Double Deposit Escrow Smart Contracts for Decentralized E-Commerce

Matin Ghaffari, Wenyuan “Sandy” Chen, Yu Huang, Alan Amirian

1 Abstract

The online marketplace and escrow system stand to benefit tremendously from blockchain technology and decentralization. Such systems save transactors money, minimize privacy risks, and give users control over their transactions, rather than relying on often unreliable, restrictive, and expensive third-party platforms. For these reasons, we implement a decentralized, Ethereum-based escrow marketplace that provides a setting for untrusted parties to initiate transactions without authority-bearing third-party arbiters in order to maintain anonymity, transaction integrity, and transparency. With blockchain technology, we are able to leverage smart contracts and use Ether to ensure that both parties remain compliant with their agreement, while also minimizing many of the aforementioned risks and costs of traditional third-party platforms. By implementing smart contracts through the Remix IDE and deploying them to an Ethereum network, and with the use of cryptocurrency wallets such as MetaMask, we are able to run transactions anonymously and securely store them on the Ethereum blockchain. This protocol creates a strong monetary incentive for both parties to go ahead with their transaction, while also keeping their transaction private and secure. More specifically, our smart contract allows one seller to sell items to any quantity of buyers at once. Moreover, using Javascript, HTML, and CSS, we designed and built a website that serves as the front end of our smart contract, so that users don’t have to interact on Etherscan, which requires some knowledge of the programming language Solidity. Simply put, we are providing a layer of abstraction for all

parties while also simplifying the process for them. Thus, the website can serve as a decentralized e-commerce application, where sellers and buyers can use their Metamask wallets to perform secure and efficient commerce.

2 Introduction

The emergence of Web3.0, a new concept of the World Wide Web, introduces the newfound ideas of decentralization and blockchain technologies which display the potential to improve safety, costs, and privacy in various industries, such as the online marketplace and escrow system. Currently, online marketplace exchanges like Amazon require the private information of users, a cut of the seller's profits ranging from 6%-45% of the total amount of the sale,[6] and are not very resistant to online faults and attacks. All of these pose risks to buyers and sellers while also costing sellers and buyers more money. However, a Web 3.0-influenced, decentralized, Ethereum-based escrow system would only require one's public wallet address and negligible gas fees instead of a substantial portion of the seller's profits. Trusted third-party firms such as Amazon and eBay were critical prior to blockchain technology and smart contracts since they address the issue of trust between unknown buyers and sellers in a digital marketplace. These third parties would establish trust between the buyer and seller by acting as a mediator that ensures that the buyer will pay and the seller will deliver. However, with smart contracts, the open source code may replace such firms as mediators, while also eliminating the need for trust between transactors, but without the aforementioned drawbacks of centralized third parties.

Additionally, decentralized platforms are more tolerant of faults and more resistant to attacks, making them a better alternative than traditional online marketplace and escrow platforms.[1] For example, attacks such as DDOS are more difficult to execute against decentralized networks because nodes are spread all across the world, making DDOS attacks

very expensive to conduct.[4] Since blockchains are distributed on multiple nodes, there is no central entity managing the chain, but rather a decentralized peer-to-peer network that prevents single points of failure, which third parties are often vulnerable to. A decentralized peer-to-peer network allows for each node to verify blocks or reject blocks that are tampered with and come to a consensus with the rest of the network before the block is added throughout the network. This idea makes blockchain much more secure and reliable than third parties, since dishonesty or attacks would be less feasible, as they would require tampering with all the blocks on the chain and getting the network's consensus, in turn eliminating single points of failure. Furthermore, data leaks are a growing concern, since third-party firms often store their users' personal information, which makes them a common target for hackers and security breaches, greatly jeopardizing the privacy of their users. However, since cryptocurrency wallets have no links to personal information, blockchain significantly reduces data privacy concerns since only the transactors' wallet addresses and transaction details are stored on the blockchain.

