

DEERWALK INSTITUTE OF TECHNOLOGY

Tribhuvan University

Institute of Science and Technology



E-AUCTION USING BLOCKCHAIN

A FINAL PROJECT REPORT

Submitted to

Department of Computer Science and Information Technology

DWIT College

*In partial fulfillment of the requirements for the Bachelor's Degree in Computer
Science and Information Technology*

Submitted by

Sulav Baskota

5-2-1175-42-2018

17th April, 2023

DWIT College
DEERWALK INSTITUTE OF TECHNOLOGY

SUPERVISOR’S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by SULAV BASKOTA entitled “**E-AUCTION USING BLOCKCHAIN**” in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for the evaluation.

.....

Ms. Shristi Awale

Program Head - Sophomore Year

DWIT College

DWIT College
DEERWALK INSTITUTE OF TECHNOLOGY

STUDENT'S DECLARATION

I hereby declare that I am the only author of this work and that no sources other than that listed here have been used in this work.

.....

Sulav Baskota

17th April, 2023

DWIT College
DEERWALK INSTITUTE OF TECHNOLOGY

LETTER OF APPROVAL

This is to certify that this project prepared by SULAV BASKOTA entitled “**E-AUCTION USING BLOCKCHAIN**” in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied.

In our opinion it is satisfactory in the scope and quality as a project for the required degree.

<p>.....</p> <p>Ms. Shristi Awale Project Supervisor Program Head - Sophomore Year DWIT College</p>	<p>.....</p> <p>Mr. Nawaraj Poudel Associate Professor Central Department of Computer Science and Information Technology, Kirtipur</p>
<p>.....</p> <p>Mr. Niresh Dhakal Project Coordinator Associate Director of Undergrad Program DWIT College</p>	<p>.....</p> <p>Mr. Hitesh Karki Campus Chief DWIT College</p>

ACKNOWLEDGEMENT

I would like to express my sincere thanks to Ms. Shristi Awale, for her invaluable guidance and support in completing my project on "E-Auction using Blockchain". Her expertise, dedication, and motivation have deeply inspired me throughout the project. She provided me with the necessary tools and methodologies to carry out the research and present it as accurately and clearly as possible. It was an honor and privilege to work and learn under her guidance. I am extremely grateful for everything she has offered me.

Besides my advisor, I would like to express my gratitude to the rest of my project supervisor committee: Mr. Hitesh Karki, Mr. Ritu Raj Lamsal, Mr. Bijay Babu Regmi, Mr. Niresh Dhakal, Mr. Saurav Gautam, Mr. Kumar Lamichhane, and Mrs. Sarada Pohkrel, for their insightful comments, encouragement, and challenging questions. Without their support and suggestions, this project would not have been completed successfully.

I would also like to thank our institution, Deerwalk Institute of Technology, for providing me with this amazing opportunity to work on this project. It has been a great learning experience and a significant milestone in my academic journey.

Sulav Baskota

TU Reg. No: 5-2-1175-42-2018

17th April, 2023

ABSTRACT

This paper describes a blockchain-based blind auction system built using Solidity and Ethereum. The system incorporates a sealed-bid mechanism, ensuring anonymity, and a smart contract for transparency and security. The project report discusses the design and development process, including the smart contract functionalities, challenges faced, and solutions implemented. The proposed system addresses issues of transparency and fairness in online auctions, and aims to ensure equal chances of winning, prevent collusion, simplify the bidding process, and increase efficiency. Blockchain technology provides a decentralized, secure, and transparent method for conducting auctions, eliminating the need for intermediaries. The system proposed in this paper has the potential to revolutionize the online auction industry, providing benefits to both buyers and sellers. Several recent research papers have explored the implementation of blockchain technology in the online auction process, with proposed schemes including smart contracts and zero-knowledge proof.

Keywords: *Blockchain; Cryptography; Decentralized-Application; E-Auction; Ethereum; Hashing; Keccak-256; Sealed-Bid; Smart-Contracts*

TABLE OF CONTENTS

SUPERVISOR’S RECOMMENDATION	ii
STUDENT’S DECLARATION	iii
LETTER OF APPROVAL	iv
ACKNOWLEDGEMENT	v
ABSTRACT.....	vi
TABLE OF CONTENTS.....	vii
LIST OF FIGURES	x
LIST OF TABLES	xi
LIST OF ABBREVIATIONS	xii
CHAPTER 1: INTRODUCTION	1
1.1. Introduction.....	1
1.2. Problem Statement	2
1.3. Objectives	2
1.4. Scope and Limitation	3
1.5. Development Methodology	3
1.6. Report Organization.....	4
CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW	6
2.1. Background Study.....	6
2.2. Literature Review.....	8
2.3. Current System.....	9
2.4. The Problem with Current System.....	10
CHAPTER 3: SYSTEM ANALYSIS.....	11

3.1. Requirement Analysis	11
3.1.1. Functional Requirement	11
3.1.2. Non-functional Requirement	12
3.2. Feasibility Analysis.....	13
3.2.1. Technical Feasibility	13
3.2.2. Operational Feasibility	13
3.2.3. Economic Feasibility	14
3.2.4. Schedule Feasibility	14
3.3. Analysis.....	15
3.3.1. Class Diagram	15
3.3.2. Sequence Diagram	16
3.3.3. Activity Diagram	17
CHAPTER 4: SYSTEM DESIGN.....	19
4.1. Design	19
4.1.1. Component Diagram.....	19
4.1.2. Deployment Diagram.....	20
4.2. Algorithm Details.....	21
4.2.1. Keccak-256	21
4.2.2. Suitability of Algorithm.....	26
CHAPTER 5: IMPLEMENTATION AND TESTING	28
5.1. Implementation	28
5.1.1. Tools Used	29
5.1.2. Implementation Details of Modules.....	30
5.2. Testing.....	35

5.2.1. Test Cases for Unit Testing.....	35
5.2.2. Test Cases for System Testing.....	37
5.3. Result Analysis	38
CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATIONS.....	39
6.1. Conclusion	39
6.2. Future Recommendations	39
CHAPTER 7: REFERENCES	41
APPENDIX I	43
APPENDIX II	46

LIST OF FIGURES

Figure 1: Incremental Development Model	4
Figure 2: Use Case Diagram of the System	12
Figure 3: Project Gantt Chart.....	15
Figure 4: Class Diagram of System Smart Contracts	15
Figure 5: Sequence Diagram of the System.....	16
Figure 6: Activity Diagram of the System	18
Figure 7: Component Diagram of the System	19
Figure 8: Deployment Diagram of the System	20
Figure 9: The Theta Step.....	22
Figure 10: The Rho Step	22
Figure 11: The Pi Step	23
Figure 12: The Chi Step.....	23

LIST OF TABLES

Table 1: Project Task Duration	14
Table 2: The Round Constants RC[i].....	26
Table 3: The Rotation Offsets r[x,y]	26
Table 4: Admin Smart Contract Functions	30
Table 5: BlindAuctionFactory Smart Contract Functions	31
Table 6: Test Cases for Unit Testing of the Application	35
Table 7: Test Cases for System Testing of the Application.....	37

LIST OF ABBREVIATIONS

ABI	Application Binary Interface
CID	Content Identifier
DApp	Decentralized Application
ER	Entity-Relationship
EVM	Ethereum Virtual Machine
FTC	Federal Trade Commission
ID	Identification
IPFS	InterPlanetary File System
JS	JavaScript
JSON	JavaScript Object Notation
NIST	National Institute of Science and Technology
PDF	Portable Document Format
RPC	Remote Procedure Call
SHA	Secure Hash Algorithm
UI	User Interface
UML	Unified Modelling Language
URL	Uniform Resource Locator
USA	United States of America
VS	Visual Studio
WBES	World Business Environment Survey
XOR	Exclusive OR

CHAPTER 1: INTRODUCTION

1.1. Introduction

The emergence of blockchain technology has revolutionized the way in which online transactions take place. Its decentralized and immutable nature provides a secure and transparent platform for conducting various transactions. One of the most promising applications of blockchain technology is in the field of online auctions.

Depending on whether the auction price is open or closed, traditional auctions are split into sealed-bid auctions and open-bid auctions in various markets. In traditional auctions, there is a lack of transparency, and the bidders may face issues of collusion, shill bidding, and bid sniping. However, with the use of blockchain technology, these issues can be eliminated, and online auctions can be made more secure and transparent.

In this project, we have developed a blockchain-based online auction platform using Solidity and Ethereum. The auction platform is designed to ensure a fair and secure auction environment for bidders. It incorporates a sealed-bid mechanism, which ensures the anonymity of the bidders, and a smart contract, which ensures the security and transparency of the auction process. The platform also ensures that the highest bidder wins the auction, and the auction rules are enforced automatically, eliminating the need for a middleman.

The project report presents an in-depth analysis of the blockchain-based online auction platform. The report outlines the problem statement and provides a detailed explanation of the auction platform's design and development process. The report also highlights the challenges faced during the development process and the solutions that were implemented to overcome them. Finally, we have evaluated the platform's performance and analyzed its effectiveness in providing a secure and transparent auction environment.

Overall, the blockchain-based online auction platform developed in this project has the potential to revolutionize the online auction industry. It ensures a fair and secure auction environment, eliminates the need for intermediaries, and provides transparency to the auction process. The platform can be used in various industries, including e-commerce, real estate, art, and charity, to name a few.

1.2. Problem Statement

Online auctions have gained immense popularity in recent years, providing buyers and sellers with a convenient and efficient way of transacting. However, the current online auction systems suffer from issues of transparency and fairness. According to the World Business Environment Survey (WBES) conducted by the World Bank, about 60% of companies have admitted to bribery and manipulation of the auction results [1]. In Addition, in 2019 the Federal Trade Commission (FTC) of USA reported that internet auction fraud accounted for a total loss of \$9.4 million in USA [2]. This indicates that the current auction systems are vulnerable to fraudulent activities, and the bidders participating in such auctions may not get a fair chance to win the auction.

In addition, buyers participating in an online auction based on an English auction scheme suffer from a phenomenon known as "The winner's curse". This represents a scenario where a winner overpaid to win the auction. In traditional auctions, the highest bidder wins the auction, and there is a lack of transparency in the auction process. As a result, bidders may bid beyond the actual value of the item, leading to the winner's curse.

To address these issues, there is a need for a secure and transparent online auction system that can provide a fair and efficient trading environment for buyers and sellers. The use of blockchain technology can provide the necessary security and transparency in the auction process, ensuring that the bidders get a fair chance to win the auction. In this project, we aim to develop a blockchain-based blind auction system that provides an anonymous bidding mechanism and eliminates the issues of bribery, manipulation, and the winner's curse.

1.3. Objectives

The proposed blockchain-based blind auction system aims to overcome the challenges faced by traditional auction systems by ensuring transparency, fairness, and security. The objectives of the proposed system are as follows:

1. To ensure that all bidders have an equal chance of winning the auction.
2. To maintain the anonymity of bidders and protect their information.
3. To prevent collusion between bidders.

4. To simplify the bidding process and make it easier for bidders to participate.
5. To increase the efficiency of the auction process and reduce the time and cost required to conduct the auction.

1.4. Scope and Limitation

The proposed system provides a secure and transparent platform for conducting auctions. The use of blockchain technology ensures that the auction is conducted in a decentralized manner, eliminating the need for intermediaries. This system can be used to auction a variety of products or services such as antique items, jewelry, art pieces, rare collectibles, automobiles, etc. By providing a transparent and fair environment for bidding, this system can significantly reduce the risk of fraud and manipulation, as all transactions are recorded on the blockchain and can be easily audited, and thus has the potential to replace the current auction system.

Despite its potential benefits, a blockchain based blind auction system has some limitations. One of the biggest limitations is the immaturity of the technology. Blockchain technology is still in its early stages of development, and there are still many challenges that need to be addressed before it can be widely adopted.

Another limitation is the long time required for transaction confirmation. Each transaction on the blockchain must be confirmed by multiple nodes, which can take several minutes or even hours in some cases. This delay can be a significant obstacle for time-sensitive auctions.

