Auction House Using Smart Contracts

By

Raja Amad Iftikhar

Supervised By

Dr. Muhammad Usman

Department of Computer Sciences

Quaid-i-Azam University

Islamabad

2014-2018

ACKNOWLEDGMNET

In the name of ALLAH, Most Beneficent, Most Merciful.

First of all, thanks to Almighty Allah, through his mercy I was able to complete my project.

I would like to express my deepest gratitude to all those who provided me the possibility to complete this project. A special gratitude I give to my respected teacher and supervisor Dr. Muhammad Usman, whose guidance and kindness helped me to coordinate in my project. Furthermore, I would like to give thanks to all faculty members of computer science department for their help, giving proper attention and time.

I would like to express my gratitude to my parents and all family members specially my elder brother Raja Usman Iftikhar for his encouragement, which helped me throughout my life.

My thanks and appreciation also goes to my class fellows and friends, who have willingly helped me out with their abilities.

Raja Amad Iftikhar
2014-2018

Abstract

Auction House Using Smart Contracts is a decentralized application which operates using the power of blockchain technology and Smart Contracts. People can buy and sell products through auction without depending upon any third-party service like eBay. Smart Contracts allow this application to run in trust-less and decentralized environment. People no longer need to provide their private information like name, password, email address to access features of website. Multisignature escrow service is implemented in which two out of three people which are buyer, seller and arbiter (person who finalized the auction) vote to either release fund to seller or refund amount to seller in case of any product related problem.

Table of Contents

List of Figures

List of Tables

"Education is the most powerful weapon which you can use to change the world."

. *Nelson Mandela (1918-2013)*

Chapter 1

Introduction

This chapter first introduces the Auction House Using Smart Contracts. It then highlights the problem that has been addressed in this work. It also elaborates project organization and project planning. Finally, this chapter elucidates the scope and objectives of this project.

1.1 Problem Definition

The project is named as "Auction House using Smart Contract" which is an online auction system. Currently there are many online auction systems on the internet. Three of the most famous auction websites are: eBay.com, asteinrete.com and onsale.com. All of these systems follow the centralized model. Users give their private information by registering on these systems and the information is stored by the systems in their centralized databases which is insecure. For buying and selling people have to rely on a third party, and if the central server goes down, the whole website goes down. For online payments, people rely on banks, which hold cash of user. If banks are compromised, all user's cash can go to zero. Moreover, websites have private data of users, which is not a good thing. These websites can give away private data of user to other parties for business purposes.

1.2 Proposed Solution

Auction House Using Smart Contracts is designed to make web-based system which is decentralized and distributed using blockchain technology. System will help users to sell their items on website without giving away their private data such as username, email address, etc. Moreover, this system will use cryptocurrency for buying and selling of auctions. Auction House Using Smart Contracts uses Ethereum protocol. Ethereum is a software running on a network of computers that ensures that data and small computer programs called smart contracts are replicated and processed on all the computers on the network, without a central coordinator. Ethereum has a blockchain, which contains blocks of data (transactions and smart contracts). The blocks are created or mined by some participants and distributed to other participants who validate them. The web-based system is designed in such a way that it removes the involvement of third party like banks for payment of item bought by the user and the need of storing user's private data in database. This system utilizes blockchain technology to store all the transactions ever executed by users in a trustless environment. User is hidden behind a wallet address which is the public key of user and it is used to sign transactions performed following the rules of public key cryptography. User pays by using cryptocurrency residing on his wallet. Smart contracts are used to make agreements between two parties during buying and selling of auctions. Auction House Using Smart Contracts removes the need of banks or any other third party to perform payments.

1.3 Scope

The system should provide major functionalities such as add item for create auction, time limit of created auction, place bid, reveal bid, finalize auction, display the currently running auctions and multisignature escrow service. The system should allow users to perform all activities of Auction House Using Smart Contracts. The system should put cost on the user for creating new auctions. The system should display the ethereum status that is users address, total balance they have in their wallet. The system should allow users to reveal their bids they placed on auction. The system should allow users to display the current status of their item they have placed for bidding. The system should only allow a third person (except buyer and seller) to finalize the auction after all the bids have been revealed. The person who finalized the auction will act as a third party in multisignature escrow service to confirm whether item has been sent to seller or not. The system should take the ownership of item from user and give the ownership to smart contract while it is being auctioned. If nobody buys the item, the ownership should come back to its original owner. This system should only be used by users who have created accounts for decentralized applications.

1.4 Objectives

The primary objective of the system is to create a decentralized web application using smart contracts. Smart contracts will be stored on block-chain. This system will remove the need of third party for storing private data of users. The users will perform transactions using their public key. User will access the whole functionality of the system. This type of system will increase security because it will be based on public key cryptography which will secure all transactions. If any of the node on the network fails, it will not have any effect on the functionality of the system. Users will buy items using ethereum software currency which is ether.

1.5 Project Organization

This section explains about which process model will be followed, what are major roles and responsibilities and which tools and techniques will be used in order to develop the system. Its main reason is to create an environment that fosters interactions among the team members with a minimum amount of disruptions, overlaps, and conflict.

1.5.1 Software Process Model

The software process model to be followed is Evolutionary Spiral Model. The reason behind choosing this model is because of a new technology being used, student has no previous experience on this technology and requirements can change with time because smart contracts technology is still evolving, there might be requirements which cannot be fulfilled at current stage, so it is reasonable to use this approach. This approach is quite suitable as compared to other approaches of software process model as waterfall model is used when requirements are well understood. The incremental approach is used when requirements are well understood and increments can easily be defined. Evolutionary prototyping model is used when requirements are not well understood. Rapid Application Development model is used when the system is divided in modules. The approach of component-based development follows when quality components are available. Spiral model is basically a risk driven process model generator for software projects. The basic steps followed in this model are; plan, determine goals, evaluate risks, develop and test. The advantages of using this approach are; high amount of risk analysis hence, avoidance of risk is enhanced, additional functionality can be added at a later date. The main disadvantage of using this approach is; project's success is highly dependent on the risk analysis phase.

1.5.2 Roles and Responsibilities

The "Auction House Using Smart Contracts" is a single student project. Perform communication with supervisor for a clear understanding of requirements. Meet the stakeholders for requirement gathering. The supervisor has very important role in refinements of requirement and testing of the system according to the given requirements. Implementation of the project according to requirements and verify that system. Test the system and finally deployment of that system for the end users. The following are the roles and responsibilities of the student formulation are; project plan, requirements specification, analysis, architecture specifications, component or object specification, source code, test plan and final deliverable.

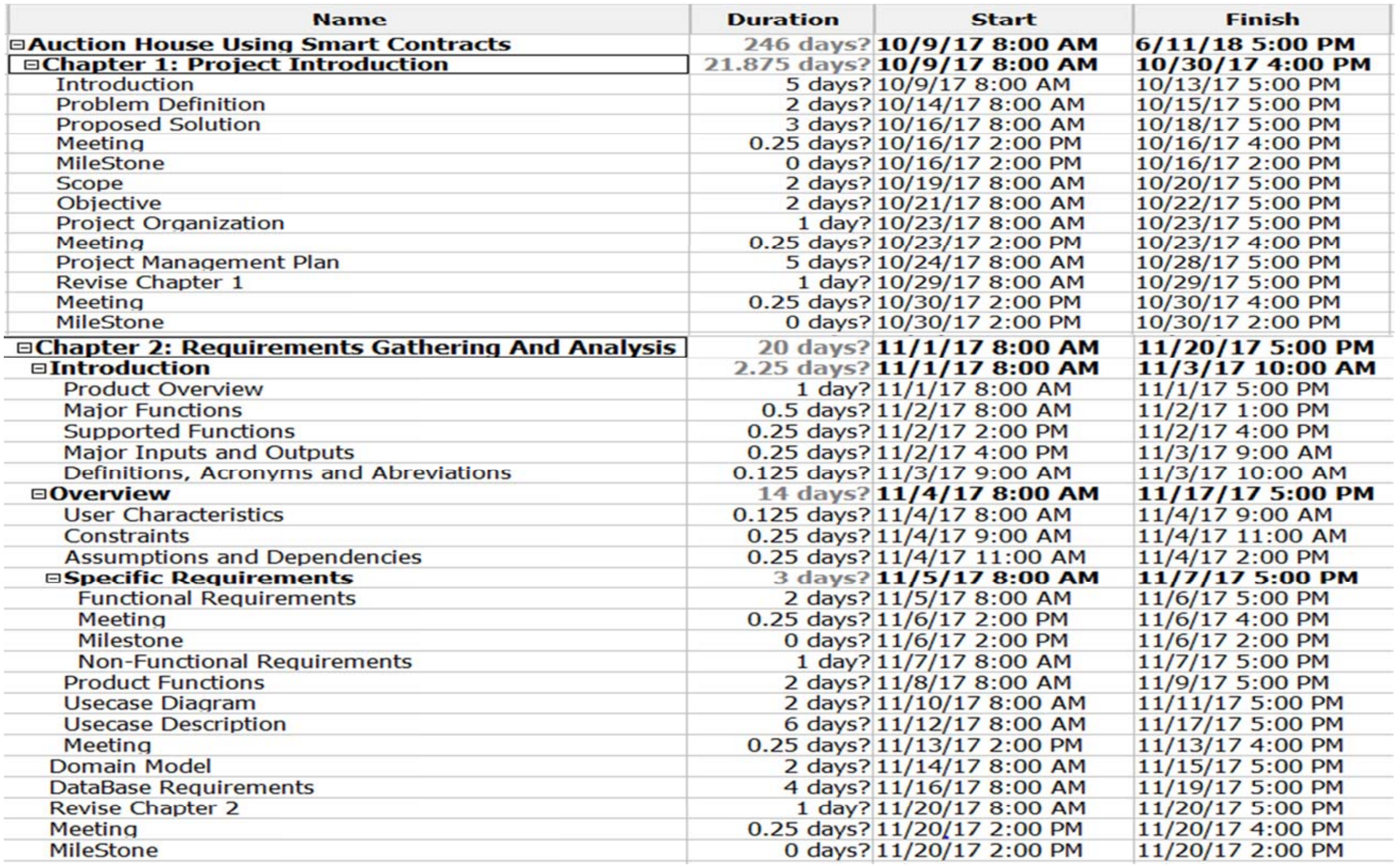
1.5.3 Tools and Techniques

Tools that are used for implementation of this system are visual studio code, Remix-Solidity IDE, Ganache blockchain service, Meta-Mask, truffle test network, mongodb, Inter Planetary File System. Argo UML tool and Microsoft Visio for UML diagrams such as use case diagram, class diagram, activity diagram, domain model and Entity relationship diagram and for writing documentation Microsoft word is used. For designing a plan of the system, project libre is used.

The smart contracts will be developed using Solidity programming language. Html will be used to create front end of application. CSS will be used as cascading style sheets to style the contents. Bootstrap, JavaScript, JQuery will also be used.

