✓	X	X
StockX [23]	Secondhand	X	X	X
Tangible [21]	Retailer	✓	✓	X
Aura B.C. [22]	Both	✓	✓	X
This paper	Both	✓	✓	✓

III. WEB3 APPLICATION MODELING PRINCIPLES

A. Defined concerns and Web3 solutions

1) Evidence of Trust: The issues surrounding retail and secondhand markets are primarily rooted in a lack of transparency and trust, as well as inconsistencies in documentation and practices. In the retail sector, these problems manifest in the form of non-standardized receipts and unethical return policies. On the other hand, the secondhand market is plagued by vague product descriptions, imbalanced access to information, and unreliable seller reputation metrics.

Integrating Web3 principles into these markets' infrastructure can notably address these problems. Blockchain technology offers a transparent, tamper-proof record of transactions, addressing documentation inconsistencies in the retail market. Smart contracts, acting as a decentralized trust and reputation system, can help tackle the transparency and trust issues in secondhand markets. These technologies enable users to verify their identities, create trustworthy transactions, and make informed decisions based on shared feedback and ratings.

2) Evidence of Authenticity: Authenticity concerns are a major drawback in secondhand markets, with prevalent issues such as counterfeits and questionable product quality. These problems are amplified by the lack of effective regulations and enforcement by e-commerce platforms.

One of Web3's features is the use of NFTs to establish product ownership. NFTs provide a decentralized, verifiable proof of ownership that is not tied to a specific retailer or platform. Users can claim ownership securely through their digital wallets, ensuring the portability and accessibility of their assets across different platforms.

Moreover, the use of permissionless blockchains allows for continued, independent asset management even if a specific retailer or platform ceases operations. This promotes resilience in the face of potential service disruptions and further assures users of the security and longevity of their assets.

B. Functional requirements

TABLE II
FUNCTIONAL REQUIREMENTS AND THEIR DESCRIPTION

Functionality	*Description*
Purchase Item and Receive NFT	Upon purchasing items from a retailer, users should receive an NFT that serves as a digital proof of ownership, storing relevant product details like name, price, brand, category and serial number.
Overview of NFTs	Users should have access to an overview of all their NFTs and the associated information for potential resale.
Access Product Information	By referencing an NFT marked for sale, users should be able to access product details such as description, purchase date, ownership history, and any linked warranties or services.
Transaction and Transfer of NFT	During a transaction, the user should be able to securely transfer the NFT to a new owner. This ensures that the digital record corresponds with the physical exchange and updates with new transaction data.
Rate a Transaction	Post-transaction, the buyer should be able to review their experience, providing feedback for the seller or retailer which is publicly available across e-commerce platforms.
Remove Broken Products	Users should have the option to remove the NFTs of worn out or broken products, preventing clutter from obsolete NFTs.

C. Non-functional requirements

Scalability: The system must support a growing number of users and transactions without major performance deterioration.

Robustness: The system's smart contracts should manage unexpected situations or inputs, safeguarding the platform's integrity. NFTs within the system should be designed to exist indefinitely, preventing spontaneous disappearance or destruction.

Security: The system should offer protection against fraud and user data breaches. Key elements include secure storage, identity verification, and defined access roles for system administrators, trusted retailers, and NFT owners.

Reliability: A high-availability, low-downtime operation is required to ensure consistent user access.

Performance: Swift responses to user requests and efficient transaction management are paramount. This involves optimization of smart contracts for resource usage and gas cost, leading to quick, low-cost transactions.

Usability: An intuitive user interface is crucial for broad adoption. This involves a simple workflow and clear instructions for NFT transactions, as well as an easily accessible overview of owned NFTs.

Upgradability: The smart contracts should feature a mechanism for logic upgrades, enabling bug fixes and new feature additions without full system redeployment.

Privacy: Smart contracts should respect user privacy, minimizing the exposure of personal information.