1.5. Development Methodology

The development process of the proposed system followed the Incremental Development model. According to this approach, the contract development was carried out in increments, and after each increment, it was tested on a local blockchain node to ensure that it was working as expected. Once all the contract functions were developed and tested, a basic UI was created and gradually improved and connected to the contract functions. Finally, the entire UI was redeveloped for a cleaner and more optimized look and feel. This development process provided an opportunity for early testing and feedback, which helped to identify and fix any issues more efficiently.

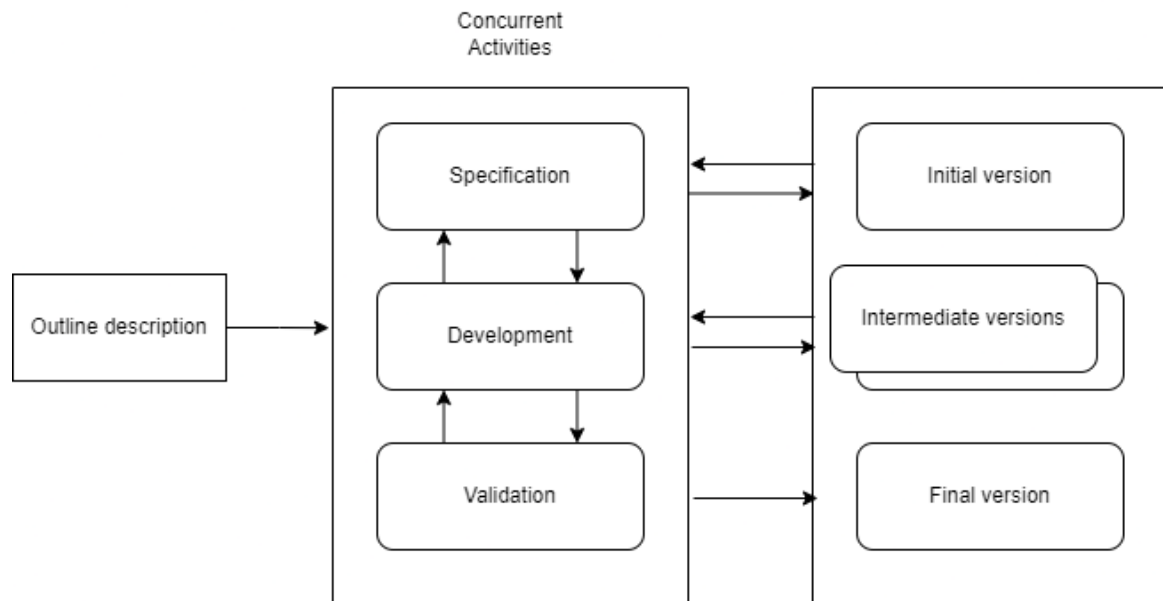


Figure 1: Incremental Development Model

1.6. Report Organization

This project analyses the problem with existing online auction platform and develops a decentralized system to solve some issues with the existing system. This paper includes the following sections:

Preliminary Section: This section consists of the title page, abstract, table of contents, list of figures, and list of tables.

Introduction: In this section, the overview of the project, problem statement, its objectives, its scope and limitations, and the development methodology used are discussed.

Background Study and Literature Review: This section discusses about the findings in other paper related to this system and discusses in brief the current system and its problem.

System Analysis: This section talks about the functional and non-functional requirements of this project. It also includes diagrams that help to elaborate on the overall design of the system proposed in this project.

System Design: This section deals with the actual designs of the system. On the basis of the designs provided here, the implementation stage can proceed further.

Implementation and Testing: This section is where the actual coding part of the system is described. It also describes some limitation or problems that may have arisen during coding and how it was mitigated.

Conclusion and Future Recommendation: It summarizes the concrete point of the report and provides an insight at the key points of the document. This section also addresses the limitations of the entire system and suggests some workaround or recommendations that could be followed to construct an even efficient system.

References: This section is the end-note of the document and it highlights where certain sources of relevant data and related information were extracted.

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1. Background Study

Blockchain technology has emerged as a decentralized solution to many centralized systems, including auctions. In a blockchain network, each transaction is recorded as a block of data, which is verified and added to the chain of previous blocks in a tamper-evident way. Each block contains a unique cryptographic hash that links it to the previous block in the chain, creating a digital record of all transactions that have taken place on the network.

When a new transaction is initiated, it is broadcast to the network of nodes, which use a consensus mechanism to validate and confirm the transaction. This validation process ensures that the transaction is legitimate, and that the sender has the necessary funds or assets to execute the transaction.

Once the transaction is validated, it is added to a new block of data, which is then broadcast to the network of nodes for further validation and confirmation. Once the block has been verified and added to the chain, it cannot be altered or deleted without consensus from the network.

The transparency and accountability of the blockchain come from the fact that the entire history of all transactions is stored in the ledger and can be viewed and audited by anyone on the network. This transparency makes it difficult for any single entity to manipulate or alter the data, as any changes to the ledger would be visible to everyone on the network.

Auctions are a popular mechanism for buying and selling goods and services, where buyers bid against each other for the item being sold. The different types of auctions are:

- English Auction: In this type of auction, the auctioneer starts with a low price and gradually increases the price until there is only one bidder left, who wins the item.
- Dutch Auction: In a Dutch auction, the auctioneer starts with a high price and gradually lowers the price until a bidder accepts the price, and wins the item.

- **Sealed Bid Auction:** In this type of auction, each bidder submits a sealed bid, and the highest bidder wins the item.
- **Reverse Auction:** In a reverse auction, the seller starts with a high price, and the bidders compete to offer the lowest price.
- **Blind Auction:** In a blind auction, the bidders do not know what the other bidders are offering, and the highest bidder wins the item.

In a blind auction, each bidder submits a sealed bid without knowing the value of the other bids. After all the bids have been submitted, the auctioneer opens the bids and awards the item to the highest bidder. The steps involved in conducting a blind auction is:

1. The auctioneer announces the terms and conditions of the auction, including the item being auctioned, the bidding rules, and the deadline for submitting bids.
2. Each bidder submits a sealed bid with their offer for the item.
3. Once all the bids have been received, the auctioneer opens the bids and determines the highest bidder.
4. The auctioneer notifies the highest bidder and awards the item to them.

Blind auctions are a popular mechanism for conducting fair and transparent bidding processes, as they eliminate any bias or collusion among bidders, and ensure that each bidder has an equal chance of winning the item being auctioned.

Traditionally, online auctions are conducted through web applications that follow a centralized architecture where the auctioneer controls the proceedings. In such systems, the auctioneer has the power to manipulate the auction in their favor, and participants have no way to verify the fairness of the auction process.

A decentralized blockchain based online auction system on the other hand provides a transparent and secure method for conducting auctions that eliminates the need for a trusted third party, where the bids are recorded on a blockchain, making it impossible to alter or delete the data.

Thus, a blockchain-based blind sealed auction system should ensure the transparency and fairness of auction proceedings, return the bids to unsuccessful bidders, transfer the winning bid to the seller automatically, use a decentralized architecture, and ensure the

anonymity of the participants to increase the trust in the system and encourage more people to participate in the auction. All of these requirements are fulfilled by a decentralized application allowing for the auction process to be conducted in a decentralized, secure, and transparent way, providing benefits to both the buyers and sellers.

2.2. Literature Review

Blockchain technology has the potential to revolutionize the online auction process by providing secure, transparent, and decentralized systems. This literature review focuses on several recent research papers that have explored the implementation of blockchain technology in the online auction process.

The paper titled, "A Blockchain-Based Sealed-Bid e-Auction Scheme with Smart Contract and Zero-Knowledge Proof" proposed a blockchain-based sealed-bid e-auction scheme with smart contract and zero-knowledge proof. The system utilizes a smart contract to automate the auction process, and a zero-knowledge proof to verify the validity of the bids without revealing their content [3]. The system also allows the auctioneer to set a reserve price, which must be met before the auction can be closed.

Another paper, "A Novel Auction Blockchain System with Price Recommendation and Trusted Execution Environment", proposes a blockchain-based auction system that aims to provide a more efficient and secure auction process by utilizing a blockchain-based distributed ledger and a trusted execution environment [1]. The system also includes a price recommendation engine that provides bidders with an estimate of the fair price for the item being auctioned.

In the paper titled "Blockchain based Smart Contract for Bidding System," the authors conduct an in-depth examination of a bidding system that utilizes blockchain technology. They explore the advantages and difficulties associated with integrating blockchain into the auction process [4]. The authors contend that the implementation of blockchain in bidding can eliminate intermediaries, decrease fraudulent activity, and facilitate a trustworthy and transparent bidding process.

In the article titled "Effective Consensus-Based Distributed Auction Scheme for Secure Data Sharing in the Internet of Things," the authors suggest a distributed auction scheme

that utilizes consensus to enhance security while sharing data in the Internet of Things [5]. The paper introduces a decentralized data trading model that utilizes a reverse auction to tackle the issue of trust without relying on a centralized auctioneer. The bidders reach consensus on the outcome of the auction to ensure confidentiality and circumvent the intrinsic performance limitations of blockchain.

The paper "Implementing Decentralized Auctions Using Blockchain Smart Contracts" proposes a general framework for decentralized auctions leveraging Ethereum smart contracts to trace and track bids, decentralized storage systems to upload documents related to bidding and trusted timer oracles that act as gateway between smart contract and external data feeds [6].

In conclusion, these research papers suggest that blockchain technology can significantly improve the auction process by providing transparency, security, and efficiency. The use of smart contracts, consensus mechanisms, and trusted execution environments can enhance the fairness and privacy of the auction process.

2.3. Current System

The current system of online auctions involves using a website or application to facilitate the buying and selling of goods or services through bidding. The seller lists their item with a starting price and interested buyers bid on the item until the auction ends, usually with the highest bidder winning the item. Payments are then made online through secure payment gateways, and the item is shipped to the buyer.

In the context of Nepal, the implementation of online auctions is still limited. Only partial implementations of online auction systems have been developed. One example of a partial online auction platform used in Nepal is online-auction.state.gov, which is used by the U.S. embassy to auction old goods [7]. Another partial online auction platform in Nepal is bolpatranepal.com [8]. This platform aggregates auctions related to tender published on newspapers and makes them available for the general public.

2.4. The Problem with Current System

The current system of online auctions faces several issues worldwide, including a lack of transparency and security, potential for bid manipulation, and the need for intermediaries, among others. In Nepal, the situation is even more limited, with only partial implementation of online auction systems that either provide limited services with no online payment options or only aggregating tenders information.

A blockchain-based decentralized blind auction platform can address these issues by providing a transparent and secure environment for conducting auctions. The use of a smart contract ensures the rules of the auction are enforced automatically, making the process more efficient and transparent. The sealed-bid mechanism ensures anonymity for the bidders, reducing the risk of bid manipulation. In Nepal, such a platform can also overcome the limitations of traditional auction methods and allow for a wider range of goods and services to be sold via an online auction.

CHAPTER 3: SYSTEM ANALYSIS

3.1. Requirement Analysis

The following features were added to the system after a careful analysis of existing papers and systems to identify key features and functionalities that are necessary for a successful auction platform:

- Sellers shall create an auction with valid auction duration and minimum bid amount.
- Only admins can evaluate an auction for verification.
- Bidders shall be allowed to bid on verified ongoing auctions.
- Automatic payout to seller is triggered when the auction ends with a valid highest bid and auction is closed.
- Only bidders who placed a bid can make a withdraw request.

3.1.1. Functional Requirement

The functional requirements of the project are:

- Sellers shall be able to add items to be auctioned.
- Item shall be verified by an auction admin.
- Sellers shall be able cancel an auction.
- Buyers shall be able to place bid on an auctioned item.
- Bidders shall be able to withdraw their bids after reveal period.
- Sellers and bidders shall be able to view the auction result.
- Losing bidders shall be refunded their bidding amount.
- Seller shall be transferred the payable amount.
- Auction manager shall be able to add and remove auction admins.