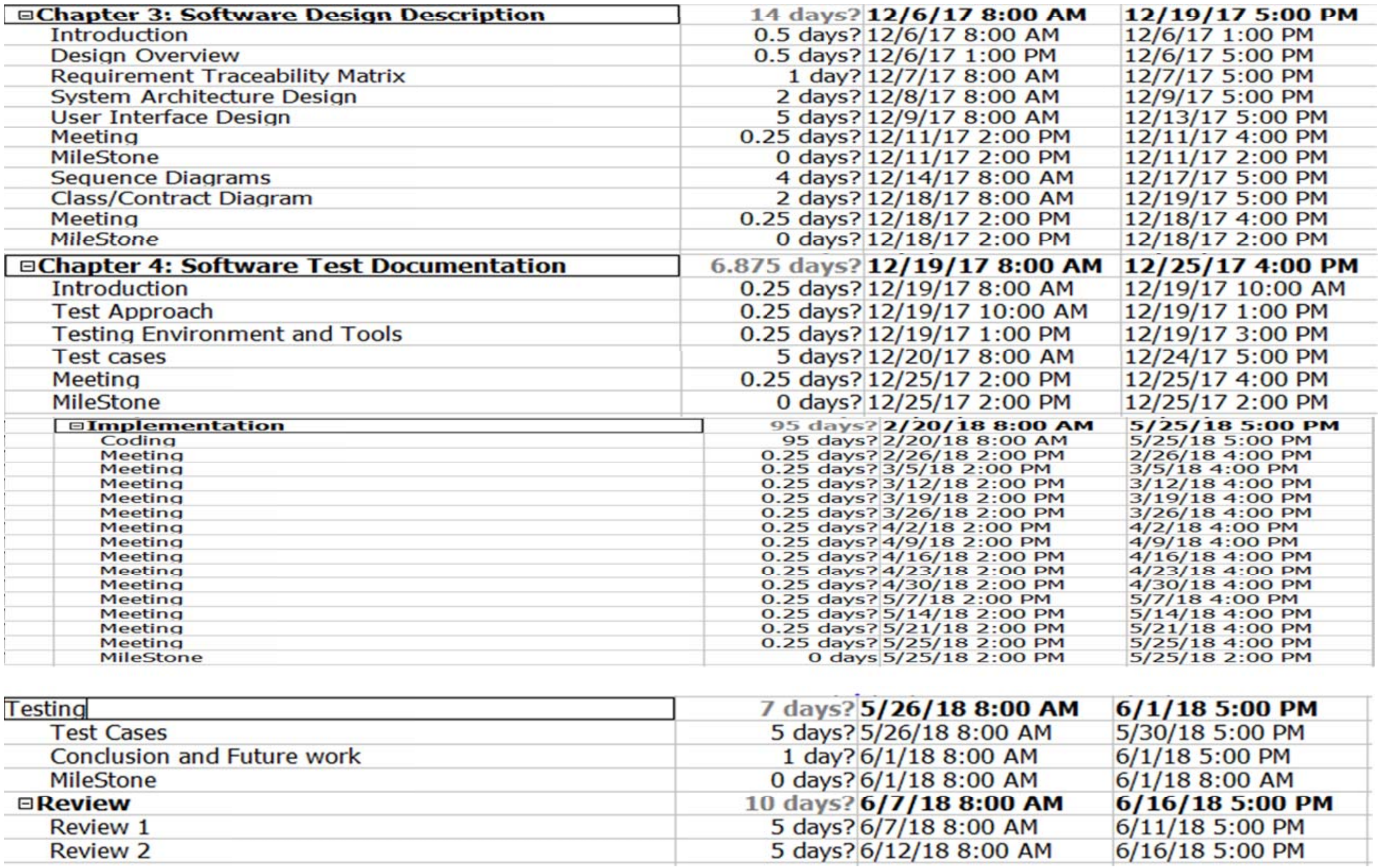
1.6 Project Management Plan

Table 1.1 (1/2) Project Management Plan



Name	Duration	Start	Finish
⊟**Auction House Using Smart Contracts**	246 days?	**10/9/17 8:00 AM**	**6/11/18 5:00 PM**
⊟**Chapter 1: Project Introduction**	21.875 days?	**10/9/17 8:00 AM**	**10/30/17 4:00 PM**
Introduction	5 days?	10/9/17 8:00 AM	10/13/17 5:00 PM
Problem Definition	2 days?	10/14/17 8:00 AM	10/15/17 5:00 PM
Proposed Solution	3 days?	10/16/17 8:00 AM	10/18/17 5:00 PM
Meeting	0.25 days?	10/16/17 2:00 PM	10/16/17 4:00 PM
MileStone	0 days?	10/16/17 2:00 PM	10/16/17 2:00 PM
Scope	2 days?	10/19/17 8:00 AM	10/20/17 5:00 PM
Objective	2 days?	10/21/17 8:00 AM	10/22/17 5:00 PM
Project Organization	1 day?	10/23/17 8:00 AM	10/23/17 5:00 PM
Meeting	0.25 days?	10/23/17 2:00 PM	10/23/17 4:00 PM
Project Management Plan	5 days?	10/24/17 8:00 AM	10/28/17 5:00 PM
Revise Chapter 1	1 day?	10/29/17 8:00 AM	10/29/17 5:00 PM
Meeting	0.25 days?	10/30/17 2:00 PM	10/30/17 4:00 PM
MileStone	0 days?	10/30/17 2:00 PM	10/30/17 2:00 PM
⊟**Chapter 2: Requirements Gathering And Analysis**	20 days?	**11/1/17 8:00 AM**	**11/20/17 5:00 PM**
⊟**Introduction**	2.25 days?	**11/1/17 8:00 AM**	**11/3/17 10:00 AM**
Product Overview	1 day?	11/1/17 8:00 AM	11/1/17 5:00 PM
Major Functions	0.5 days?	11/2/17 8:00 AM	11/2/17 1:00 PM
Supported Functions	0.25 days?	11/2/17 2:00 PM	11/2/17 4:00 PM
Major Inputs and Outputs	0.25 days?	11/2/17 4:00 PM	11/3/17 9:00 AM
Definitions, Acronyms and Abreviations	0.125 days?	11/3/17 9:00 AM	11/3/17 10:00 AM
⊟**Overview**	14 days?	**11/4/17 8:00 AM**	**11/17/17 5:00 PM**
User Characteristics	0.125 days?	11/4/17 8:00 AM	11/4/17 9:00 AM
Constraints	0.25 days?	11/4/17 9:00 AM	11/4/17 11:00 AM
Assumptions and Dependencies	0.25 days?	11/4/17 11:00 AM	11/4/17 2:00 PM
⊟**Specific Requirements**	3 days?	**11/5/17 8:00 AM**	**11/7/17 5:00 PM**
Functional Requirements	2 days?	11/5/17 8:00 AM	11/6/17 5:00 PM
Meeting	0.25 days?	11/6/17 2:00 PM	11/6/17 4:00 PM
Milestone	0 days?	11/6/17 2:00 PM	11/6/17 2:00 PM
Non-Functional Requirements	1 day?	11/7/17 8:00 AM	11/7/17 5:00 PM
Product Functions	2 days?	11/8/17 8:00 AM	11/9/17 5:00 PM
Usecase Diagram	2 days?	11/10/17 8:00 AM	11/11/17 5:00 PM
Usecase Description	6 days?	11/12/17 8:00 AM	11/17/17 5:00 PM
Meeting	0.25 days?	11/13/17 2:00 PM	11/13/17 4:00 PM
Domain Model	2 days?	11/14/17 8:00 AM	11/15/17 5:00 PM
DataBase Requirements	4 days?	11/16/17 8:00 AM	11/19/17 5:00 PM
Revise Chapter 2	1 day?	11/20/17 8:00 AM	11/20/17 5:00 PM
Meeting	0.25 days?	11/20/17 2:00 PM	11/20/17 4:00 PM
MileStone	0 days?	11/20/17 2:00 PM	11/20/17 2:00 PM

Table 1.1(2/2) Project Management Plan



⊟**Chapter 3: Software Design Description**	14 days?	**12/6/17 8:00 AM**	**12/19/17 5:00 PM**
Introduction	0.5 days?	12/6/17 8:00 AM	12/6/17 1:00 PM
Design Overview	0.5 days?	12/6/17 1:00 PM	12/6/17 5:00 PM
Requirement Traceability Matrix	1 day?	12/7/17 8:00 AM	12/7/17 5:00 PM
System Architecture Design	2 days?	12/8/17 8:00 AM	12/9/17 5:00 PM
User Interface Design	5 days?	12/9/17 8:00 AM	12/13/17 5:00 PM
Meeting	0.25 days?	12/11/17 2:00 PM	12/11/17 4:00 PM
MileStone	0 days?	12/11/17 2:00 PM	12/11/17 2:00 PM
Sequence Diagrams	4 days?	12/14/17 8:00 AM	12/17/17 5:00 PM
Class/Contract Diagram	2 days?	12/18/17 8:00 AM	12/19/17 5:00 PM
Meeting	0.25 days?	12/18/17 2:00 PM	12/18/17 4:00 PM
MileStone	0 days?	12/18/17 2:00 PM	12/18/17 2:00 PM
⊟**Chapter 4: Software Test Documentation**	6.875 days?	**12/19/17 8:00 AM**	**12/25/17 4:00 PM**
Introduction	0.25 days?	12/19/17 8:00 AM	12/19/17 10:00 AM
Test Approach	0.25 days?	12/19/17 10:00 AM	12/19/17 1:00 PM
Testing Environment and Tools	0.25 days?	12/19/17 1:00 PM	12/19/17 3:00 PM
Test cases	5 days?	12/20/17 8:00 AM	12/24/17 5:00 PM
Meeting	0.25 days?	12/25/17 2:00 PM	12/25/17 4:00 PM
MileStone	0 days?	12/25/17 2:00 PM	12/25/17 2:00 PM
⊟**Implementation**	95 days?	**2/20/18 8:00 AM**	**5/25/18 5:00 PM**
Coding	95 days?	2/20/18 8:00 AM	5/25/18 5:00 PM
Meeting	0.25 days?	2/26/18 2:00 PM	2/26/18 4:00 PM
Meeting	0.25 days?	3/5/18 2:00 PM	3/5/18 4:00 PM
Meeting	0.25 days?	3/12/18 2:00 PM	3/12/18 4:00 PM
Meeting	0.25 days?	3/19/18 2:00 PM	3/19/18 4:00 PM
Meeting	0.25 days?	3/26/18 2:00 PM	3/26/18 4:00 PM
Meeting	0.25 days?	4/2/18 2:00 PM	4/2/18 4:00 PM
Meeting	0.25 days?	4/9/18 2:00 PM	4/9/18 4:00 PM
Meeting	0.25 days?	4/16/18 2:00 PM	4/16/18 4:00 PM
Meeting	0.25 days?	4/23/18 2:00 PM	4/23/18 4:00 PM
Meeting	0.25 days?	4/30/18 2:00 PM	4/30/18 4:00 PM
Meeting	0.25 days?	5/7/18 2:00 PM	5/7/18 4:00 PM
Meeting	0.25 days?	5/14/18 2:00 PM	5/14/18 4:00 PM
Meeting	0.25 days?	5/21/18 2:00 PM	5/21/18 4:00 PM
Meeting	0.25 days?	5/25/18 2:00 PM	5/25/18 4:00 PM
MileStone	0 days	5/25/18 2:00 PM	5/25/18 2:00 PM
Testing	7 days?	**5/26/18 8:00 AM**	**6/1/18 5:00 PM**
Test Cases	5 days?	5/26/18 8:00 AM	5/30/18 5:00 PM
Conclusion and Future work	1 day?	6/1/18 8:00 AM	6/1/18 5:00 PM
MileStone	0 days?	6/1/18 8:00 AM	6/1/18 8:00 AM
⊟**Review**	10 days?	**6/7/18 8:00 AM**	**6/16/18 5:00 PM**
Review 1	5 days?	6/7/18 8:00 AM	6/11/18 5:00 PM
Review 2	5 days?	6/12/18 8:00 AM	6/16/18 5:00 PM

