A SECURE ESCROW PLATFORM FOR DECENTRALIZED TRANSACTIONS BETWEEN UNTRUSTED PARTIES

A PROJECT REPORT

Submitted by

SRICHARAN S [211419104265]

SRI SANJAY S [211419104268]

VIJAY P K [211419104302]

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI – 600 123.

ANNA UNIVERSITY: CHENNAI – 600 025

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

(An Autonmous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **“A SECURE ESKROW PLATFORM FOR DECENTRALIZED TRANSACTIONS BETWEEN UNTRUSTED PARTIES”** is the bonafide work of **“SRICHARAN S (211419104265), SRI SANJAY S (211419104268), VIJAY P K (211419104302)”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.L.JABASHEELA,M.E.,Ph.D.,	**Mr.M.MOHAN, B.E.,M.Tech.,(Ph.D.),**
PROFESSOR	**SUPERVISOR**
HEAD OF THE DEPARTMENT	**ASSISTANT PROFESSOR (GRADE 1)**
Department of Computer Science and Engineering	Department of Computer Science and Engineering
Panimalar Engineering College, Poonamallee, Chennai – 600 123	Panimalar Engineering College, Poonamallee, Chennai – 600 123

Certified that the above candidates were examined in the End Semester Project Viva-Voce Examination held on..........................

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **SRICHARAN S (211419104265), SRI SANJAY S (211419104268), VIJAY P K (211419104302)** hereby declare that this project report titled **"A SECURE ESCROW PLATFORM FOR DECENTRALIZED TRANSACTIONS BETWEEN UNTRUSTED PARTIES"** , under the guidance of **Mr.M.MOHAN,B.E., M.Tech.,(Ph.D.),** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our beloved Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHI KUMAR,M.E.,Ph.D and Dr.SARANYASREE SAKTHI KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. L. JABASHEELA , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Mr.M.MOHAN,B.E., M.Tech.,(Ph.D.),** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

ABSTRACT

In a decentralized payment system, trust is a critical element, especially when parties involved don't know or trust each other. Fortunately, decentralized escrow protocols offer a reliable solution for parties to make safe payments without the need for intermediaries. One of the popular decentralized escrow protocols is Eskro. This protocol enables parties to transact safely, knowing that their tokens are securely held until the conditions for payment are met. Before any transaction can take place, the tokens are sent to an escrow, which is a third-party smart contract that holds the tokens until the payment terms are satisfied. To ensure that both parties fulfill their obligations, both the product or service and the payment must be made as agreed. This guarantees that one party can't back out of the deal at the expense of the other party. The Eskro protocol helps build trust among parties by ensuring that everyone fulfills their end of the deal, which is crucial in decentralized transactions. In some cases, the payment terms may depend on external information, such as when a product is shipped. To address this, the oracle pattern can be utilized to provide the escrow with the necessary information, allowing it to execute the transaction only when the conditions are met. Once the smart contract code is deployed on the blockchain, it becomes immutable, meaning it can't be changed. This feature ensures the safety and reliability of the escrow functionality, making it virtually impossible for any party to tamper with the transaction. This provides all parties involved with the peace of mind that they won't be taken advantage of in the transaction. Overall, escrow protocols are an essential element of decentralized transactions, providing a secure and transparent way for parties to exchange goods or services without the need for a centralized intermediary.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1. GENERAL

The concept of a blockchain protocol is a revolutionary development that operates on top of the Internet, allowing for peer-to-peer (P2P) value transactions without the need for a central intermediary such as a bank or other clearing institutions. It does this by utilizing a P2P network of computers, all of which run the blockchain protocol and hold an identical copy of the ledger of transactions. This machine consensus allows for trust in the outputs of the system without the need for trust in any actor within it, as the blockchain technology ensures that the ledger of transactions is cryptographically secured and cannot be tampered with or revised. This eliminates the need for trusted third parties like banks or other intermediaries, enabling people and institutions who reside in different countries and are subject to different jurisdictions to interact over the Internet without the need for legally binding agreements.

Cryptographically secured P2P networks have been discussed in theoretical papers since the 1980s, and the consensus mechanism that enables distributed control over the ledger is called "Proof-of-Work," which combines economic incentives and cryptography. Blockchain is a public ledger of transactions that is shared, trusted, and continuously growing, with cryptographically secured data that is immune to tampering or revision.

B2P for procure to pay solutions are helping to alleviate the problems affecting the supply chain process. It makes the whole procedure paperless, saves time, reduces costs, helps ensure timely payments, and improves business liquidity. Blockchain technology offers real-time access to relevant information, data verification, integration with ERP, and advanced payment matching, helping companies address problems with unscheduled procurement, unauthorized access to information, and compliance issues. P2P is a decentralized network communications model that consists of a group of devices (nodes) that store and share files collectively. Each node acts as an individual peer, and the P2P communication happens without any central administration or server. P2P architecture is suitable for various use cases and can be classified into structured, unstructured, and hybrid peer-to-peer networks.

Crypto currency, particularly Bitcoin, is becoming increasingly popular worldwide, with an estimated 300 million users. It was created to exchange value anonymously and directly among users with a P2P transaction model that eliminates the need for intermediaries. This laid the foundation for the peer-to-peer blockchain technology that powers Bitcoin and all cryptocurrencies. Escrow Protocol is a Blockchain-based Web3 platform where investors can fund start-ups with extended control of their financial contributions, and funding is released based on the successful completion of milestones.

P2P architecture manages cryptocurrency transactions and is the fundamental element of blockchain technology. P2P mobile payments and security offer mobile developers numerous opportunities, but they are inherently less secure than centralized systems based on trusted servers. P2P configurations involve collaborative transactions or communications between two parties without the involvement of a central authority or administrator. Unlike traditional client/server networks, P2P networks enable all interconnected nodes to store, distribute and upload files, without any centralized authority or single point of failure.

A Blockchain protocol operates on top of the Internet, on a P2P network of computers that all run the protocol and hold an identical copy of the ledger of transactions, enabling P2P value transactions without a middleman though machine consensus. The concept of Blockchain first came to fame in October 2008, as part of a proposal for Bitcoin, with the aim to create P2P money without banks. Bitcoin introduced a novel solution to the age-old human problem of trust. The underlying blockchain technology allows us to trust the outputs of the system without trusting any actor within it. People and institutions who do not know or trust each other, reside in different countries, are subject to different jurisdictions, and who have no legally binding agreements with each other, can now interact over the Internet without the need for trusted third parties like banks, Internet platforms, or other types of clearing institutions. Ideas around cryptographically secured P2P networks have been discussed in the academic environment in different evolutionary stages, mostly in theoretical papers, since the 1980s. "Proof-of-Work" is the consensus mechanism that enables distributed control over the ledger. It is based on a combination of economic incentives and cryptography. Blockchain is a shared, trusted, public ledger of transactions, that everyone can inspect but which no single user controls. It is a distributed database that maintains a continuously growing list of transaction data records, cryptographically secured from tampering and revision.

The ills troubling the supply chain process are alleviating, owing to B2P for procure to pay solutions. It improves business liquidity, makes the whole procedure paperless, saves time, reduces costs, and helps ensure timely payments.

Blockchain offers real-time access to relevant information, verification of data, integration with ERP, and advanced payment matching. It helps companies deal with problems in unscheduled procurement, threats of unauthorized access to information, and compliance-related issues. Commonly known as P2P is a decentralized network communications model that consists of a group of devices (nodes) that collectively store and share files where each node acts as an individual peer. In this network, P2P communication is done without any central administration or server, which means all nodes have equal power and perform the same tasks. P2P architecture is suitable for various use cases and can be categorized into structured, unstructured, and hybrid peer-to-peer networks. The unstructured peer-to-peer networks are formed by nodes randomly from connection to each other, but they are inefficient than structured ones. In structured peer-to-peer systems, the nodes are organized, and every node can efficiently search the network for the desired data. Hybrid models are actually a combination of P2P and client-server models, and when compared to the structured and unstructured P2P systems, these networks tend to present improved overall performance.

Crypto currency is an exploding asset around the globe, with an estimated 300 million users worldwide. The most popular crypto, Bitcoin was created to exchange value anonymously and directly among users with a peer-to-peer (P2P) transaction model that eliminates the need for a central intermediary such as a bank or broker. This P2P model laid the foundation for the peer-to-peer blockchain technology which powers Bitcoin and all cryptocurrencies. Escrow Protocol is a Blockchain-based Web3 platform where investors can fund start-ups with extended control of their financial contributions. Funding is released based on the projects' successful completion of Milestones. We secure investors' funds through the time-honoured practice of putting funds into escrow; then releasing pay-outs for pre-determined project developments.

While funds are held in Escrow, waiting for pay out, we allocate investment funds to stable coin Yield-Farming Protocols. The global investment now adays is leveraging day by day. It fragmented the way of sustaining the world. The massive growth in technology has play a key role in making the system easy and smooth. The business leaders and employees mostly get benefitted from this. The p2p payment gateway with escrow protocol has a huge potential for the users. Specifically , The businesses are in the verge of making it more effective and smarter. The term P2P refers to decentralized networks of interconnected computer systems containing peers, or nodes. All nodes are equal, and the exchange of data occurs without a central server — that is, each computer or node can act as both a file server and a client. For example, when acting as a client, a node downloads data from other participants; and when it's acting as a server, it can be a downloading source.

Put simply, the peers or participating computer systems can simultaneously consume and provide resources on the same network. These resources can be files, storage, access to a scanner or printer, or processing power. There is no centralized authority, and no single point of failure. All interconnected nodes can engage in storing, distributing and uploading files. Transactions are peer-to-peer — P2P — meaning that they take place directly between the two parties involved. Each node in a network shares files with every other node without going through a central authority or administrator. As mentioned, nodes play the dual roles of client and server to other nodes on the network. P2P networks differ from traditional client/server networks, where clients request specific resources from central servers.

As the fundamental element of blockchain technology, P2P architecture manages cryptocurrency transactions. Cryptocurrencies leverage the power of peer-to-peer blockchain technology, as they can be exchanged or transferred without the help of any central body.

It's clear that blockchain is a revolutionary piece of technology, but how can mobile developers harness the power of blockchain? Well, there are many opportunities emerging around peer-to-peer mobile payments and security. Peer-to-peer mobile payments (and other transactions or communications) are inherently less secure than a centralized system that's based on a trusted server. With a peer-to-peer configuration, you have a collaborative comprised of computers and devices that share information directly, without a centralized server in place. This translates into significant vulnerability during the data transmission process. There is also a lot of vulnerability surrounding the data that's stored on a device—or node—that's part of a peer-to-peer network because these devices usually lack the encryption capabilities and high-level security measures that would be in place on a centralized server.

P2P is a technology that is based on a very simple principle, and that is the concept of decentralization. The peer-to-peer architecture of blockchain allows all cryptocurrencies to be transferred worldwide, without the need for any middle-man or intermediaries, or central server. With the distributed peer-to-peer network, anyone who wishes to participate in the process of verifying and validating blocks can set up a Bitcoin node. Blockchain is a decentralized ledger tracking one or more digital assets on a peer-to-peer network. When we say a peer-to-peer network, it means a decentralized peer-to-peer network where all the computers are connected in some way, and where each maintains a complete copy of the ledger and compares it to other devices to ensure the data is accurate. This is unlike a bank, where transactions are stored privately and are managed only by the bank.

An outcome of the implementation of blockchain technology in P2P is the elimination of paperwork or, at the least, reducing paperwork to the minimum. When everything is recorded on a distributed ledger accessible to everyone on the network, it does away with the requirement of a paper trail. Blockchain eliminates the need for paperwork in the authorization and authentication process in procurements. Many developers use (or would like to use) decentralized peer-to-peer mobile networks for their applications, as P2P brings many benefits. Peer-to-peer networks tend to be faster and more reliable, with little chance of an outage or downtime since it's unlikely that hundreds or thousands of nodes would crash at once. P2P networks are also relatively simple and affordable, both to establish and maintain. Blockchain makes it possible for developers to leverage the benefits of a peer-to-peer mobile network securely. Plus, this high-level security helps promote greater confidence amongst the app's users.

Already, we're beginning to see mobile apps that leverage blockchain technology. The glyph is a digital marketplace that allows users to buy and sell items, with payments processed with blockchain technology. Fold is another significant player that is utilizing blockchain to allow users to purchase home goods, groceries, and other items from large retailers such as Whole Foods and Target. The sky is truly the limit when it comes to leveraging blockchain for financial transactions and potentially some new and innovative uses. IOS 10 now allows for blockchain payments thanks to a partnership with Circle. This opens the door to a new realm of Apple-friendly apps that involve peer-to-peer mobile payments.

1.2 PROBLEM DESCRIPTION

A Secure Escrow Platform For Decentralized Transactions Between Untrusted Parties.

Every time buyers and sellers exchange goods or services for money, there is a risk of fraud. This risk is present in both online and offline transactions and has become a significant concern in today's digital age. One of the primary challenges of these transactions is deciding which party should send out the goods or money first. This issue has sparked a constant debate, with no clear resolution in sight. These transactions happen every day, ranging from peer-to-peer marketplaces like eBay to freelancing work, and their combined global value is estimated to reach $100 trillion annually. However, with the constant threat of fraud, trust issues arise between parties, making it challenging to conduct transactions smoothly. Fraudulent transactions come in different forms, including chargebacks, identity theft, fake payments, and non-delivery of goods or services. These scams can result in significant financial losses for either party involved, leading to a loss of trust in online transactions. To address this issue, various solutions have been proposed, with decentralized escrow protocols emerging as a reliable solution. With decentralized escrow protocols, tokens are sent to an escrow, which is a third-party smart contract that holds the tokens until the conditions for payment are met. The escrow ensures that both the product or service and the payment are made by all parties involved in the transaction, providing a secure and reliable way for parties to exchange goods or services without intermediaries. Despite the numerous advantages offered by decentralized escrow protocols, there is still a need to educate the public on the importance of securing their transactions and the need for reliable transaction systems. This way, we can help build trust and confidence in online transactions and eliminate fraudulent activities that continue to plague the industry.

1.3 ORGANIZATION OF REPORT

The proposed work is organized into several chapters, outlined below: Chapter 1 serves as the introduction, outlining the objectives and scope of the proposed system.

Chapter 2, the literature survey, provides an overview of research conducted on the existing system and the issues it faces. It also proposes a new system that attempts to improve on the shortcomings of the existing one.

Chapter 3 focuses on the system design and the roles of different modules in the proposed system.

Chapter 4 delves into the specifics of each module, providing detailed descriptions.

Finally, Chapter 5 concludes the work and includes a timeline chart for PHASE II, summarizing the efforts undertaken and the findings of the proposed system. It also identifies the various shortcomings and limitations of the proposed system.

CHAPTER 2
LITERATURE SURVEY

2.1 LITERATURE SURVEY

There have been many works to create a secure escrow platform for decentralized transactions between untrusted parties. However there have been certain limitations in each of the techniques used.

Haya r sahib, Khaled Saleh [1] et al "Blockchain-based physical delivery of proof system", Khalifa university conference and electronics department,2018

The survey was a very good process to find out a good solution. Reviewing all the concepts related to it is really good practice. With the widespread e-commerce, the need for a trusted system to ensure the delivery of traded items is crucial. Current proof of delivery (Pod) systems lacks transparency, traceability, and credibility. These systems are mostly centralized and rely on trusted third parties (TTP) to complete the delivery between sellers and buyers. TTP can be costly, a single point of failure, and subject to hacking, privacy evasion, and compromise. The blockchain is immutable, trusted, and decentralized ledger with logs and events that can be used for transparency, traceability, and tracking. In this paper, we present absolution and a general framework using the popular permissionless Ethereum blockchain to create a trusted, decentralized Pod system that ensures accountability, auditability, and integrity. The solution uses Ethereum smart contracts to prove the delivery of a shipped item between a seller and a buyer irrespective of the number of intermediate transporters needed. In our proposed solution, all participating entities are incentivized to act honestly by using double deposit collateral. Automated payment in ether is an integral part of a solution to tonsure that every entity gets its intended share of ether upon successful delivery. An arbitration mechanism is also incorporated if a dispute arises during the shipping process. In this paper, we show how we implemented, verified, and tested the proper functionality of our Pod solution. We also provide security analysis and give estimates of the cost consumption of ether gas. We made the full code of the Ethereum smart contracts publicly available at GitHub.