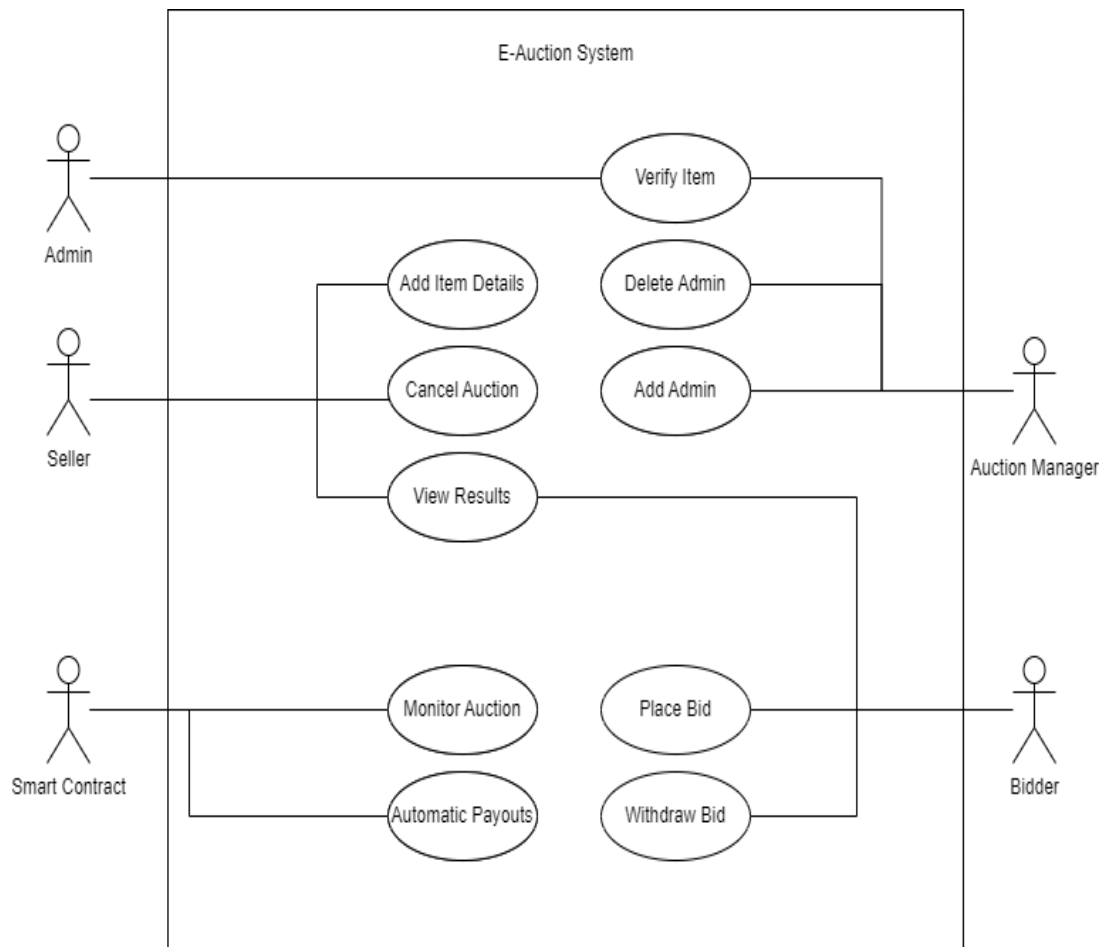


Figure 2: Use Case Diagram of the System

3.1.2. Non-functional Requirement

The non-functional requirements of the system are listed below:

- The system must be highly secure and protect the anonymity of bidders during the auction process.
- The system must be able to handle a large number of bids and transactions without compromising performance.
- The system must be easy to use, even for users who may not be familiar with blockchain technology.
- The system must be reliable and available at all times to ensure that bidders can participate in auctions as desired.
- The user's system must have stable Internet connection.

3.2. Feasibility Analysis

The project's overall feasibility was assessed in detail after completing the requirement elicitation and analysis process to help to evaluate the viability of the system.

3.2.1. Technical Feasibility

The system proposed in this paper is a DApp based on the Ethereum blockchain and Ethereum Smart Contract. Ethereum Smart Contracts, written in Solidity, were used to automate the auction process and store information on the blockchain. Development tools such as Hardhat and Ganache that allow developers to test and deploy the smart contracts in a local blockchain network were used to ensure that the contracts functioned as intended. NextJS, a React-based framework was used to build the user interface and handle user interactions, along with React-Moralis, a JavaScript library was used to interact with the local blockchain and the smart contracts deployed on it. Similarly, Metamask browser extension, a wallet provider was used to provide users with a user-friendly interface for sending and receiving cryptocurrencies, as well as signing transactions and interacting with the deployed smart contracts. All the tools used to build the system are open source and thus the project is technically feasible.

3.2.2. Operational Feasibility

The proposed system is implemented as a web-based application and thus is accessible to users through a web browser. User can access the services provided by the system such as creating an auction or placing a bid on an auction item after connecting their wallet to the application.

Creating an auction requires the user to fill out a form and pay for the transaction using their wallet. Users can place bid on an item by providing three inputs: deposit amount, bid amount and secret pass phrase, during the auction period. To qualify their bid, the users have to reveal their bid amount along with their pass phrase during the reveal period. After the reveal period, users can withdraw their deposit or withdrawable amount by clicking on the withdraw button. Thus, the system allows users to use its services in a manner similar to normal web-based application and as such is operationally feasible.

3.2.3. Economic Feasibility

The development of the project was done entirely using open source and freely available software and resources making it economically feasible. Smart Contracts were deployed on a local blockchain in the local machine using Hardhat and Ganache, and test ethers were used to enable transactions for testing. The project also used freely available Metamask browser extension as a wallet provider. Thus, economic feasibility was guaranteed as no direct costs were necessary for successfully completing and constructing the project.

3.2.4. Schedule Feasibility

The schedule feasibility analysis is carried out by evaluating the Gantt chart for this project. The Gantt chart in Figure 3 and data in Table 1 provides a clear and detailed overview of the project timeline, including all the tasks required to complete the project and their dependencies. The estimated durations for each task appear to be reasonable and there is sufficient time allocated for each task to be completed without undue pressure. Overall, the Gantt chart demonstrates that this project is well-planned, structured and should be feasible within the desired timeframe.

Table 1: Project Task Duration

Activity	Plan Start	Plan Duration	Actual Start	Actual Duration
Research	1	2	1	3
Smart Contract Development	4	1	4	2
Smart Contract Deployment	6	1	7	1
UI Design	8	3	8	4
Web3 Integration	12	2	11	3
System Testing	14	2	14	2
Documentation	10	3	12	1

E-Auction Using Blockchain

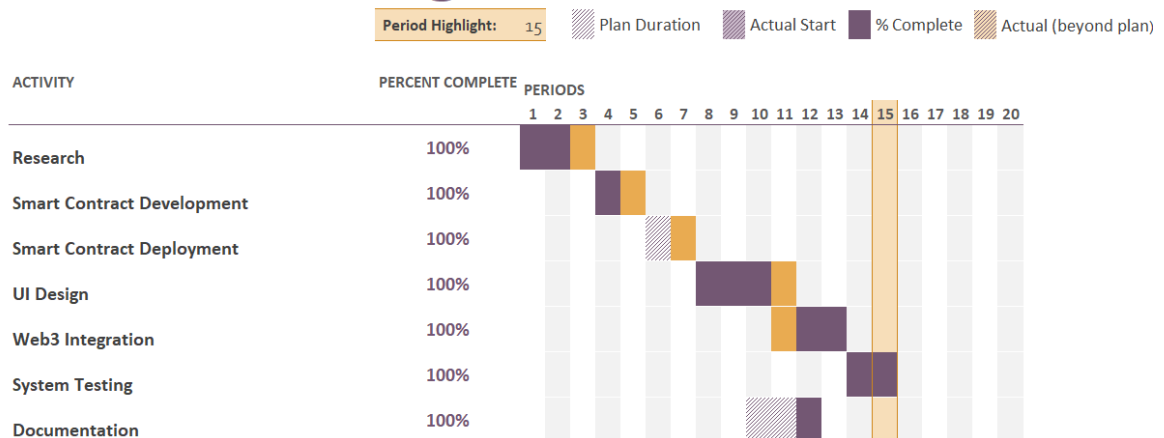


Figure 3: Project Gantt Chart

3.3. Analysis

Since blockchain based systems follow object-oriented approach, class diagram, sequence diagram and activity diagram have been generated for the analysis of the system.

3.3.1. Class Diagram

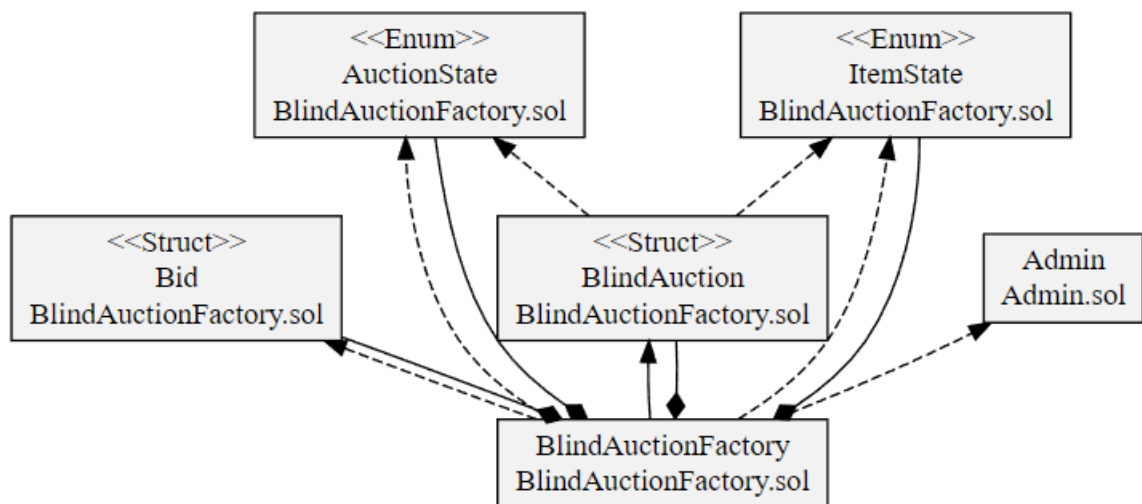


Figure 4: Class Diagram of System Smart Contracts

The system consists of two classes, an Admin class and a BlindAuctionFactory class. The BlindAuctionFactory class uses the BlindAuction and Bid structures to store information about the different auctions and bids placed on these auctions, and makes use of the

AuctionState and ItemState enumerators to keep track of the auction state and item shipment state respectively. These classes are implemented as smart contracts.