A decentralized escrow smart contract would greatly reduce the need for human interaction and minimize the risk of litigation, in turn reducing costs as well. Via smart contract code, users may have more trust and a greater sense of reliability, as smart contracts are open source and transparent, unlike third-party firms that subject their users to rely on often obscure and undisclosed methods. Thus, by deploying a smart contract through the Ethereum network (Goerli test Ethereum network) using blockchain technology, buyers and sellers may interact with the contract through actions such as making deposits to initiate a transaction, canceling transactions, confirming transactions, and sending receipts. Ultimately, we propose that smart contracts are a more viable option than third-party arbiters for a trustless marketplace exchange since our smart contract will securely hold funds until the item is received by the buyer and all

conditions of the smart contract are met. Using an escrow mechanism that requires buyers and sellers to make double deposits, which creates a trustless system since the extra deposit locked in the contract acts as collateral, giving the transactors a strong incentive to complete the transaction. For example, the seller will be incentivized to ship the item of desire in time, whereas the buyer will be incentivized to confirm the purchase because they both want to have their escrow deposit back.

In coordination with our smart contract, our application will include a user-friendly front end that will make it easy for buyers and sellers to complete transactions. The process of running our smart contract through Remix Ethereum is not a straightforward process, especially for those who do not have the technical expertise. This can be a serious roadblock in marketing our application to the public. Thus, we decided that it would be best to simplify the process while adding a level of abstraction, via our front-end web application. Furthermore, in the implementation of our front-end, we incorporated a database to store all relevant information such as buyers' shipping address (only visible to the seller) and the active contracts associated with the user.

The smart contract process combined with our user friendly front-end web application will provide users with all of the aforementioned benefits of blockchain technology in an efficient and convenient manner.

3 Overview

According to the Ethereum website, Ethereum is the "foundation for building apps and organizations in a decentralized, permissionless, censorship-resistant way." The EVM (Ethereum Virtual Machine) is a computer that the network works with, and anyone on the network may request computations from the computer. These computations are verified by nodes, or others on

the network and committed to the updated "state" of the Ethereum network. These EVM state updates come with a cost of Ether, the native cryptocurrency of Ethereum, some of which is rewarded to nodes that verify these updates to keep the network secure. All of this data is stored within blocks that are then stored on the blockchain, where a user can track all of their previous transactions. This makes the blockchain "traceable" and prevents issues such as "double-spending" where the same currency is used twice.

Smart contracts are one of the main benefits of the Ethereum blockchain. They are Ethereum accounts, just like Ethereum wallet addresses' that can send and receive transactions, but instead of being controlled by a user, "they are deployed to the network and run as programmed."[3] Simply put, smart contracts use a snippet of code to enforce the conditions that are previously defined on them. In the case of our project, smart contracts would run the transaction between the two parties who use our service. For example, if user A would like to sell an item to user B, they would note the transfer in the smart contract along with the transfer of Ether (the currency that our system will use). Upon completion of the condition in the smart contract, user A will receive the Ether and user B will receive the item that they desired from user A.

This is a basic summary of the smart contract, but when implemented, there will be other aspects as well such as an Ether amount working as collateral, to ensure both parties stay honest and do not cheat each other of the item or Ether. Upon completion of a smart contract, the transaction will be stored on the Ethereum blockchain and can always be verified by the involved parties. The whole process will be orchestrated through the front end of our decentralized application where the users begin by connecting to their Metamask account. After connecting, users can deploy and search for smart contracts which will begin the process of the decentralized

arrangement. From there, users fill the necessary fields and have access to the functions they need to complete their transaction. This data, and all other relevant data is stored in a database in order to facilitate the product searching process and other core functionalities of our app that require the active smart contract addresses associated with accounts.

4 Literature Review

Several decentralized crypto-currency-based trading systems similar to our model have been deployed. These platforms have been developed for the aforementioned advantages of removing the mediating third party from commodity exchanges, which eliminates any potential high transaction fees, restrictions, or censorship while also ensuring the anonymity and data security of the transactors.[5]

For example, in N. I. Schwartzbach's (2020) paper, it is indicated that smart contracts can work as the arbiter and only invoke if disputes arise.[8] Normally, the buyer will transfer the value of goods in cryptocurrency as escrow and notify the contract when receiving the goods. The money as escrow will then be transferred to the seller, which terminates the contract. However, if disputes occur during the transaction, one party will place money of arbitrary value as a wager. The other party will also place the same amount of money as a wager to counter this dispute. This paper develops an arbiter based on mathematical deduction. According to the paper, the arbiter will judge in favor of the honest party based on evidence. The winner of the dispute will gain back the cost of the goods plus the wager, while the loser gains nothing. However, errors in judgment will always happen because the judgment is based on probability, thus a smart contract working as an arbiter in this case isn't the best approach as the stakes for errors may be high. This differs from our smart contract protocol for a decentralized exchange because, in our protocol, both buyers and sellers will need to put twice the amount of the transaction as escrow,

which incentivizes the completion of the transaction and reduces the chance of arbitration. Moreover, if disputes happen, both parties are incentivized to resolve the dispute because only when the dispute is resolved can both parties get their escrow back.