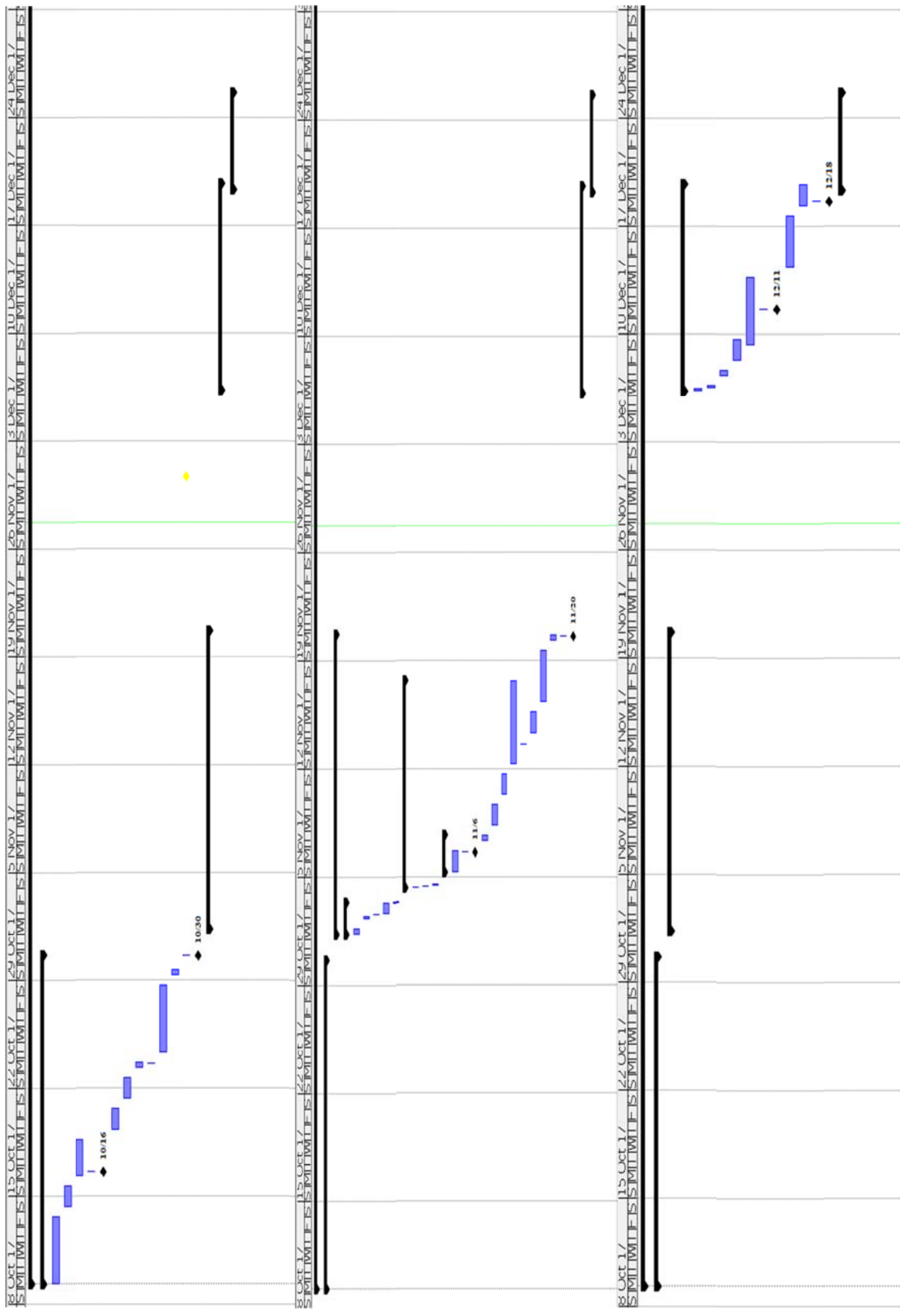


Figure 1.1(1/2) Project Gantt Chart

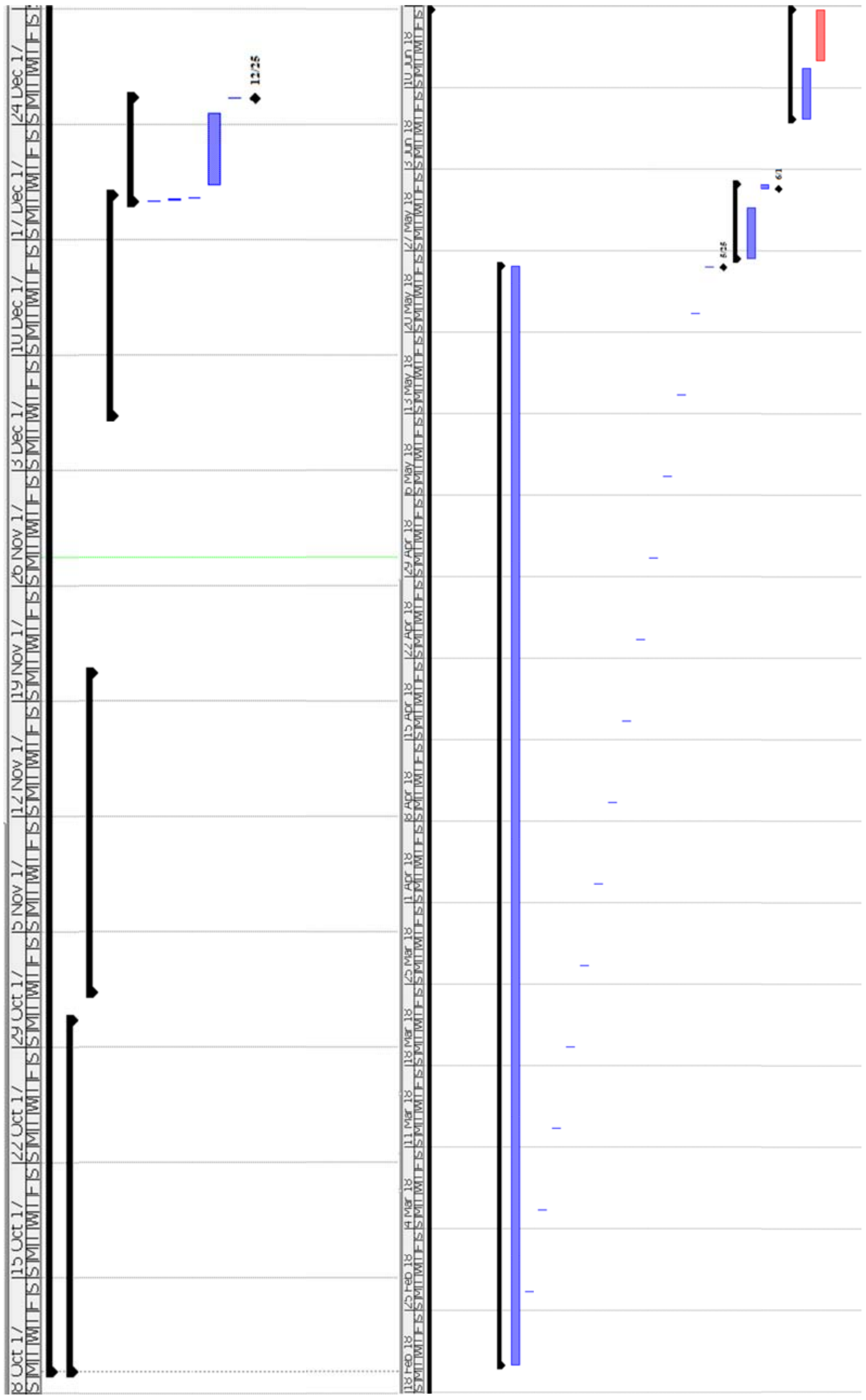


Figure 1.1 (2/2) Project Gantt Chart

1.6.1 Project Deliverables

- Software Process Management Plane
- Software Requirements Specification
- Software Design Description
- Software Test Documentation

1.6.2 Risks and Contingencies

Smart Contracts technology is still evolving, some of the features like uploading image in blockchain and displaying it might not be implementable.

1.7 Report Structure

Chapter 1 has briefly described the introduction of the system, what are actual problem and proposed solution, its scope, objective, the organization of project, and project management plan. Chapter 2 describes what the blockchain technology is. Chapter 3 describes the requirements of the system and to decide what the system should do and what the system should not do. Briefly describes complete details of functional and non-functional requirements specifications for the system. Chapter 3 gives the description of software design. It entails the description about the chosen architecture design, user interface design. Briefly describes the interaction and relation between the between actor and system objects through sequence and class diagram. Chapter 4 gives the description of the testing of the system. Briefly describe the test approaches, test plan, testing tool and environment and finally the test cases of the system.

“Employ your time in improving yourself by other men’s writings, so that you shall gain easily what others have laboured hard for.” *Socrates (470 BC – 399 BC)*

Chapter 2

Ethereum and Technologies Used in this Project

The purpose of this chapter is to give a short introduction to Ethereum. It also gives introduction to tools and technologies used in this project.

2.1 The Origin of Ethereum

Bitcoin blockchain is primarily used for sending money between various parties on the blockchain without the need for a central authority like bank. A 19 year old developer by name Vitalik Buterin wanted to apply idea of decentralization (no central authority) to more than just money transfer. He wanted to build applications that could run globally without any central authority in control. For example, if you are a Facebook user, Facebook as a company owns your data and they have the right to remove your account if they wish. In 10 years, Facebook could shut down in which case all your data could be lost.

Vitalik Buterin proposed to add a scripting language to Bitcoin to be able to build applications. However, after failing to gain agreement from the Bitcoin development team, in January 2014, he officially published the white paper proposing the development of a new platform with a more general scripting language. A team formed shortly and one of the developers, Dr. Gavin Wood soon released the Ethereum yellow paper, which covered the Ethereum Virtual Machine (EVM), the runtime environment that executes all of the smart contracts. The development of the platform was funded through a crowdsale in July–August 2014, with the participants buying the Ethereum value token (ether).

2.2 Global Computer

Ethereum is a public, blockchain based distributed computing platform. It can be thought of as one big computer made up of small computers around the world. You can write applications and run them on this global computer. The platform guarantees that your application will always run without any downtime, censorship, fraud or third-party interference. Apart from running applications, Ethereum blockchain can also transfer money between 2 parties without a central authority.

All these computers (also called nodes) are connected to one another and have a full copy of the code and data**.** When you deploy your code on to the Ethereum blockchain, the code is replicated across all the nodes in the network. When your application stores any data, even that data is replicated across all the nodes. There are thousands of nodes in the network and it is almost impossible for anyone to stop all the nodes. This ensures your application to be always accessible.

2.3 Client Server Architecture VS Ethereum Architecture

One of the best ways to understand Ethereum is by comparing it with a traditional client/server architecture, using example of web application.

A typical web application consists of server side code which is usually written in a programming language like Java, C#, Ruby, Python. The frontend code is implemented using HTML/CSS/JavaScript. This entire application is then hosted on a hosting provider like Microsoft Azure, Google Cloud Platform.

Users interact with the web application using a client such as web browser or through an API. Note that there is one web application which is centralized and all the clients interact with this one application**.** When a client makes a request to the server, the server processes the request, interacts with the database and/or cache, reads/writes/updates the database and returns a response to the client.

In Ethereum, there is no central server to which all clients connect to. This means, in an ideal decentralized world, every person who wants to interact with a dapp (Decentralized Application) will need a full copy of the blockchain running on their computer/phone etc. That means, before you can use an application, you have to download the entire blockchain and then start using the application.

We don't live in an ideal world and it is unreasonable to expect everyone to run a blockchain server to use these apps. But the idea behind decentralization is to not rely on a single/centralized server. So, the community has come up with solution like Meta-Mask, where you don't have to spend lot of your hard disk and RAM downloading and running a full copy of the blockchain.

The Ethereum blockchain has 2 main components:

1. **Database**: Every transaction in the network is stored in the blockchain. When you deploy your application, it is considered as a transaction. If you have for example a Voting application that allows anyone to vote for candidates, a vote for a candidate would be considered a transaction. All these transactions are public and any one can see this and verify. This data can never be tampered with. To make sure all the nodes in the network have same copy of the data and to ensure no invalid data gets written to this database, Ethereum uses an algorithm called Proof of Work to secure the network.

2. **Code**: The database aspect of blockchain just stores the transactions. But where is all the logic to vote for candidate, retrieve the total votes etc. In Ethereum world, developer write the logic/application code (called contract) in a language called Solidity. The developer then uses the solidity compiler to compile it to Ethereum Byte Code and then deploy that byte code to the blockchain (There are few other languages that are used to write contracts but solidity is by far the most popular and relatively easier option). So, not only does Ethereum blockchain store the transactions, it also stores and executes the contract code.

So basically**,** the blockchain stores your data, stores the code and also runs the code in the EVM (Ethereum Virtual Machine).

To build web based Dapps, Ethereum comes with a handy javascript library called web3.js which connects to blockchain node. So, one can just include this library in any framework like reactjs, angularjs etc and start building.

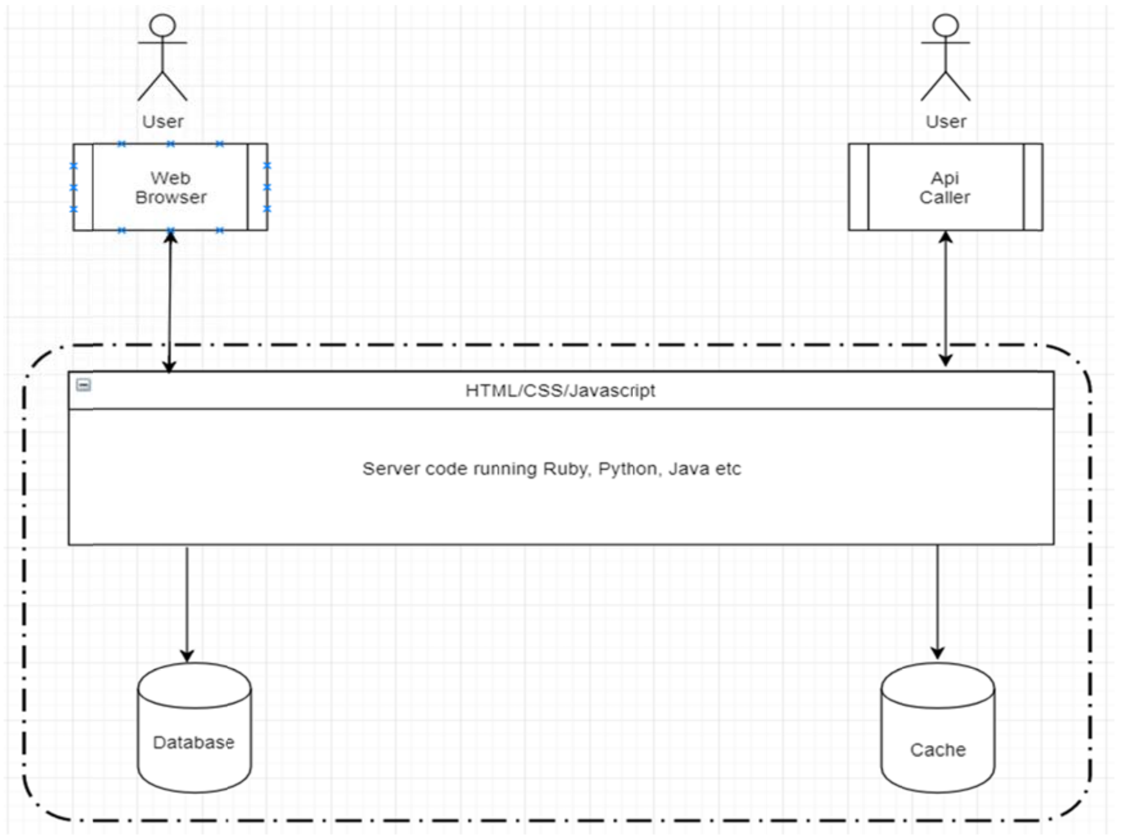


Figure 2.1 High level View of Client/Server Architecture

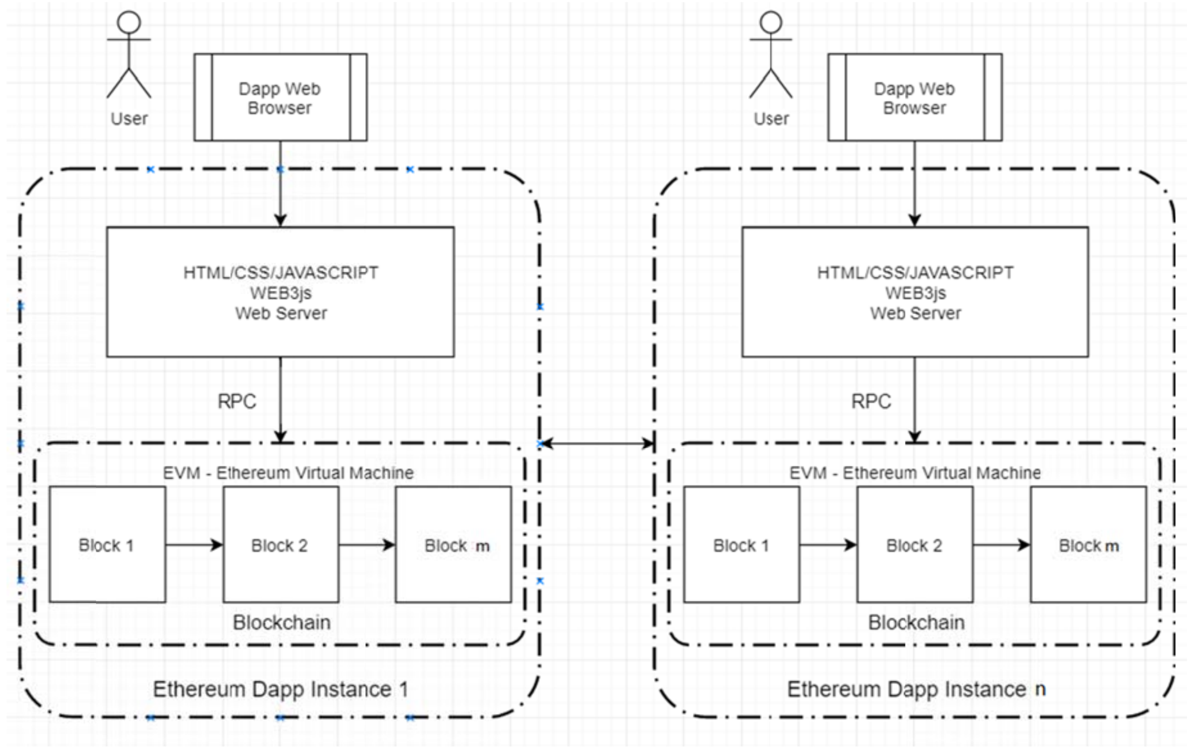


Figure 2.2 Ethereum Architecture

2.4 Ethereum Concepts

There are concepts used in Ethereum which should be understood.