Shingling wang, ixia yang, yawling Jahan, [2] et al "Auditable Protocols for Fair Payment and Physical Asset Delivery Based on Smart Contracts", Xi'an university of technology conference in science,2019

The survey before the result is a very good step to find out a better output. With the rapid development of electronic information technology, online transaction will gradually surpass traditional market transaction, among which online payment and asset delivery become the focus of attention. But in fact, due to the incomplete third-party payment mechanism and the intrusion risk of various charging Trojan, it is easy to cause a trust crisis. The existing centralized framework often leads to information asymmetry between the two parties. Therefore, how to realize the fairness of payment and the auditability of assets in the distributed system is a challenging problem. The emerging blockchain technology provides a new method with its openness, transparency and verifiability. Existing researches do not provide a complete shopping model for consumers, most of which focuses on payments or only on asset delivery. In this paper, we propose an auditable fair payment and physical asset delivery protocol based on smart contracts. Three types of smart contracts are designed to achieve reliable and fair payment among merchants, consumers, and logistics companies. The traceability and auditability of blockchain provide an effective method to audit assets and data sharing in the whole transportation. In view of the phenomenon of goods being switched, the way of "pre-verification" is added. In order to prevent illegal elements from fake pickup codes, inducing consumers to conduct illegal operations, and causing property loss, in our system the pickup codes are generated by consumers to reduce the risk of fraud. In addition, our plan designs a complete return process for the first time, providing a better service experience and higher efficiency for consumers. Finally, all the contracts involved in the scheme are implemented and deployed on the Ethereum test network. The results of security analysis and evaluation showed that our scheme was improved in cost, with high security and availability.

W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng and V. C. M. Leung [3] et al "Decentralized Applications: The Blockchain-Empowered Software System," in IEEE Access,2018

Blockchain technology has attracted tremendous attention in both academia and capital market. However, overwhelming speculations on thousands of available cryptocurrencies and numerous initial coin offering scams have also brought notorious debates on this emerging technology. This paper traces the development of blockchain systems to reveal the importance of decentralized applications (daps) and the future value of blockchain. We survey the state-of-the-art daps and discuss the direction of blockchain development to fulfill the desirable characteristics of daps. The readers will gain an overview of dap research and get familiar with recent developments in the blockchain.

N. Z. Aitzaz and D. Voinovich [4] et al "Security and Privacy in Decentralized Energy Trading Through Multi-Signatures,

Blockchain and Anonymous Messaging Streams," in IEEE Transactions on Dependable and Secure Computing, 2018

Smart grids equipped with bi-directional communication flow are expected to provide more sophisticated consumption monitoring and energy trading. However, the issues related to the security and privacy of consumption and trading data present serious challenges. In this paper, we address the problem of providing transaction security in decentralized smart grid energy trading without reliance on trusted third parties. We have implemented a proof-of-concept for a decentralized energy trading system using blockchain technology, multi-signatures, and anonymous encrypted messaging streams, enabling peers to anonymously negotiate energy prices and securely perform trading transactions. We conducted case studies to perform security analysis and performance evaluation within the context of the elicited security and privacy requirements.

Nasheed khan, Faiza lousily [5] et al "Blockchain smart contracts: Applications, challenges, and future trends", Peer-to-peer networking applications , 2021

In recent years, the rapid development of blockchain technology and cryptocurrencies has influenced the financial industry by creating a new crypto-economy. Then, next-generation decentralized applications without involving a trusted third-party have emerged thanks to the appearance of smart contracts, which are computer protocols designed to facilitate, verify, and enforce automatically the negotiation and agreement among multiple untrustworthy parties. Despite the bright side of smart contracts, several concerns continue to undermine their adoption, such as security threats, vulnerabilities, and legal issues. In this paper, we present a comprehensive survey of blockchain-enabled smart contracts from both technical and usage points of view. To do so, we present a taxonomy of existing blockchain-enabled smart contract solutions, categorize the included research papers, and discuss the existing smart contract-based studies. Based on the findings from the survey, we identify a set of challenges and open issues that need to be addressed in future studies. Finally, we identify future trends.

Enes Erden, Momin kibbe, kernel akala [6] et al "A Bitcoin payment network with reduced transaction fees and confirmation times", Computer networks and technology conference,2021

The high transaction fees and confirmation times made Bitcoin unfeasible for many applications when the payments are in small amounts and require instant approval. As a result, many other cryptocurrencies were introduced for addressing these issues, but the Bitcoin network is still the most widely used payment system. Without doubt, to benefit from its network of users, there is a need for novel solutions that can successfully address the problems about high transaction fees and transaction verification times. Recently, payment network ideas have been introduced including the Lightning Network (LN) which exploits off-chain bidirectional payment channels between parties. As off-chain links can be configured to perform aggregated transactions at certain intervals without writing to blockchain, this would not only reduce the transaction fees but also decrease the verification times significantly. Nevertheless, LN still deploys relay nodes which charge fees and its growth leads certain nodes to become monopolies which defeat the very purpose of decentralization. Despite the thought that LN will provide a "scale-free network mechanism" along with high decentralization, how to form such a network among multiple parties is never studied before. Therefore, in this paper, by exploiting the LN, we offer to form a purely decentralized payment network which will allow Bitcoin to handle a lot of transactions with enough capacity. The idea is to form a network of retailers (i.e., nodes) which do business with Bitcoin by connecting them through off-chain links (i.e., payment channels) based on the business needs. The problem is initially modeled as a network optimization but since scalability would be a concern the next steps follow a heuristic approach where certain links are pruned to force evenly distributed payment flows while minimizing the total investments made to create initial off-chain links. The evaluations demonstrate the extent of scalability of the network along with the trade-offs between the distribution of flows and the initial flow capacities of the nodes.

Reza Trapdoor, Peja ghazi [7] et al "Block by block: A blockchain-based peer-to-peer business transaction for international trade", Technological forecast and social change conferences, 2022

The survey is good practice before going into finding the solution, it ensure a good output for the problem. Heterogeneity and complicated processes, risk of information leakage, and higher costs are some of the challenges that stem from third-party involvement in business transactions. This study proposes a novel mechanism to address the shortcomings of third-party-dependent transactions in the context of international trade. Moreover, we provide business process modeling, deployed in a business transaction scenario, to furnish a deeper perspective on the working of the mechanism based on Business Process Model and Notation (BPMN) 2.0 standards and guidelines. By analyzing and identifying blockchain roles and capabilities, this study proposes a blockchain technology-based letter of credit (BTLC), which is a mechanism providing letters of credit (LCs) that incorporate the benefits of blockchain and smart contracts.

Steven Goldfaden, Joseph Bonneau, Rosario Gennaro & Arvind Narayanan [8] et al "Escrow Protocols for Cryptocurrencies: How to Buy Physical Goods Using Bitcoin", International Conference on Financial Cryptography and Data Security, 2017

The survey should be conducted because it has a very positive impact on the final output. We consider the problem of buying physical goods with cryptocurrencies. There is an inherent circular dependency: should be the buyer trust the seller and pay before receiving the goods or should the seller trust the buyer and ship the goods before receiving payment? This dilemma is addressed in practice using a third party escrow service. However, we show that naive escrow protocols introduce both privacy and security issues. We formalize the escrow problem and present a suite of schemes with improved security and privacy properties. Our schemes are compatible with Bitcoin and similar blockchain-based cryptocurrencies.

Z. Hong, Z. Wang, W. Cai and V. C. M. Leung [9] et al "Connectivity-aware task outsourcing and scheduling in D2D networks", Proc. 26th Int. Conf. Compute. Common. Newt. (ICCCN),2017

With the flourishing of smart mobile devices (e.g., smartphones, tablets), development of mobile cloud computing has received more and more attentions from both industry and academia. Compared with the traditional way of executing large- scale computational tasks on powerful desktop computers and the cloud, mobile cloud computing is featured by the ubiquitous availability, flexibility, and low-cost. However, this feature also brings challenges when we build the satisfactory mobile computing system. First, the computational power on a mobile device is not comparable with that on a personal computer such that many computation-intensive tasks cannot be independently handled by mobile devices. Second, offloading computational tasks to the cloud introduces additional monetary costs (e.g. wireless communication cost, computational service cost), which may be pricy for users. In this paper, we propose a novel connectivity- aware task scheduling paradigm to enable mobile device users to accomplish computation-intensive tasks cooperatively in the device-to-device (D2D) network by incorporating the "fog" - aggregate of computational powers in the ad-hoc. A super node at the base station is responsible for scheduling cooperation tasks based on user mobility. To further enhance the quality of experience (Quek) for the users, we propose a lightweight heuristic algorithm to perform task scheduling to ensure low cooperative task execution time. Simulation results show that our cooperative paradigm efficiently reduces the average task execution time for mobile device users in the D2D network.

M. Wuhrer and U. Zdun, "Smart contracts: Security patterns in the Ethereum ecosystem and solidity" [10] et al Proc. Int. Workshop Blockchain Oriented Soft. Eng. (IWBOSE), 2018.

The survey is a very important dynamics to specify a better result in the last and it should be very unique as of the requirements. Smart contracts that build up on blockchain technologies are receiving great attention in new business applications and the scientific community, because they allow untrusted parties to manifest contract terms in program code and thus eliminate the need for a trusted third party. The creation process of writing well performing and secure contracts in Ethereum, which is today's most prominent smart contract platform, is a difficult task. Research on this topic has only recently started in industry and science. Based on an analysis of collected data with Grounded Theory techniques, we have elaborated several common security patterns, which we describe in detail on the basis of Solidity, the dominating programming language for Ethereum. The presented patterns describe solutions to typical security issues and can be applied by Solidity developers to mitigate typical attack scenarios.

CHAPTER 3
SYSTEM SPECIFICATIONS

3.1 Exsisting System

In the existing system, the size of the proof stored in the blockchain for each transaction is extremely large. The existing system is not trustable and reliable. The existing system requires the use of third party and charges extra fees than necessary. The existing escrow protocols are neither transparent not secure, untrusted parties who wish to carry out a transaction face several challenges. Firstly, they may not trust each other, and there is a possibility that one party may fail to meet their obligation, leading to disputes and potential fraud. Secondly, they may have to rely on a third-party intermediary to hold the funds until the transaction is complete, which adds an extra layer of complexity and cost to the transaction.

Furthermore, the existing system has several limitations. For instance, it is not decentralized, meaning that it relies on a centralized entity to hold the funds and oversee the transaction. This centralization makes the system vulnerable to hacking and other security threats. Additionally, the intermediaries may take a long time to process the transaction, leading to delays and inefficiencies. Moreover, the existing system does not provide transparency or accountability. The parties involved in the transaction have to rely on the intermediaries to carry out the transaction in good faith, which is not always guaranteed. In case of a dispute, the intermediaries may not be impartial, and there may not be an adequate mechanism for resolving the dispute.

Overall, the existing system has several shortcomings that make it less than ideal for carrying out secure and efficient transactions between untrusted parties.

3.2 Proposed System

The parties involved in the transaction need to ensure that both the agreed product/service is delivered and payment is made. One party should not be able to default on the transaction at the expense of the other party.

- Trust and security – An escrow smart contract reduces the risk of fraud by acting as a neutral party and ensuring proper escrow logic execution.
- Transparency – Operations happening in the system are transparent as relevant transactions are accessible to all blockchain participants.
- Efficiency – Blockchain eliminates the need for third parties, which in turn helps to reduce the transaction cost and enhances service efficiency.

The proposed system is a secure escrow platform for decentralized transactions between untrusted parties. It eliminates the need for a third-party intermediary by using smart contracts on a blockchain network to hold the funds until the transaction is complete. The platform ensures that both parties fulfill their obligations before releasing the funds, providing a secure and reliable transaction process. The smart contract code is immutable, ensuring that the escrow functionality is safe and cannot be altered once the contract is deployed on the blockchain. This provides a level of transparency and trust that is not possible with the existing system.

In summary, the proposed system offers a more secure, efficient, and cost-effective solution for decentralized transactions between untrusted parties.

3.3 Hardware Environment

A server or a network of servers to run the platform. Adequate storage for storing transaction records and smart contracts. Security hardware such as firewalls, intrusion detection and prevention systems, and other security appliances. Encryption hardware to encrypt and decrypt sensitive information. In the existing system, when two untrusted parties want to carry out a transaction, they either have to trust each other or rely on a third-party intermediary to hold the funds until the transaction is complete. This third-party intermediary charges a fee for their services, which can be significant for large transactions. Additionally, the intermediary may not always act in the best interest of both parties, creating a potential for fraud or other disputes.

3.4 Software Environment

- Operating system: The platform can run on any operating system such as Linux, Windows, or macOS.
- Web server: The platform may require a web server such as Apache or Nginx to serve web pages and APIs.
- Programming languages: The platform may require programming languages such as Solidity for writing smart contracts, JavaScript for client-side programming, and Python for server-side programming.
- Database: The platform may require a database to store transaction records and smart contracts. Popular options include MySQL, MongoDB, and PostgreSQL.

- Blockchain platform: The platform may be built on a blockchain platform such as Polygon, which allows for the creation of decentralized applications and smart contracts.

CHAPTER 4

SYSTEM DESIGN

4.1 PROPOSED SYSTEM

4.1.1 SYSTEM FLOW DIAGRAM

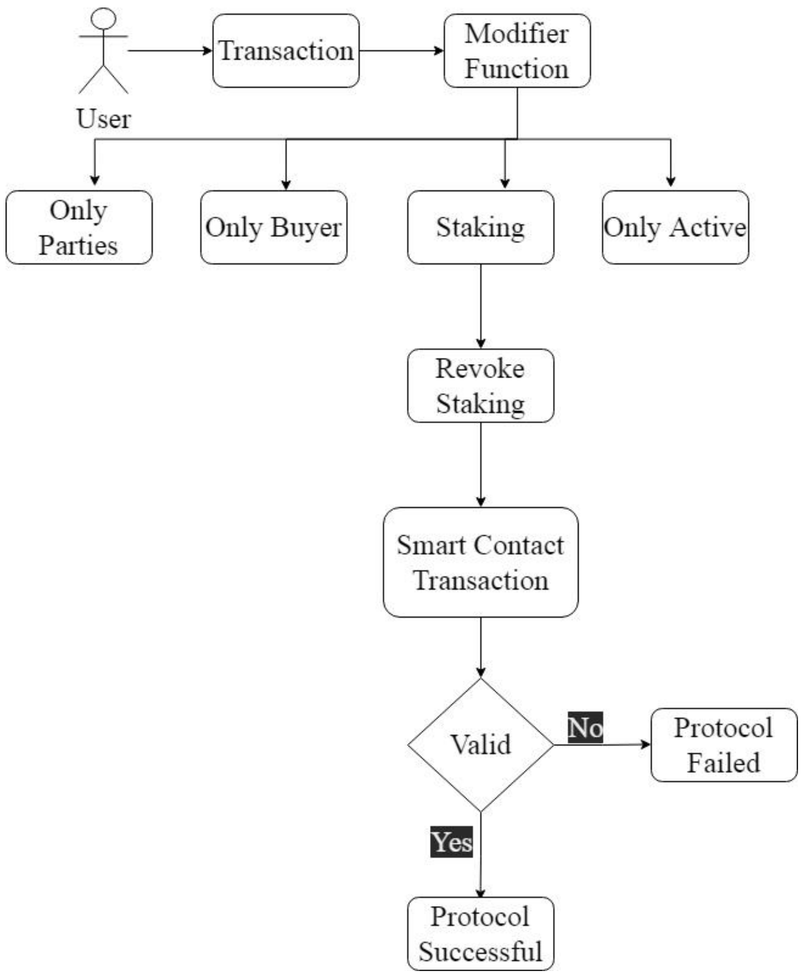


Fig 4.1 Block diagram of the proposed system.