IV. DESIGN AND IMPLEMENTATION

Our decentralized system introduces an approach to track and validate the ownership of physical goods through a corresponding digital copy, leveraged by the Ethereum blockchain. When a customer purchases an item in-store, a corresponding digital copy, represented as a Non-Fungible Token (NFT), is minted and assigned to the customer's cryptocurrency address. This NFT, transferred to the customer's digital wallet, serves as a proof of ownership, see Figure 1. This digital copy, besides demonstrating ownership, encapsulates key information about the product, ensuring traceability and authenticity.

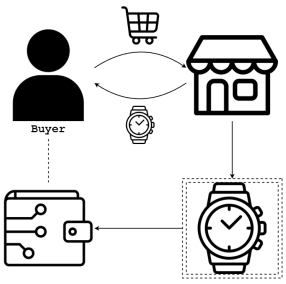


Fig. 1. This model illustrates the process of a user buying a watch in the retail market using an NFT. The retail seller creates an NFT and transfers ownership of the NFT to the buyer's crypto wallet.

The system architecture is designed to facilitate secure transactions and transparent ownership validation. The initial step of an NFTs lifecycles starts when a retailer invokes the **purchase** function, passing the customer's Ethereum address as input. This action mints a new NFT under the customer's address. To prevent unregulated NFT creation, the minting process is limited to authorized retailers, preserving the platform's authenticity. An authorized seller is implemented as a smart contract called: *Trusted Seller*.

If the owner decides to resell the product, they call the **transfer** function with the token ID and the new owner as input, transferring the ownership to a new user. On the secondhand market, the seller hands over the item and its corresponding NFT, establishing ownership for the new buyer, see Figure 2.

To list the item for sale, a user calls the putItemForSale function with the tokenID as input. A potential buyer can then call the **getOwner** function with the tokenID to verify the current owner and the item's authenticity. This structure makes it unattractive to sell a product without the corresponding digital copy NFT, ensuring the system's integrity. Users can

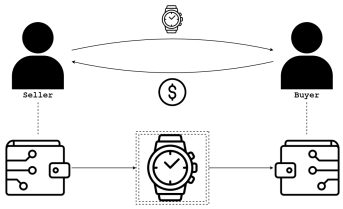


Fig. 2. This model demonstrates the sale of a watch in the secondhand market. The seller transfers ownership of the NFT to the new owner's crypto wallet as part of the transaction.

verify the item's legitimacy through the associated NFT in their crypto wallet. The NFT serves as a digital twin of the physical product, offering an immutable proof of ownership and a transparent track record.

After a transaction, a user can submit a review by using the **newReview** function with the transaction as input, allowing the system to gather customer feedback and build a reliable review repository.

From a logical perspective, we have four components:

- **User:** Requires an identity in the Ethereum blockchain, accomplished by assigning a unique address to each user.
- **Digital Copy:** A digital representation of a physical item, managed through a mint function, transfer function, and burn function for creating, transferring, and deleting items, respectively.
- **Trusted Seller:** Retailers who generate and transfer digital NFT copies upon a purchase using the purchase function.
- **Reviews:** Manages user reviews and user scoring on the blockchain.

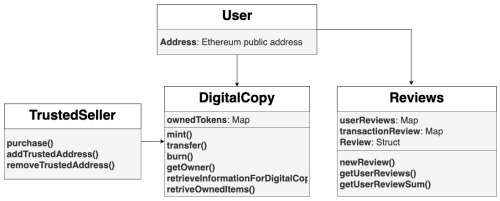


Fig. 3. Logical view of the interactions between stakeholders and the corresponding tasks they perform in our architecture

Summary of interconnections between software and hardware components:

- **Ethereum Node:** Stores smart contracts and maintains the system state.
- **Smart Contracts:** Include "SystemManager", "TrustedSeller", "DigitalCopy", "Users", and "Reviews", each with distinct roles in the system.
- **Web-Browser Unit:** Facilitates interaction with the Ethereum blockchain via the ethers.js API.

A. Implementation of smart contracts

In the implementation stage, we focus on smart contracts, developed in Solidity. We have actively used modifiers as access control to secure against fraudulent system use. The system comprises several interconnected smart contracts, each with a unique role: SystemManager, TrustedSeller, DigitalCopy, Reviews, and Users. Instead of discussing the several hundred lines of code for each contract, we summarize their primary functionality, while the complete code is available on our open Github repository.