2.4.1 Smart Contract

In Ethereum applications are developed using solidity programming language and then they are deployed on the blockchain. These applications are called smart contracts.

In general, contract is a written agreement between two or many parties that is intended to be enforced by law**.** If we take this written contract and translate it in to code and deploy on the blockchain, we get digital contracts. But the true power of this code on the blockchain is that it can enforce the agreement between parties and that is the reason they are called "smart contracts". Once a contract is deployed to the blockchain, it can neither be stopped nor modified. That is how the agreement is enforced.

2.4.2 Ether and Denominations

In the real world, each country has its own currency like USD, INR, RNB, GBP, EUR, PKR etc, each blockchain has its own currency. In the case of Ethereum blockchain, the native currency is called Ether. There are exchanges where you can convert Ether to any other fiat currency like USD or EUR.

In the real world, currencies have various denominations. For example, a US Dollar is equal to 100 cents and it has various denominations such as pennies (1 cent), nickel (5 cents), dime (10 cents), quarter (25 cents). Depending on your country and currency, you probably have various denominations as well.

Ether also has various denominations. The only two a developer or investor should keep in mind are Ether and Wei. Wei is the lowest denomination and this is the denomination used in smart contracts.

2.4.3 Addresses

To login to any website like Facebook, you usually use an email/username and password. Your username is your identity in Facebook and you use your username/password to authenticate with Facebook.

In Ethereum blockchain, address is your identity. An Ethereum address looks like this:

001d3f1ef827552ae1114027bd3ecf1f086ba0f9. An address has a corresponding private key. An address is also called a public key. One can think of private key as a password that only you know. You need this pair of address + private key to interact with the blockchain. Few key things should be kept in mind:

1. Ethereum address is public and you can share it with anyone in the world.
2. The private key should never ever be shared with anyone.
3. The address and private key are not stored in any database. Only you are in control of them.

2.4.4 Gas

One can deploy contracts on the Ethereum blockchain and execute transactions on it. However, there is a cost associated with each interaction. You have to pay Ether to the miners in the network to execute a transaction on the blockchain.

Who decides how much Ether to pay for a transaction? The answer is, the yellow paper has specification on how many units of work a transaction has. For example, if your transaction is to simply add two numbers, that is 3 units of work. If it is multiplication, that would be 5 units of work and so on. This unit of work is called gas.

2.4.5 Byte Code

Smart contract code is usually written in a high level programming language such as Solidity. This code gets compiled to something called the EVM bytecode which gets deployed to the Ethereum blockchain. This is very similar to a programming language like Java where the code gets converted to JVM Byte code. The Ethereum runtime environment only understands and can execute the bytecode.

One of the benefits of this design is that it gives developers option to use other programming languages to implement smart contracts. Currently, there are a handful of languages like Vyper (similar to Python) that compiles down to the EVM bytecode.

2.4.6 Ethereum Virtual Machine

The Ethereum Virtual Machine (EVM) is a simple but powerful, Turing complete 256bit Virtual Machine that allows anyone to execute arbitrary EVM Byte Code. The EVM is part of the Ethereum Protocol and plays a crucial role in the consensus engine of the Ethereum system. It allows anyone to execute arbitrary code in a trust-less environment in which the outcome of an execution can be guaranteed and is fully deterministic.

When you install and start the geth, parity or any other client, the EVM is started and it starts syncing, validating and executing transactions.

2.5 Tools and Technologies

2.5.1 Geth/Parity Clients

Geth is the official client software provided by the Ethereum Foundation. It is written in the Go programming language. When you start geth, it connects to other Ethereum clients (also called nodes) in the network and downloads a copy of the blockchain. It will constantly communicate with other nodes to keep it's copy of the blockchain up to date. It also has the ability to mine blocks and add transactions to the blockchain, validate the transactions in the block and also execute the transactions. It also acts as a server by exposing APIs you can interact with through RPC. It also comes with **a** JavaScript client (geth console) that can be used to connect to the blockchain.

2.5.2 Web3js and Truffle

Web3js is a very popular JavaScript library that is used to interact with the Ethereum blockchain. One can use this JavaScript library in any frontend framework to build a user facing decentralized application**.**

Just like we have frameworks for web application development such as Ruby on Rails, Python/Django etc, Truffle is one of the most popular frameworks used to develop dapps. They abstract away lot of the complexities of compiling and deploying your contract on the blockchain. It also has an in-built testing framework which one can use to test your contracts.

2.5.3 Ganache

Developers usually use an in-memory blockchain called ganache for development of Dapps. One can either install a command line version called ganache-cli or the GUI version. Another nice thing about ganache is that you get 10 test accounts loaded with 100 Ether for your testing**.**

2.5.4 Meta-Mask

MetaMask is a bridge that allows you to visit the decentralized web of tomorrow in your browser today. It allows you to run Ethereum decentralized applications right in your browser without running a full Ethereum node. MetaMask includes a secure identity vault, providing a user interface to manage your identities on different sites and sign blockchain transactions. Metamask is just a Developer Preview right now, and has not been released to the general public. So, it is recommended not to put serious funds in it, but use it to help prepare your decentralized applications.

2.5.5 Remix

Remix is a browser IDE you can use to code your smart contracts. Not only can you use it to as an editor but it can be used to compile and deploy your contracts to various networks and interact with them directly from the IDE. It has many features to select various compiler versions, debug your contracts and so on.

2.5.6 Inter Planetary File System

Interplanetary File System (IPFS) is a peer-to-peer distributed file system that seeks to connect all computing devices with the same system of files. It also is a protocol designed to create a permanent and decentralized method of storing and sharing files.

Just like anyone can run an Ethereum node, anyone can run an IPFS node and join the network to form a global file system. The files are replicated across many nodes and it is almost impossible to lose access to your files and is also censorship resistant. This is similar to how your contract and data associated with it is stored on all the Ethereum nodes across the network.

In this project large data files like image and description will be stored in IPFS and their hash in blockchain.

2.5.7 MongoDB

MongoDB is a free and open source cross platform documented oriented database program. Classified as a no No-Sql database program. In MongoDB records can be replicated on multiple nodes. In this project MongoDB will be used to store products data and their hashes on blockchain to maintain integrity of the system. Products will be queried and displayed through MongoDB instead of blockchain in order to reduce the burden on the blockchain for querying thousands of products again and again.

“Education is the key to unlock the golden door of freedom” George Washington(1860-1943)

Chapter 3

Requirement Gathering and Analysis

The purpose of the requirement gathering and analysis phase is to clear the requirements of the system and to decide what the system should do and what the system should not do. To clear the requirements like function and non-functional requirements for the system and understand the major inputs and outputs for the system.

3.1 Product Overview

“Auction House Using Smart Contracts” is a system for buying and selling any type of item using vickery auction. This system is basically a web-based application. It can run on every system. This system will be implemented using ethereum software. The user can use this system for different purposes such as buying and selling. The user will be able to buy or sell an item by placing bid which will be ether. The highest bidder will get the item and the rest of the bidders will be able to get their ether back which they placed for bidding. Placing an item for selling will cost the seller some ether. System is decentralized so database will be on every computer which is block-chain. Users can create their account(s) using MetaMask or mist browser. The account of the user will contain the users ether and address (public key). Private key stays on user’s computer.

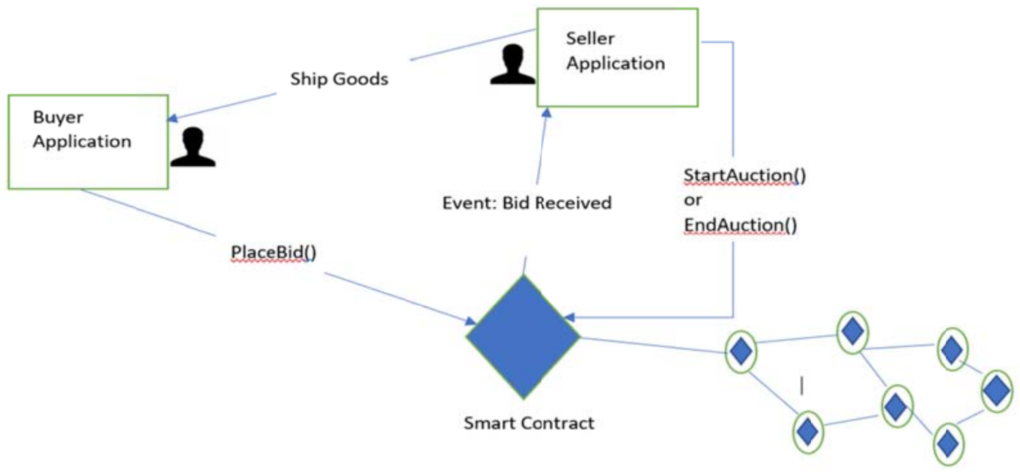


Figure 3.1 Product High Level View

3.2 Major Functions

Winner of the auction will pay the amount proposed by second highest bidder. After winner is decided, amount will be stored in a contract. Amount can only be unlocked when the contracts requirement is complete. If nobody places bid on item, auction will end without a winner. Moreover, displaying the ethereum status which contains; user network address, amount in ether user has. Display the user his active auctions and other active auctions placed by other users on the system.

3.3 Major Inputs and outputs

The user gives some inputs to system and system will generate the response to those inputs respectively.

3.3.1 Major Inputs

User enters item details. User enters description of item, image, starting price in ether, item deadline in the form of date and time, then selects create auction. User selects start auction to place the item for bidding. User cancels the auction by selecting cancel auction button.

3.3.2 Major Outputs

User's active auction and other users' auctions are displayed. Bid successfully placed message is displayed.

3.4 Definitions, Acronyms and Abbreviations

Table 3.1 Abbreviations

Smart Contracts	A set of promises, specified in digital form, including protocols within which the parties perform on these promises.
Solidity	Programming language for smart contracts
BlockChain	A massive distributed database which lives on every computer
Ethereum	Ethereum is a software running on a network of computers that ensures that data and small computer programs called smart contracts are

	replicated and processed on all the computers on the network, without a central coordinator. The vision is to create an unstoppable censorship-resistant self-sustaining decentralized world computer.
Auction	An auction is a process of buying and selling goods or services by offering them up for bid, taking bids, and then selling the item to the highest bidder.
Eth/Ether	Ethereum’s inbuilt native cryptocurrency, used for paying for smart contracts to run.
Truffle	Truffle is a development environment, testing framework for ethereum.
MetaMask	MetaMask is a bridge that allows you to visit the distributed web of tomorrow in your browser today. It allows you to run ethereum decentralized applications right in your browser without running a full ethereum node.
Vickery Auction	A Vickrey auction is a type of sealed-bid auction. Bidders submit written bids without knowing the bid of the other people in the auction. The highest bidder wins but the price paid is the second-highest bid.

3.5 Overview

The rest of topics contain the detailed information about functional, non-functional, the overall functionality of the system, use cases and their description, domain model, and database explanation.

3.6 User Characteristics

Users of this system can be any person. This is assumed that all the users have basic knowledge of computer or laptop and knowledge of web application. Users must have knowledge of how to use web-based applications and can able to perform certain tasks. Users must know how to create ethereum wallet to access the decentralized application.

3.7 Constraints

Auction House Using Smart Contracts is a web-based system. It is not efficient to store large images and description about product in blockchain. For maintaining integrity store the hash of images and description in blockchain. Moreover querying the blockchain again and again for displaying products or applying search on products. So, use a separate database other than blockchain to handle this issue. In vickery auction people cannot see each other bid placed on product. Bids can only be seen when users reveal their bids after auction ends. In blockchain world all transactions are public, so bid placed by users will also be public and can be seen by other users. Implement a way to hide the bids so that no user can see others bids. Put a specific time in which users can reveal their bid. After the time is over no bids can be reveal anymore for the currently running auction.

3.8 Assumptions and Dependencies

This decentralized web-based application depends upon the availability of internet and having an ethereum account. It is assumed that the users have any computer or laptop to access the system.