Furthermore, Zimbeck's (2014) paper established a trustless escrow system that relies on the concept of two-party double deposits and the world's first decentralized smart contract protocol, BitHalo.[8] BitHalo was developed to address the issues of early smart contract protocols, which lacked a feasible mechanism for trustless transactions. Additionally**,** transactions on early smart contract protocols like the one in Schwartzbach's paper were often vulnerable to transaction malleability, where raw transaction data may be changed or exchanged before broadcasting, ultimately invalidating the transaction. Such vulnerabilities allowed for extortion and attacks such as double-spend attacks.[8] As a result, BitHalo introduced a two-party double deposit protocol that not only addressed these vulnerabilities mentioned above but also introduced a mechanism for trustless online marketplaces that rely on code rather than a mediating entity.

BitHalo's smart contract protocol for a decentralized exchange is similar to ours, since twice the amount of the transaction is held as escrow to incentivize honest behavior and the completion of the terms of the smart contract. According to BitHalo's design principles, if any of the parties break the commitment, or do not finish the transaction in a timely manner, the escrow will time out and destroy itself; thus, both parties lose money. This leads BitHalo to be considered the "mother of unbreakable contracts."

However, one fundamental difference is that our smart contract protocol will run on the Ethereum blockchain whereas BitHalo is software-based and uses Bitcoin as currency. However, the contract is not on the Bitcoin blockchain (i.e., a record of contracts not stored on

blockchain).[2] It is important to consider that BitHalo's off-blockchain decentralized smart contract protocol has the advantage of not bloating the blockchain and allowing for greater complexity contracts, whereas Ethereum smart contracts have a max size of 25 KB due to gas limitations.[3, 2] On the other hand, the benefits to our protocol via Ethereum smart contracts is that they are able to execute on blockchain and stored in a distributed manner along with cryptographic mechanisms for a greater degree of security in terms of being tamper-proof and immutable. Moreover, our protocol also has the benefit of running on the Ethereum blockchain, which is proof-of-stake, resulting in lower overhead and a significantly lower carbon footprint than other blockchains that often run on proof-of-work.

5 Infrastructure

5.1 High-Level Infrastructure

- *Remix-Ethereum*: Remix-Ethereum is where we write our smart contract code. From Remix-Ethereum, we are able to deploy our smart contract. Any changes in code must be updated here and redeployed.
- *MetaMask Wallet*: The buyer/seller must connect to their MetaMask wallet, in order to utilize the smart contract, as the currency (Ether) is contained in the MetaMask wallet. The MetaMask wallet is also used for its unique ID (public key) to recognize an account (unique identifier for security purposes such as verifying sufficient funds).
- *Goerli Test Networ*k: This is the network where our smart contracts run until our protocol is ready for production. It is a test Ethereum network that replicates the Ethereum Network (EVM - Ethereum Virtual Machine) and uses the test currency we are using, Goerli Test Ether. We use a test network and test Ethereum due to the expense of using

actual Ethereum for contract deployment and testing. Despite developing our Dapp using the Goerli Test Network, for the sake of our demo, we used the Polygon Mumbai Test Network because the Goerli Test Network was under heavy traffic at the time due to the upcoming Ethereum update.

- *Etherscan.io*: We now use etherscan.io as a dashboard to monitor active contracts. Previously, prior to the development of our website, this was also where we ran, verified, (ensuring the bytecode for contract execution is the same as when it was deployed), published, read from, wrote to, and monitored smart contracts. However, we can now do all the core functionalities through our website and leave Etherscan.io just for monitoring.
- Javascript: JavaScript helped us set up and connect our smart contract to our website, or front end, essentially working as the bridge for our decentralized application. In particular, the Node.js and Express.js packages were used for connecting with MetaMask, our database, and our smart contract code.
- Parse Server: Parse is a back end that we used in conjunction with Node.js to set up our database and store all relevant data. We are using a NoSQL database via Parse using MongoDB that contains entries for each user. It's used to record user information necessary to complete a transaction such as buyer wallet address, item contract address, quantity to purchase, etc. One thing to note is that there is no private information stored except for the buyer's shipping address which is only visible to respective sellers while the buyer has an active transaction.