1) SystemManager Contract: Capable of approving and evicting trusted sellers, ensuring only credible sellers are operational within the ecosystem. Comprised of two private mappings, the SystemManager forms a list of trusted sellers and tracks all active DigitalCopy contracts. It verifies retailer existence and allows the deployment of multiple DigitalCopy contracts. Additionally, it initiates the deployment of Users and Reviews contracts. The ' Authorized ' modifier limits access to functions such as adding and removing TrustedSeller contracts.

2) TrustedSeller Contract: Designed for retailers, enabling the minting of new digital copies through the purchase function. The purchase function incorporates three requirements: verifying buyers' user existence, confirming that the purchased product does not already possess a related NFT and ensuring only authorized accounts act on the reseller's behalf. Its constructor initializes the trusted seller's name and the SystemManager contract's address. The contract includes additional functions for retrieving the retailer's name, executing transfers, and managing address authorizations. Additionally, it provides the flexibility to modify and disclose the associated DigitalCopy contract by authorized users.

3) DigitalCopy Contract: The digital copy contract is an ERC721 contract developed for providing the users with control and easy management over their NFTs. It defines structures for storing NFT information and includes functions for minting, transferring ownership, destroying, and retrieving information.

4) Users Contract: Manages user accounts and identities on-chain. This contract ensures that an identity cannot have multiple users, maintaining a one-to-one relationship between physical users and on-chain accounts. It creates a hash of every new user and stores it inside a mapping. It provides functionality for registering new users, and assists the other contracts with verifying user existence.

5) Reviews Contract: Stores all user reviews on-chain, providing a transparent and immutable feedback mechanism. It contains functions for creating a new review as well as fetching and calculating the sum of all review ratings about a user.

V. Testing and Results

A. Deployment

In Figure 4, we detail the deployment of the SystemManager contract on the sepolia testnet, pointing out five key aspects. The transaction hash is the unique identifier for this state change in the smart contract. The "From" field represents the public address we used to deploy the contract, and the "To" field matches the calculated smart contract address from Section 5.3. The gas price at deployment was approximately 1.5 gwei. The gas usage for deployment was 5,155,282 gas. The nonce, though less critical, is visible and set to zero, marking the account's first interaction with this network.

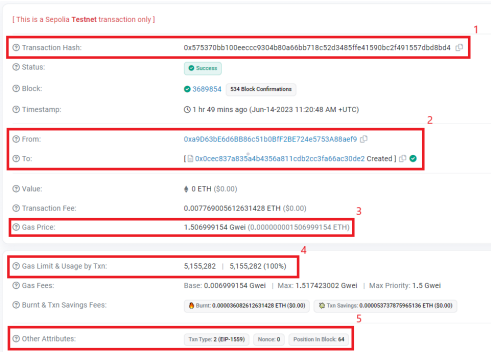


Fig. 4. Screenshot of SystemManger deployment on Etherscan

B. Scenarios

We constructed four test scenarios to test the system's capabilities. We introduce one fictive trusted seller: Trusted watches, and two users: Ola and Kari.

1) **NFT Minting and Duplicate Prevention:** Here, we examine the retailer Trusted Watches as they attempt to resell a previously minted NFT watch. This scenario prevents duplicate NFT creation and upholds the system's integrity. It fulfils the functional requirements of "Purchase Item and Receive NFT" and "Overview of NFTs".
2) **Listing, Selling, and Ownership Transfer:** This scenario tracks Ola's unsuccessful retrieval of a not-for-sale item, followed by Kari's successful sale and ownership transfer. We observe the process of listing, selling, and transferring ownership of an item. It fulfils the functional requirements of "Access Product Information" and "Transaction and Transfer of NFT".
3) **Review and Rating:** Ola writes a product review impacting Kari's seller rating. We analyze the review system's overall function and the impact of reviews on seller ratings. It fulfils the functional requirement of "Rate a Transaction".
4) **NFT Deletion:** Kari unsuccessfully tries to delete Ola's NFT and later successfully burns her damaged NFT. We evaluate the NFT deletion process and its role in maintaining system integrity. It fulfils the functional requirement of "Remove Broken Products".