3.9 Specific Requirements

3.9.1 Functional Requirements

Functional requirements are the software capabilities that must be present in order for the user to carry out the services provided by the system, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. The system allows user to access the system who have ethereum account with some ether plus connected to any test network like ropsten or any other test network. The system gives the proper message of any invalid entry. Main Functional Requirements are:

1. Implement Vickery auction
2. Highest bidder i.e. winner of the auction will pay the amount equal to second highest bidder
3. Bid placed by users should not be visible to others
4. After auction ends all bidders must reveal their bids. Not revealing the bid will be considered as a loss for that user.
5. Users who lost the auction will get their invested amount back. In case if a person has not revealed his bid, his invested bid will not be sent back to his account.
6. Only a user except buyer and seller will finalize the auction after bid reveal time ends.
7. Implement a three-person voting multisignature escrow service in which two out of three people (buyer, seller and the person who finalized auction) will vote to either

send amount to seller of the product or send back the amount to buyer of the product in case of any fraud or any other situation which includes product related problems.

8. The person who finalized the auction can only participate in voting including buyer and seller.
9. Amount should only be transferred to its rightful honour when voting is complete.

3.9.2 External Interface Requirement

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication.

3.9.3 User Interfaces

Through user interface user will be able to interact with the system. This will be Web-based application for the user. User can use this application through internet. User would have ethereum account to access this application.

3.9.3.1 Create Auction

Input: Name, Description, image, Initial price, auction start and end time (using date and time)

Output: Auction created successfully message.

3.9.3.3 Place Bid

Input: Bid value

Output: Bid placed successfully message

3.10.4 Software Interfaces

Auction House Using Smart Contracts is decentralized web-based application and it will be implemented in Ethereum software therefore, this system can run on any operating system. The internet is required to access the system. The system can be accessed through browsers like google chrome, Firefox which support meta-mask plugin extension or mist browser.

3.10.5 Communication Protocol

Communication protocols required for this system are; Hypertext transfer protocol [http] for communication over the internet.

3.11 Software System Attributes

3.11.1 Reliability

System should be reliable. There should be no occurrence of the failure. The system should be able to work properly all-time means the extent to which it works as and when needed. The system should give the proper response to every query performed by user.

3.11.2 Availability

System should be available to every user at any time. All the users should able to access the system at any time as it does not depend upon the centralized server.

3.11.3 Security

Since the system is decentralized, all of user's private data is on his computer. System should be based on public key cryptography. User should only be able to access the system through user own credentials and any other user should not be able to access to the user private data. All transactions are signed by private key of user and transactions can be verified using user's public key to hold integrity in the system. User should not lose his private key otherwise his data will be stolen by hackers.

3.11.4 Maintainability

There should be aspect of maintainability for the system. In some cases, maintainability involves a system of continuous improvement learning from the past in order to improve the ability to maintain systems, or improve reliability of systems based on maintenance experience. The application should be easy to extend. The code should be written in a way that it favours implementation of new functions. In order for future functions to be implemented easily to the application. System is based on new technology so improvements will occur with time.

3.11.5 Portability

This is a web-based system so the main purpose of developing web based system is to improve the portability of system. To improve portability, system should run on variety of platforms and variety of connection speeds. System should be lightweight therefore that it can run on a machine with slow internet connection. To make the web application lightweight, simple libraries and tools should be used at developing phase.

3.11.6 Performance

Since this system is going to be decentralized and web based, one performance requirement is the storage space. Higher storage space means more space for blockchain and bigger workspace for user therefore higher the storage, better the performance. Performance requirement by the user side is, web application should be developed as a lightweight web application.

3.12 Product Functions

3.12.1 Add Item

User will give details about the item like name, image, description.

3.12.2 Create Auction

User will provide selling detail for item like starting bid, time limit for item being sold to the system.

3.12.3 Place Bid

User can place bid for item after seller has started the auction and before deadline is reached.

3.12.4 Display User Auctions

User can view his currently running auctions.

3.12.5 Display All Users Auctions

User can see all of the other users currently running auctions.

3.12.6 Reveal Bid

Users can reveal their bids given a specific time after auction time ends, so that winner can be decided.

3.12.7 Finalize Auction

Auction will be finalized after bid reveal time ends, then winner will be decided

3.12.8 Release amount to seller

User will vote to transfer amount to seller.

3.12.9 Refund amount to buyer

User will vote to refund amount to buyer

3.13 Use Case Diagram

List of use-cases mentioned in use case diagram are described in detail, therefore we are able to look more precisely that how user can interact with system to perform tasks. There will be only one user which is anonymous buyer as well as an anonymous seller meaning buyer and seller cannot know true identity of each other.

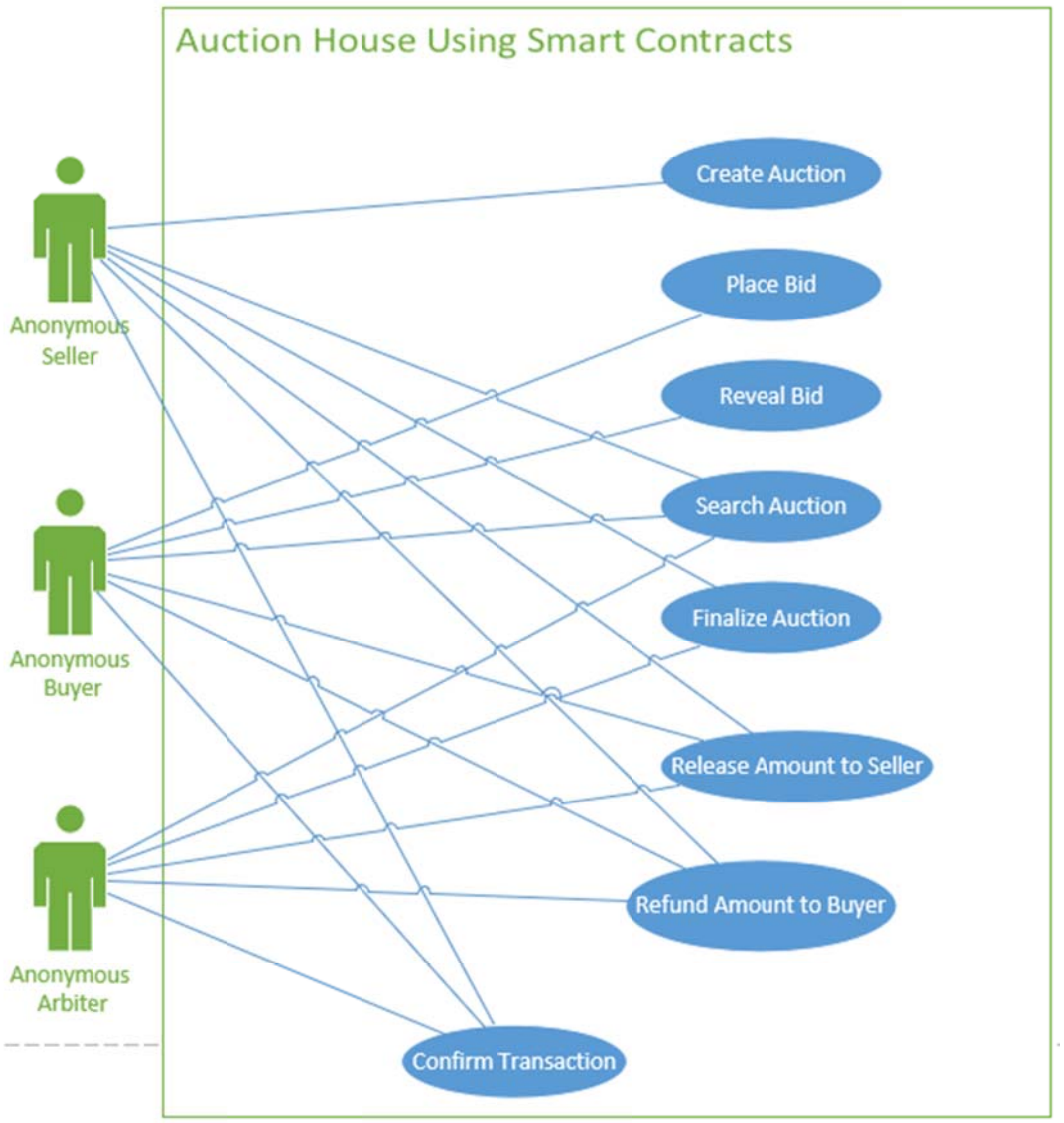


Figure 3.2 Usecase Diagram

3.14 Use Case Description

3.14.1 Use case 1: Create Auction

Table 3.2 Create Auction

ID	**UC1**
Name	Create Auction
Primary Actor	Anonymous Seller
Pre-Conditions	i. User has an ethereum account ii. User has ether in his account to make transaction iii. User is connected with any network instance associated with ethereum
Post-Conditions	i. Information about the item and his owner address is stored in the blockchain and other databases ii. Auction time Starts
Main Success Scenario	1. User enters data in the given field (item name, description, image, start Price) 2. User selects Create Auction 3. User provides ether for Create Auction transaction
Alternative flows or Extensions	1a**.** If user has less ether than ether required for transaction 1. Auction fail message will be displayed. 2a**.** If user selects add item before entering details 1. System gives error message and asks user to enter item details.
Frequency	Could be nearly continuous until user has enough ether

3.14.2 Use case 2: Place Bid

Table 3.3 Place Bid

ID	**UC2**
Name	Place Bid
Primary Actor	Anonymous Buyer
Pre-Conditions	i. User has an ethereum account ii. User has ether in his account to make transaction iii. User is connected with any network associated with ethereum iv. Auction time of item is not over
Post-Conditions	i. Bid is successfully placed and amount is deducted from user account ii. User bid is encrypted by the system
Main Success Scenario	1. User enters amount of bid he wants to send 2. User selects Place Bid
Alternative flows or Extensions	1a**.** If user has less ether than ether required for transaction 2. Auction fail message will be displayed. 2a**.** If user selects Place Bid before entering details 2. System gives error message and asks user to enter Bid
Frequency	Could be nearly continuous until user has enough ether

3.14.3 Use case 3: Reveal Bid

Table 3.4 Reveal Bid

ID	**UC3**
Name	Reveal Bid
Primary Actor	Anonymous Buyer
Pre-Conditions	1. User has placed bid on the item 2. User has enough ether to perform the transactions 3. Auction time is finished
Post-Conditions	i. User bid is revealed by the system
Main Success Scenario	1. User enter bid amount he sent earlier 2. User selects reveal bid
Alternative flows or Extensions	1a**.** if user does not have enough ether 1. Transaction will fail 1b**.** if user enters wrong bid amount 2. Transaction will fail
Frequency	Can be performed multiple times for a single item

There will be a specific time in which bidders can reveal their bid

3.14.4 Use case 4: Finalize Auction

Table 3.5 Finalize Auction

ID	**UC4**
Name	Finalize Auction
Primary Actor	Anonymous Arbiter
Pre-Conditions	i. User is not the buyer or seller of the product ii. Reveal Bid time is over
Post-Conditions	i. Auction is finalized and winner is decided ii. Users who did not win the auction will be sent back their invested bid iii. Escrow service is created which locks the amount in it.
Main Success Scenario	1. User selects finalize Auction
Alternative flows or Extensions	1a**.** if user does not have enough ether 1. Transaction will fail
Frequency	Only one time for single item

3.14.5 Use case 5: Release Amount to Seller

Table 3.6 Release Amount to Seller

ID	**UC5**
Name	Release Amount to Seller
Primary Actor	Anonymous Seller, Anonymous Buyer, Anonymous Arbiter
Pre-Conditions	1. amount is locked in Escrow
Post-Conditions	Amount from the Escrow is released
Main Success Scenario	1. User selects Release amount to seller option 2. Repeat step 1 until at least two of three users have voted to release amount to seller
Alternative flows or Extensions	1a. If nobody votes to release amount 1. Amount stays locked in the escrow
Frequency	Only one for a single item

3.14.6 Use case 6: Search Auction

Table 3.7 Search Auction

ID	**UC6**
Name	Search Auction
Primary Actor	Anonymous Buyer, Anonymous Seller, Anonymous Arbiter
Pre-Conditions	None
Post-Conditions	Searched auction is displayed to user
Main Success Scenario	1. User selects a category 2. Relevant item(s) are short listed 3. Item(s) are displayed to users
Alternative flows or Extensions	1a) If no item is found according to users search category 1. No items are displayed to user
Frequency	Could be nearly continuous

3.14.7 Use case 7: Refund Amount to Buyer

Table 3.8 Refund Amount to Buyer

ID	**UC7**
Name	Refund Amount to Buyer
Primary Actor	Anonymous Seller, Anonymous Buyer, Anonymous Arbiter
Pre-Conditions	1. Amount is locked in Escrow
Post-Conditions	i. Amount from the escrow has been released
Main Success Scenario	1. User selects Release amount to seller option 2. Repeat step 1 until at least two of three users have voted to release amount to seller
Alternative flows or Extensions	1a. If nobody votes to release amount 1. Amount stays locked in the escrow
Frequency	Only one time for single item

3.14.8 Use case 8: Confirm Transaction

Table 3.9 Confirm Transaction

ID	**UC8**
Name	Confirm Transaction
Primary Actor	Anonymous Seller, Anonymous Buyer, Anonymous Arbiter
Pre-Conditions	1. User has a account in Meta-Mask 2. User is accessing functionality of a Smart Contract
Post-Conditions	i. Transaction is processed
Main Success Scenario	1. User perfroms any functionality like Placing Bid or Creating Auction 2. Meta-Mask will show the cost to perform transaction 3. User can select to confirm or reject transaction
Alternative flows or Extensions	1a. If user selects Reject transaction 1. Transaction will be rejected
Frequency	Can occur as many times until user accesses the system to perform any functionality

3.15 Domain Model

Domain Model contains four objects, Anonymous User, Auctioned item, Bid and Escrow. Zero or many Items are sold by one user and one or many items get one or many bids. One or many items have only zero or one successful bid. Zero or many bids are placed by one user. Only three participants can vote in escrow to release amount in order to send it to its rightful honour. Object Bid contains attribute fake amount and encrypted amount for example, fake ether placed by user to confuse other buyers and encrypted ether which is the actual ether he placed on the item. Object item contains attributes name, image, description, category, starting price of item, and auction end time of item. Anonymous user contains attributes Ethereum_address which is user's public key and ether. Escrow contains information about highest bid value, buyer address (who is the winner of auction), seller address and arbiter address (the person who finalized the auction). These participants will vote to either release the amount to seller or refund the amount to buyer.