4.1.2 ARCHITECTURE DIAGRAM

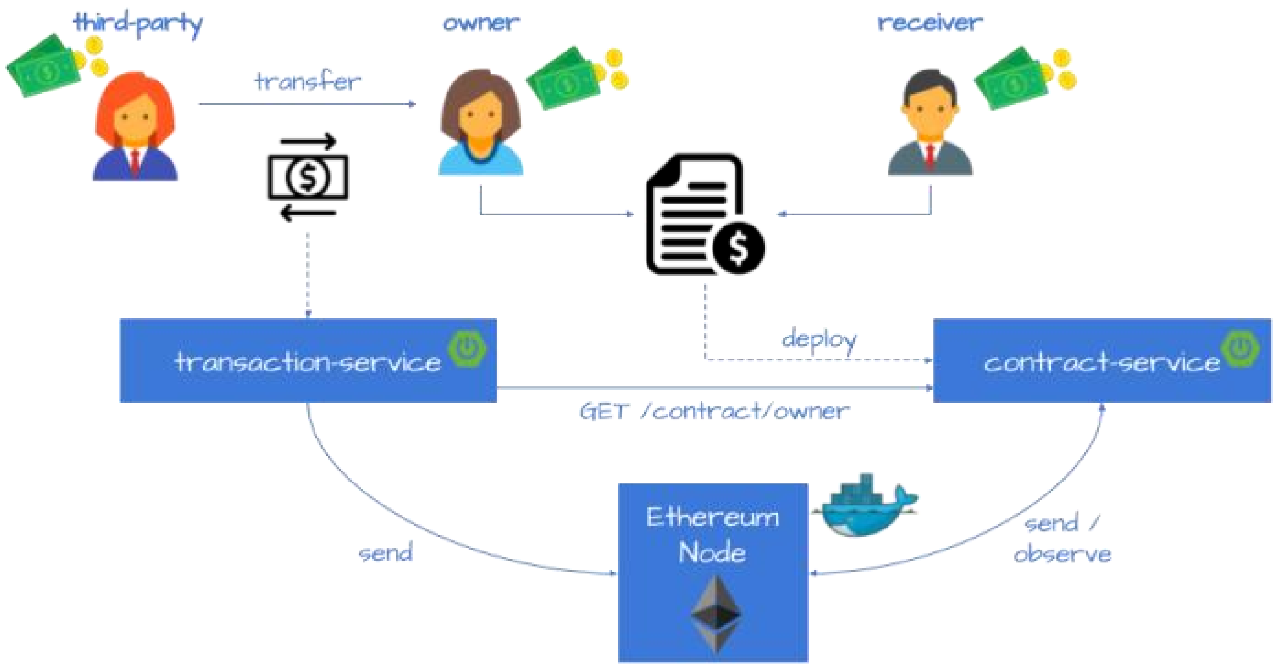


Fig 4.2 Architecture diagram of the proposed system.

4.1.3 USE CASE DIAGRAM

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system. The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

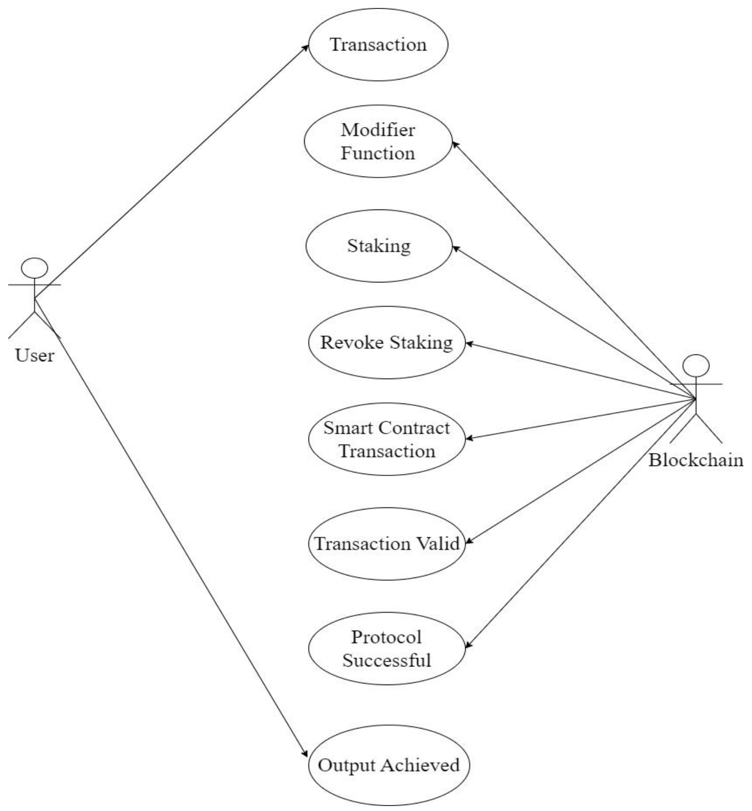


Fig 4.3 Use case Diagram

4.1.4 ACTIVITY DIAGRAM

An activity diagram is a kind of graphical representation that may be used to depict events visually. It is made up of a group of nodes that are linked to one another by means of edges. They are able to be connected to any other modeling element, which enables the behavior of activities to be replicated using that methodology. Simulations of use cases, classes, and interfaces, as well as component collaborations and component interactions, are all made feasible with the help of this tool.

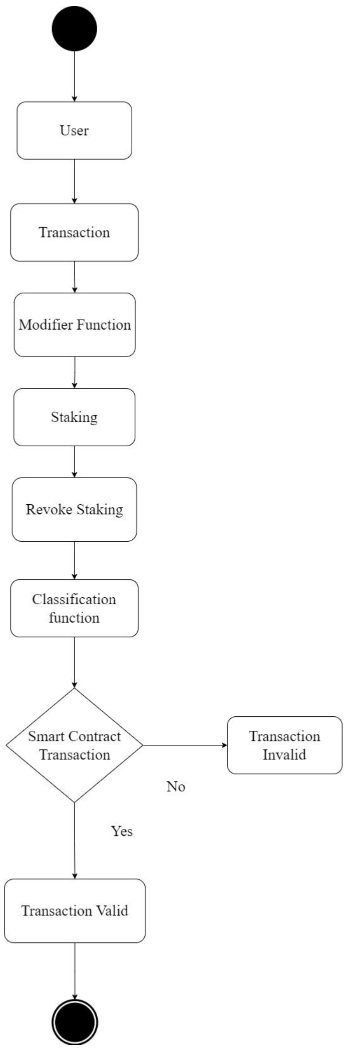


Fig 4.4 Activity Diagram

4.1.5 CLASS DIAGRAM

A static view of an application is depicted in the class diagram. It displays the properties, classes, functions, and relationships of the software system to provide an overview of the software system. It organizes class names, properties, and functions into a discrete compartment to aid in program development.

The following are the functions of class diagrams:

1. Define the main responsibilities of the system.

2. Serves as a basis for component diagrams and deployment. Use forward and reverse engineering.

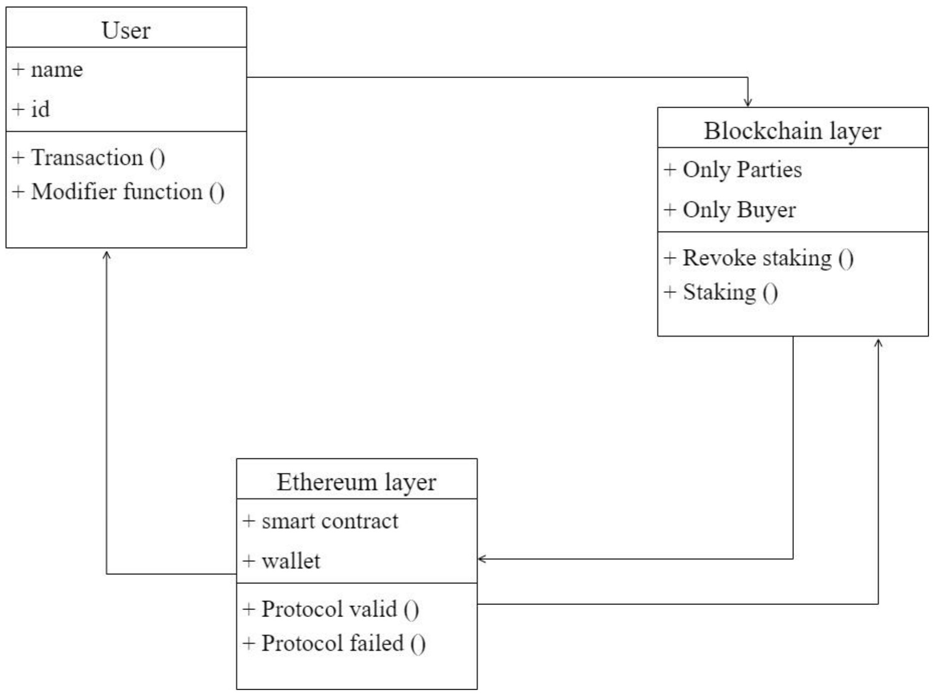


Fig 4.5 Class Diagram

4.1.6 SEQUENCE DIAGRAM

The sequence diagram, also called the event diagram, describes the flow of messages in the system. It helps to visualize various dynamic parameters. He describes the communication between two rescue lines as a series of events arranged in time in which these rescue lines participated during the performance. The lifeline is represented by a vertical bar in UML while the message flow is represented by a vertical dotted line that crosses the bottom of the page. It includes both repetitions and branches.

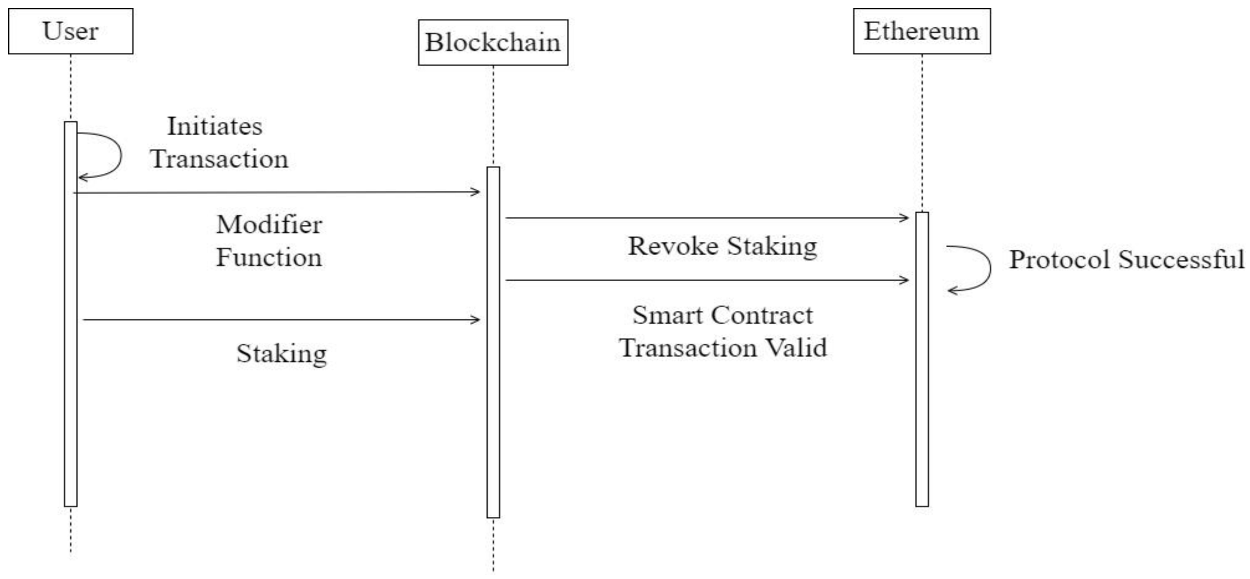


Fig 4.6 Sequence Diagram

4.1.7 COMPONENT DIAGRAM

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that are often used to model the static implementation view of a system.

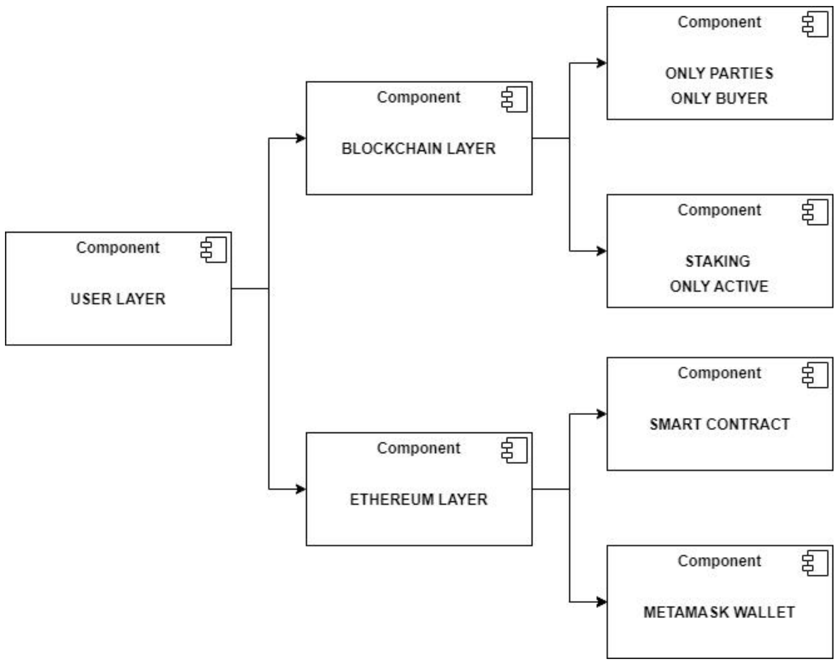


Fig 4.7 Component Diagram

CHAPTER 5

SYSTEM ANALYSIS

5.1 Feasibilty Study

5.1.1 Economic Feasibility

Developing a secure escrow platform for decentralized transactions between untrusted parties would require a significant investment. The following economic aspects need to be considered:

Development costs: The cost of developing the platform would depend on the complexity and size of the platform. The technical team, hardware, and software requirements would need to be evaluated to determine the cost of development.

Operational costs: The platform would require ongoing maintenance, security, and support, which would add to the operational costs.

Revenue generation: The platform would generate revenue by charging a fee for each transaction. The fee charged should be reasonable and competitive with existing platforms.

Based on market research, it is feasible to develop the platform as there is a growing demand for secure escrow platforms for decentralized transactions.

5.1.2 Technical Feasibility

To develop a secure escrow platform for decentralized transactions, the following technical aspects need to be considered:

Smart contract development: A smart contract is essential for implementing the escrow mechanism in a decentralized platform. A technical team with expertise in smart contract development would need to be assembled to create the platform.

Blockchain technology: The platform would need to be built on a blockchain network to ensure immutability, transparency, and security.

Security: The platform must be designed with high-security standards to prevent hacking and other malicious activities.

All of these technical aspects are feasible as there are already existing blockchain-based escrow platforms in the market, which proves that the technology is mature enough to handle the proposed platform.

5.1.3 Social Feasibility

The social feasibility of the platform would depend on the following factors:

User adoption: The success of the platform would depend on its user adoption. It would need to be marketed effectively to attract users and gain their trust.

Customer support: The platform would require a dedicated customer support team to provide assistance to users.

Regulatory compliance: The platform would need to comply with relevant regulations in the jurisdictions where it operates.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 SYSTEM IMPLEMENTATION

6.1.1 The Escrow Contract Implementation (to Facilitate Transaction)

The Eskro.sol contract is used to facilitate the transaction between two trustless parties.

- SafeMath.sol

  SafeMath.sol from OpenZeppelin is imported here.(It validates if an arithmetic operation will result in integer overflow/underflow).

- Modifiers-

  onlyParties() ,onlyBuyer(),onlyActive() modifiers are defined to make sure that the associated function can be called by only certain people or can be called under only certain circumstances.

- stake() -

  This Function is used for the parties to stake some amount of ETH that they can't withdraw until both parties reach a consensus and agree to either cancel or proceed with their trade.

- revokeStake()-

  This allows parties to withdraw their staked fund if the contract has not been locked yet.

- cancel()-

  This function is used to cancel transactions. This can be revoked by the revokeCancellation() function.

- confirm()-

  This functions confirms that seller has confirmed & honored the transaction.After hitting this function the contract is set to an inactive state and a state change event is emitted

Libraries Used and Imported

1. Hardhat

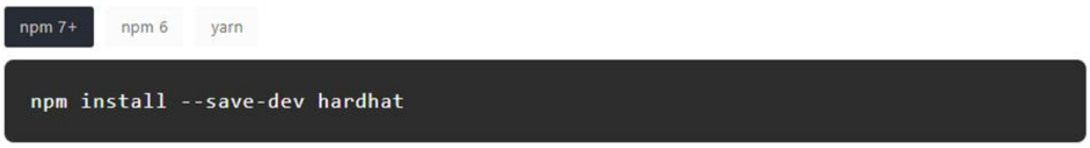
   Hardhat is a specific environment for development where Ethereum software can be built. It is made up of different parts that work together to make a full development environment for editing, compiling, debugging, and deploying smart contracts and dApps. Hardhat Runner is the main part of Hardhat that you interact with. It is a flexible and extensible task runner that helps you manage and automate the repetitive tasks that come with building smart contracts and decentralized applications (dApps).

   The main ideas behind Hardhat Runner are tasks and plugins. When you use the command line to run Hardhat, you are running a task. For example, the built-in compile task is run when you type npx hardhat compile. Tasks can call other tasks, which makes it possible to set up complex workflows. Existing tasks can be changed by users or plugins, which makes those workflows flexible and easy to add to.

   Hardhat is used in your project by setting it up locally. So, your environment will be easy to copy, and you won't have to deal with version conflicts in the future.

To install it, go to an empty folder, run npm init, and follow the instructions. You can use a different package manager, like yarn, but we suggest using npm 7 or later because it makes it easier to install Hardhat plugins.