C. Quality attributes

Our Ethereum-based implementation features scalability, supporting concurrent interactions and increasing transactions without significant performance degradation. Transaction throughput, however, depends on network congestion and user-paid gas price.

Robustness is achieved through the immutability of smart contracts, which are designed to handle unexpected situations using modifiers and require statements. For instance, the NFT cannot be destroyed unintentionally or multiple NFTs minted for the same product.

Security is prioritized with contracts designed to ensure only authorized users access certain operations. The SystemManager contract controls trusted retailer inclusion and DigitalCopy contract functionality but does not have any control over the NFTs. Only trusted retailers can mint NFTs, and only current NFT owners can transfer ownership. System compromise could occur if the SystemManager owner's wallet is breached.

Reliability stems from the Ethereum network's own reliability, promising high availability and minimal downtime. Considering Ethereum's decade-long operation and widespread use, we find it sufficiently reliable.

Performance optimization was considered in smart contract design, favoring mappings over arrays to reduce computational cost. While there's room for optimization in gas cost for transfers involving multiple items, our PoC solution provides a satisfactory user experience with swift transaction handling.

Usability-wise, we've created a functional interface showcasing system core functionality. The design allows intuitive user interaction, even for those unfamiliar with blockchain technology. The system is accessible via Ethereum-compatible wallets and can be incorporated into existing e-commerce platforms. Trust in blockchain technology from society and wallet system dependency remain potential challenges.

User privacy is upheld, with no personal information beyond names stored on the blockchain, although verification of user individuality, such as verifying personal identifier, has not been implemented. Visibility of items is restricted unless they're for sale, although the public nature of blockchain transactions means interactions can be viewed and analyzed. Privacy considerations for future development might include data encryption or deployment on a permissioned blockchain.

D. Gas Cost

Table III presents the gas and USD costs incurred during various scenarios. The costs hinge on gas price and gwei value (assumed at 22 gwei and $0.0000018). Gas costs differ due to the diverse computational demands of operations. 'Deploy SystemManager,' the most computation-heavy operation, costs 5,155,282 gas or $204. This initial setup cost is a one-time expense. Other high-cost operations, such as the deployment of 'DigitalCopy' and 'TrustedSeller,' are also one-time setup costs.

Frequent operations like 'Purchase 1' and 'Purchase 2' have high individual costs ($20.2 and $19). The difference in gas usage here, around 31,000 gas, is due to different variable lengths. A longer name, such as "Cosmograph Daytona," requires more computational resources than a shorter one like "Submariner." Simulating all variables set to an unrealistic length of 69 words, the purchase cost has a maximum cost of $108.

User creation operations, 'Create user in Users (1st)' and 'Create user in Users (2nd),' exhibit relatively low, consistent gas costs. The computation cost depends on name length and user identifier and is a one-time registration fee. For a highly unlikely scenario with 69 words in both name and identifier, the cost remains manageable at $16.6.

The deployment cost for a system manager totals to $308 implementation cost for a retailer equals $36 and for the user purchase costs are around $20 and later transfers of NFTs $10 footnoteFor a minimal system using previously stated prices. However, calculating these cost on average historical prices from 2022 onwards, they are roughly $1047, $122, $68 and $28 respectively.