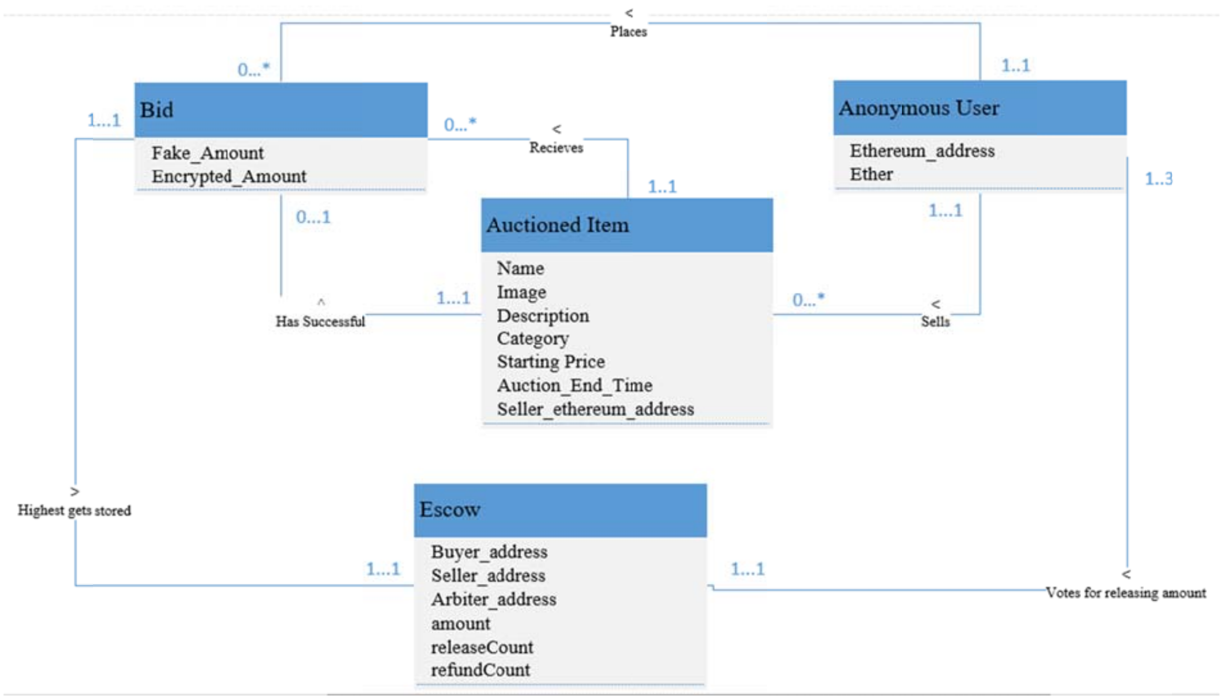
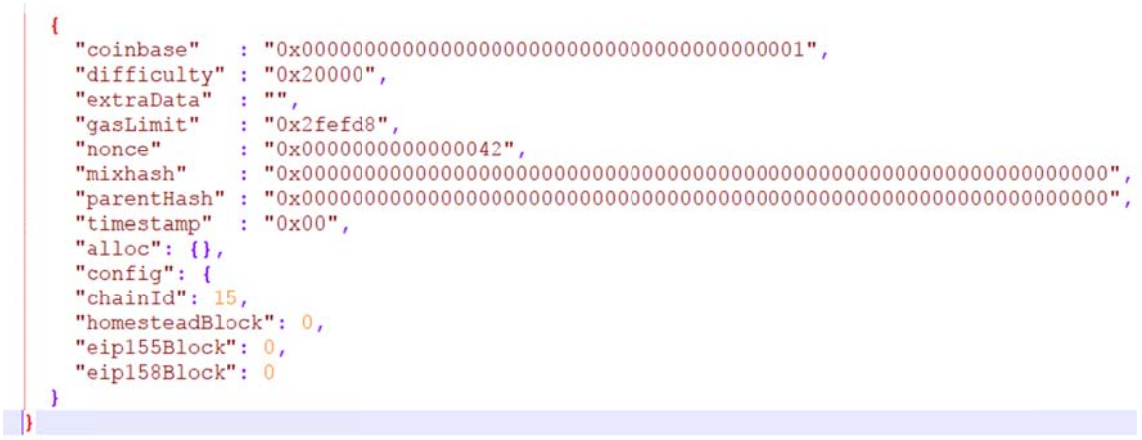


Figure 3.3 Domain Model

3.16 Database Requirements

System under development is based on the next web which is called web3. System is decentralized and it has a massive distributed database which is called blockchain. Blockchain is available on every computer, unlike the traditional databases which are installed on the server side. Working with decentralized applications, they assume that you have access to the blockchain on your computer. Ethereum downloads the whole chain ledger. In current scenario system will create its own block-chain using ganache network, later on application can be brought on ethereum block-chain to bring it live on the public network. Genesis Block of block-chain is:



```
{
  "coinbase"   : "0x0000000000000000000000000000000000000001",
  "difficulty" : "0x20000",
  "extraData"  : "",
  "gasLimit"   : "0x2fefd8",
  "nonce"      : "0x0000000000000042",
  "mixhash"    : "0x0000000000000000000000000000000000000000000000000000000000000000",
  "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp"  : "0x00",
  "alloc": {},
  "config": {
  "chainId": 15,
  "homesteadBlock": 0,
  "eip155Block": 0,
  "eip158Block": 0
 }
}
```

Figure 3.4 Genesis Block

Attributes are on left side with their values in genesis block on right side.

3.16.1 Coinbase

- It is an address (160-bit address)
- It is set in the miner
- It is where ether go during mining
- Also called beneficiary or ether-base

3.16.2 Difficulty

A scalar value corresponding to the difficulty level of the block. This can be calculated from the previous block difficulty level and the timestamp.

In simple words: A number that increases the mining time.

3.16.3 Extra data

An optimal free, but max 32 – byte long space to conserve smart things for eternity.

3.16.4 Gas limit

A scalar equal to the current limit of gas expenditure per block.

In simple words: A number that stops too complex contracts from executing.

3.16.5 Nonce

A 64 – bit hash which proves combined with the mix-hash that a sufficient amount of computation has been carried out on the block.

In simple words: A hash that proves that a certain block has been mined

3.16.6 Mix hash

A 256 – bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block.

In simple words: Combined with the nonce it proves that proof of work is done.

3.16.7 Parent hash

The keccak 256 – bit hash of the parent block’s header in it’s entirely. It is a complete hash of the parent block’s header including nonce and mix-hash. It points to the parent block. The foundation of a block-chain. Only in the genesis block it can be zero.

3.16.8 Timestamp

A scalar value equal to the reasonable output of Unix’s time () at this block’s inception. Used to verify order of the blocks within the chain, also if the time between two blocks is getting too long the difficulty decreases. If the time is shorter then difficulty is automatically increased.

3.16.9 Alloc.

Allows defining a list of pre-filled wallets. (Pre-allocate accounts with ether)

3.16.10 Config.

Chain configuration field. It is the core configuration which determines the block-chain settings.

Chain-id: Tells about the id of chain

Home-stead-block: Homestead is the second major release of ethereum. The value zero means that you are using this release.

EIP: It stands for Ethereum Improvement Proposal, where developers propose ideas on how to improve ethereum and contribute to this project.

3.16.11 Gas

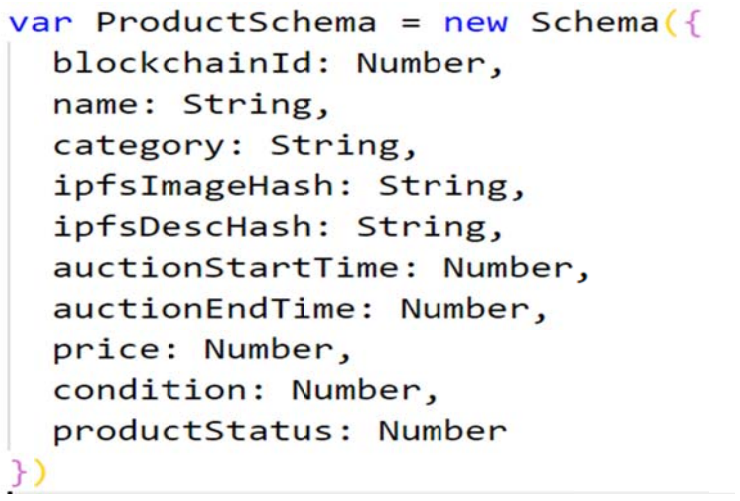
Gas is a unit, which represents the amount paid for the code (Smart Contract) execution. Efficient code consumes less gas.

New blocks created will contain same attributes. New block will contain hash of previous block(s) which creates the whole chain.

This chapter gives complete information of requirement gathering and analysis to clear the requirements of the system and to decide what the system should do and what the system should not do.

The second database will be the **Inter Planetary File System** IPFS for storing large data like product image and its description, to reduce the size of blockchain storage, only hash of the product images and description will be stored in blockchain to maintain integrity of the system.

The third database will be the **Mongo-Db** for storing all the product related information and displaying that information in front end, because it is not a good idea to query the blockchain again and again to display hundreds of products information. The products specific data except image and description (only their hash will be stored), will be stored in blockchain as well. The data in blockchain and Mongo-dB will remain consistent. Mongo-Db schema for product is:



```
var ProductSchema = new Schema({
  blockchainId: Number,
  name: String,
  category: String,
  ipfsImageHash: String,
  ipfsDescHash: String,
  auctionStartTime: Number,
  auctionEndTime: Number,
  price: Number,
  condition: Number,
  productStatus: Number
})
```

Figure 3.5 Mongo-Db Product Schema

"Knowledge is of no value unless you put it into practise." *Anton Chekhov (1860-1904)*

Chapter 4
Software Design Description

This chapter first gives the complete description of software design. It then elaborates the architectural design and detailed description of components of system. Finally, this chapter elucidates the user interface design and interaction diagrams mainly system sequence diagrams and class/contract diagram.

4.1 Introduction

Software Design Description (SDD) is the representation of a software design to be used for communication design information to its stakeholders. It shows how the software system will be structured to satisfy the requirements. The SDD is performed in two stages. The first is a preliminary design in which the overall system architecture and data architecture is defined. In the second stage, that is the detailed design stage, more detailed data structures are defined and algorithms and codes are developed for the defined architecture.

4.1.1 Design Overview

Software design is an iterative process through which requirements are translated into a blueprint for constructing the software. It shows how end user can interact with the system therefore it mainly focuses on user interface design. Design begins with requirement model and at each stage, software design work products are reviewed for clarity, correctness, completeness and consistency with the requirements and with one another. Software design sits at the technical kernel of software engineering and is applied regardless of the software process model that is used. The requirements are translated clearly through designing class diagram, sequence diagram, system sequence diagram and user interface interactions.

4.1.3 Requirement Traceability Matrix

Table 4.1 Requirement Traceability Matrix

Requirement Id	**Requirement Name**	**Sequence Diagram**	**Test Case**	**Class/Contract Diagram**	**Interface**
UC:1	Create Auction	Yes	Yes	Yes	Yes
UC:2	Place Bid	Yes	Yes	Yes	Yes
UC:3	Reveal Bid	Yes	Yes	Yes	Yes
UC:4	Search Auction	No	Yes	No	Yes
UC:5	Finalize Auction	Yes	Yes	Yes	Yes
UC:6	Release Amount to Seller	Yes	Yes	Yes	Yes
UC:7	Refund Amount to Buyer	Yes	Yes	Yes	Yes
UC:8	Confirm Transaction	Yes	No	No	No

4.2 System Architecture Design

4.2.1 Chosen System Architecture

The chosen architecture for this system is Ethereum Architecture as explained in chapter 2. Interacting between components of system is shown in diagram. In current context architecture will contain user interfaces, for data storage there will be BlockChain, Mongo-Db and IPFS.