Once the project is ready, it is necessary to run



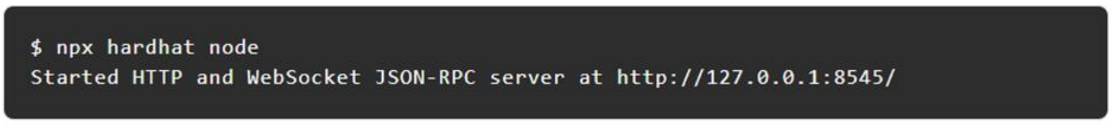
```
npm install --save-dev hardhat
```

To use local installation of Hardhat, we used npx to run it (i.e. npx hardhat).

Connecting a wallet or Dapp to Hardhat Network

By default, Hardhat will spin up a new in-memory instance of Hardhat Network on startup. It's also possible to run Hardhat Network in a standalone fashion so that external clients can connect to it. This could be MetaMask, your Dapp front-end, or a script.

To run Hardhat Network in this way, we must run npx hardhat node:



```
$ npx hardhat node
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/
```

This will expose a JSON-RPC interface to Hardhat Network. To use it, we connect wallet or application.

2. Chai

Chai is a BDD/TDD assertion library for node and the browser that works well with any javascript testing framework. Chai has a number of interfaces, so the developer can choose the one that feels best. Chain-capable BDD styles use clear, expressive language, while the TDD assert style has a more traditional feel.

3. Mocha

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.

We installed Mocha by the following codes:

Install with npm globally:

```
$ npm install --global mocha
```

or as a development dependency for your project:

```
$ npm install --save-dev mocha
```

Working with Mocha

```
$ npm install mocha
$ mkdir test
$ $EDITOR test/test.js #
```

In editor:

```
var assert = require('assert');
describe('Array', function () {
  describe('#indexOf()', function () {
    it('should return -1 when the value is not present', function () {
      assert.equal([1, 2, 3].indexOf(4), -1);
    });
  });
});
```

Back in the terminal:

```
$ ./node_modules/mocha/bin/mocha

  Array
    #indexOf()
      ✓ should return -1 when the value is not present
  1 passing (9ms)
```

Setting up a test script in package.json:

```
"scripts": {
  "test": "mocha"
}
```

4. Openzeppelin

OpenZeppelin provides a complete suite of security products to build, manage, and inspect all aspects of software development and operations for Ethereum projects.

```
$ npm install @openzeppelin/contracts
```

OpenZeppelin Contracts features a stable API, which means the contracts won't break unexpectedly when upgrading to a newer minor version.

Importing The Contracts

Once installed, we used the contracts in the library by importing them:

```
// contracts/MyNFT.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract MyNFT is ERC721 {
        constructor() ERC721("MyNFT", "MNFT") {
        }
    }
```

Using OpenZeppelin Contracts

Good software is built around modules and libraries that can be used more than once. OpenZeppelin Contracts has a lot of useful building blocks that can be used to make smart contracts. And we can feel safe building on them because they have been audited many times and their security and correctness have been put to the test.

6.1.2 Escrow Contract creating Smart-Contract Implementation

The Escrow contract to facilitate the transaction between two trustless parties are created is created by a contract called EsKroFactory.sol

- createContract() -

  It takes buyer's address,seller's address,price as parameters.Creates contract and emits ContractCreated() event after successful creation of Escrow contract.

- getContractsByIndex()-

  It returns index of the contract that is created

- getContractsByAddress()-

  It returns address of the contract

- getContracts()-

  It returns all the contracts

6.1.3 Front-end

The ABI & address of the deployed contract is used along with the help of Ethers.js library to connect the frontend with the smart-contract.

Components- The code is divided into various components to keep the project clean and maintainable

- Wallet.js-

  This is used as the connect button which helps users to connect their Metamask wallet with the web-app.

- Contract.js-

  stake(),withdrawStake(),cancel(),revokeCancel(),confirm() functions are implemented in this component.The buttons for staking related all activities are handled by this component.

- ContractCreation.js-

  Creation of the contract is handled in this component. - A form for creating the Escrow contract is implemented here.It is connected with the EscrowFactory contract and It passes the input data in the associated functions of the Factory contract and thus a Escrow contract is created.

Contract ABI Specification

The standard way to interact with contracts in the Ethereum ecosystem, both from outside the blockchain and between contracts, is through the Contract Application Binary Interface (ABI). Based on the type of data, as described in this specification, the data is encoded. The encoding does not describe itself, so it needs a schema to be decoded.

We assume that a contract's interface functions are strongly typed, known at the time of compilation, and static. We assume that all contracts will be able to compile with the interface definitions of any other contracts they call.

This specification doesn't talk about contracts whose interfaces are dynamic or whose details are only known at runtime.

Ether.js Library

The ethers.js library aims to be a complete and compact library for interacting with the Ethereum Blockchain and its ecosystem. It was originally designed for use with ethers.io and has since expanded into a more general-purpose library.

Metamask

Metamask acts as a bridge between websites and the Ethereum blockchain. The smart contracts in the blockchain check to see if the node in the network can access that data or not. When a local blockchain network is built, the Metamask wallet can take care of the nodes. Metamask is a cryptocurrency wallet that can be added to Google

Chrome. It is a way to communicate with the Ethereum blockchain. Metamask is used to store Ethereum accounts. It is safe and easy to use, and can also be used to deploy to the main Ethereum networks.

Frontend - React

React.js

React, also called React.js or ReactJS, is a free and open-source JavaScript library for building user interfaces from UI components. Frameworks like Next.js can be used to build single-page, mobile, or server-rendered apps that are based on React.

Security

The smart contract, its state data, and transaction data should only be shown to the parties involved in the network. If they are shown to the rest of the world, real estate deals can be attacked and become unsafe. EscrowChain takes over Ethereum's security system and adds to it. As a smart contract, any programming logic can be added to Ethereum, and the program can run forever, which uses a lot of computing power and resources.

Ethereum stops this from happening by setting a limit on the amount of gas that can be used for the computation. Each transaction has a gasLimit and a gasPrice that determine how many steps a contract will go through when it is executed. When the gasLimit is reached, the contract execution stops. Ethereum makes sure that a contract can't be stopped in the middle of its execution. If a Transaction's available Gas runs out or if the contract's execution is stopped, the transaction goes back to where it was before. This keeps attackers from being able to take advantage of a state in between.

To stop Replay Attacks, Ethereum uses a unique value called a "nonce" that increases with each transaction. This makes it impossible for anyone to add blocks back to the ledger. Each transaction is followed by a Transaction Receipt, which is a hash that is saved in the block. The receipt has a hash that shows the state after the transaction and the total amount of gas used in the block that contains the transaction receipt. This lets each block in the blockchain be checked and confirmed step by step, which can help find suspicious transactions.

CHAPTER 7

METHODOLOGIES

7. Methodologies

A smart contract can act as an escrow, holding the money until the conditions for payment are met. First, we make a smart contract that spells out the settlement process and conditions. Either the seller or the buyer could set up and use this smart contract. Second, the buyer sends the token(s) to the smart contract for the escrow. Third, when the conditions for token release are met by giving the desired product or server, the escrow smart contract is told about the event.

Lastly, the escrow checks that the conditions have been met and give the tokens to the seller. If the buyer doesn't tell the escrow about the event within the time limit or if the event shows that the product or service wasn't delivered according to the terms agreed upon, the tokens are sent back to the buyer. If the payment conditions depend only on data on the blockchain, a delegated call can be made to the escrow contract to let it know when the product or service should be sent.

If the payment terms depend on outside information, like when a product is shipped, the oracle pattern can be used to give the escrow the information it needs. As soon as the smart contract code is put on the blockchain, it can't be changed. This makes sure that the escrow functionality is safe. This gives everyone involved in the trade the peace of mind that they won't be taken advantage of.

CHAPTER 8

SYSTEM TESTING

8.1 System Testing

8.1.1 Functional Testing

This type of testing is performed to ensure that the system meets its functional requirements. In this case, the platform's functionality will be tested to ensure that tokens are sent to the escrow, the escrow holds the tokens until payment conditions are met, and the escrow releases the tokens to User B once the conditions are fulfilled.

Functional testing is a type of testing that verifies that each function of the software application operates in conformance with the requirement specification. Here are some sample test cases and expected outputs for the functional testing of the Secure Escrow Platform for Decentralized Transactions between Untrusted Parties:

Test Case 1: Verify that the escrow smart contract receives the tokens sent by the buyer

Input: The buyer sends 100 tokens to the escrow smart contract.

Output: The smart contract successfully receives the 100 tokens.

Test Case 2: Verify that the seller can claim the tokens from the escrow smart contract after the product has been shipped

Input: The seller ships the product and updates the shipment information on the smart contract.

Output: The smart contract releases the tokens to the seller.

Test Case 3: Verify that the buyer can receive a refund if the product is not delivered or is not as described

Input: The buyer raises a dispute and provides evidence that the product was not delivered or was not as described.

Output: The smart contract refunds the tokens to the buyer.

Test Case 4: Verify that the oracle pattern is used to provide external information to the smart contract

Input: The smart contract is programmed to release the tokens to the seller once the shipment is confirmed.

Output: The oracle pattern is used to obtain the shipment confirmation from an external source and release the tokens to the seller.

Test Case 5: Verify that the smart contract cannot be tampered with once it is deployed on the blockchain

Input: The smart contract is deployed on the blockchain.

Output: The smart contract code is immutable and cannot be altered, ensuring that the escrow platform remains secure.

The expected outputs for each test case should match the requirements and specifications outlined in the design phase of the project. By conducting thorough functional testing, the team can ensure that the system performs as intended and meets the needs of the users.

8.1.2 Performance Testing

This type of testing is performed to ensure that the system performs as expected under different loads and stress levels. Performance testing for the escrow platform would involve testing how many transactions the platform can handle at a given time, how quickly the transactions are processed, and how well the platform can handle sudden spikes in usage.

Test Case 1: Load Testing Objective: To verify the system's behavior under heavy loads.

Input:

Increase the number of transactions and users.

Monitor the system's performance and behavior.

Record the response time and the system's performance under heavy loads.

Output:

The system should be able to handle a large number of transactions and users without any issues.

The response time should not significantly increase, even under heavy loads.

Test Case 2: Stress Testing Objective: To verify the system's behavior under stress conditions.

Input:

Apply stress to the system by increasing the number of transactions and users beyond the system's capacity.

Monitor the system's performance and behavior.

Record the response time and the system's performance under stress conditions.

Output:

The system should be able to recover from stress conditions without crashing.

The response time should not significantly increase, even under stress conditions.

Test Case 3: Scalability Testing Objective: To verify the system's behavior as the number of transactions and users increase.

Input:

Increase the number of transactions and users.

Monitor the system's performance and behavior.

Record the response time and the system's performance as the number of transactions and users increase.

Output:

The system should be able to handle an increasing number of transactions and users without any issues.

The response time should not significantly increase as the number of transactions and users increase.

Test Case 4: Endurance Testing Objective: To verify the system's behavior over an extended period.

Input:

Run the system for an extended period (e.g., 24 hours).

Monitor the system's performance and behavior.

Record the response time and the system's performance over the extended period.

Output:

The system should be able to operate without any issues over an extended period.

The response time should not significantly increase over the extended period.

Test Case 5: Concurrent Testing Objective: To verify the system's behavior when multiple transactions occur simultaneously.

Input:

Simulate multiple transactions occurring simultaneously.

Monitor the system's performance and behavior.

Record the response time and the system's performance under concurrent testing.

Expected Output:

The system should be able to handle multiple transactions occurring simultaneously without any issues.

The response time should not significantly increase under concurrent testing.

8.1.3 Security Testing

This type of testing is performed to ensure that the system is secure and cannot be compromised by attackers. Security testing for the escrow platform would involve testing for vulnerabilities such as smart contract bugs, denial-of-service attacks, and other security threats.

Test case 1: SQL injection attack

Input: Malicious code entered into a text field

Output: System should reject the input and display an error message.

Test case 2: Cross-site scripting attack

Input: Malicious code entered into a text field

Output: System should reject the input and display an error message.

Test case 3: Brute force attack

Input: Multiple incorrect login attempts

Output: System should block the user after a certain number of attempts to prevent unauthorized access.

Test case 4: Man-in-the-middle attack

Input: Intercepted network traffic

Output: System should encrypt all data transmitted over the network to prevent interception.

Test case 5: Denial of Service (DoS) attack

Input: Flood the system with excessive traffic

Output: System should detect and block the source of the traffic to prevent a DoS attack.

8.1.4 Usability Testing

This type of testing is performed to ensure that the system is easy to use and understand for the end-users. Usability testing for the escrow platform would involve testing how easy it is for users to initiate transactions, make payments, and receive tokens.

Test Case 1: Login Process

User should be able to easily find the login button on the homepage.

User should be able to input their credentials without any confusion or difficulty.

User should receive clear feedback on whether their login attempt was successful or not. Output: Users were able to find the login button quickly and easily, and had no issues inputting their credentials. The login feedback was clear and concise.

Test Case 2: Product/Service Listing

User should be able to easily find the button to list their product/service.

User should be able to add all necessary information without any confusion or difficulty.

User should be able to preview their listing before submitting it. Output: Users were able to find the listing button quickly and easily, and had no issues adding all necessary information. The preview feature was helpful and allowed users to make sure their listing looked how they wanted it to.

Test Case 3: Payment Process

User should be able to see the payment amount and all associated fees before agreeing to the transaction.

User should be able to choose from a variety of payment methods.

User should receive clear feedback on whether their payment was successful or not. Output: Users were able to see the payment amount and associated fees clearly, and had no issues selecting a payment method. The payment feedback was clear and concise.

Test Case 4: Dispute Resolution

User should be able to easily report a problem with a transaction.

User should be able to clearly explain the problem and provide any necessary evidence.

User should receive clear feedback on the status of their dispute. Output: Users were able to report a problem easily and had no issues explaining the issue and providing evidence. The dispute feedback was clear and kept users informed throughout the process.

Overall, the usability testing indicated that the secure escrow platform was easy to use and understand, with clear feedback throughout the various processes.

8.1.5 Compatibility Testing

This type of testing is performed to ensure that the system is compatible with different software and hardware environments. Compatibility testing for the escrow platform would involve testing how well the platform works on different operating systems, browsers, and devices.

Test case 1: Compatibility with different browsers.

Input: Accessing the platform on different browsers such as Chrome, Firefox, Safari, and Edge.

Output: The platform should work smoothly on all the tested browsers, and all the features should work as expected.

Test case 2: Compatibility with different operating systems.

Input: Accessing the platform on different operating systems such as Windows, macOS, and Linux.

Output: The platform should work smoothly on all the tested operating systems, and all the features should work as expected.

Test case 3: Compatibility with different devices

Input: Accessing the platform on different devices such as desktop, laptop, tablet, and mobile.

Output: The platform should work smoothly on all the tested devices, and all the features should work as expected.

Test case 4: Compatibility with different network environments.

Input: Accessing the platform on different network environments such as LAN, WAN, VPN, and mobile data.

Output: The platform should work smoothly on all the tested network environments, and all the features should work as expected.

Test case 5: Compatibility with different versions of the blockchain technology

Input: Accessing the platform on different versions of the blockchain technology such as Ethereum, Hyperledger, and Corda.

Output: The platform should work smoothly on all the tested blockchain technologies, and all the features should work as expected.

CHAPTER 9

CONCLUSION

9. CONCLUSION

The escrow systems that we suggest are completely orthogonal to the method in which the files are shared and distributed among peers. The escrow service and the verification of the content can be done by the peer-to-peer service, or by a separate non-affiliated service. We hope that our study can stimulate more future research endeavors on this crucial problem in the blockchain.