TABLE III
GAS COSTS FOR DIFFERENT OPERATIONS AND THEIR COST IN USD (SORTED BY COST) USING THE GAS COST OF 22 GWEI AND GWEI VALUE OF $0.00000180

Operation	Gas Cost	Cost in USD
Deploy SystemManager	5,155,282	$204
Deploy DigitalCopy in SystemManager	2,636,386	$104
Deploy TrustedSeller	847,738	$33.6
Purchase 1	511,163	$20.2
Purchase 2	479,667	$19
Transfer item in DigitalCopy	214,148	$8.5
New review in Reviews	213,752	$8.5
Create user in Users (2nd time)	92,019	$3.6
Create user in Users (1st time)	91,983	$3.6
Change DigitalCopy in TrustedSeller	56,044	$2.2
Add TrustedSeller in SystemManager	47,247	$1.9
Put item for sale in DigitalCopy (2nd)	46,012	$1.8
Put item for sale in DigitalCopy (1st)	46,000	$1.8
Burn item in DigitalCopy	39,520	$1.6

1) Transaction Cost: We further experimented with transaction cost variations. The cost of transfers primarily hinges on the number of items owned by a user, in this case, one who owns 50 identical items. These items are transferred sequentially to another user, with costs reducing as the owned item list shortens. The position of an item in the list influences the gas required for transfer: items further back in the list cost more to transfer as the longer iterations require more computational operations.

The first transfer also incurs a higher cost due to the setup of the item list for the receiving user. This setup demands about an extra 36,500 gas. After this setup, each transfer cost decreases consistently by 2,339 gas, reflecting the reduction in items in the list.

The last transfer is slightly less costly, requiring 168,195 gas (or $6.7), due to no need for list iteration, thus demanding less computational effort and gas.

This makes it possible to calculate the worst-case expected cost of a transfer depending on the amount of NFTs the seller has. Disregarding the cost of the initial setup if the receiver has yet to receive an NFT as well as the decrease in the cost of transferring the last item, we get these equations:

$$\widehat{GasCost} = NumberOfOwnedItems \times 2400 + 177795$$

$$\widehat{Price} = GasCost \times GasPrice \times \frac{EtherValue}{10^9}$$

Using a gas price of 22 gwei, an ether value of $1,800 and having 100 items:

$$\widehat{Price} = (100 \times 2400 + 177795) \times 22 \times \frac{1800}{10^9} = 16.54$$

This results in a worst-case cost of $16.5 for this transfer. It is worth mentioning that these are estimators, and other factors, such as the item's price, also affect the gas cost.

Using the historical charts of gas prices, we performed a simulation for the function costs of our system. The simulation is shown in Figure 5.

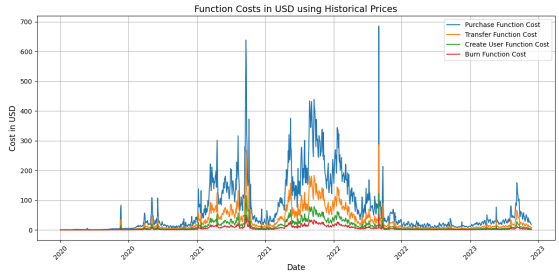


Fig. 5. Simulation charts of functions cost based on historical gas prices in USD.

VI. CONCLUSIONS

In conclusion, our research underscores the successful execution of a POC solution which effectively navigates key issues pervasive in traditional reseller markets, covering troublesome documentation, unethical practices, lack of transparency, trust issues, and authenticity concerns. The system that we developed successfully satisfies functional and non-functional aspects, representing a transparent, trustworthy, and authentic alternative to existing practices in the reseller markets. Our approach leverages Web3 technology, incorporating the blockchain's immutability and the unique properties of NFTs to create a secure, transparent, and fair reseller market, eliminating the need for a third party. Each NFT, tethered to the sale of an item, functions as an authenticator, fostering trust between involved parties and establishing verifiable product ownership. This solution significantly reduces the likelihood of counterfeit item sales and builds trust.

Future work includes exploring DAO integration, decentralized identification systems, upgradability options, adaptable interfaces for different product categories, contract limitations, data protection during event emission, private/permissioned networks, and affordability optimization.

REFERENCES

[1] (2023) Introduction to Web3. [Online]. Available: https://ethereum.org/en/web3/

[2] M. White. (2022) Web3 is going great. [Online]. Available: https://web3isgoinggreat.com/

[3] M. Raikwar, D. Gligoroski, and K. Kralevska, "SoK of used cryptography in blockchain," *IEEE Access*, vol. 7, pp. 148 550–148 575, 2019.