3.3.2. Sequence Diagram

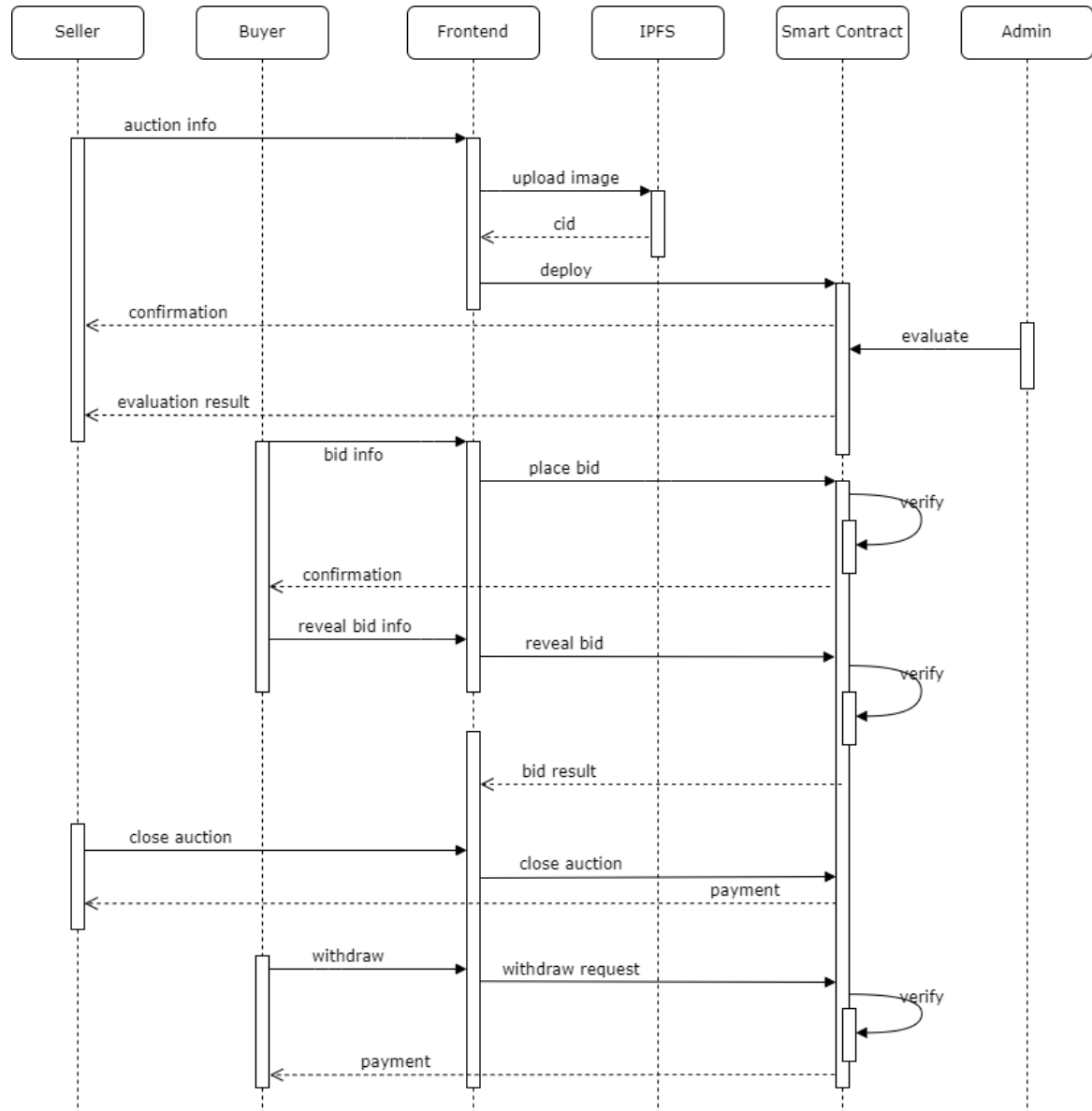


Figure 5: Sequence Diagram of the System

The auction process begins after the seller uploads the auction information to Frontend of the application. The Frontend of the application then uploads the auction images to the IPFS and obtains the CID for the uploaded images and then sends this CID along with other auction information such as title, timing details to the BlindAuctionFactory contract to register the auction. The auction is then evaluated by an admin.

To place a bid, interested bidders send bid information through the frontend to the smart contract, which is verified and validated by the contract before it is accepted. The bidder must input three parameters for placing a valid bid, a deposit amount that is higher than minimum bid amount, a true bid amount and a secret pass phrase that is used for securing the true bid amount. This is done by hashing the value of the true bid amount with the secret pass phrase. This hash value is then sent to the contract for storage.

To reveal a bid, bidders who have placed a bid on the item should disclose their true bid amount and secret pass phrase. This information is sent to the contract for verification. If the hash of the provided true bid and secret pass phrase match with the stored hash provided in the bidding phase, it is successfully revealed and then considered for calculating the highest bid.

After the auction duration has passed the, auction result is displayed, and seller can close the auction to change the auction state and transfer the winning amount to their wallet. The bidders can also send request to transfer their refundable amount to their wallets. If a bid is unsuccessful the total amount is refunded, where as if it is successful, only the difference between deposit and true bid for the winning bid is refunded.

3.3.3. Activity Diagram

The auction process starts with the upload of information by the seller to the smart contract. The item is then registered if the necessary information is provided. The item is then evaluated, where it is either rejected or verified by an admin. If the item has been rejected it has to be returned back to the seller and the auction process ends. If it has been verified, the system checks if auction start time has passed, if it has auction starts accepting bids, else it waits for the start time to pass. The system during this period also checks if auction end time has elapsed, if it has, it stops accepting bids, else it will continue to accept new bids.

Once the auction has ended, the result of the auction is calculated for the bids that were revealed. The unsuccessful bids are then marked to be returns to the respective bidders. The auction then checks if the auction was successful or not. If it was successful, the winning bid is transferred to the seller's wallet, and the item is shipped to the winner by the admin, else the item is returned back to the seller.

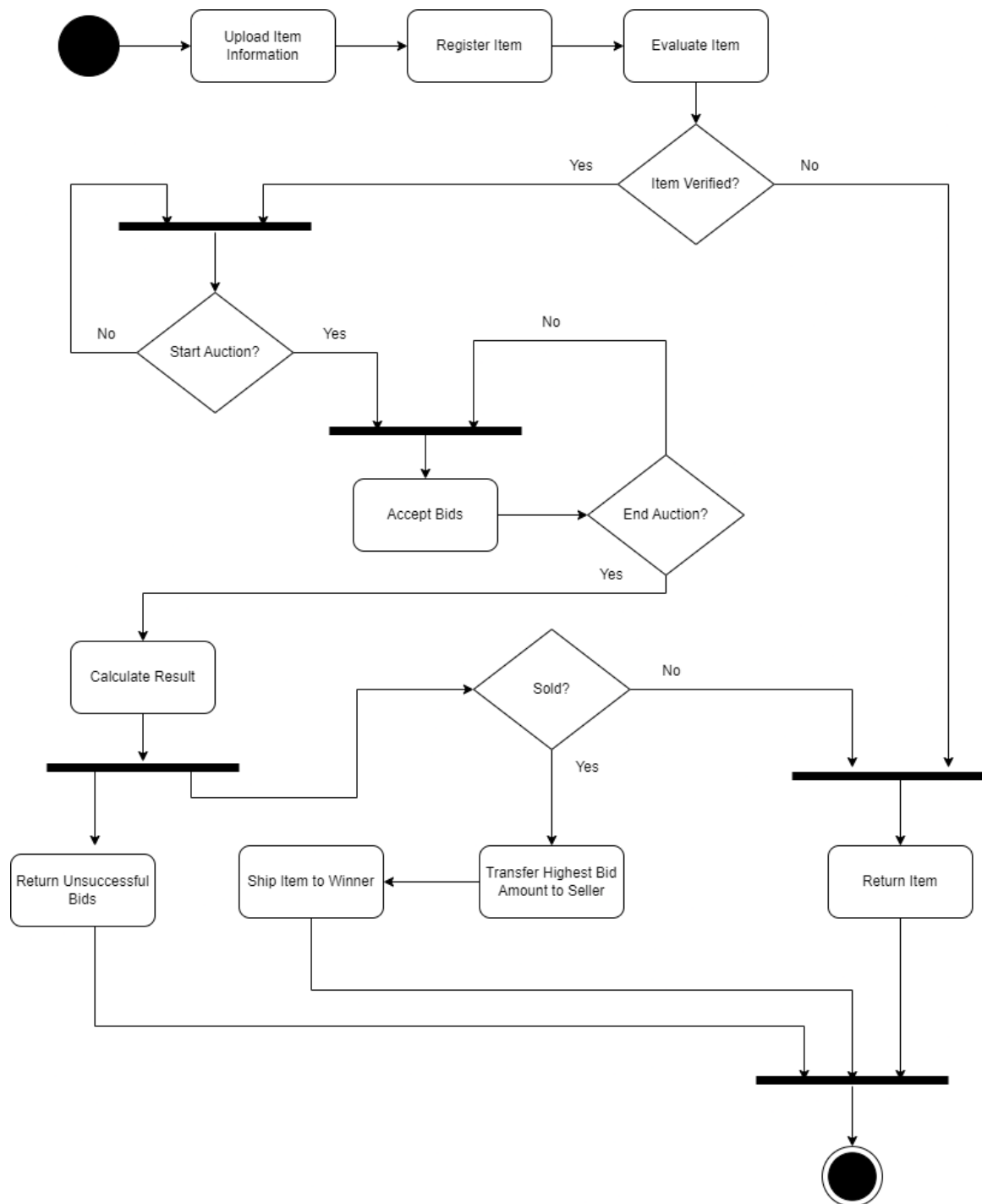


Figure 6: Activity Diagram of the System

CHAPTER 4: SYSTEM DESIGN

4.1. Design

The design of the system is illustrated using UML diagram such as component diagram and deployment diagram.

4.1.1. Component Diagram

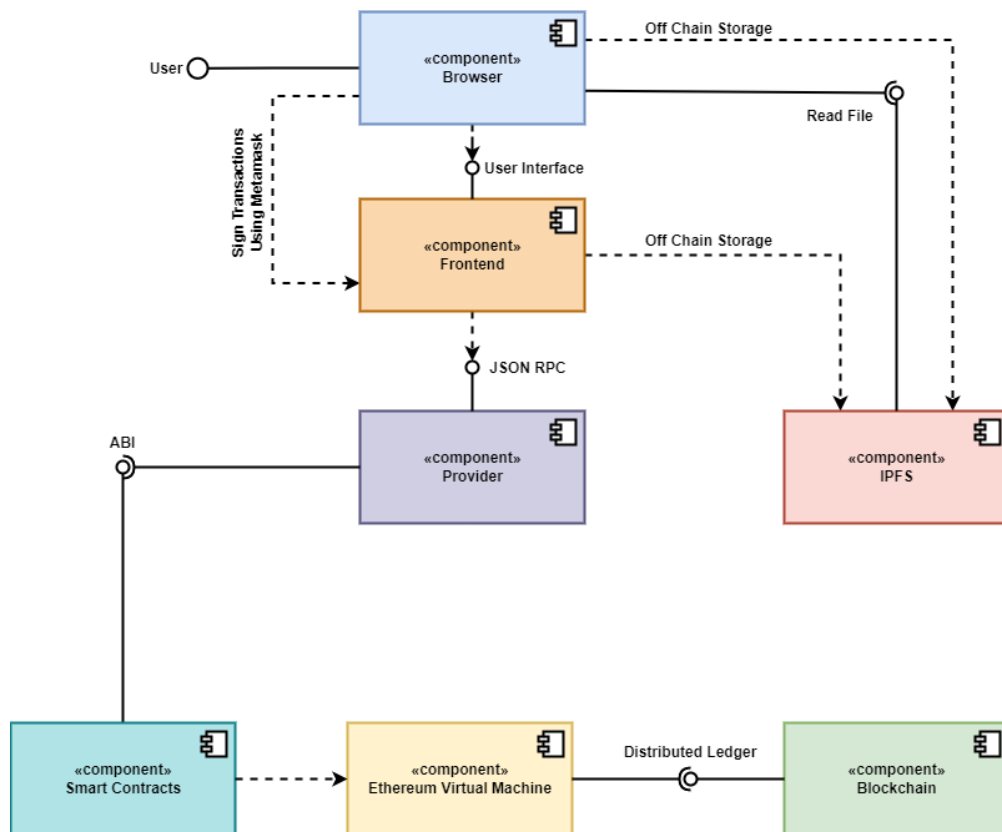


Figure 7: Component Diagram of the System

The user interacts with the frontend using a browser component, which renders the frontend in a web browser. The frontend component is responsible for providing the user interface and handling user input. To read and write image data, the browser component interacts with the IPFS component. To sign transactions, the user makes use of the Metamask signer. The frontend communicates with the Metamask provider component using a JSON RPC interface. The smart contract component is deployed on a blockchain. The Metamask

provider component allows the frontend to access the ABI of the deployed smart contract, which defines how the frontend can interact with the contract's functions and variables. The Ethereum Virtual Machine component then executes the smart contract code on the blockchain.