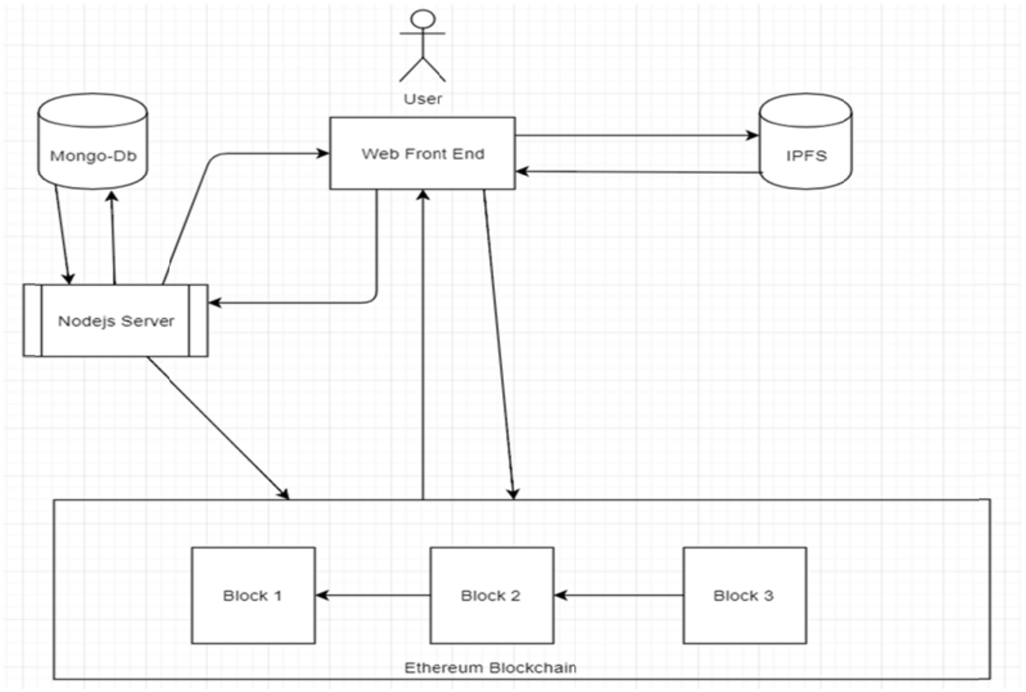


Figure 4.1 Ethereum Architecture of Application

4.3 User Interface Design

User interface design creates an effective communication between user and a computer. User interface design begins with the identification of user, task, and environmental requirements.

4.3.1 Description of User Interface Design

Table 4.2 User Interface Characteristics Description

Characteristic	**Description**
Window	Multiple windows allow different information to be displayed simultaneously on the user's screen.
Icon	Icons different types of information. On some systems, icons represent files; on others, icons represent processes.
Menu	Commands are selected from a menu rather than typed in a command language.
Pointing	A pointing device such as a mouse is used for selecting choices from a menu or indicating items of interest in a window.

Interface 1: Index Page

Index page shows Ethereum status and currently running auctions. User can click on the button "Create an Auction" to go to the next page where item can be added and auction can be created. Interface is shown in figure 4.2.

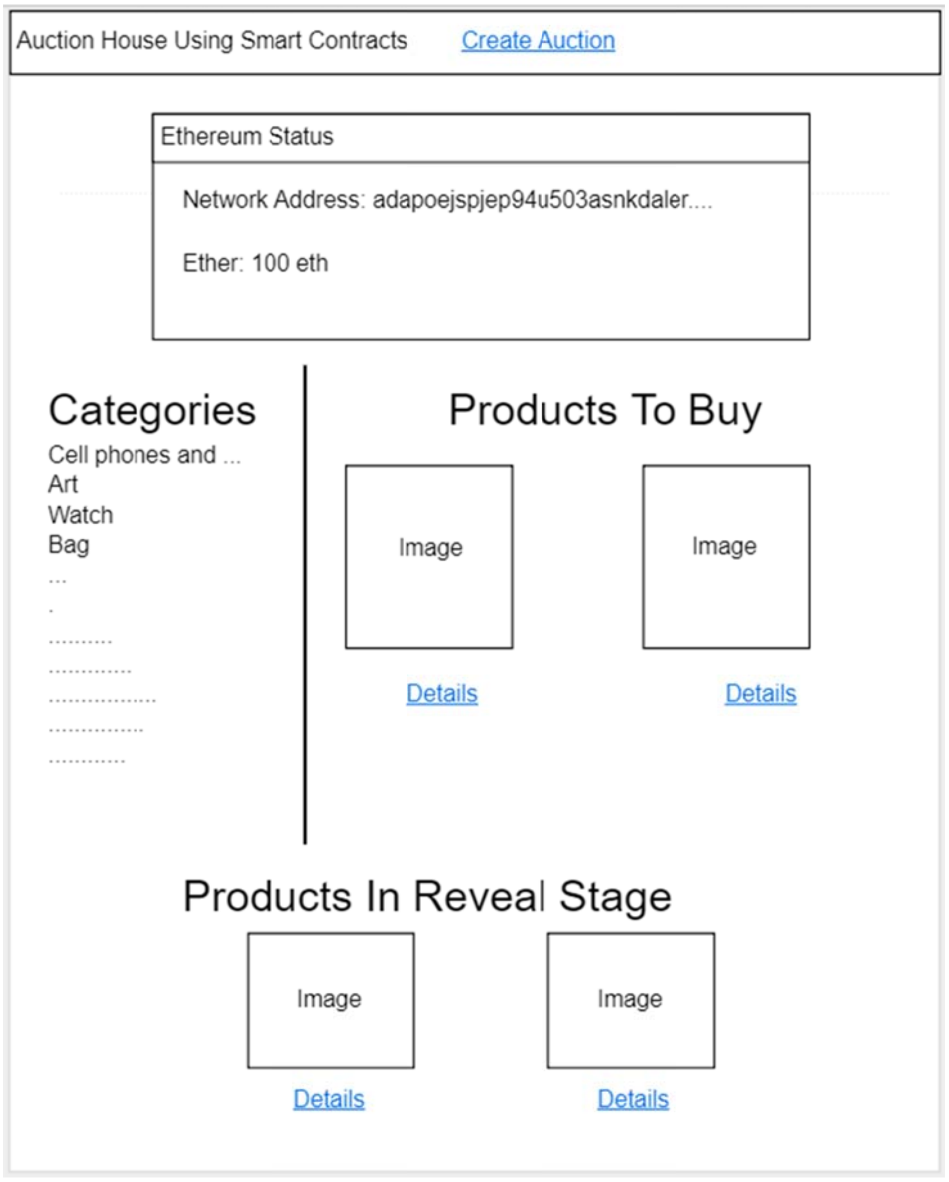
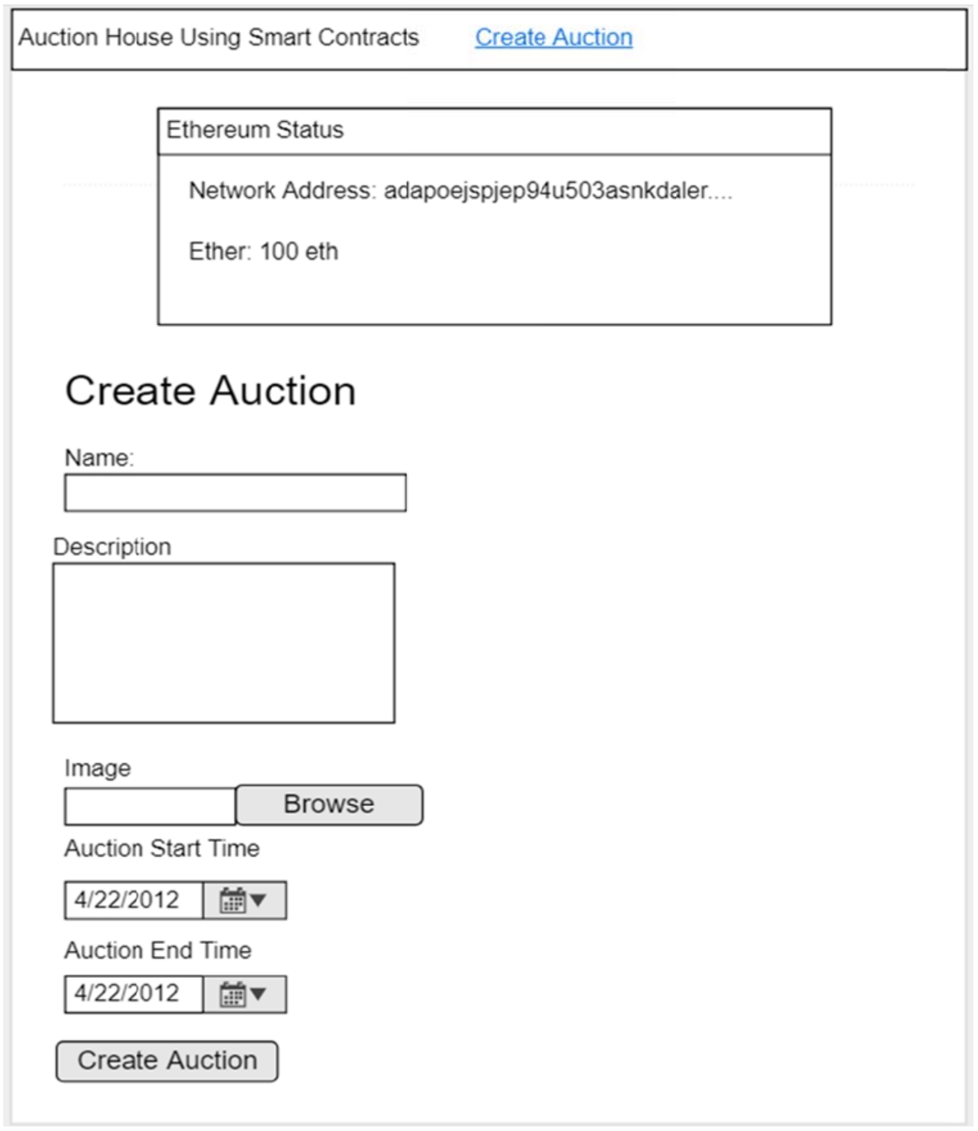


Figure 4.2 Index Page Interface

Interface 2: Create New Auction

This interface shows ethereum status and two forms. First form is for adding item and second form is for item bidding detail. Interface is shown in figure 4.3.


Auction House Using Smart Contracts Create Auction

Ethereum Status

Network Address: adapoejspjep94u503asnkdaler....

Ether: 100 eth

Create Auction

Name:

Description

Image

Auction Start Time

4/22/2012

Auction End Time

4/22/2012

Figure 4.3 Create New Auction Interface

Interface 3: Place Bid

This page shows the detail of item currently being auctioned. User can place bid for this item and click on “Place Bid”. This interface is shown in Figure 4.4.

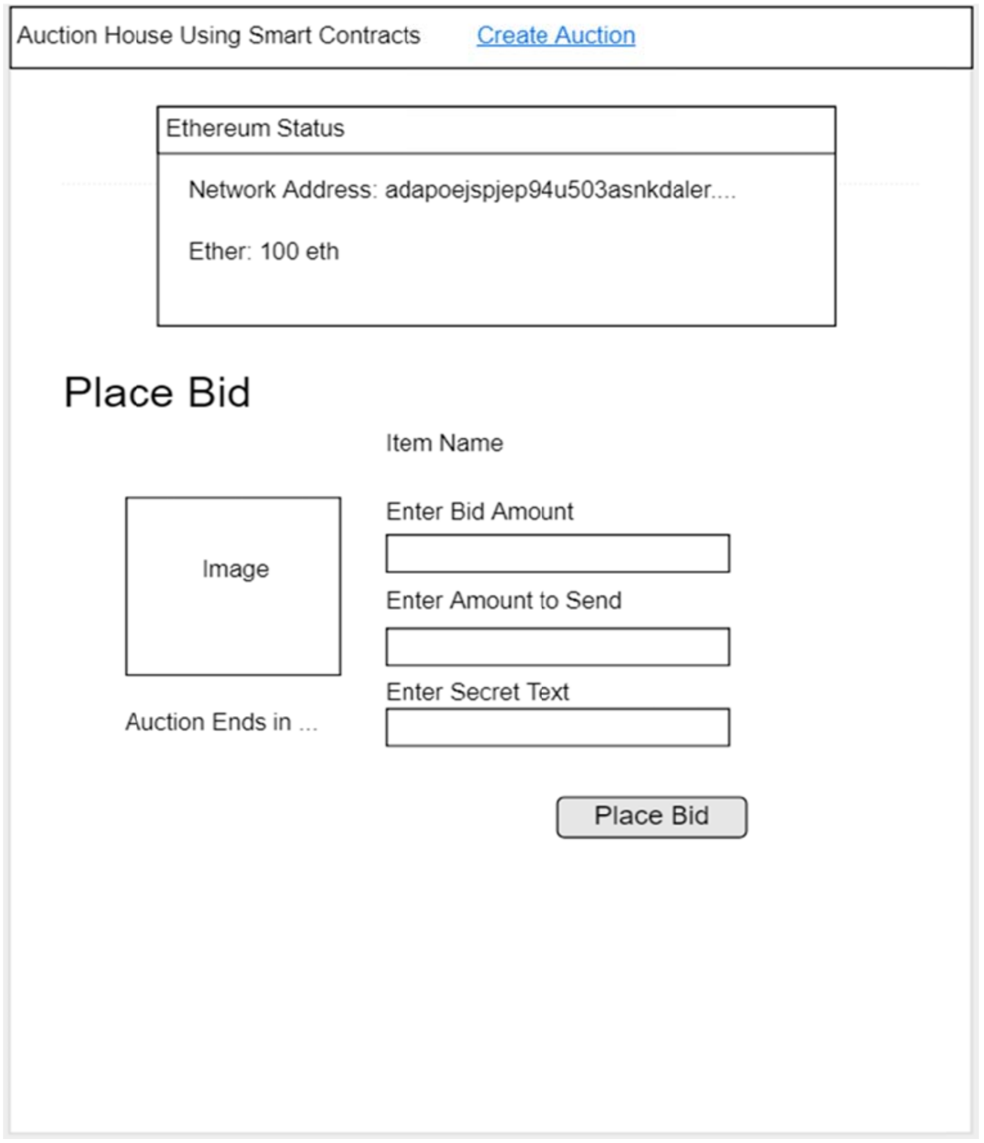


Figure 4.4 Place Bid Interface

Interface 4: Reveal Bid Interface

Bidders can reveal their bid after auctions in a specific time. Interface is shown in figure 4.5.