[4] K. Goel and N. R. Patel, "Digital receipts: A viable replacement for the printed receipts on thermal papers," *International Journal Of Innovative Research And Development, Guna*, vol. 2, no. 12, pp. 38–41, 2013.

[5] V. Vadde, C. Nithya, and A. P. Surhonne, "An nfc based innovation for paperless retail transactions and digital receipts management," in *2015 Annual IEEE India Conference (INDICON)*. IEEE, 2015, pp. 1–6.

[6] I. A. Hamilton. (2021) One Amazon warehouse reportedly throws out 130,000 products a week, including some that are brand new. An expert blames its giant third-party retail business. [Online]. Available: https://tinyurl.com/2udpeyh6

[7] Y. Hristova, "The second-hand goods market: Trends and challenges," *Izvestia Journal of the Union of Scientists - Varna. Economic Sciences Series*, vol. 8, pp. 62–71, 01 2019.

[8] A. Brooks, "Stretching global production networks: The international second-hand clothing trade," *Geoforum*, vol. 44, pp. 10–22, 2013, global Production Networks, Labour and Development. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0016718512001091

[9] D. Guiot and D. Roux, "A second-hand shoppers' motivation scale: Antecedents, consequences, and implications for retailers," *Journal of retailing*, vol. 86, no. 4, pp. 355–371, 2010.

[10] J. A. Heinonen, T. J. Holt, and J. M. Wilson, "Product counterfeits in the online environment: An empirical assessment of victimization and reporting characteristics," *International Criminal Justice Review*, vol. 22, no. 4, pp. 353–371, 2012.

[11] C. Forman, A. Ghose, and B. Wiesenfeld, "Examining the relationship between reviews and sales: The role of reviewer identity disclosure in electronic markets," *Information systems research*, vol. 19, no. 3, pp. 291–313, 2008.

[12] S. M. Lee and S. J. Lee, "Consumers' initial trust toward second-hand products in the electronic market," *Journal of computer information systems*, vol. 46, no. 2, pp. 85–98, 2005.

[13] L. N. Leonard and K. Jones, "Trust in C2C Electronic Commerce: Ten Years Later," *J. of Computer Information Systems*, vol. 61, no. 3, pp. 240–246, 2021. [Online]. Available: https://tinyurl.com/2p9ep878

[14] S. He, B. Hollenbeck, and D. Proserpio, "The market for fake reviews," *Marketing Science*, vol. 41, no. 5, pp. 896–921, 2022.

[15] F. Dini and G. Spagnolo, "Buying reputation on ebay: Do recent changes help?" *International Journal of Electronic Business*, vol. 7, no. 6, pp. 581–598, 2009.

[16] J. P. Kennedy, "Counterfeit products online," *The Palgrave handbook of international cybercrime and cyberdeviance*, pp. 1001–1024, 2020.

[17] J. Treadwell, "From the car boot to booting it up? ebay, online counterfeit crime and the transformation of the criminal marketplace," *Criminology & Criminal Justice*, vol. 12, no. 2, pp. 175–191, 2012.

[18] B. Shen, X. Xu, and Q. Yuan, "Selling secondhand products through an online platform with blockchain," *Transportation Research Part E: Logistics and Transportation Review*, vol. 142, p. 102066, 2020. [Online]. Available: https://tinyurl.com/bdf9paze