4.1.2. Deployment Diagram

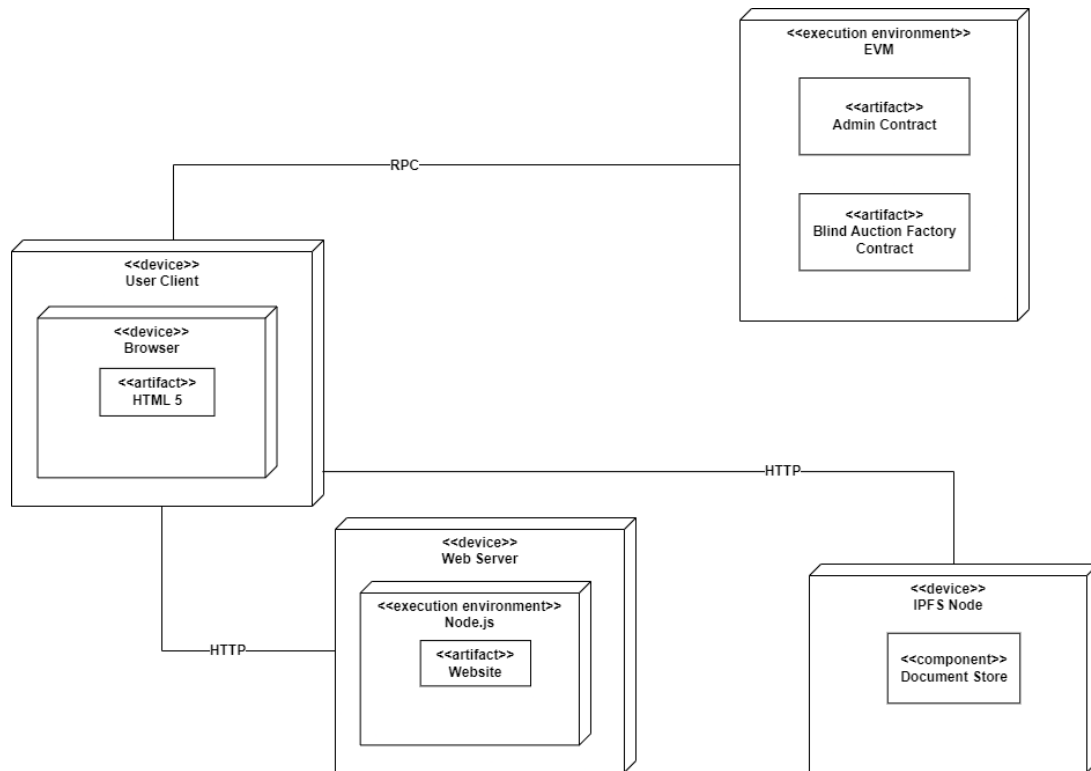


Figure 8: Deployment Diagram of the System

A User client device can access the system using a web browser that renders the HTML for the website. This is accomplished through a HTTP connection to the Webserver that runs the Node.js execution environment for the website.

The user client accesses the images for the auctions stored in a document store in IPFS using an HTTP connection and also uses this connection to upload new images for an auction to the document store.

The user client connects to an EVM using an RPC connection to communicate with the Ethereum blockchain and interact with smart contracts. The EVM includes two contracts: the Admin smart contract for managing admins and the Blind Auction Factory contract for managing auctions and the bidding process.

4.2. Algorithm Details

The different algorithms that have been used for the implementation of the blind auction system on a blockchain network have been described below.

4.2.1. Keccak-256

Keccak-256 is a cryptographic hashing algorithm belonging to the Keccak family of hash functions. It is used in Ethereum and Solidity. Each of the SHA-3 functions is based on an instance of the Keccak algorithm that NIST selected as the winner of the SHA-3 Cryptographic Hash Algorithm Competition [9]. Keccak-256 produces a 256-bit hash value and is used for various purposes, including address generation, contract creation, and transaction signing. Solidity uses the keccak256 function to compute the Keccak-256 hash of a given input.

Steps in Keccak-256 algorithm:

1. The algorithm operates on a state of 1600 bits, which is represented as a 5x5 matrix of 64-bit words. Each element of the state array is called a "lane". The state array is treated as a 3-dimensional array of bytes, with the first dimension being the lane index, the second dimension being the row index, and the third dimension being the column index.
2. The input message is first padded to ensure that its length is a multiple of the block size of 136 bytes or 1088 bits.
3. The Keccak-256 algorithm consists of a series of rounds, where each round applies a set of transformations to the state. The number of rounds for Keccak-256 is 24.
4. Each round consists of five steps: **theta** (θ), **rho** (ρ), **pi** (π), **chi** (χ), and **iota** (γ).
5. The theta step:
 - a. Compute the parity of each column of the state array. This means that for each column, we calculate the bitwise **XOR** of all the bits in that column and store the result in a new array called **C**.
 - b. Compute the XOR of the three neighboring columns of **C**. For each column **i** of **C**, we compute the bitwise **XOR** of columns **i-1**, **i**, and **i+1**, where the indices are taken modulo 5 to ensure that the computation is circular. This result is stored in a new array called **D**.

- c. Replace each column of the state array with the **XOR** of the original column and the corresponding column of **D**.

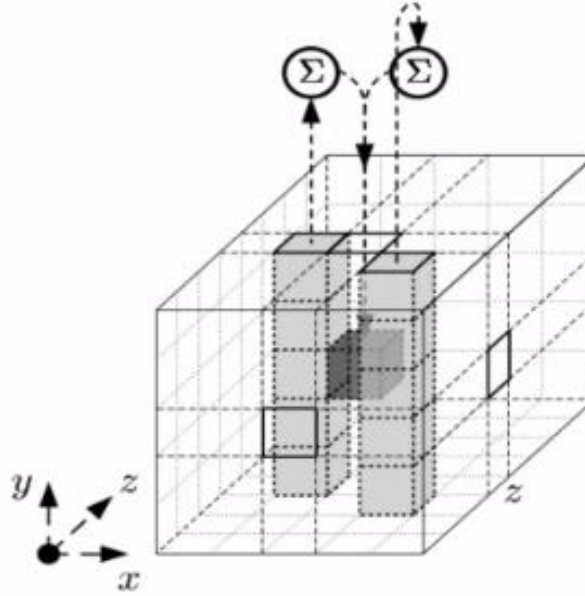


Figure 9: The Theta Step [10]

6. In the rho step, each word is rotated by a fixed amount based on its position in the matrix.

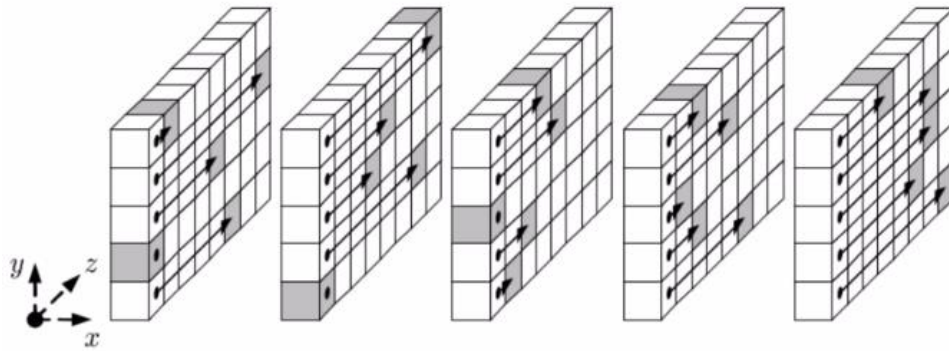


Figure 10: The Rho Step [10]

7. In the pi step, the words are rearranged based on a fixed permutation.

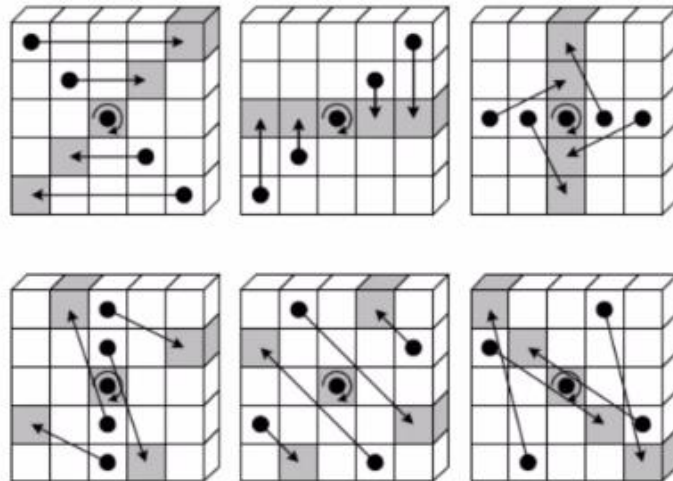


Figure 11: The Pi Step [10]

8. In the chi step, each word is **XOR**ed with a function of two of its neighboring words. This step adds a non-linear aspect to the permutation round. It combines the row elements using only three bitwise operators: **AND**, **NOT** and **XOR**.

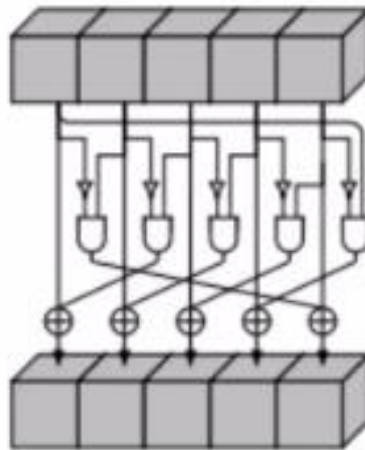


Figure 12: The Chi Step [10]

9. Finally, in the iota step, a round constant is **XOR**ed to one of the words in the state to break any symmetry caused by the other modules.
10. After 24 rounds of these operations, the resulting state is hashed to produce the final output of 256 bits.

Padding scheme in Keccak-256:

The padding scheme used in Keccak-256 is as follows:

1. Append a single "1" bit to the end of the data.
2. Append "0" bits to the data until the length of the data, in bits, is congruent to 1088 minus 128 modulo 1088.
3. Add 128 bits representing the length of the original message to the end of the padded message.

Pseudo-code description of the permutations:

Keccak-f[b](A) {

 For i in 0 to n-1 do

$A = \text{Round}[b](A, RC[i])$

 End for

 Return A

}

Round[b](A, RC) {

 # θ step

 For x in 0 to 4 do

$C[x] = A[x,0] \text{ xor } A[x,1] \text{ xor } A[x,2] \text{ xor } A[x,3] \text{ xor } A[x,4]$

 End for

 For x in 0 to 4 do

$D[x] = C[x-1] \text{ xor } \text{rot}(C[x+1], 1)$

 End for

```

For x,y in (0 to 4, 0 to 4) do

     $A[x,y] = A[x,y] \text{ xor } D[x]$ 

End for

#  $\rho$  and  $\pi$  steps

For x,y in (0 to 4, 0 to 4) do

     $B[y,2x+3y] = \text{rot}(A[x,y], r[x,y])$ 

End for

#  $\chi$  step

For x,y in (0 to 4, 0 to 4) do

     $A[x,y] = B[x,y] \text{ xor } ((\text{not } B[x+1,y]) \text{ and } B[x+2,y])$ 

End for

#  $\iota$  step

 $A[0,0] = A[0,0] \text{ xor } RC$ 

return A

}

```

The operations involving the indices in the pseudo-code shown are performed with modulo 5. The array A represents the entire permutation state, with $A[x,y]$ representing a specific lane within that state. Intermediate variables $B[x,y]$, $C[x]$, and $D[x]$ are also used during the calculations. The rotation offsets are denoted by the constants $r[x,y]$, and the round constants are represented by $RC[i]$. The function $\text{rot}(W,r)$ is used to shift bits cyclically, moving the bit at position i to position $i+r$ (with the lane size taken into account) [11].

Table 2: The Round Constants RC[i] [11]

RC[0]	0x0000000000000001	RC[12]	0x000000008000808B
RC[1]	0x0000000000008082	RC[13]	0x800000000000008B
RC[2]	0x800000000000808A	RC[14]	0x8000000000008089
RC[3]	0x8000000080008000	RC[15]	0x8000000000008003
RC[4]	0x000000000000808B	RC[16]	0x8000000000008002
RC[5]	0x0000000080000001	RC[17]	0x8000000000000080
RC[6]	0x8000000080008081	RC[18]	0x000000000000800A
RC[7]	0x8000000000008009	RC[19]	0x800000008000000A
RC[8]	0x000000000000008A	RC[20]	0x8000000080008081
RC[9]	0x0000000000000088	RC[21]	0x8000000000008080
RC[10]	0x0000000080008009	RC[22]	0x0000000080000001
RC[11]	0x000000008000000A	RC[23]	0x8000000080008008

Table 3: The Rotation Offsets r[x,y] [11]

	x = 3	x = 4	x = 0	x = 1	x = 2
y = 2	25	39	3	10	43
y = 1	55	20	36	44	6
y = 0	28	27	0	1	62
y = 4	56	14	18	2	61
y = 3	21	8	41	45	15

4.2.2. Suitability of Algorithm

The advantage of Keccak-256 over other hashing algorithms is that it is not vulnerable to the same types of attacks as other hashing algorithms like SHA-1 and SHA-2. In 2017, Google announced that it had successfully generated two PDFs with different contents but

the same SHA-1 hash [12]. This was a significant security issue, as many systems still rely on SHA-1 for digital signatures and other security-related tasks. In contrast, Keccak-256 has not been found to have any such vulnerabilities.