CHAPTER 10

APPENDIX

10.1 SCREENSHOTS

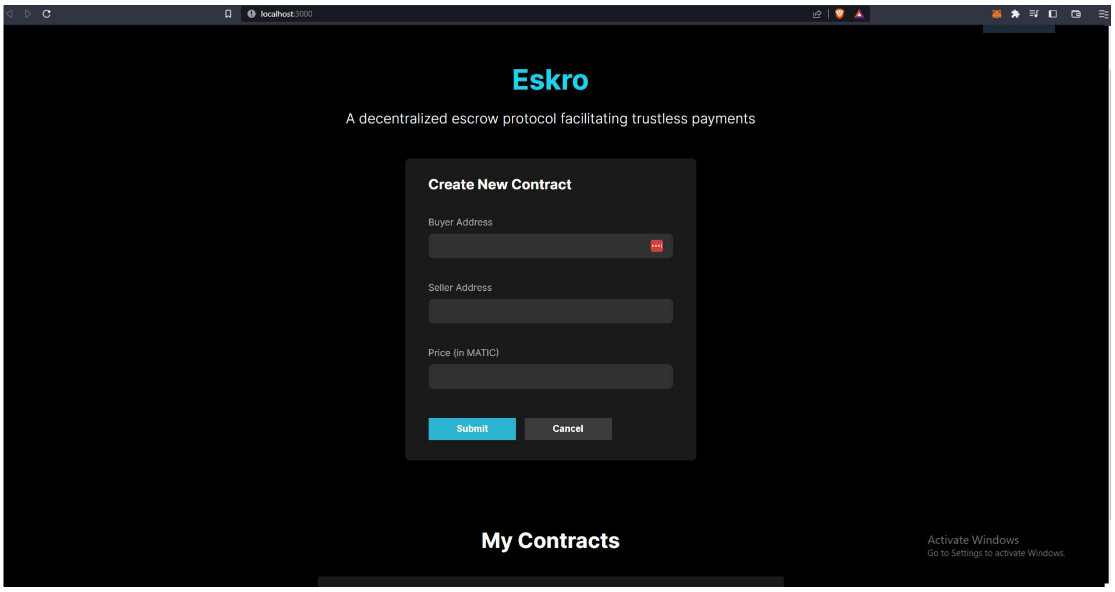
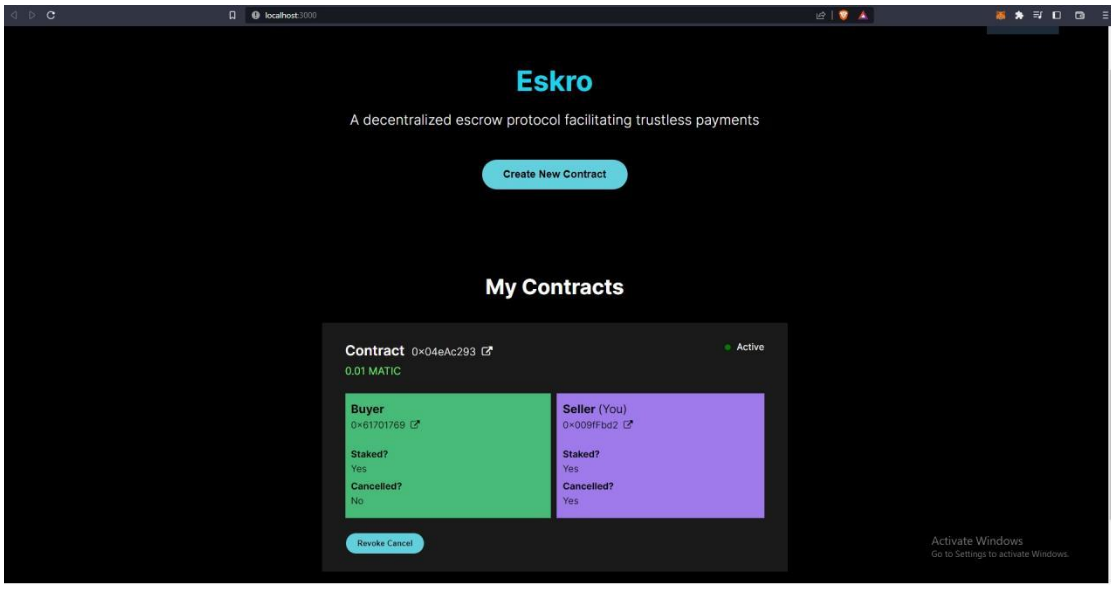


Fig 10.1 Main

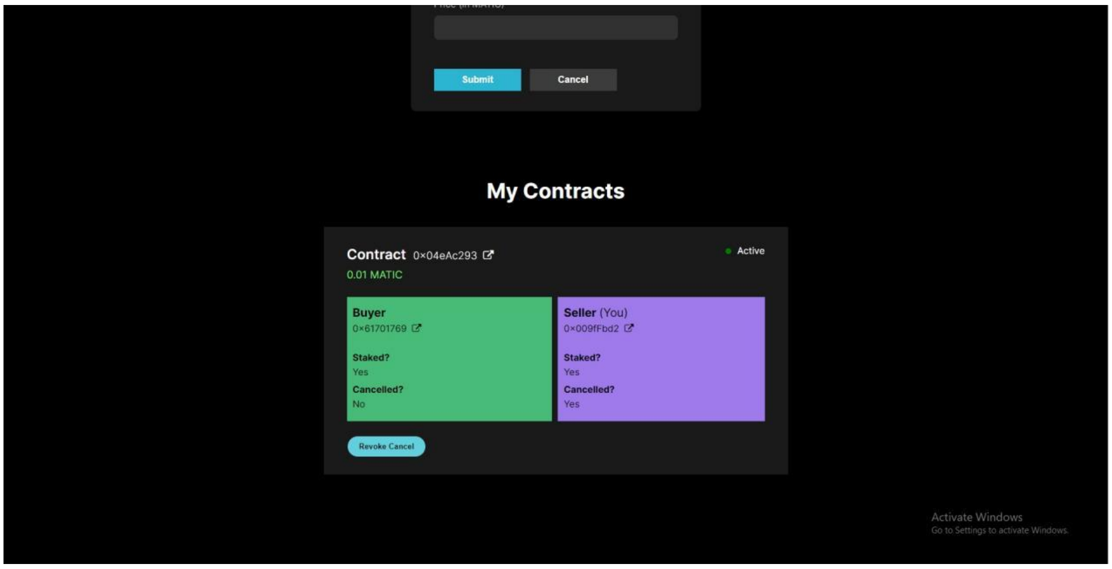

localhost:3000
Eskro
A decentralized escrow protocol facilitating trustless payments
Create New Contract
My Contracts
Revoke Cancel
Activate Windows
Go to Settings to activate Windows.

Contract 0x04eAc293
Active
0.01 MATIC

Buyer
0x61701769
Staked?
Yes
Cancelled?
No

Seller (You)
0x009fFbd2
Staked?
Yes
Cancelled?
Yes

Fig 10.2 New Contract



My Contracts

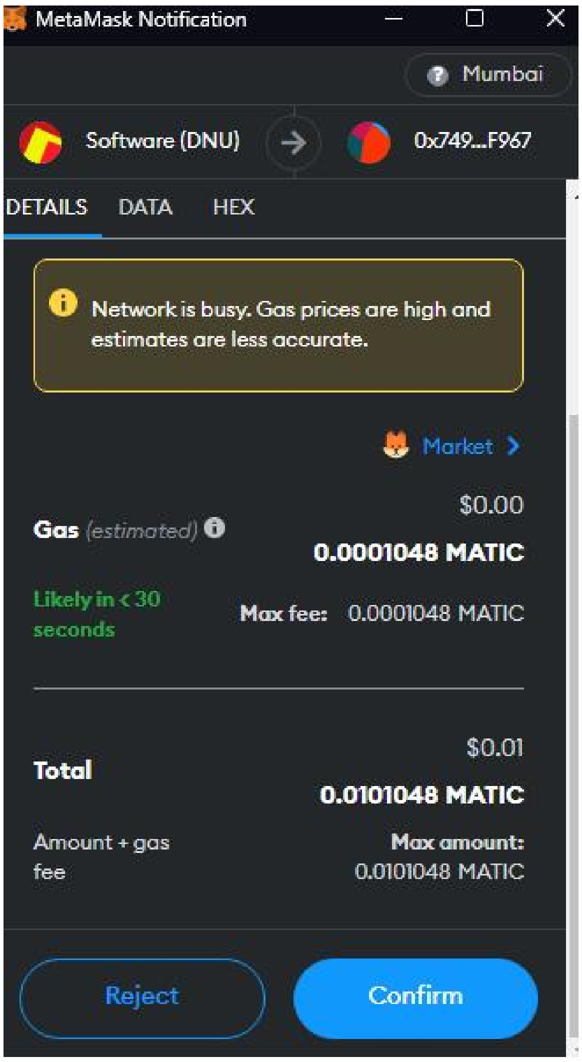
Contract 0×04eAc293 · Active

0.01 MATIC

Buyer
0×61701769
Staked?
Yes
Cancelled?
No

Seller (You)
0×009fFbd2
Staked?
Yes
Cancelled?
Yes

Fig 10.3 My Contracts



Network is busy. Gas prices are high and estimates are less accurate.

Gas *(estimated)* $0.00 0.0001048 MATIC

Likely in < 30 seconds

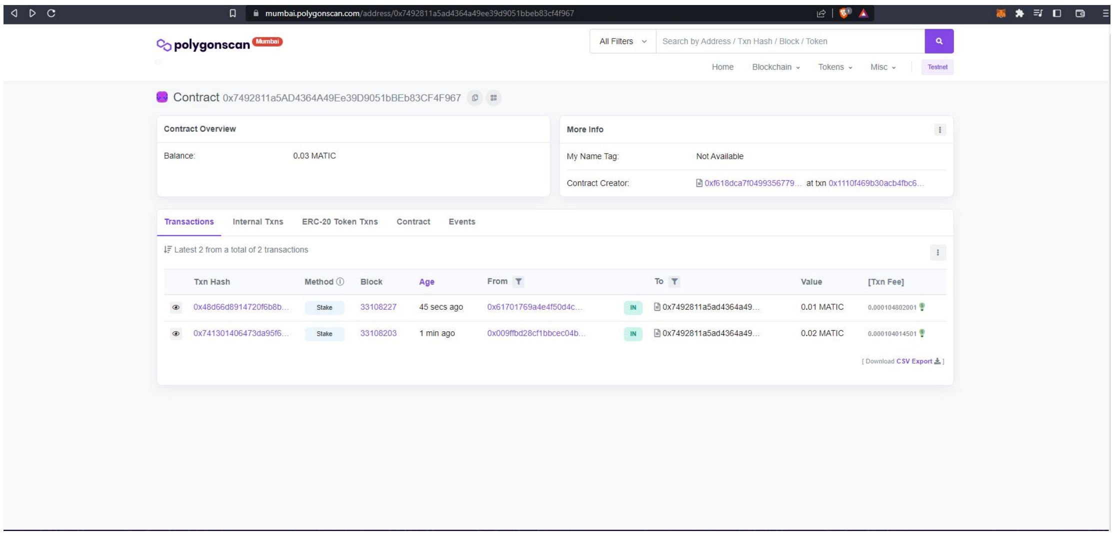
Max fee: 0.0001048 MATIC

Total $0.01 0.0101048 MATIC

Amount + gas fee

Max amount: 0.0101048 MATIC

Fig 10.4 Stake Transaction



mumbai.polygonscan.com/address/0x7492811a5ad4364a49ee39d9051bbeb83cf4f967

Contract 0x7492811a5AD4364A49Ee39D9051bBEb83CF4F967

Contract Overview

Balance: 0.03 MATIC

More Info

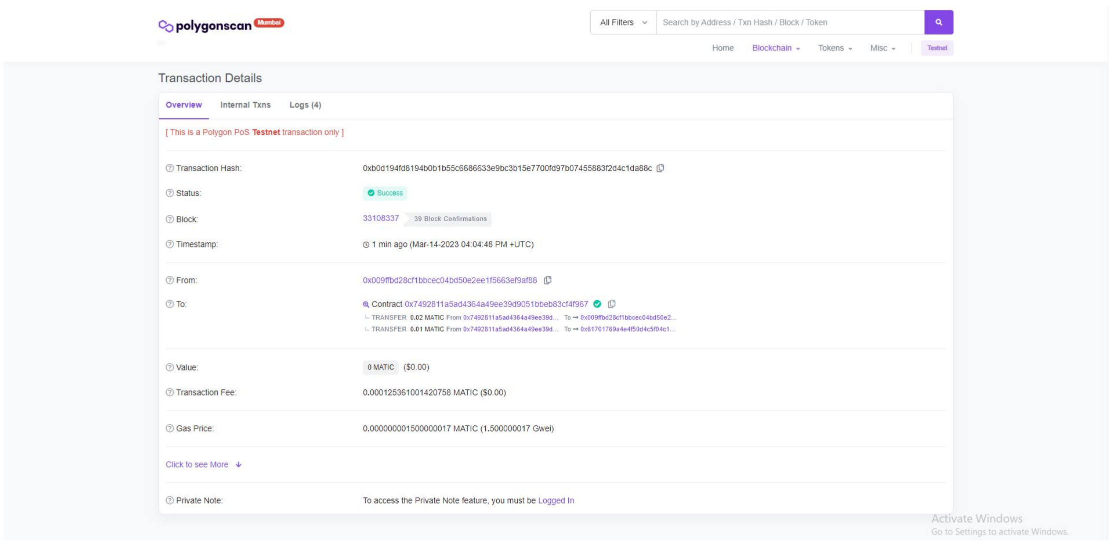
My Name Tag: Not Available

Contract Creator: 0xf618dca7f0499356779… at txn 0x1110f469b30acb4fbc6…

Latest 2 from a total of 2 transactions

	Txn Hash	Method	Block	Age	From		To	Value	[Txn Fee]
	0x48d66d8914720f6b8b…	Stake	33108227	45 secs ago	0x61701769a4e4f50d4c…	IN	0x7492811a5ad4364a49…	0.01 MATIC	0.000104802001
	0x741301406473da95f6…	Stake	33108203	1 min ago	0x009ffbd28cf1bbcec04b…	IN	0x7492811a5ad4364a49…	0.02 MATIC	0.000104014501

Fig 10.5 Block Explorer

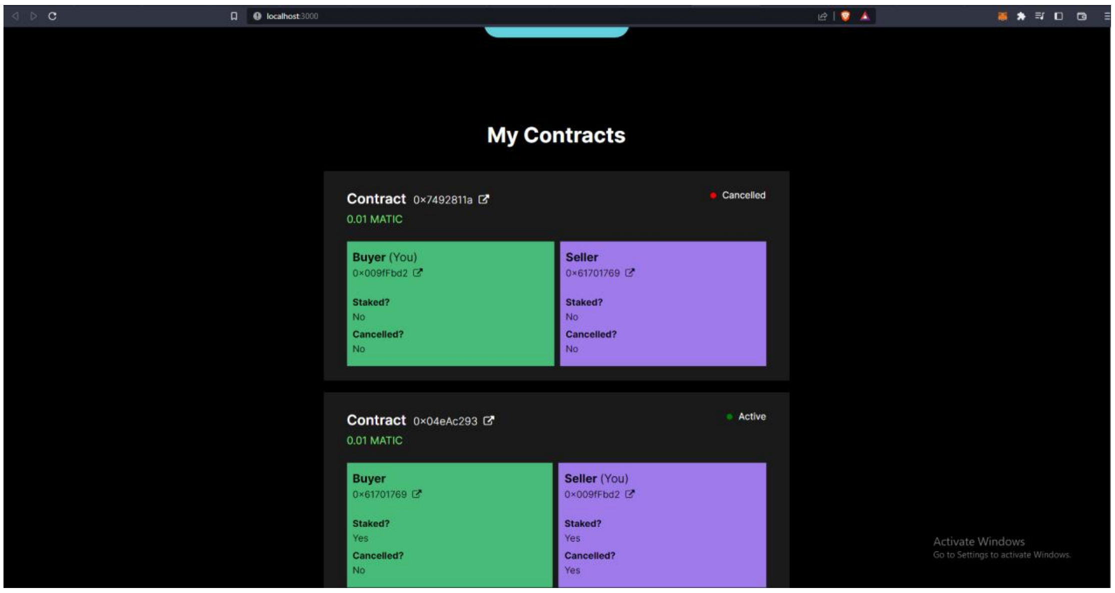


Transaction Details

[This is a Polygon PoS **Testnet** transaction only]

Transaction Hash:	0xb0d194fd8194b0b1b55c6686633e9bc3b15e7700fd97b07455883f2d4c1da88c
Status:	Success
Block:	33108337 39 Block Confirmations
Timestamp:	1 min ago (Mar-14-2023 04:04:48 PM +UTC)
From:	0x009ffbd28cf1bbcec04bd50e2ee1f5663ef9af88
To:	Contract 0x7492811a5ad4364a49ee39d9051bbeb83cf4f967 TRANSFER 0.02 MATIC From 0x7492811a5ad4364a49ee39d… To → 0x009ffbd28cf1bbcec04bd50e2… TRANSFER 0.01 MATIC From 0x7492811a5ad4364a49ee39d… To → 0x61701769a4e4f50d4c5f04c1…
Value:	0 MATIC ($0.00)
Transaction Fee:	0.000125361001420758 MATIC ($0.00)
Gas Price:	0.000000001500000017 MATIC (1.500000017 Gwei)
Private Note:	To access the Private Note feature, you must be Logged In