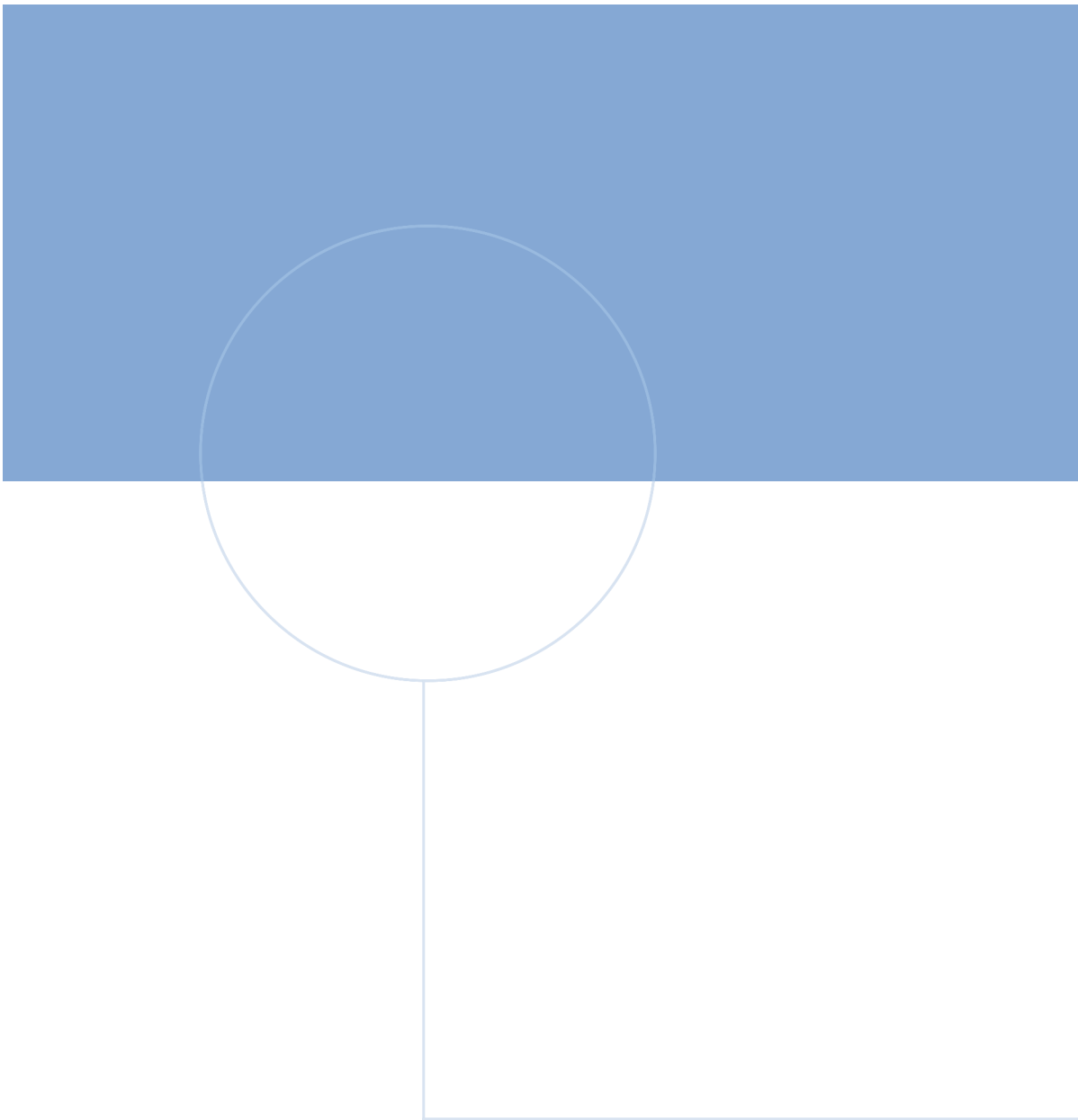
[19] S. Huang, D. Hou, Z. Peng, Y. Dong, and J. Zhang, "A trustable and traceable blockchain-based secondhand market with committee consensus," in *2023 IEEE 8th International Conference on Big Data Analytics (ICBDA)*, 2023, pp. 72–76.

[20] S. Panda and S. Satapathy, *An Investigation into Smart Contract Deployment on Ethereum Platform Using Web3.js and Solidity Using Blockchain*, 05 2021, pp. 549–561.

[21] (2023) Tangible. [Online]. Available: https://www.tangible.store/

[22] (2023) Auralb. [Online]. Available: https://auraluxuryblockchain.com/

[23] (2023) Stockx. [Online]. Available: https://stockx.com/



NTNU

Norwegian University of
Science and Technology