5.2 Smart Contract Design Principles

In order to ensure that both parties are incentivized to complete the transaction, the buyer and the seller will each deposit twice the amount of ETH into the contract as escrow. The ETH payment will remain locked inside the contract until the buyer confirms receipt of the promised item. After the buyer confirms receiving the item, the buyer will be sent their deposit back minus the cost of the item and next the seller will be able to "claim payment", so that they will be sent their deposit back plus the profit from the transaction. Currently, each seller deploys one contract for one item onto the Goerli test Ethereum network and each buyer can only have one ongoing transaction associated with one item but have no limit on the quantity to buy except for stock limitations. There is also no limit on the buyers for each contract or on the stock quantity for each item. Once the seller deploys the contract and adds an item to the unique contract, the item and seller information is immutable. Since each item listing is immutable once contract is deployed smart contract is set forth (set forth via AddItem() function), our e-commerce platform is similar to NFT platforms which similarly use a smart contract for each NFT listed to ensure immutability. We will determine the success of our protocol by analyzing the time elapsed between the time transactions were initiated until the time they were resolved along with analyzing the distribution of successful transactions and cancellations. Thus, shorter times elapsed for transactions and higher rates of successful resolutions will be our metric for evaluating success. Furthermore, once our protocol goes into production, we intend to record transaction details in order to conduct analysis that will optimize our protocol. A potential approach to optimizing our protocol will be finding an optimal value for the escrow deposit amount. For instance, by setting the deposit amount as a factor that is more than the item value but no more than twice the amount of the item, depending on the transaction details and patterns

learned from our analysis, in order to achieve shorter contract completion times, more contract completions, and incentivize users to use our platform. For our current implementation our value factor for the escrow is 2x and will allow for multiple buyers as detailed in section 6.3.

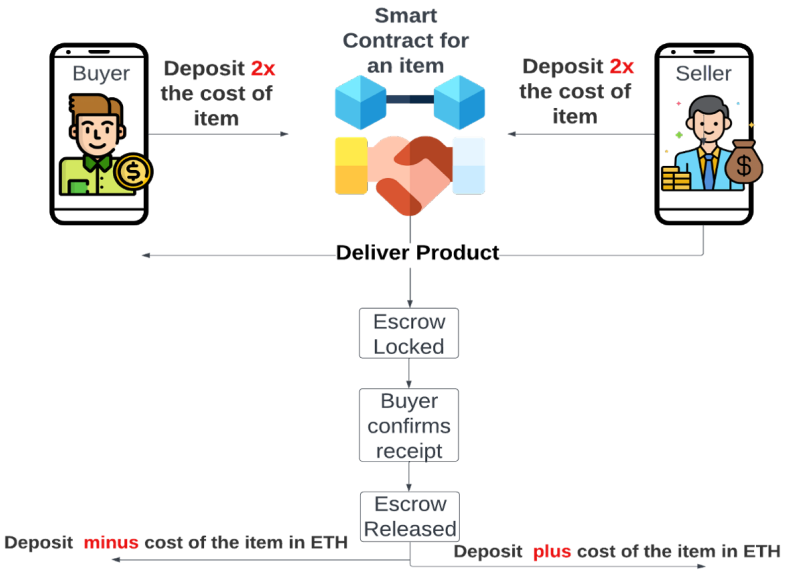

Figure 1. Diagram of Proposed Decentralized Escrow Smart Contract Protocol

5.3 Decentralized Web Application Design Principles

The front end of our application, known as the dApp or the decentralized application will provide an easy-to-use interface solution to access the functions and capabilities of our smart contract code. In the back end, a Parse database using MongoDB will record transaction information and users' contract addresses in order to facilitate our website functionalities and the transaction process.

A user can be either buyer or seller. As buyers, they can shop the marketplace and search for items of desire. At the same time, they can also deploy new contracts to list items for sale. Both buyers and sellers have dashboards where they access and act on their active and completed transactions. If they decide to deploy a new contract, they must provide all necessary

information, such as the total escrow amount, the quantity of the item, and the price of each individual item, as well as other relevant details. Following this, the seller will then need to connect to their MetaMask account wallet after which they will have access to all of their necessary functions to work with the contract as a seller.