Fig 10.6 Cancel Details



My Contracts

Contract 0×7492811a
● Cancelled
0.01 MATIC

Buyer (You)
0×009fFbd2
Staked?
No
Cancelled?
No

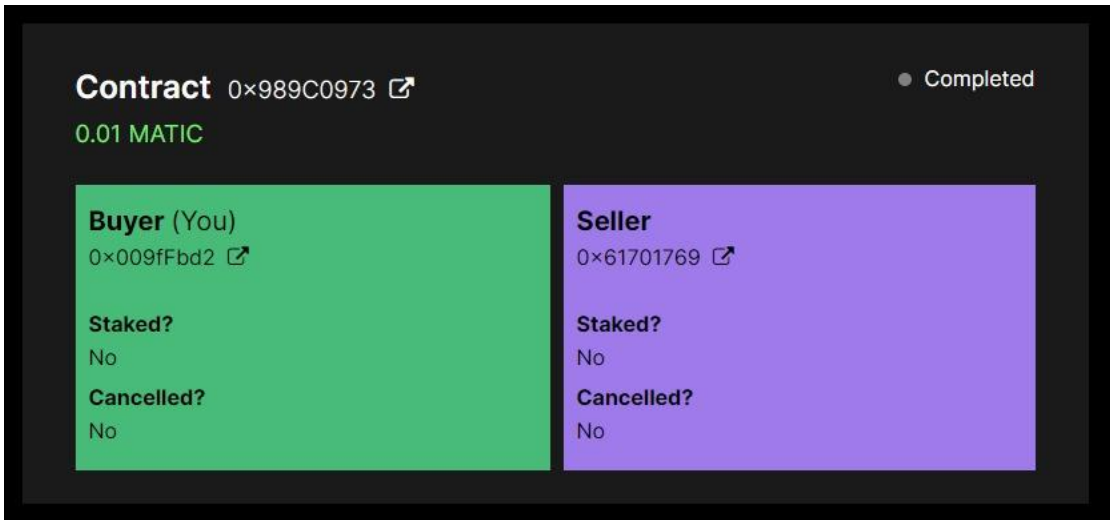
Seller
0×61701769
Staked?
No
Cancelled?
No

Contract 0×04eAc293
● Active
0.01 MATIC

Buyer
0×61701769
Staked?
Yes
Cancelled?
No

Seller (You)
0×009fFbd2
Staked?
Yes
Cancelled?
Yes

Fig 10.7 Canceled Transaction



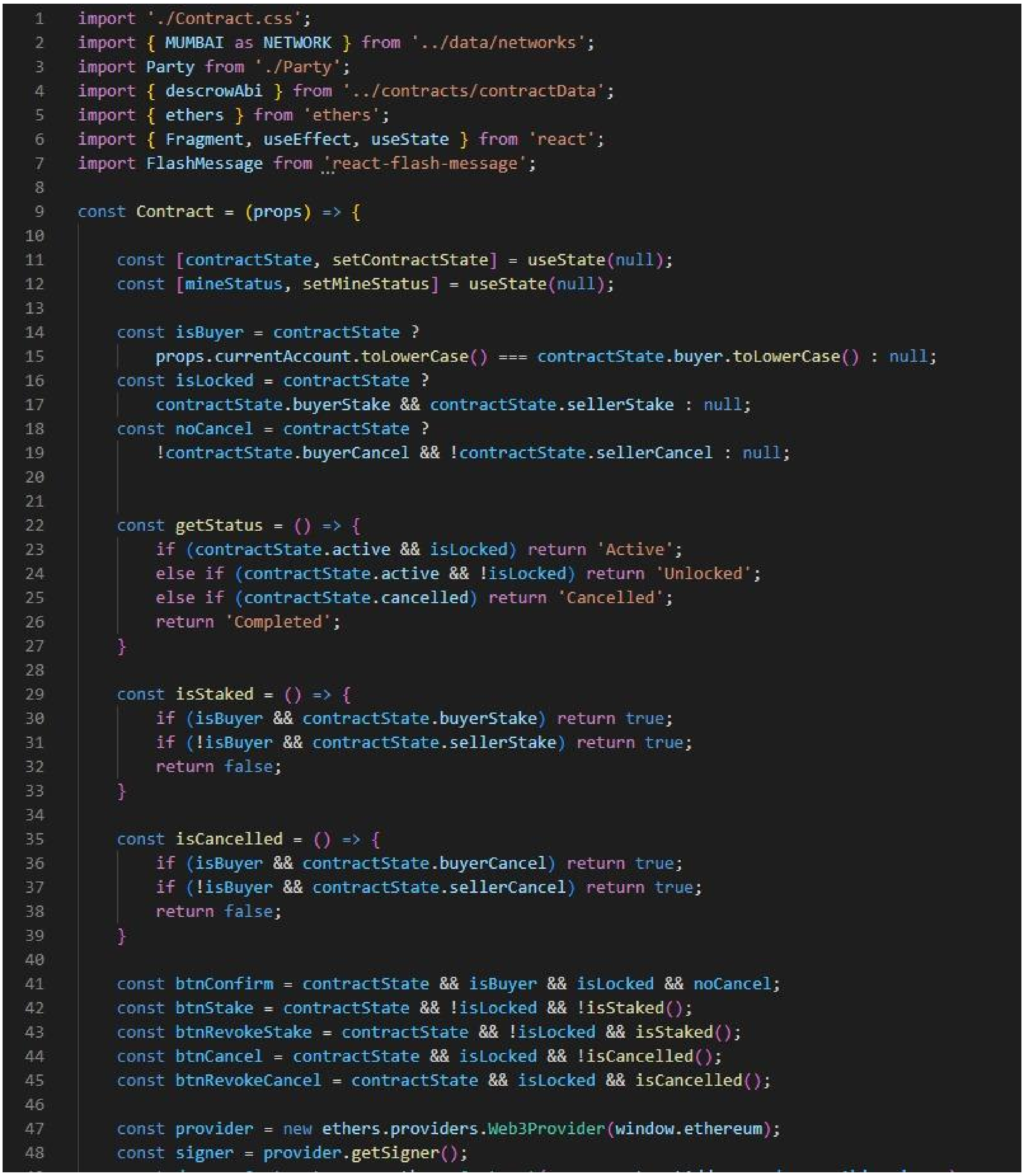
Contract 0×989C0973
● Completed
0.01 MATIC

Buyer (You)
0×009fFbd2
Staked?
No
Cancelled?
No

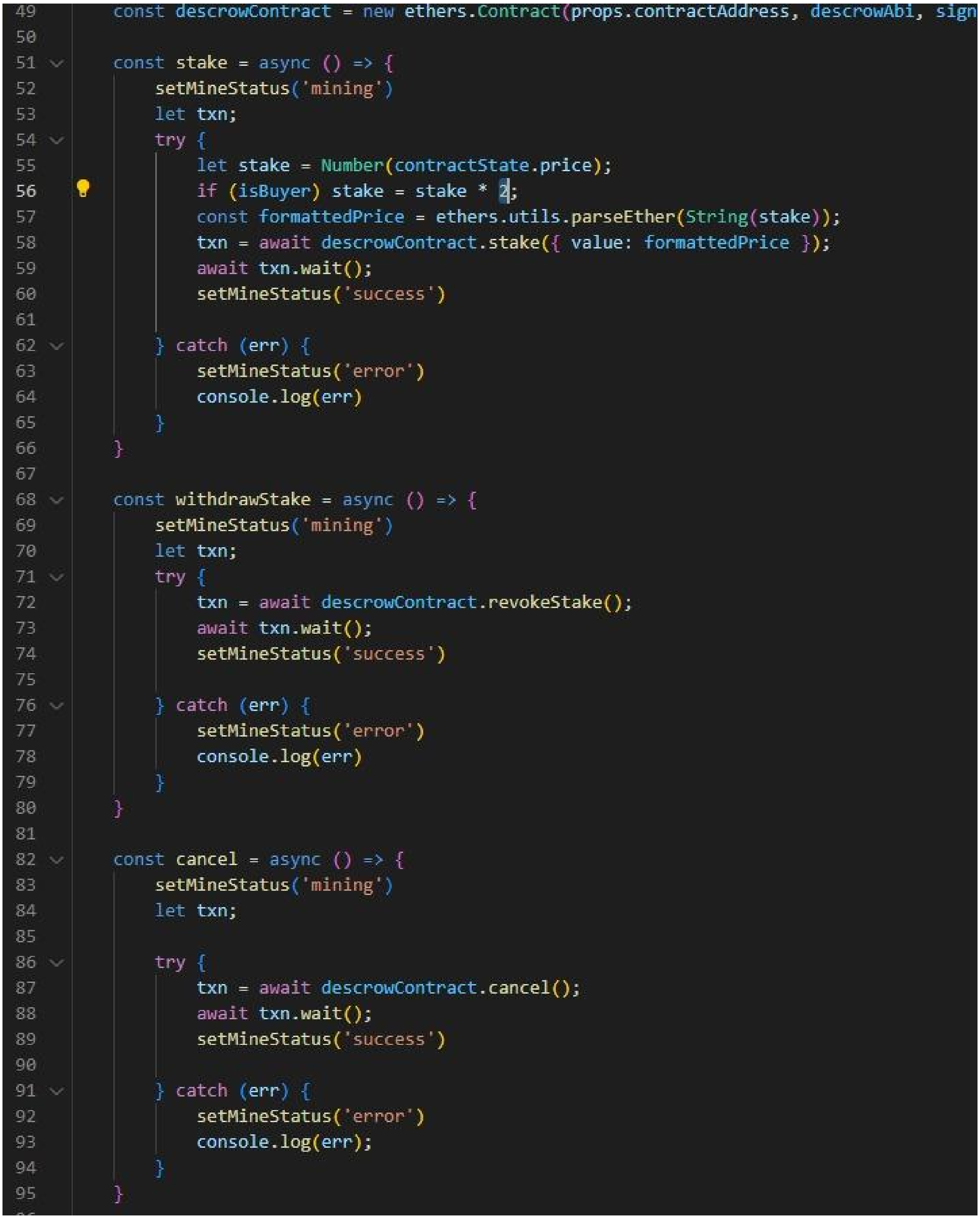
Seller
0×61701769
Staked?
No
Cancelled?
No