The use of a sponge construction and parallel processing of input data makes it more efficient and faster than other hashing computation. This is important for implementation in a blockchain network such as Ethereum, which relies on the hashing algorithm to perform a wide range of tasks, including mining, transaction validation, and smart contract execution.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1. Implementation

A blind online auction DApp system is implemented using various technologies and processes. First, the contracts are written in Solidity, a programming language used for developing smart contracts for the Ethereum blockchain. These contracts are then compiled into bytecode and deployed onto the local blockchain node using deployment scripts written in JavaScript and Hardhat, a development environment for Ethereum smart contracts.

The frontend of the website is built using JavaScript and NextJS framework. Once the web server is started, users can access the website using its URL through a web browser. To access the services, users must connect to the website using their Metamask wallet, which serves as their digital identity and enables them to interact with the blockchain.

Sellers can create an auction by providing the required information such as auction start time, end time, minimum bid amount, images, auction title, and description. Images uploaded by sellers are stored in IPFS, a decentralized storage system. The creation of an auction involves storing its information in the associated **BlindAuctionFactory** contract deployed to the local blockchain. The auction is then verified or rejected by the admin. If verified, it is displayed on the website.

Bidders can bid on the verified auction during the bidding period, which starts after the auction start time has passed and before the auction end time has elapsed. To bid, a bidder must provide a deposit amount, bid amount, and a secret passphrase used to encrypt the bid amount. A bidder can bid only once.

After the auction end time has elapsed, the auction enters the reveal bid phase. During this period, bidders reveal their bid by providing the true bid and secret passphrase. These values are then compared to their hashed value stored in the auction smart contract. If the hashes are the same, the bid is valid and considered for evaluation as the highest bid.

After the reveal period has ended, the contract can evaluate the valid bids to determine the highest bidder. The unsuccessful bidders can withdraw their deposit amount at this point. The highest bid is then transferred to the seller's wallet, and the result is announced. The highest bidder can also withdraw the remaining amount from their deposit at this point. Finally, the auction is marked as closed.

5.1.1. Tools Used

The following tools, libraries, framework, and programming languages were used during the development and deployment of the system proposed in this paper.

Visual Studio Code: VS Code is a free and open-source code editor. It has been used for writing and debugging code.

Draw.io: Draw.io is a web-based software used for making diagrams and charts, such as flowcharts, network diagrams, ER diagrams, UML diagrams and so on. It has been used to create the various diagrams used in this report.

Solidity 2 UML: A visualization tool for solidity. It has been used to generate the class diagrams used in this report.

Solidity: Solidity is a statically typed high-level programming language used for writing smart contracts on blockchain platforms, such as the Ethereum blockchain. It has been used to write smart contracts for this project.

JavaScript: JavaScript is a high-level, dynamic programming language used for developing web-based applications. It is used for adding interactivity and dynamic effects to web pages and web applications. It has been used to develop the front of the application and write deployment scripts in hardhat for deploying contract to the local blockchain.

Hardhat: It is a development environment for Ethereum software. It was used for editing, compiling, debugging and deploying the smart contracts.

Ganache: It is a personal, local blockchain platform used for developing and testing Ethereum-based applications. It was used to create a test blockchain network to deploy the smart contracts used in this project.

React.js: A JavaScript library for building user interfaces. This library was used to build frontend components and pages.

Nest.js: A Server-Side Rendering React.js framework. The frontend of the project was built on top of this framework. It was used to handle frontend routing.

Mantine UI: A popular React UI framework. This package was used to implement styling to the pages built using React.js.

ThirdWeb: ThirdWeb is a web3 development framework that provides a set of React hooks for connecting to wallet providers, storing and retrieving files from IPFS, querying data from the blockchain, and executing smart contract functions into React applications.

IPFS: It is a distributed file system that allows for permanent, decentralized storage of files, and provides a way to create decentralized applications that are not reliant on centralized servers. It has been used to store auction images uploaded by sellers.

Metamask: It is a browser extension that allows users to interact with the Ethereum blockchain and other compatible blockchains. It provides a digital wallet and a user interface for users to manage their accounts, sign transactions, and interact with DApps that are built on top of the blockchain. It is used to connect with the local test network and use the account address to execute transactions in the project.

5.1.2. Implementation Details of Modules

5.1.2.1. File: Admin.sol

This is a solidity smart contract that is used to store address of super admin and admin users. It provides methods for super-admin to add and remove admin users.

Table 4: Admin Smart Contract Functions

Brief Description	Function Name
Register an address as admin.	registerAdmin()
Remove an address as admin.	unregisterAdmin()
Check if an address is admin.	isAdmin()
Check if an address is super admin	isSuperAdmin()

Return list of admin addresses.	getAdmins()
Default function.	fallback()

registerAdmin(): It allows the super admin to register a new admin by providing their address. It checks if the address is valid and if the admin is already registered before adding them to the list of admin addresses.

unregisterAdmin(): It allows the super admin to unregister an existing admin by providing their address. It checks if the address is valid, if the admin is the super admin, and if the admin is already registered before removing them from the list of admin addresses.

isAdmin(): It checks whether a given address is registered as an admin or not.

isSuperAdmin(): It checks whether the caller is the super admin or not.

getAdmins(): It returns an array of all the registered admin addresses. This function can only be called by the super admin.

fallback(): This is a default function that is executed when the contract receives a transaction that does not match any of the defined functions.

5.1.2.2. File: BlindAuctionFactory.sol

This contract stores information about all auction items created on the platform, and is responsible for handling the auction proceedings.

Table 5: BlindAuctionFactory Smart Contract Functions

Brief Description	Function Name
Used to create an auction.	createBlindAuctionContract()
Called by an admin to verify the auction.	verifyAuction()
Used by an admin to reject the auction.	rejectAuction()
Stores valid bid.	bid()
Called by bidder to reveal a bid.	reveal()
Internal function to place a bid.	placeBid()

Allows bidders to refund their bid.	withdraw()
Function to end the auction.	auctionEnd()
Internal function to allow unrevealed bids to be refunded.	refundBids()
Function to cancel an auction.	cancelAuction()
Allows winner to provide shipping address.	updateShippingAddress()
Allows admin to update shipment status of item.	updateShipmentStatus()
Returns all auctions stored in the contract.	getBlindAuctions()
Returns a particular auction information.	getBlindAuctionById()
Returns shipping address information.	getShippingAddress()
Provides auction information to other functions.	getBlindAuction()
Default function.	fallback()

createBlindAuctionContract(): This function is used to create an auction by a seller.

verifyAuction(): This function can only be called by an admin to verify an auction. It can only be executed before auction start time has elapsed and auction has not been evaluated before.

rejectAuction(): This function can only be called by an admin to reject an auction. It can only be executed before auction start time has elapsed and auction has not been evaluated before.

bid(): This function is called by a buyer to place a bid for the auction item. It can only be called for a verified auction after auction start time has passed and before auction end time has elapsed.

placeBid(): This internal function is called by the reveal function when the reveal is valid. It checks if a bid is greater than the highest bid value and if true, set new highest bid. It also determines the amount that should be returned to the bidder after subtracting their bid from the deposit.

reveal(): This function is used to check if the hash of bid amount and secret pass phrase provided by a bidder is equal to the hash of the same previously provided by the bidder during the bidding phase and stored in the contract.

auctionEnd(): It is used to close the auction and transfer the highest bid amount to the seller's wallet address.

refundBids(): It is called by the **auctionEnd()** function to allow the deposit amount of unrevealed bids to be refunded to the associated bidders.

withdraw(): It transfers the refundable amount to bidders wallet after auction has ended.

cancelAuction(): This is an external function that can only be called by the seller to cancel an auction before it has been verified.

updateShippingAddress(): This is an external function that can only be called by the highest bidder to update the shipping address after the auction has been closed and before item has been shipped.

updateShipmentStatus(): This is an external function that can only be called by the admin to mark that an auction item has been shipped to the concerned party.

getBlindAuctions(): This function provides information about all the auctions on the platform to its caller as an array of auctions.

getBlindAuctionsById(): This is an external function that returns the details of a particular auction using its ID.

getShippingAddress(): This is an external function that can only be called by an admin user and used to get the shipment address of successful, failed or rejected auction.

getBlindAuction(): This is an internal function that returns details of an auction to other functions in the contract using the auction's ID.

fallback(): This is a default function that is executed when the contract receives a transaction that does not match any of the defined functions.

5.1.2.3. File: 01-deploy-admin.js

This file is a module written in JavaScript that exports an asynchronous function, along with some metadata, using the CommonJS module system. The function is designed to be used with the Hardhat framework for Ethereum development.

The function takes an object as an argument that contains two properties: **getNamedAccounts** and **deployments**. These properties are functions that are provided by Hardhat to assist with the deployment of contracts.

Inside the function, the **deployments.deploy** function is used to deploy a contract called "Admin" with an empty argument array. The address of the deployed contract is then logged to the console.

5.1.2.4. File: 02-deploy-blindAuctionFactory.js

This JavaScript deploys a contract named BlindAuctionFactory with the provided arguments. The address of the deployed Admin contract is obtained using **ethers.getContract()** and passed as an argument to the **BlindAuctionFactory** contract constructor. Once the deployment is complete, the address of the deployed BlindAuctionFactory contract is logged to the console.

5.1.2.5. File: 99-update-frontend.js

This is a script used to update the contract addresses and ABIs that the front-end of a DApp uses to interact with the smart contracts on the blockchain.

The script updates the contract addresses and ABIs for the following contracts:

- Admin
- BlindAuctionFactory

The updated ABIs are written to JSON files that are used by the front-end application to interact with the contracts.

5.2. Testing

5.2.1. Test Cases for Unit Testing

Table 6: Test Cases for Unit Testing of the Application

Test Case #	Test Case Description	Test Data	Expected Result	Actual Result	Pass / Fail
TU01	Creating an auction with invalid auction period information.	Title: Auction 1 Minimum Bid: 1 Start Time: 07/03/2023 15:00 End Time: 07/03/2023 14:00 Images: [photo.png] Description: Description 1 Return Address: Address 1	Invalid Auction Period error notification.	As expected	Pass
TU02	Creating an auction with correct auction period information.	Title: Auction 1 Minimum Bid: 1 Start Time: 07/03/2023 15:00 End Time: 07/03/2023 16:00 Images: [photo.png] Description: Description 1 Return Address: Address 1	Auction created by smart contract and success notification displayed to user.	As expected	Pass
TU03	Cancel an auction before	Auction ID: 0xb2cdfb8c2edb75d05dda 866308cd98d66dda9ed482 72a010aaa478cff4986135	Auction cancelled and auction state	As expected	Pass

	auction is verified.		updated as Failed.		
TU04	Place bid on an open auction before.	Deposit: 10 True Bid: 5 Secret: bidder1	Bid placed on auction item and success notification displayed	As expected	Pass
TU05	Reveal bid on an open auction after placing a valid bid with incorrect information.	True Bid: 10 Secret: bidder1	Error notification stating incorrect bid reveal information.	As expected	Pass
TU06	Reveal bid on an open auction after placing a bid with incorrect information.	True Bid: 5 Secret: bidder1	Notification stating successful bid reveal displayed.	As expected	Pass
TU07	Send second bid reveal request to the contract.	True Bid: 5 Secret: bidder1	Request denied.	Error notification stating incorrect bid reveal information.	Fail

TU08	Withdraw deposit amount after auction end.	Bidder ID: 0x450840C781Bdd965Af7 eb0a6701EaccaDcF53B6f Auction ID: 0xb2cdfb8c2edb75d05dda 866308cd98d66dda9ed482 72a010aaa478cff4986135	Transfer pending return to bidder's wallet address.	As expected	Pass
------	--	---	---	-------------	------

5.2.2. Test Cases for System Testing

Table 7: Test Cases for System Testing of the Application

Test Case #	Test Case Description	Test Data	Expected Result	Actual Result	Pass / Fail
TU01	Closing an auction after auction reveal period.	Auction ID: 0xb2cdfb8c2edb75d05dda 866308cd98d66dda9ed482 72a010aaa478cff4986135	Auction status changed to successful, winning bid amount transferred to seller, highest bidder and bid displayed.	As expected	Pass
TU02	Updating shipping address by the highest bidder.	Bidder ID: 0x450840C781Bdd965Af7 eb0a6701EaccaDcF53B6f Shipping Address: bidder1, Kathmandu, Nepal	Shipping address updated, and displayed to admins on	As expected	Pass

			the Ship Item page.		
TU03	Send withdraw request for an unrevealed bid without closing an auction.	Auction ID: 0xb2cdfb8c2edb75d05dda 866308cd98d66dda9ed482 72a010aaa478cff4986135 Bidder ID: 0x8ab5343b94f5Aa297B1 7c9BBAA2609d507cC2e9f	Request denied.	Error notification stating no pending return.	Fail
TU04	Send withdraw request for an unrevealed bid after closing an auction.	Auction ID: 0xb2cdfb8c2edb75d05dda 866308cd98d66dda9ed482 72a010aaa478cff4986135 Bidder ID: 0x8ab5343b94f5Aa297B1 7c9BBAA2609d507cC2e9f	Pending return amount returned to the bidder's wallet address.	As expected	Pass

5.3. Result Analysis

The system was tested on both Hardhat local node and Ganache local node. Throughout the testing phase, 15 user accounts were utilized, with 5 assigned for buyers, 5 for sellers, and 4 for admins and 1 for super admin.