The process for a buyer is relatively similar with some minor differences. The buyer will need to connect to their MetaMask account wallet and will then have access to all of their necessary functions to work with the contract as a buyer.

6 Methods

6.1 Setting the Contract in Motion

Our online escrow marketplace will be run through a smart contract via our front-end website, and in order to understand the process of executing our smart contract, it is crucial to distinguish between some terminology. Each party in our escrow system will have their own MetaMask wallet, which contains their Ether and their unique wallet address, which is necessary for identifying the selling party and the buying party. For the purposes of our sample, we have two metamask wallet accounts which each represent a party in our sample transaction. Henceforth, we will refer to the purchasing party's MetaMask account as the "buyer", or the "buyer's account" and the selling party's account as the "seller", or the "seller's account." The transaction must be commenced from the point of view of the seller. The seller deploys the smart contracts via our website, with the necessary Ether cost attached to it (twice the product amount for x items), to cover two times the transaction costs for escrow (not including the negligible gas fee). For example, if the seller sells ten items, the Ether cost attached will be twice the total amount of the ten items. The smart contract is then documented on Etherscan.io for the buyer

and seller to monitor. The seller must also set values for the item_price_in_ETH, item_quantity, item_name, item_details, and item_description which will be passed into respective read only functions that may be utilized by the buyer before confirming purchase.

6.2 Transaction process

Now that the seller has put the smart contract officially in motion, two times the total product price is held in the contract and a buyer must make the next step. However, if a suitable buyer is not found, the seller has the ability to terminate the contract and this will refund their Ether deposited in the contract. The transaction process is secured by the buyer states, which tells us which stage of the contract the buyer is in. The contract tracks the transaction process with three buyer states (Paid, Shipped, and Received) and the strict buyer state check before each step ensures the process runs smoothly and efficiently. Once buyers confirm their purchase and deposit two times the transaction amount into the smart contract, their state becomes “paid” in the system. With both parties double depositing, they will both bear the same risk if the contract is not completed, creating an equally strong incentive for both parties to complete the transaction. At this point, the buyer’s MetaMask account is connected to the smart contract and recognized as the purchaser, by confirming the purchase. Now, the escrows for both buyer and seller are locked in the contract, as an agreement is being initiated between the two acknowledged parties. Next, the seller ships the desired item to the buyer and the buyer state changes from “Paid” to “Shipped.” The Ether will be locked into the contract until the buyer confirms receiving the promised item agreed upon. Thus, upon receipt of the purchased item, the buyer, who is in the “Shipped” state, will confirm that they have received the purchase, at which point the buyer will be refunded half of their input into the smart contract (1x the price of the

product). Following this, the buyer state becomes “Received” and gets removed from the buyer hashmap and indicated by our parse database as “completed” buyer. This means the buyer no longer exists in the smart contract. Finally, the seller will refund their stored Ether for the transaction, to receive their whole input, as well as the revenue generated from the transaction (total of 3x the price of the product to account for the seller’s profit and their original deposit).