Fig 10.8 Completed Transaction

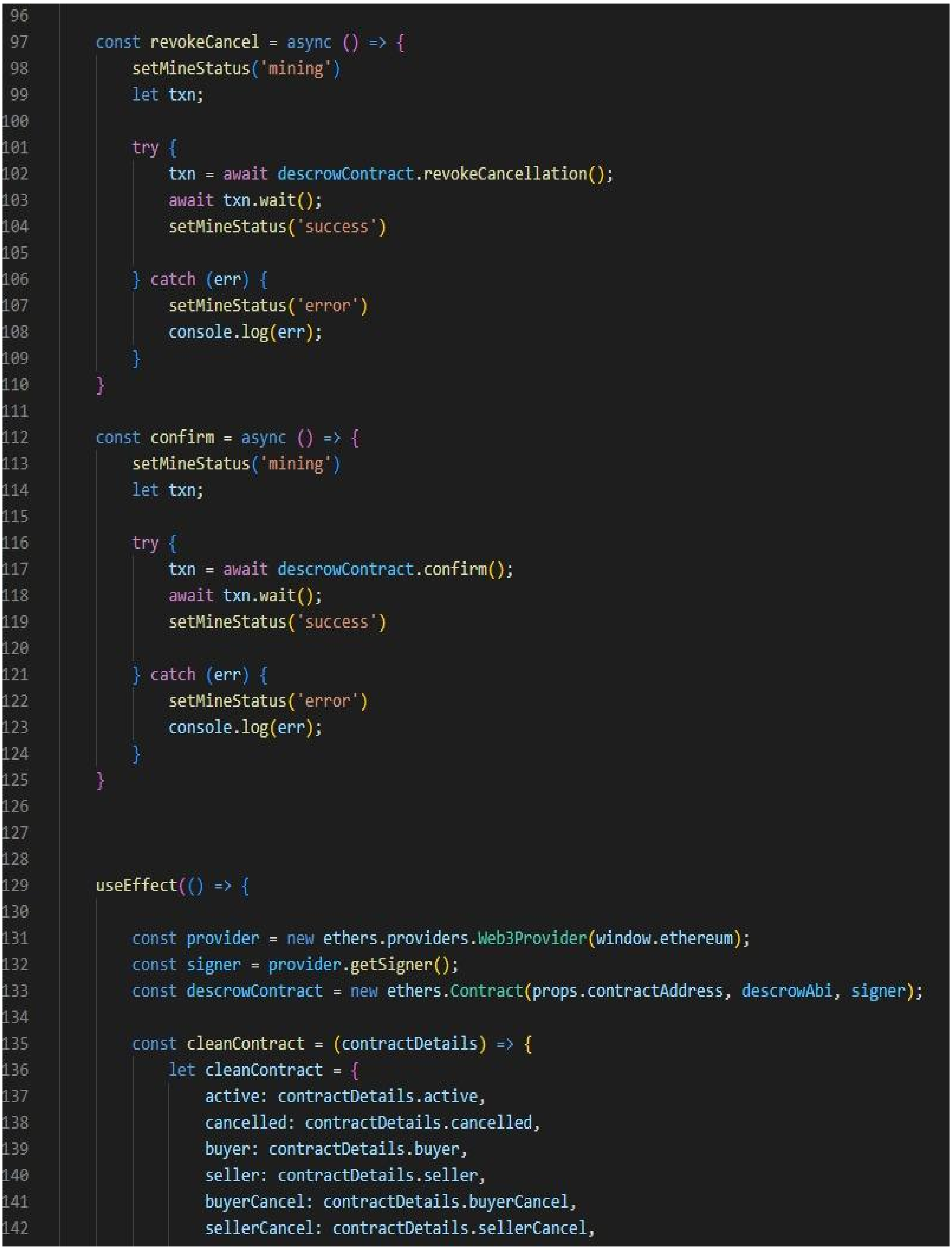
10.2 SOURCE CODE



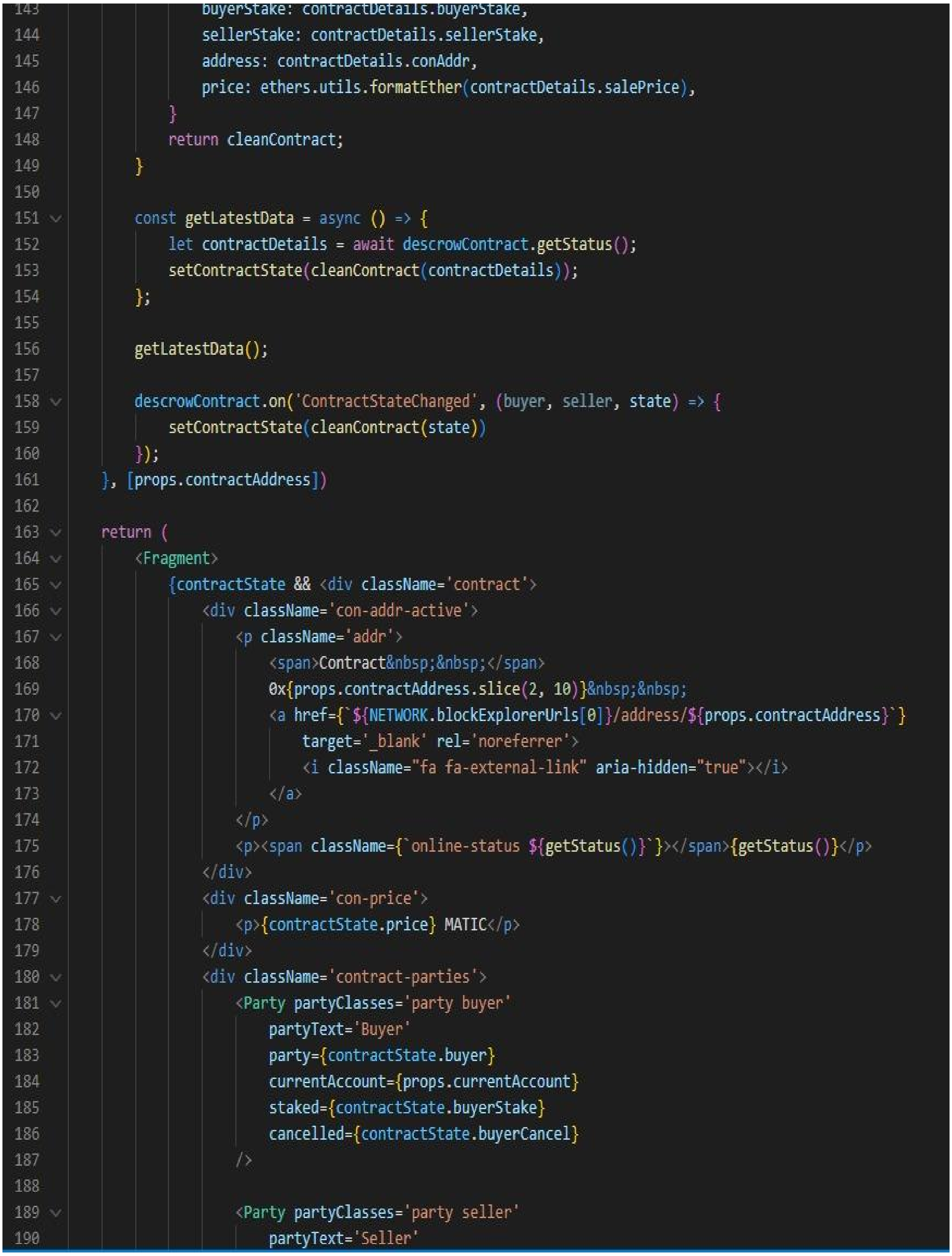
```javascript
import './Contract.css';
import { MUMBAI as NETWORK } from '../data/networks';
import Party from './Party';
import { descrowAbi } from '../contracts/contractData';
import { ethers } from 'ethers';
import { Fragment, useEffect, useState } from 'react';
import FlashMessage from 'react-flash-message';

const Contract = (props) => {

    const [contractState, setContractState] = useState(null);
    const [mineStatus, setMineStatus] = useState(null);

    const isBuyer = contractState ?
        props.currentAccount.toLowerCase() === contractState.buyer.toLowerCase() : null;
    const isLocked = contractState ?
        contractState.buyerStake && contractState.sellerStake : null;
    const noCancel = contractState ?
        !contractState.buyerCancel && !contractState.sellerCancel : null;


    const getStatus = () => {
        if (contractState.active && isLocked) return 'Active';
        else if (contractState.active && !isLocked) return 'Unlocked';
        else if (contractState.cancelled) return 'Cancelled';
        return 'Completed';
    }

    const isStaked = () => {
        if (isBuyer && contractState.buyerStake) return true;
        if (!isBuyer && contractState.sellerStake) return true;
        return false;
    }

    const isCancelled = () => {
        if (isBuyer && contractState.buyerCancel) return true;
        if (!isBuyer && contractState.sellerCancel) return true;
        return false;
    }

    const btnConfirm = contractState && isBuyer && isLocked && noCancel;
    const btnStake = contractState && !isLocked && !isStaked();
    const btnRevokeStake = contractState && !isLocked && isStaked();
    const btnCancel = contractState && isLocked && !isCancelled();
    const btnRevokeCancel = contractState && isLocked && isCancelled();

    const provider = new ethers.providers.Web3Provider(window.ethereum);
    const signer = provider.getSigner();
```

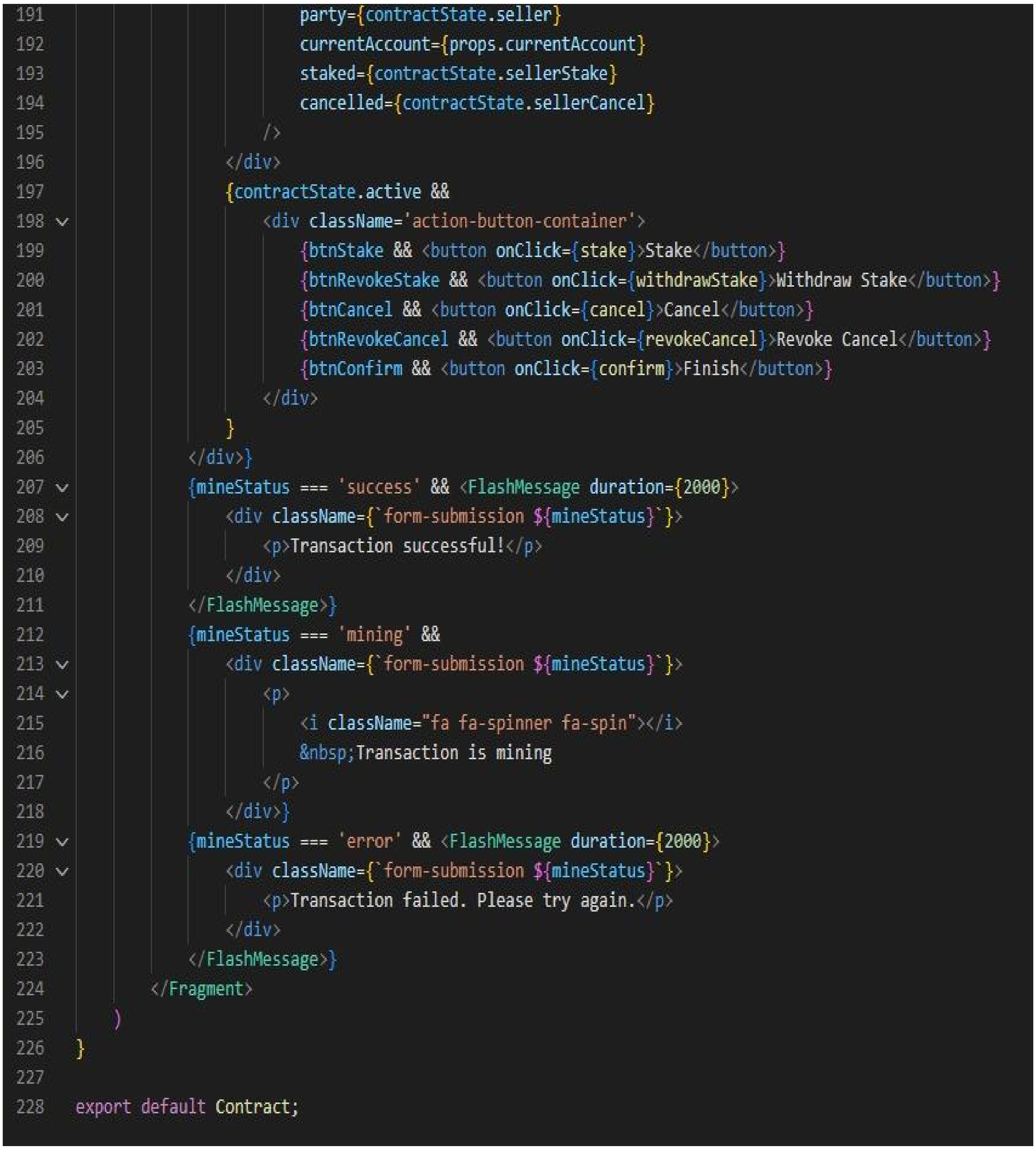


```
    const descrowContract = new ethers.Contract(props.contractAddress, descrowAbi, sign

    const stake = async () => {
        setMineStatus('mining')
        let txn;
        try {
            let stake = Number(contractState.price);
            if (isBuyer) stake = stake * 2;
            const formattedPrice = ethers.utils.parseEther(String(stake));
            txn = await descrowContract.stake({ value: formattedPrice });
            await txn.wait();
            setMineStatus('success')

        } catch (err) {
            setMineStatus('error')
            console.log(err)
        }
    }

    const withdrawStake = async () => {
        setMineStatus('mining')
        let txn;
        try {
            txn = await descrowContract.revokeStake();
            await txn.wait();
            setMineStatus('success')

        } catch (err) {
            setMineStatus('error')
            console.log(err)
        }
    }

    const cancel = async () => {
        setMineStatus('mining')
        let txn;

        try {
            txn = await descrowContract.cancel();
            await txn.wait();
            setMineStatus('success')

        } catch (err) {
            setMineStatus('error')
            console.log(err);
        }
    }
```

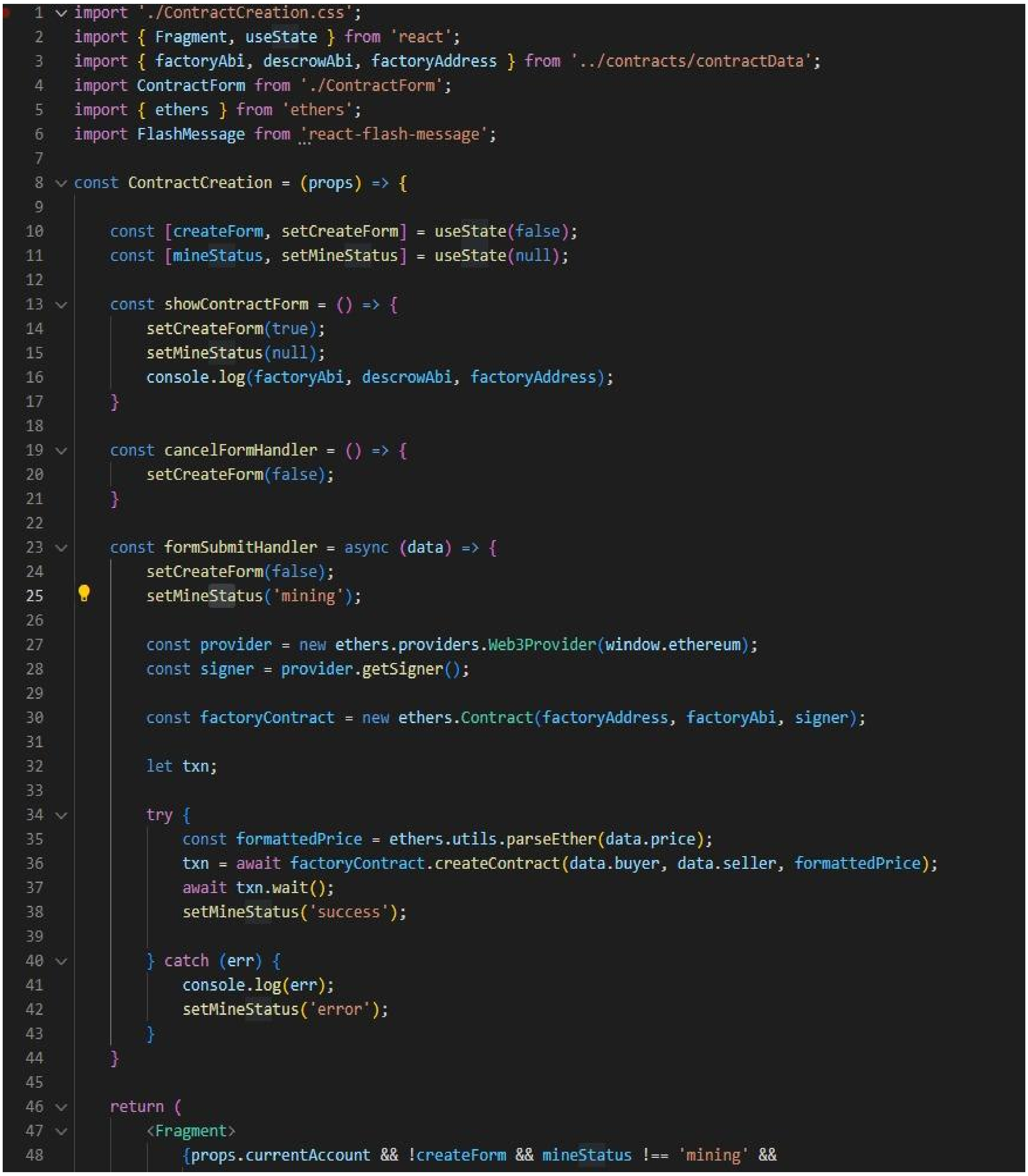


```

    const revokeCancel = async () => {
        setMineStatus('mining')
        let txn;

        try {
            txn = await descrowContract.revokeCancellation();
            await txn.wait();
            setMineStatus('success')

        } catch (err) {
            setMineStatus('error')
            console.log(err);
        }
    }

    const confirm = async () => {
        setMineStatus('mining')
        let txn;

        try {
            txn = await descrowContract.confirm();
            await txn.wait();
            setMineStatus('success')

        } catch (err) {
            setMineStatus('error')
            console.log(err);
        }
    }



    useEffect(() => {

        const provider = new ethers.providers.Web3Provider(window.ethereum);
        const signer = provider.getSigner();
        const descrowContract = new ethers.Contract(props.contractAddress, descrowAbi, signer);

        const cleanContract = (contractDetails) => {
            let cleanContract = {
                active: contractDetails.active,
                cancelled: contractDetails.cancelled,
                buyer: contractDetails.buyer,
                seller: contractDetails.seller,
                buyerCancel: contractDetails.buyerCancel,
                sellerCancel: contractDetails.sellerCancel,
```

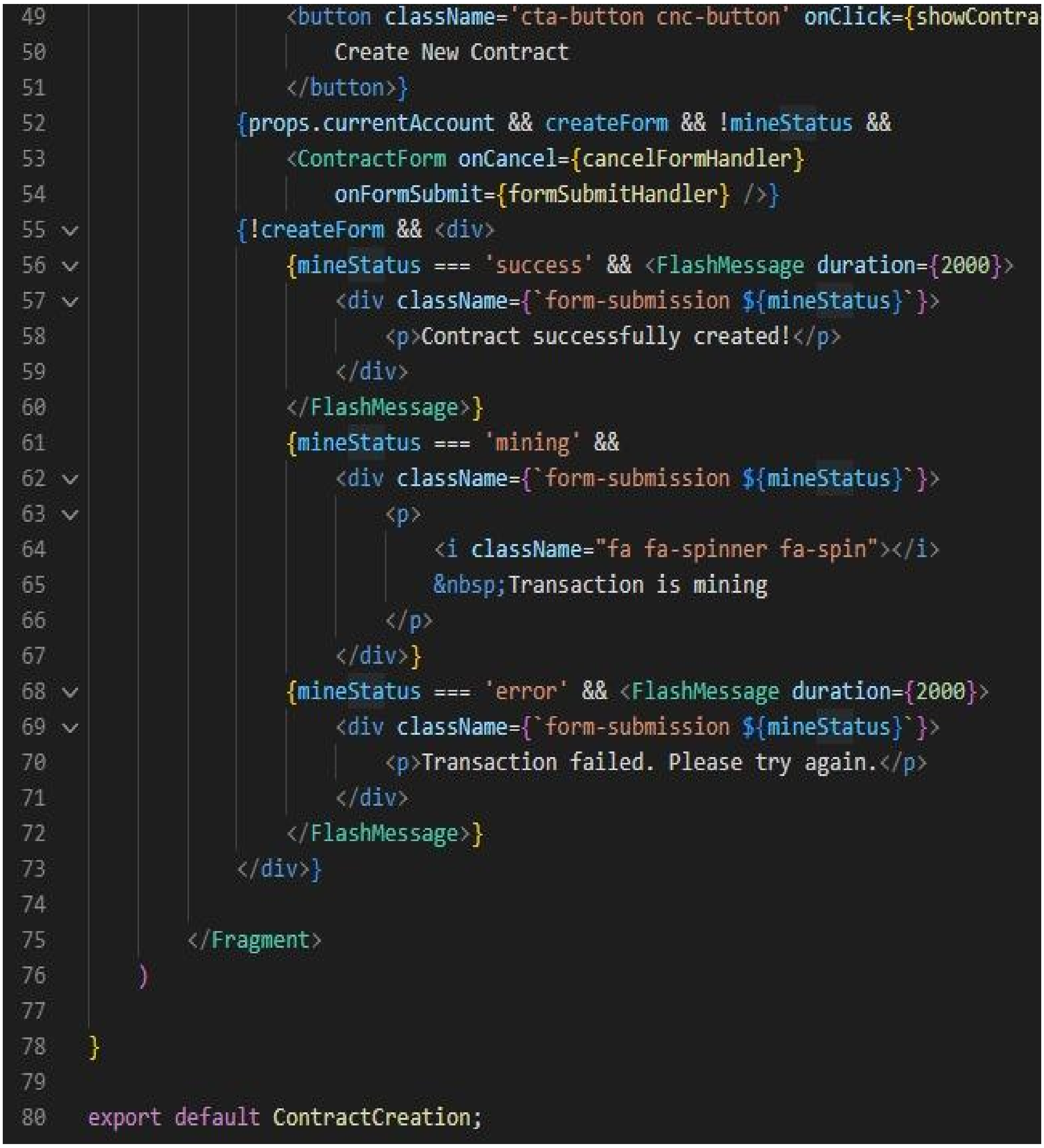

```
                buyerStake: contractDetails.buyerStake,
                sellerStake: contractDetails.sellerStake,
                address: contractDetails.conAddr,
                price: ethers.utils.formatEther(contractDetails.salePrice),
            }
            return cleanContract;
        }

        const getLatestData = async () => {
            let contractDetails = await descrowContract.getStatus();
            setContractState(cleanContract(contractDetails));
        };

        getLatestData();

        descrowContract.on('ContractStateChanged', (buyer, seller, state) => {
            setContractState(cleanContract(state))
        });
    }, [props.contractAddress])

    return (
        <Fragment>
            {contractState && <div className='contract'>
                <div className='con-addr-active'>
                    <p className='addr'>
                        <span>Contract&nbsp;&nbsp;</span>
                        0x{props.contractAddress.slice(2, 10)}&nbsp;&nbsp;
                        <a href={`${NETWORK.blockExplorerUrls[0]}/address/${props.contractAddress}`}
                            target='_blank' rel='noreferrer'>
                            <i className="fa fa-external-link" aria-hidden="true"></i>
                        </a>
                    </p>
                    <p><span className={`online-status ${getStatus()}`}></span>{getStatus()}</p>
                </div>
                <div className='con-price'>
                    <p>{contractState.price} MATIC</p>
                </div>
                <div className='contract-parties'>
                    <Party partyClasses='party buyer'
                        partyText='Buyer'
                        party={contractState.buyer}
                        currentAccount={props.currentAccount}
                        staked={contractState.buyerStake}
                        cancelled={contractState.buyerCancel}
                    />

                    <Party partyClasses='party seller'
                        partyText='Seller'
```