Multiple auctions were created using the seller accounts to test the system. The buyer accounts were then used to test the system's functionalities such as bidding, revealing bids, withdrawing unsuccessful bids, and updating the shipping address. During the test, the admin accounts were able to successfully verify or reject an auction, view the shipping address, and update the shipment status, while the super admin account was able to add or remove admin addresses as necessary. The system also correctly checked whether different user roles should be allowed to perform restricted transactions or not.

CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATIONS

6.1. Conclusion

The development of a blockchain-based blind auction system has been proposed to address the issues of transparency, fairness, and security in the current online auction systems. The proposed system aims to ensure that all bidders have an equal chance of winning the auction, maintain anonymity, prevent collusion, simplify the bidding process, and increase efficiency.

The platform incorporates a sealed-bid mechanism, smart contract, and automatic enforcement of auction rules to provide a fair and secure auction environment for bidders, and has the potential to revolutionize the online auction industry and can be adapted to various industries to provide secure and transparent transactions. The success of this project highlights the potential of blockchain technology in addressing various challenges in online transactions and provides a solid foundation for further research and development in this field.

6.2. Future Recommendations

The system can be further developed to incorporate the use of an oracle data feed to track the progress of item shipment and display it to all concerned parties. This will help ensure that all parties have visibility over the item's delivery status, which is particularly important for high-value items. Additionally, the use of an oracle can provide more accurate and up-to-date information regarding shipment status, improving transparency and accountability in the auction process. Another recommendation is to introduce the use of a custom token for transaction purposes. This will provide several benefits, such as increased efficiency and lower transaction fees.

Finally, the introduction of auctioning virtual goods can provide a new and exciting dimension to the blind auction system. This can open up opportunities for individuals or

organizations to auction virtual goods, such as digital artwork or in-game items, which can attract a wider audience and generate more revenue.

CHAPTER 7: REFERENCES

- [1] D.-H. Shih, T.-W. Wu, M.-H. Shih, W.-C. Tsai and D. C. Yen, "A Novel Auction Blockchain System with Price Recommendation and Trusted Execution Environment," *Mathematics*, 13 December 2021.
- [2] "Fraud Reports and Facts," Federal Trade Commission, 28 February 2019. [Online]. Available: <https://public.tableau.com/app/profile/federal.trade.commission/viz/FraudReports/FraudFacts>. [Accessed 19 2 2023].
- [3] H. Li and W. Xue, "A Blockchain-Based Sealed-Bid e-Auction Scheme with Smart Contract and Zero-Knowledge Proof," *Security and Communication Networks*, vol. 2021, 21 May 2021.
- [4] Y.-H. Chen, S.-H. Chen and I.-C. Lin, "Blockchain based Smart Contract for Bidding System," *Proceedings of IEEE International Conference on Applied System Innovation*, 2018.
- [5] X. Jia, X. Song and M. Sohail, "Effective Consensus-Based Distributed Auction Scheme for," *Symmetry*, vol. 14, no. 8, p. 1664, 11 August 2022.
- [6] I. Omar, H. Hasan, R. Jayaraman and K. Salah, "Implementing Decentralized Auctions Using Blockchain Smart Contracts," *Technological Forecasting and Social Change*, vol. 168, 2021.
- [7] "U.S. Embassy Online Auction," [Online]. Available: <https://online-auction.state.gov/en-US>. [Accessed 2 11 2022].
- [8] "Bolpatra Nepal," Bolpatra Nepal, [Online]. Available: <https://bolpatranepal.com/>. [Accessed 2 11 2022].
- [9] M. J. Dworkin, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," *Federal Inf. Process. Stds. (NIST FIPS)*, 4 August 2015.

- [10] G. Bertoni, J. Daemen, M. Peeters and G. V. Assche, "The KECCAK reference," 14 January 2011.
- [11] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche and R. V. Keer, "KECCAK implementation overview," 29 May 2012.
- [12] M. Stevens, E. Bursztein, P. Karpman, A. Albertini and Y. Markov, "Announcing the first SHA1 collision," Google, 23 February 2017. [Online]. Available: <https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html>. [Accessed 17 2 2023].

APPENDIX I

```
function registerAdmin(address adminAddress) external onlySuperAdmin {
    if (adminAddress == address(0)) revert("Invalid address");
    if (adminAddressMap[adminAddress] == true)
        revert("Address already resgistered");
    adminAddressMap[adminAddress] = true;
    adminAddressArray.push(adminAddress);
}

function unregisterAdmin(address adminAddress) external onlySuperAdmin {
    if (adminAddress == address(0)) revert("Invalid address");
    if (adminAddress == superAdminAddress)
        revert("Cannot delete super admin");
    if (adminAddressMap[adminAddress] == false)
        revert("Address is not admin");
    delete adminAddressMap[adminAddress];
    uint length = adminAddressArray.length;
    for (uint i = 0; i < length; i++) {
        if (adminAddressArray[i] == adminAddress) {
            adminAddressArray[i] = adminAddressArray[
                adminAddressArray.length - 1
            ];
            delete adminAddressArray[length - 1];
            adminAddressArray.pop();
            break;
        }
    }
}
```

Figure A: Implementation of registerAdmin() and unregisterAdmin() Methods

```
function bid(
    bytes32 _auctionId,
    bytes32 _blindedBid
)
{
    external
    payable
    verifiedAuction(_auctionId)
    onlyAfterStartTime(_auctionId)
    onlyBeforeEndTime(_auctionId)
    notSeller(_auctionId)
    noPreviousBid(_auctionId)
    {
        BlindAuction storage blindAuction = getBlindAuction(_auctionId);
        if (msg.value < blindAuction.minimumBid)
            revert("Bid is less than minimum bid");
        bids[_auctionId][msg.sender] = Bid({
            blindedBid: _blindedBid,
            deposit: msg.value
        });
        blindAuction.bidders.push(msg.sender);
        emit BidPlaced(_auctionId, msg.sender);
    }
}
```

Figure B: Implementation of bid() Method

```

function reveal(
    bytes32 _auctionId,
    uint trueBid,
    bytes32 secret
)
    external
    verifiedAuction(_auctionId)
    onlyAfterEndTime(_auctionId)
    onlyBeforeRevealTime(_auctionId)
{
    uint refund;
    Bid storage bidToCheck = bids[_auctionId][msg.sender];
    if (
        bidToCheck.blindedBid !=
        keccak256(abi.encodePacked(trueBid, secret))
    ) {
        revert("Incorrect true bid value or secret");
    }
    refund = bidToCheck.deposit;
    if (bidToCheck.deposit >= trueBid) {
        if (placeBid(_auctionId, msg.sender, trueBid)) refund -= trueBid;
    }
    pendingReturns[_auctionId][msg.sender] += refund;
    bidToCheck.blindedBid = bytes32(0);
    emit BidRevealed(_auctionId, msg.sender);
}

```

Figure C: Implementation of reveal() Method

```

function placeBid(
    bytes32 _auctionId,
    address bidder,
    uint value
) internal returns (bool success) {
    BlindAuction storage blindAuction = getBlindAuction(_auctionId);
    if (
        value < blindAuction.minimumBid || value <= blindAuction.highestBid
    ) {
        return false;
    }
    if (blindAuction.highestBidder != address(0)) {
        pendingReturns[_auctionId][
            blindAuction.highestBidder
        ] += blindAuction.highestBid;
    }
    blindAuction.highestBid = value;
    blindAuction.highestBidder = bidder;
    return true;
}

```

Figure D: Implementation of placeBid() Method


```

function auctionEnd(
    bytes32 _auctionId
) external verifiedAuction(_auctionId) onlyAfterRevealTime(_auctionId) {
    BlindAuction storage blindAuction = getBlindAuction(_auctionId);
    if (
        blindAuction.auctionState == AuctionState.SUCCESSFUL ||
        blindAuction.auctionState == AuctionState.FAILED
    ) revert("Auction has already been closed");
    refundBids(_auctionId);
    if (blindAuction.highestBidder != address(0)) {
        blindAuction.auctionState = AuctionState.SUCCESSFUL;
        emit AuctionSuccessful(
            _auctionId,
            blindAuction.highestBidder,
            blindAuction.highestBid
        );
        payable(blindAuction.seller).transfer(blindAuction.highestBid);
        payable(superAdminAddress).transfer(STAKE);
    } else {
        blindAuction.auctionState = AuctionState.FAILED;
        payable(blindAuction.seller).transfer(STAKE);
        emit AuctionFailed(_auctionId);
    }
}

```

Figure E: Implementation of auctionEnd() Method

```

function refundBids(bytes32 _auctionId) internal {
    BlindAuction storage blindAuction = getBlindAuction(_auctionId);
    for (uint i = 0; i < blindAuction.bidders.length; i++) {
        address bidder = blindAuction.bidders[i];
        Bid storage bidToCheck = bids[_auctionId][bidder];
        if (bidToCheck.blindedBid == bytes32(0)) continue;
        pendingReturns[_auctionId][bidder] += bidToCheck.deposit;
        bidToCheck.blindedBid = bytes32(0);
    }
}

```

Figure F: Implementation of refundBids() Method

APPENDIX II

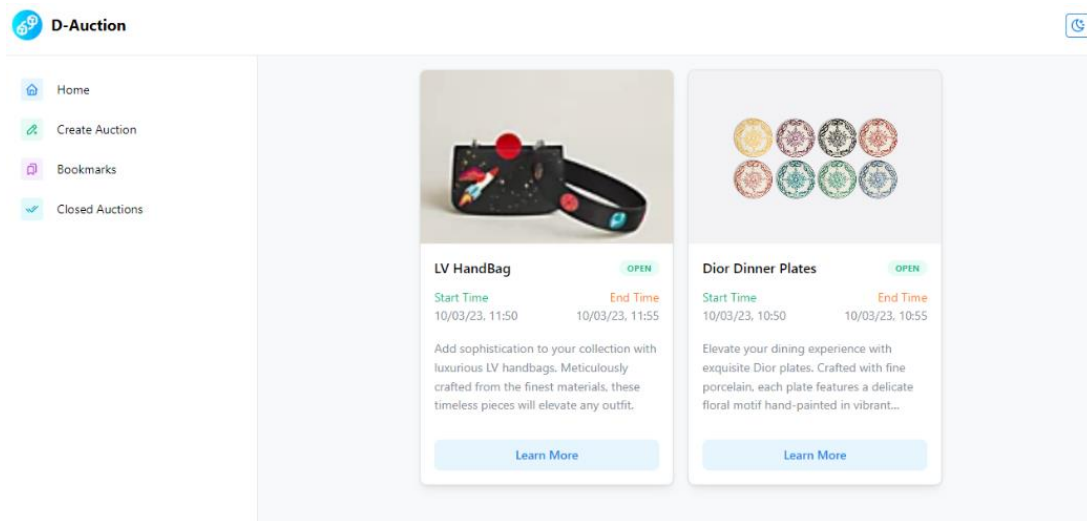


Figure G: Home Page of the Application

The screenshot displays the 'Create Auction' page with a modal form titled 'Auction Details'. The form contains the following fields: 'Title' (filled with 'Dior Dinner Plates'), 'Minimum Bid' (filled with '5'), 'Start Date' (filled with 'March 10, 2023'), 'Start Time' (filled with '10 : 40'), 'End Date' (filled with 'March 10, 2023'), 'End Time' (filled with '10 : 45'), 'Item Images' (filled with a long alphanumeric string), 'Description' (filled with a paragraph about Dior plates), and 'Return Address' (filled with '123 Main Street, Anytown, USA 12345'). A 'Submit' button is located at the bottom right of the form. The background shows a blurred view of the application interface with a 'Create Auction' button visible.