6.3 Implementation and Contract Execution Model

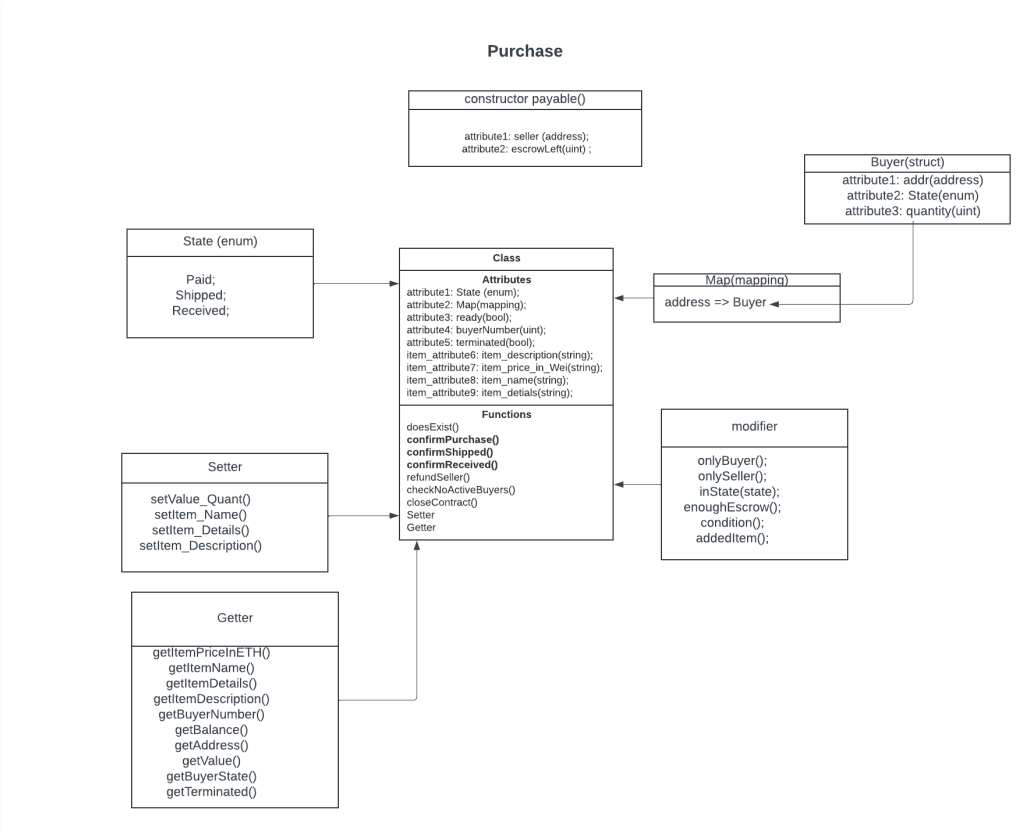

Figure 3. Class diagram of our implementation for an escrow smart contract (purchase.sol)

The purchase contract contains one class including attributes such as State, Map, maxBuyerNumber, buyerNumber, terminated (boolean), ready (boolean) and two constructor attributes (seller address and escrowLeft). The state is one of the most crucial attributes as it has three different potential values (Paid, Shipped, and Received) which will determine the operations that the contract can conduct at a given time. The variable Map is a hashmap that stores the current buyers . A constructor helps store the current escrow left using an attribute named escrowLeft, which will be in one of the modifiers to check if the escrows are sufficient enough to accept a new buyer. The sellers have to initialize the item value and item quantity and they will be able to add item attributes after the deployment of the contract through setter functions. It is important to note that there exists a condition to check that before a buyer wants to register, at least one item has been added, meaning that the seller has to process the setter function at least once after deployment. This would be before any buyers interact with the previously mentioned deployed contract. Besides all getter and setter functions, the class has 8 functions, some of which may run strictly based on the modifier check and the state check. Detailed descriptions of the functions, along with their respective logic is presented in the flow diagram in Figure 4 (see next page).