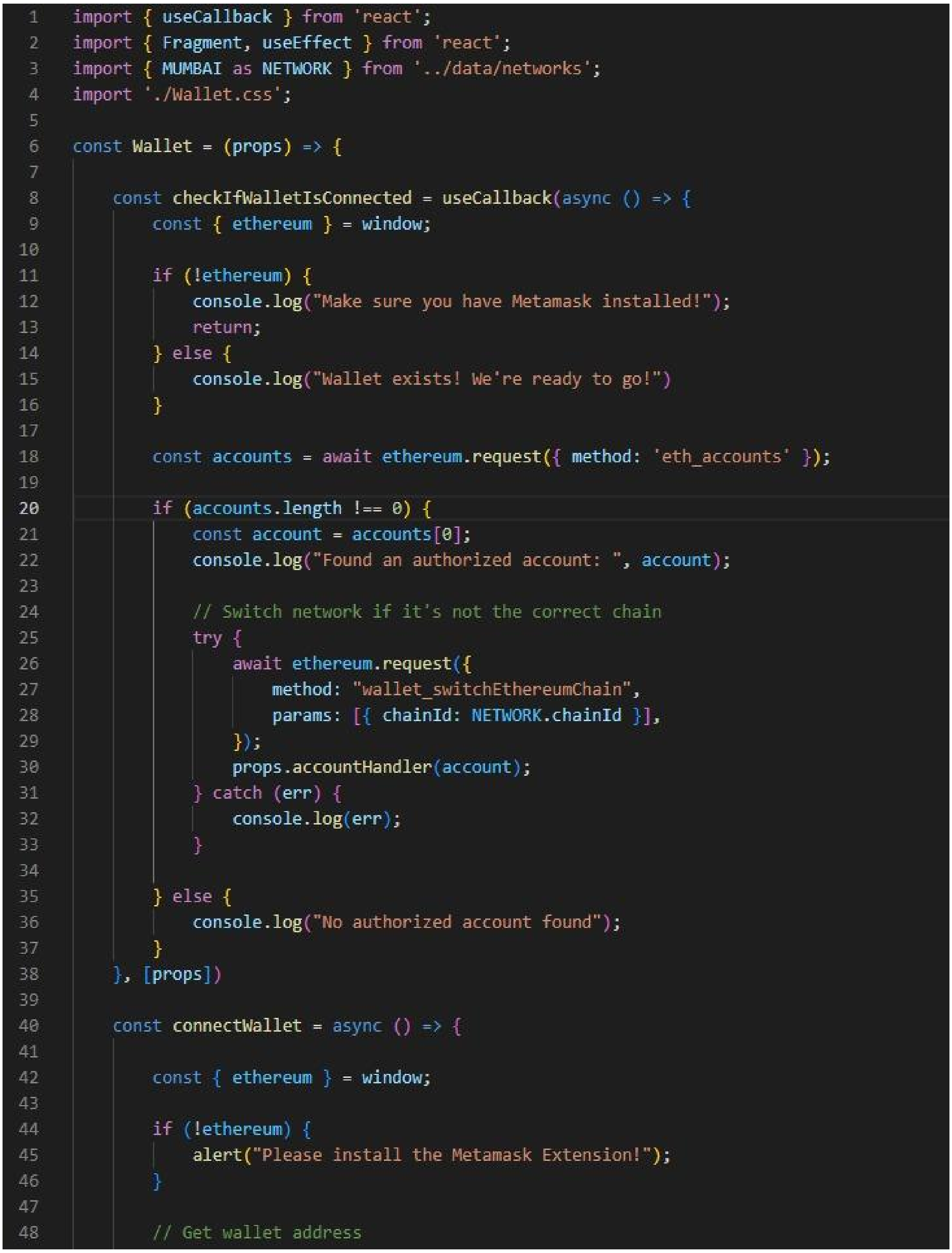
```
                        party={contractState.seller}
                        currentAccount={props.currentAccount}
                        staked={contractState.sellerStake}
                        cancelled={contractState.sellerCancel}
                    />
                </div>
                {contractState.active &&
                    <div className='action-button-container'>
                        {btnStake && <button onClick={stake}>Stake</button>}
                        {btnRevokeStake && <button onClick={withdrawStake}>Withdraw Stake</button>}
                        {btnCancel && <button onClick={cancel}>Cancel</button>}
                        {btnRevokeCancel && <button onClick={revokeCancel}>Revoke Cancel</button>}
                        {btnConfirm && <button onClick={confirm}>Finish</button>}
                    </div>
                }
            </div>}
            {mineStatus === 'success' && <FlashMessage duration={2000}>
                <div className={`form-submission ${mineStatus}`}>
                    <p>Transaction successful!</p>
                </div>
            </FlashMessage>}
            {mineStatus === 'mining' &&
                <div className={`form-submission ${mineStatus}`}>
                    <p>
                        <i className="fa fa-spinner fa-spin"></i>
                        &nbsp;Transaction is mining
                    </p>
                </div>}
            {mineStatus === 'error' && <FlashMessage duration={2000}>
                <div className={`form-submission ${mineStatus}`}>
                    <p>Transaction failed. Please try again.</p>
                </div>
            </FlashMessage>}
        </Fragment>
    )
}

export default Contract;
```



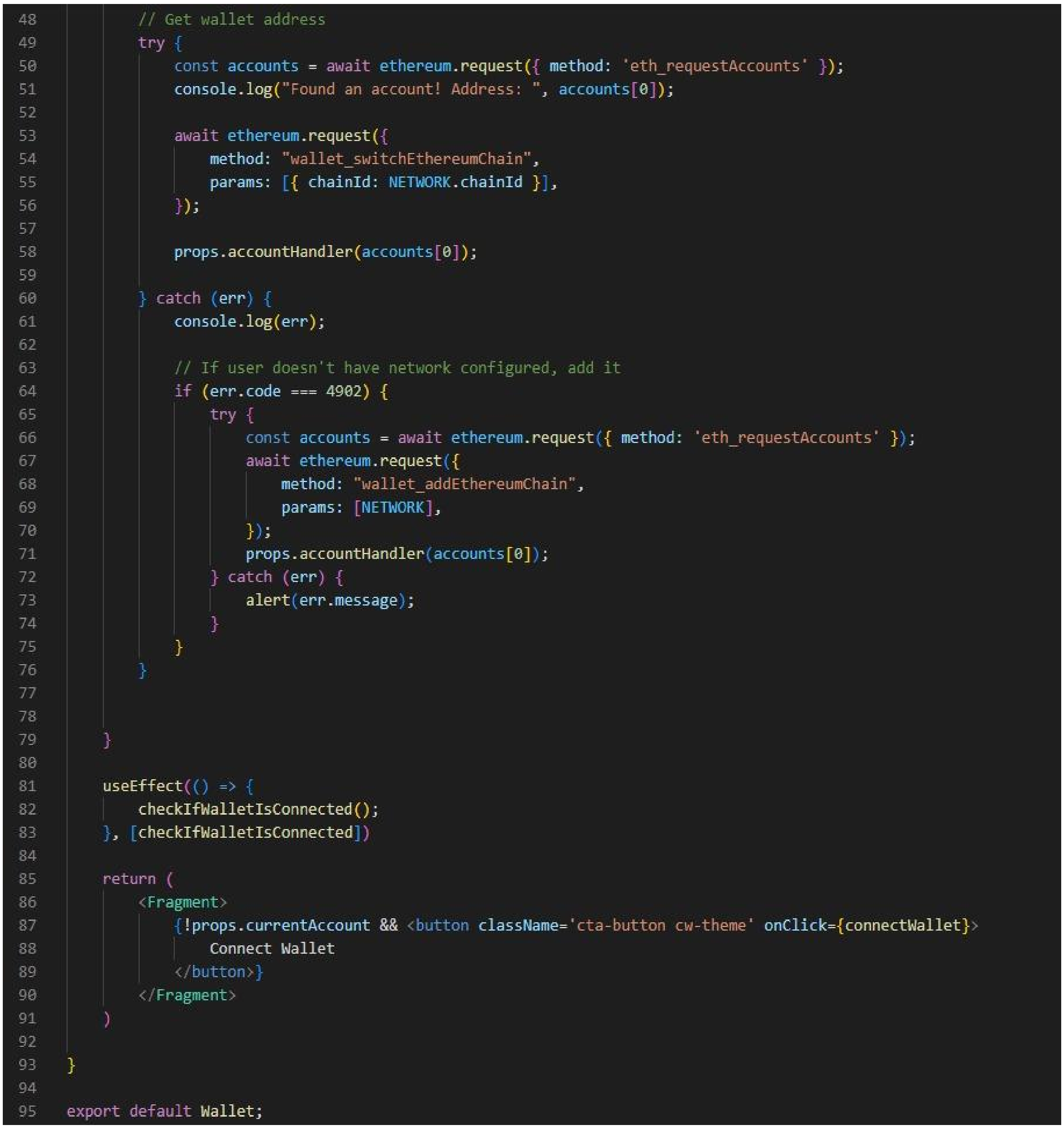
```
import './ContractCreation.css';
import { Fragment, useState } from 'react';
import { factoryAbi, descrowAbi, factoryAddress } from '../contracts/contractData';
import ContractForm from './ContractForm';
import { ethers } from 'ethers';
import FlashMessage from 'react-flash-message';

const ContractCreation = (props) => {

    const [createForm, setCreateForm] = useState(false);
    const [mineStatus, setMineStatus] = useState(null);

    const showContractForm = () => {
        setCreateForm(true);
        setMineStatus(null);
        console.log(factoryAbi, descrowAbi, factoryAddress);
    }

    const cancelFormHandler = () => {
        setCreateForm(false);
    }

    const formSubmitHandler = async (data) => {
        setCreateForm(false);
        setMineStatus('mining');

        const provider = new ethers.providers.Web3Provider(window.ethereum);
        const signer = provider.getSigner();

        const factoryContract = new ethers.Contract(factoryAddress, factoryAbi, signer);

        let txn;

        try {
            const formattedPrice = ethers.utils.parseEther(data.price);
            txn = await factoryContract.createContract(data.buyer, data.seller, formattedPrice);
            await txn.wait();
            setMineStatus('success');

        } catch (err) {
            console.log(err);
            setMineStatus('error');
        }
    }

    return (
        <Fragment>
            {props.currentAccount && !createForm && mineStatus !== 'mining' &&
```



```
                <button className='cta-button cnc-button' onClick={showContra
                    Create New Contract
                </button>}
            {props.currentAccount && createForm && !mineStatus &&
                <ContractForm onCancel={cancelFormHandler}
                    onFormSubmit={formSubmitHandler} />}
            {!createForm && <div>
                {mineStatus === 'success' && <FlashMessage duration={2000}>
                    <div className={`form-submission ${mineStatus}`}>
                        <p>Contract successfully created!</p>
                    </div>
                </FlashMessage>}
                {mineStatus === 'mining' &&
                    <div className={`form-submission ${mineStatus}`}>
                        <p>
                            <i className="fa fa-spinner fa-spin"></i>
                            &nbsp;Transaction is mining
                        </p>
                    </div>}
                {mineStatus === 'error' && <FlashMessage duration={2000}>
                    <div className={`form-submission ${mineStatus}`}>
                        <p>Transaction failed. Please try again.</p>
                    </div>
                </FlashMessage>}
            </div>}

        </Fragment>
    )

}

export default ContractCreation;
```



```
import { useCallback } from 'react';
import { Fragment, useEffect } from 'react';
import { MUMBAI as NETWORK } from '../data/networks';
import './Wallet.css';

const Wallet = (props) => {

    const checkIfWalletIsConnected = useCallback(async () => {
        const { ethereum } = window;

        if (!ethereum) {
            console.log("Make sure you have Metamask installed!");
            return;
        } else {
            console.log("Wallet exists! We're ready to go!")
        }

        const accounts = await ethereum.request({ method: 'eth_accounts' });

        if (accounts.length !== 0) {
            const account = accounts[0];
            console.log("Found an authorized account: ", account);

            // Switch network if it's not the correct chain
            try {
                await ethereum.request({
                    method: "wallet_switchEthereumChain",
                    params: [{ chainId: NETWORK.chainId }],
                });
                props.accountHandler(account);
            } catch (err) {
                console.log(err);
            }

        } else {
            console.log("No authorized account found");
        }
    }, [props])

    const connectWallet = async () => {

        const { ethereum } = window;

        if (!ethereum) {
            alert("Please install the Metamask Extension!");
        }

        // Get wallet address
```



```
        // Get wallet address
        try {
            const accounts = await ethereum.request({ method: 'eth_requestAccounts' });
            console.log("Found an account! Address: ", accounts[0]);

            await ethereum.request({
                method: "wallet_switchEthereumChain",
                params: [{ chainId: NETWORK.chainId }],
            });

            props.accountHandler(accounts[0]);

        } catch (err) {
            console.log(err);

            // If user doesn't have network configured, add it
            if (err.code === 4902) {
                try {
                    const accounts = await ethereum.request({ method: 'eth_requestAccounts' });
                    await ethereum.request({
                        method: "wallet_addEthereumChain",
                        params: [NETWORK],
                    });
                    props.accountHandler(accounts[0]);
                } catch (err) {
                    alert(err.message);
                }
            }
        }


    }

    useEffect(() => {
        checkIfWalletIsConnected();
    }, [checkIfWalletIsConnected])

    return (
        <Fragment>
            {!props.currentAccount && <button className='cta-button cw-theme' onClick={connectWallet}>
                Connect Wallet
            </button>}
        </Fragment>
    )

}

export default Wallet;
```

CHAPTER 11

BIBLIOGRAPHY

[1] Haya r sahib, Khaled Saleh, “Blockchain based physical delivery of proof system", Khalifa university conference and electronics department,2018

[2] shingling wang, ixia yang, yawling Jahan, “Auditable Protocols for Fair Payment and Physical Asset Delivery Based on Smart Contracts”, Xi’an university of technology conference in science,2019

[3] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng and V. C. M. Leung, "Decentralized Applications: The Blockchain-Empowered Software System," in IEEE Access,2018

[4] N. Z. Aitzaz and D. Voinovich, "Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams," in IEEE Transactions on Dependable and Secure Computing, 2018

[5] Nasheed khan, Faiza lousily, “Blockchain smart contracts: Applications, challenges, and future trends”, Peer-to-peer networking applications , 2021

[6] Enes Erden, Momin kibbe, kernel akala, “A Bitcoin payment network with reduced transaction fees and confirmation times”, Computer networks and technology conference,2021

[7] Reza Trapdoor, Peja ghazi, “Block by block: A blockchain-based peer-to-peer business transaction for international trade”, Technological forecast and social change conferences, 2022

[8] Steven Goldfaden, Joseph Bonneau, Rosario Gennaro & Arvind Narayanan , “Escrow Protocols for Cryptocurrencies: How to Buy Physical Goods Using Bitcoin”, International Conference on Financial Cryptography and Data Security, 2017

[9] Z. Hong, Z. Wang, W. Cai and V. C. M. Leung, "Connectivity-aware task outsourcing and scheduling in D2D networks", Proc. 26th Int. Conf. Compute. Common. Newt. (ICCCN),2017

[10] M. Wuhrer and U. Zdun, "Smart contracts: Security patterns in the Ethereum ecosystem and solidity", Proc. Int. Workshop Blockchain Oriented Soft. Eng. (IWBOSE), 2018.