Figure H: Create Auction Page of the Application

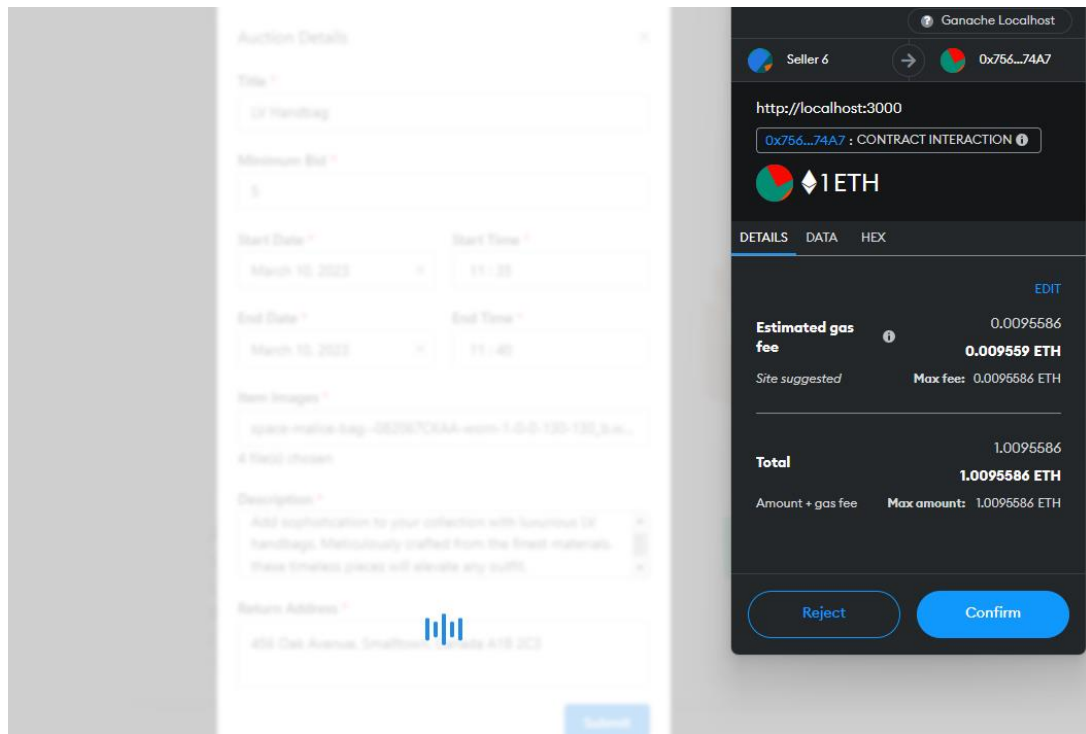


Figure I: Transaction Using Metamask

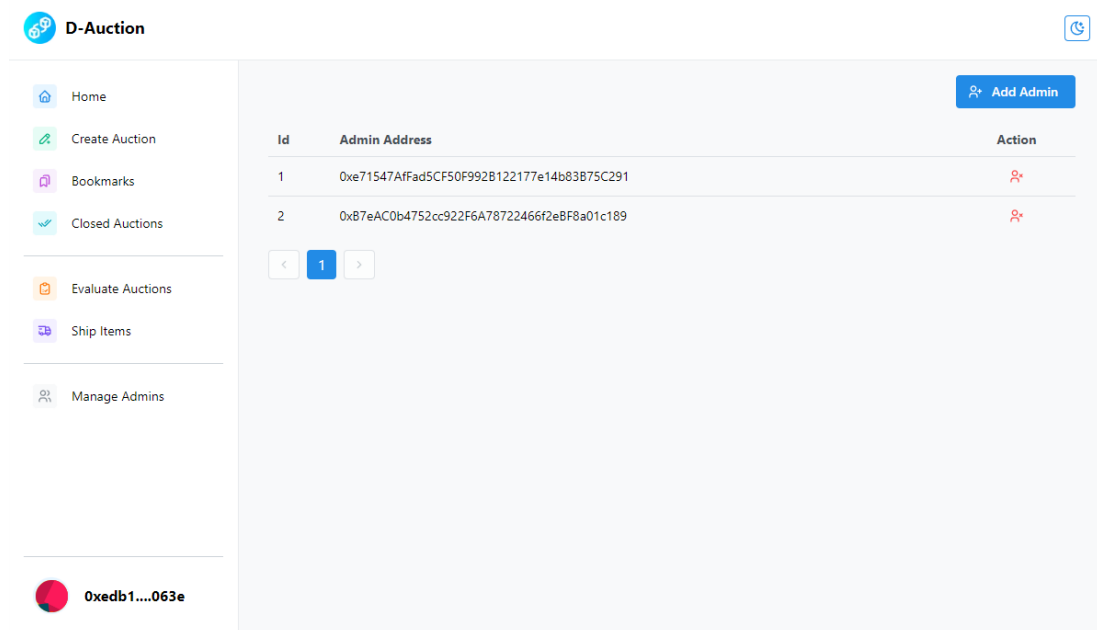



Figure J: Manage Admin Page



OPEN

Start Time

10/03/23, 11:50 AM

End Time

0037

DaysHoursMinutesSeconds

Reveal Time

10/03/23, 11:57 AM

LV HandBag

Auction Id:
0xd925fc6af2915d71d5067fd2233b0b5248d550d15c7fb749bc09ee493e59d709

Seller: 0xb6F494300440dE0d08dc41A085c688d6778ff475

Minimum Bid: 4.0 ETH

Add sophistication to your collection with luxurious LV handbags. Meticulously crafted from the finest materials, these timeless pieces will elevate any outfit.

Evaluation Message: Valid LV handbags

Evaluated By: 0xedb1d2580b82057aF77eF29fA94b7399e403063e

Highest Bidder: 0x00

Highest Bid: 0.0 ETH

Place Bid

Deposit amount *

Bid amount *

Secret Phrase *

Your secret pass phrase


Submit

Figure K: Live Auction

D-Auction

- Home
- Create Auction
- Bookmarks
- Closed Auctions
- Evaluate Auctions
- Ship Items

Dior Dinner Plates



Elevate your dining experience with exquisite Dior plates. Crafted with fine porcelain, each plate features a delicate floral motif hand-painted in vibrant colors.

Auction Item Received

Auction Evaluation Message *

Evaluation Message

VerifyReject

ID: 0xbe0556ad5df598611b7e4aa51036e91a3739dcca2ab624b8ec6035660490a99a
Seller: 0x6DbbeAA1e29836A709843f0e91e19761E093f221
Start Time: 10/03/23, 10:50
End Time: 10/03/23, 10:55

Figure L: Auction Evaluation Form

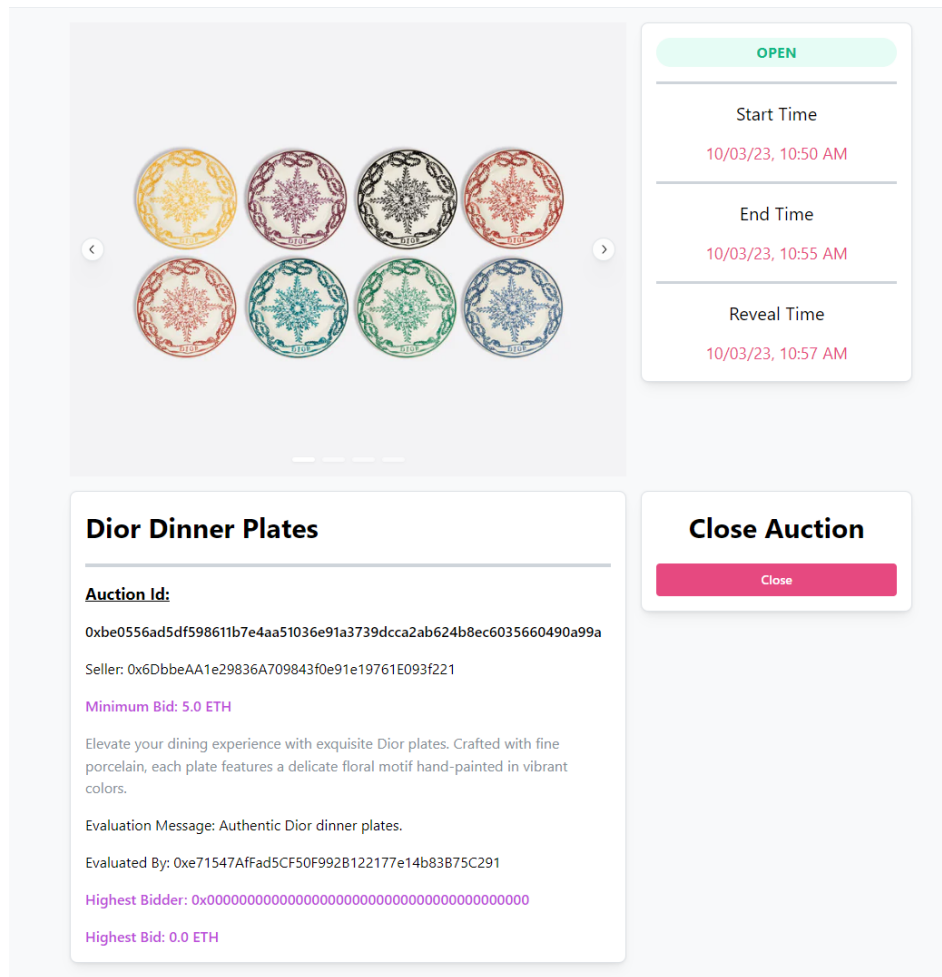


Figure M: Auction Details Page

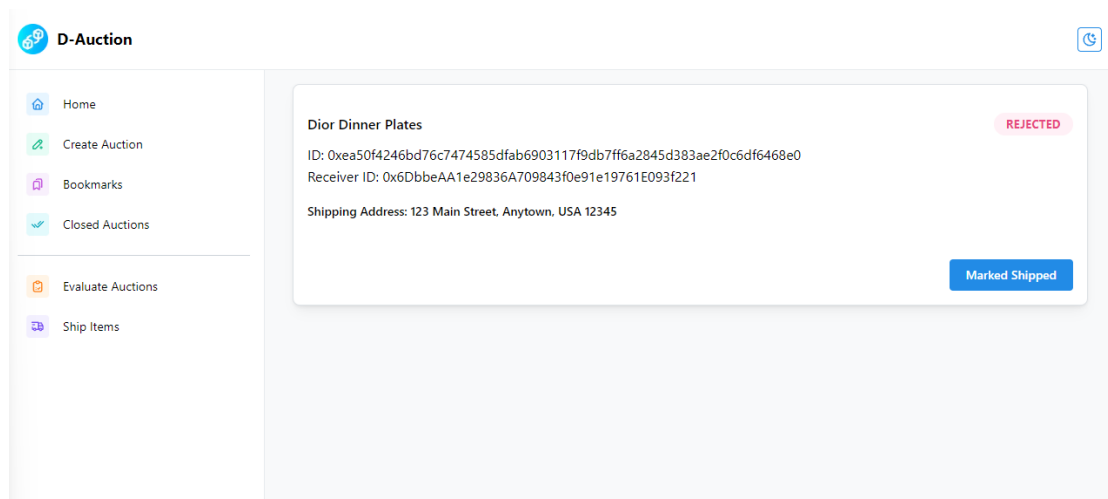


Figure N: Ship Items Page