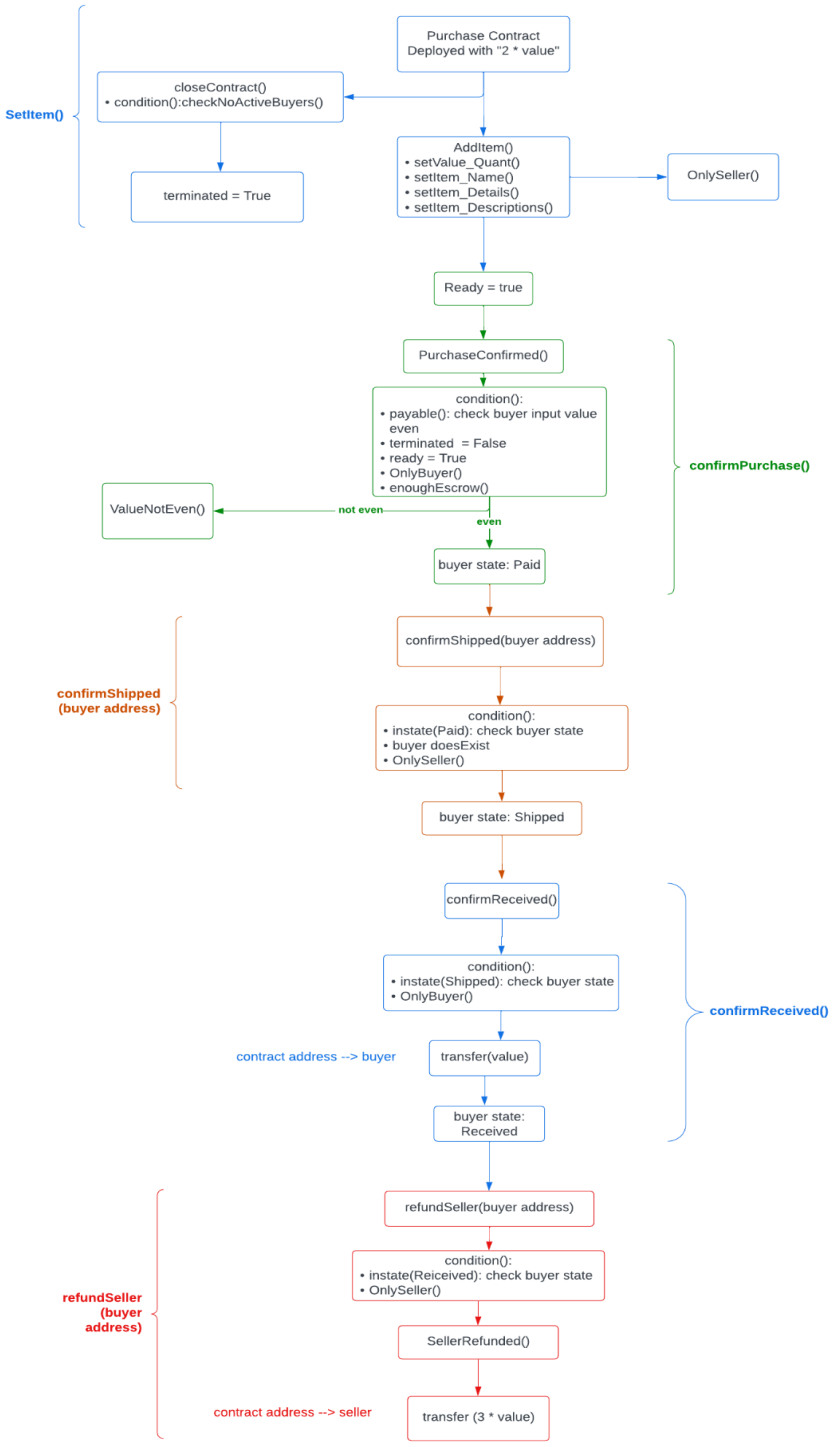

Figure 4. This diagram depicts the low-level mechanism behind each function in our smart contract. The high-level idea of each function is explained in section 6.1 and 6.2 above.

7 Results

Our app can be found online, at blockbazaar.net and we have provided a walkthrough video demonstration of its functionalities and the transaction process.

8 Conclusion

Our platform decentralizes peer-to-peer e-commerce using blockchain and smart contracts for increased security, transparency, and cost-effectiveness; allowing for a trustless marketplace. The combination of our easy to navigate front end combined with our smart contract code makes it a simple process for any individual to safely and securely process transactions without third parties and their overbearing costs and safety concerns. However, the true limitless potential of decentralization lies deeper in the idea that we can safely eliminate the need for third-party mediation in not only centralized marketplaces but also in other industries such as finance, insurance, real estate, supply chain, gaming, and more. Thus, decentralization promises to be the future in all the aforementioned industries as it quells privacy fears and eliminates unnecessary costs, while putting the decision completely in the hands of the two opposing parties, rather than relying on a third party for anything.

10 Limitations and Future Work

1) Our platform currently uses the Ethereum blockchain network for all transactions. However, to provide a more convenient user experience, we are storing some information such as public wallet address, product info, and shipping info (for physical goods) in our own database. In the future, we plan to take our decentralization efforts a step further by

migrating all stored information to a decentralized file system, such as the InterPlanetary File System (IPFS). By doing so, we can ensure that files, such as images, cannot be modified once they are stored on a fully decentralized system. This stands in contrast to the content that we currently store in our own database, which is editable by both backend developers and external hackers. We understand that certain user data, such as shipping info, needs to be kept private and secure. To achieve this, we plan to encrypt sensitive data before storing it on IPFS. Specifically, we will apply a specific hash function to the data and store the encrypted result on IPFS. Only the seller of a particular transaction will have the key to the hash function, ensuring that the shipping address is only visible to them, just like it is now.

By transitioning all data storage to a decentralized file system, we aim to transform our platform from a partially web2-based application to a fully decentralized web3-based app. Although transactions are already decentralized, this move will make our app even more secure and trustworthy.

2) Currently, we have implemented a double deposit escrow policy to ensure the honesty and validity of our transactions Currently online marketplaces largely favor the interests of buyers rather than sellers on their platform. This is because sellers have to give companies a cut of their profits, while buyers bare little to no risk in placing orders due to customer services that often provide safeguards such as refunds that eliminate buyers' concerns when shopping on the online realm. With our double deposit system, buyers and sellers would have to potentially deposit very significant amounts of Ether which they

may prefer to spend/save/invest otherwise or may not afford; Thus, going forward, we must find a compromising solution to attract users, by making our platform feasible and comfortable to buyers. Ultimately, we fear that some consumers may be unwilling to use our service when they can find centralized alternatives that let consumers hold their money themselves. At the least, we plan to minimize this deposit which will be done through statistical testing and machine learning models once our service is running to find the optimal deposit rate to ensure that trust is maximized while the collateral deposit stays low enough for buyers to still be comfortable and incentivized to use our platform.

3) Due to the decentralized and non-profit nature of our service, we cannot afford any unforeseen circumstances in regards to transactions. Thus, to avoid losing items in transit, we would consider partnering with a transportation service and provide tracking to our consumers to ensure that nothing gets lost in transit. This is crucial because if an item is lost, the smart contract would not be complete and would need to be voided, and the seller's product would not be recovered. Therefore, it is a priority for us to ensure that this scenario is avoided.

4) We would also like to include a section for product reviews, to resemble other e-commerce websites. We plan to implement a decentralized review system into our platform to provide more reliable and transparent product reviews and ratings. This is significant, because current e-commerce platforms have vulnerabilities in their review systems, since they are not decentralized, allowing for reviews and ratings to easily be dishonestly manipulated. Ultimately, using blockchain and smart contract technology, we

can ensure the credibility of reviews, which in turn will make our platform more attractive to buyers since our products will be more reputable and transparent.[9]

5) Finally, despite the promising signs from our project, there are still edge cases that we still need to cover due time limitations. For example, one buyer can only have one ongoing transaction associated with one item. In the future, we need to enable multiple ongoing transactions of the same item for a particular buyer. Also, we need to implement a refund system where the buyer can return the item he doesn't like and both parties can claim their escrows.

References

1. Ahsan, J. (2019, March 21). *Centralized vs. decentralized: The best (and worst) of both worlds*. LinkedIn. Retrieved October 30, 2022, from https://www.linkedin.com/pulse/centralized-vs-decentralized-best-worst-both-worlds-jiyad-ahsan
2. Bigi, G., Bracciali, A., Meacci, G., & Tuosto, E. (1970, January 1). *Validation of decentralised smart contracts through game theory and formal methods: Semantic scholar*. Retrieved October 30, 2022, from https://www.semanticscholar.org/paper/Validation-of-Decentralised-Smart-Contracts-Through-Bigi-Bracciali/2722e4c3bbabaaebd282b298224bf24d56a6d67f
3. *Introduction to smart contracts*. ethereum.org. (2022, September 1). Retrieved October 30, 2022, from https://ethereum.org/en/developers/docs/smart-contracts/

4. Moreland, K. (2022, October 25). *What are ddos attacks?* Ledger. Retrieved October 30, 2022, from https://www.ledger.com/academy/what-is-a-ddos-attack.
5. Prasad, R. V., Dantu, R., Paul, A., Mears, P., & Morozov, K. (2018). *A decentralized marketplace application on the ethereum blockchain*. IEEE Xplore. Retrieved October 30, 2022, from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8537821
6. Dunne, Chris (2023). "What It Costs to Sell on Amazon in 2023." *The Fastest Amazon Repricer | Repricer.com*, 10 Mar. 2023, https://www.repricer.com/blog/amazon-seller-fees/.
7. Schwartzbach, N. I. (2020, August 24). *An incentive-compatible smart contract for Decentralized Commerce*. arXiv.org. Retrieved December 2, 2022, from https://arxiv.org/abs/2008.10326
8. Zimbeck, D. (2014). *Two party double deposit trustless escrow in cryptographic networks and ...* Retrieved October 31, 2022, from https://cryptochainuni.com/wp-content/uploads/BitHalo-whitepaper-two-party-double-deposit-trustless-escrow-in-cryptographic-networks-bitcoin.pdf
9. A. Avyukt, G. Ramachandran and B. Krishnamachari, "A Decentralized Review System for Data Marketplaces," 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Sydney, Australia, 2021, pp. 1-9, doi: 10.1109/ICBC51069.2021.9461149.