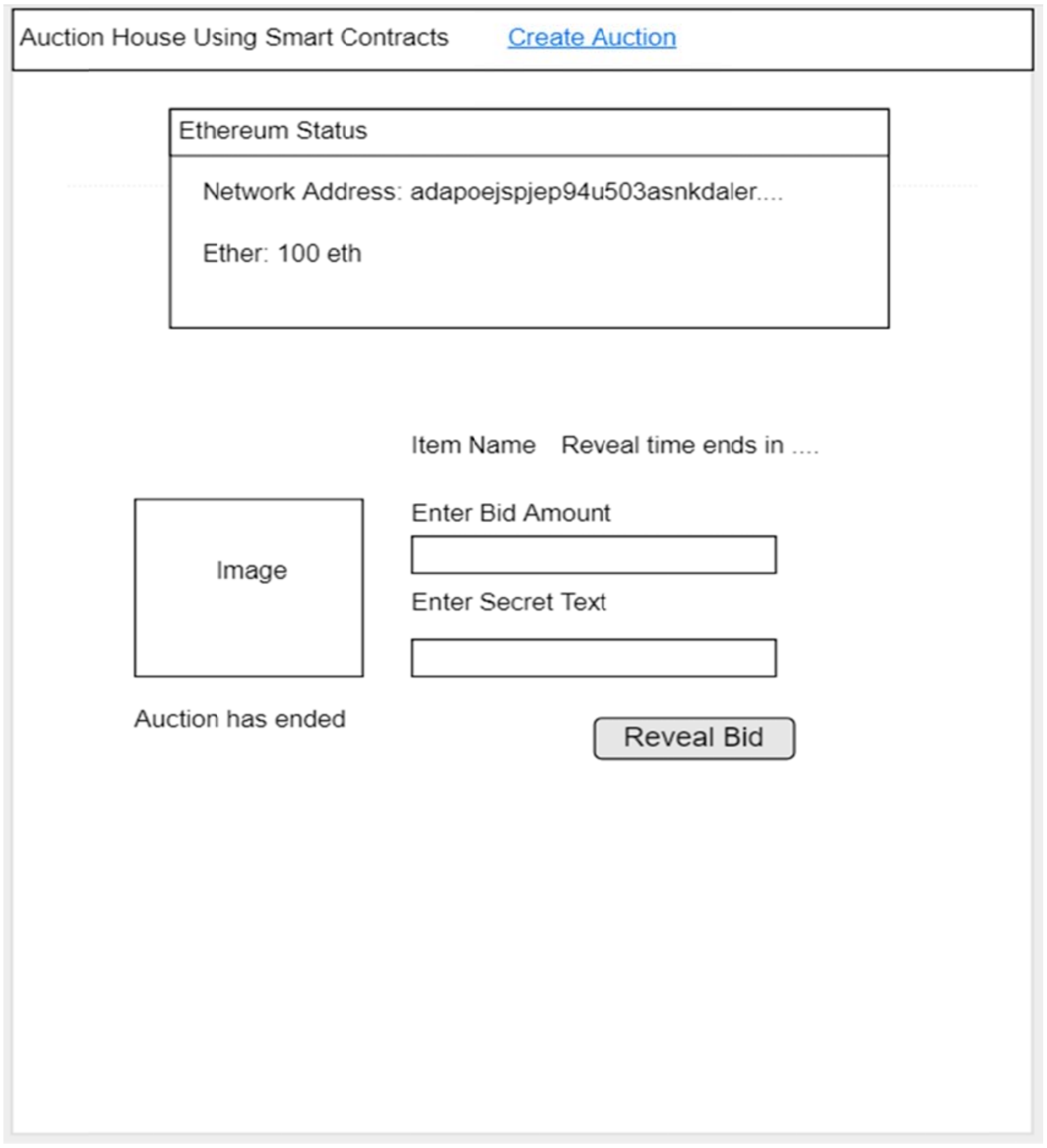


Figure 4.5 Reveal Bid Interface

Interface 5: Finalize Auction Interface

User can finalize auction using this interface. Only a person except buyer and seller can finalize the auction. Interface is shown in figure 4.6.

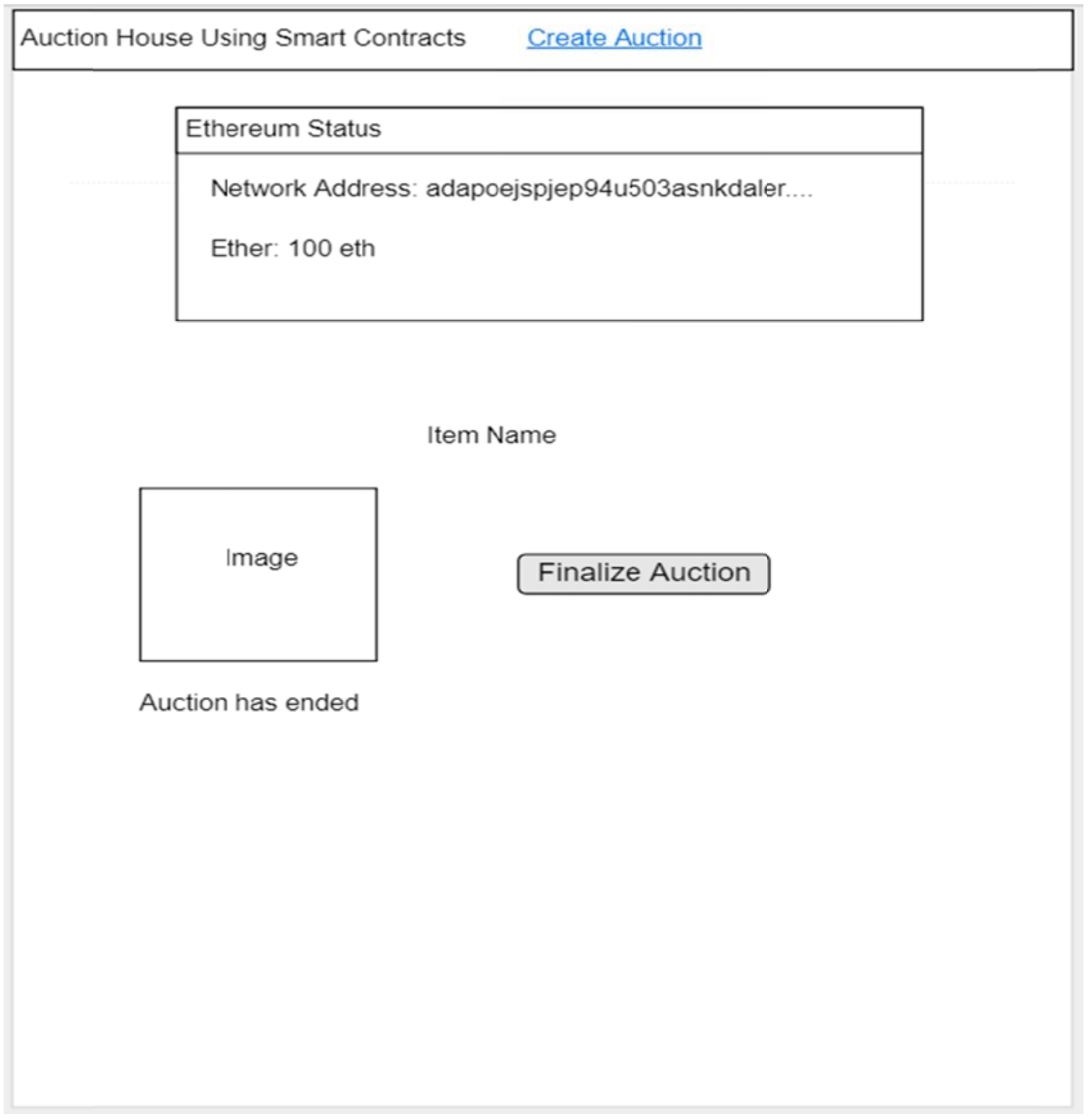
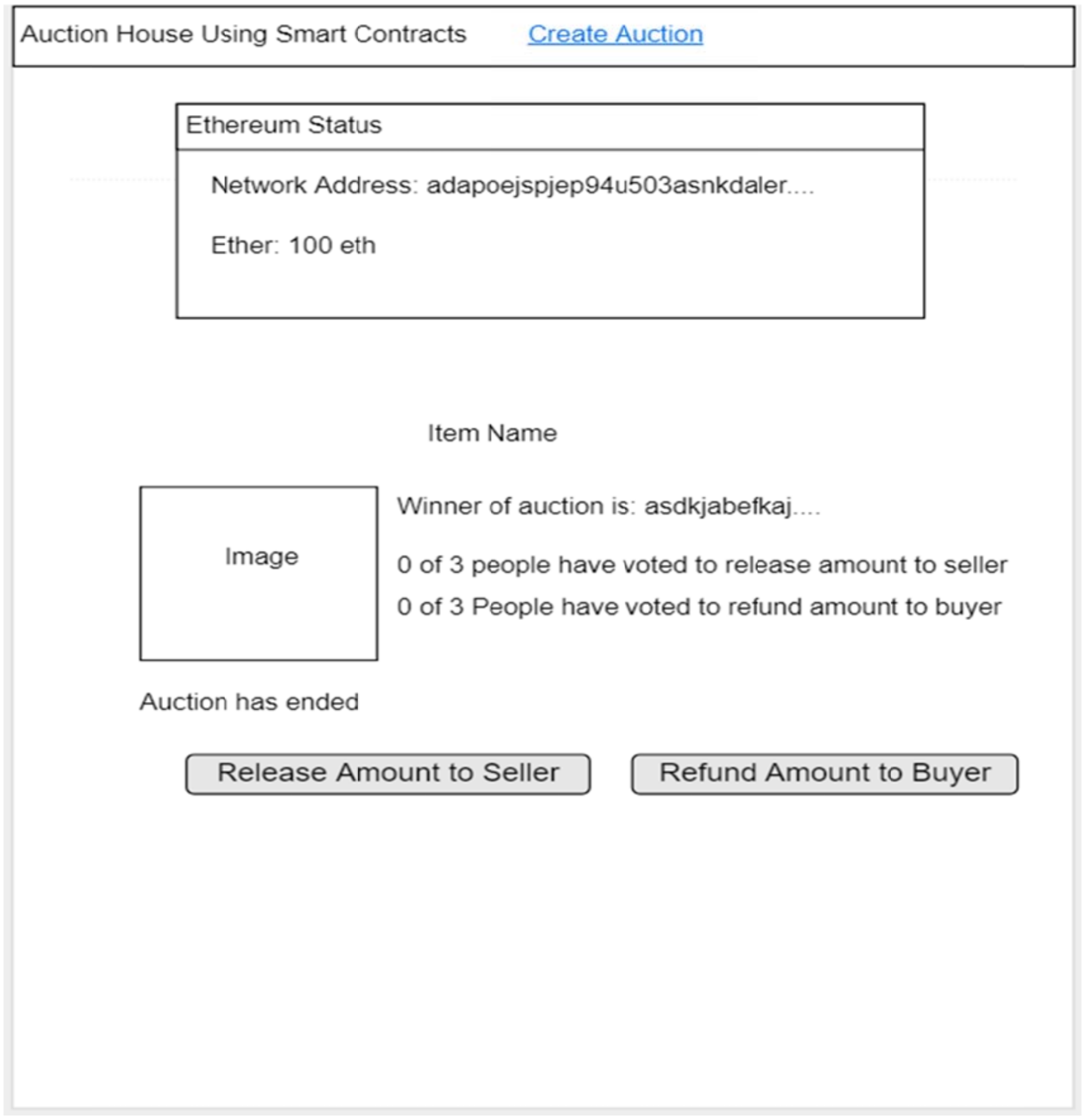


Figure 4.6 Finalize Auction Interface

Interface 6: Release/Refund Amount Interface

The person who finalized the auction, buyer and seller can vote using this interface. Interface is shown in figure 4.7.



Network Address: adapoejspjep94u503asnkdaler....

Ether: 100 eth

Item Name

Winner of auction is: asdkjabefkaj....

0 of 3 people have voted to release amount to seller

0 of 3 People have voted to refund amount to buyer

Auction has ended

Figure 4.7 Release Amount Interface

4.4 Sequence Diagram

Sequence diagram are used to model the interaction between the actors and the objects in a system and interaction between the objects themselves. A sequence diagram shows interactions that take place during a particular use case or use case scenario. Decentralized controlled structure is used for making sequence diagrams. In the decentralized control structure participating objects directly communicate with other objects without any controlling object [4].

4.4.1 Create Auction

The user adds auction details and addProductToStore() method of smart contract is called, Meta-Mask asks for confirmation before signing the transaction, after confirmation and successfully storing the transaction in blockchain, item successfully added message is returned to user. Adding product also creates an event which of new Product which stores data in Mongo-Db as well. Sequence diagram is shown in figure 4.8

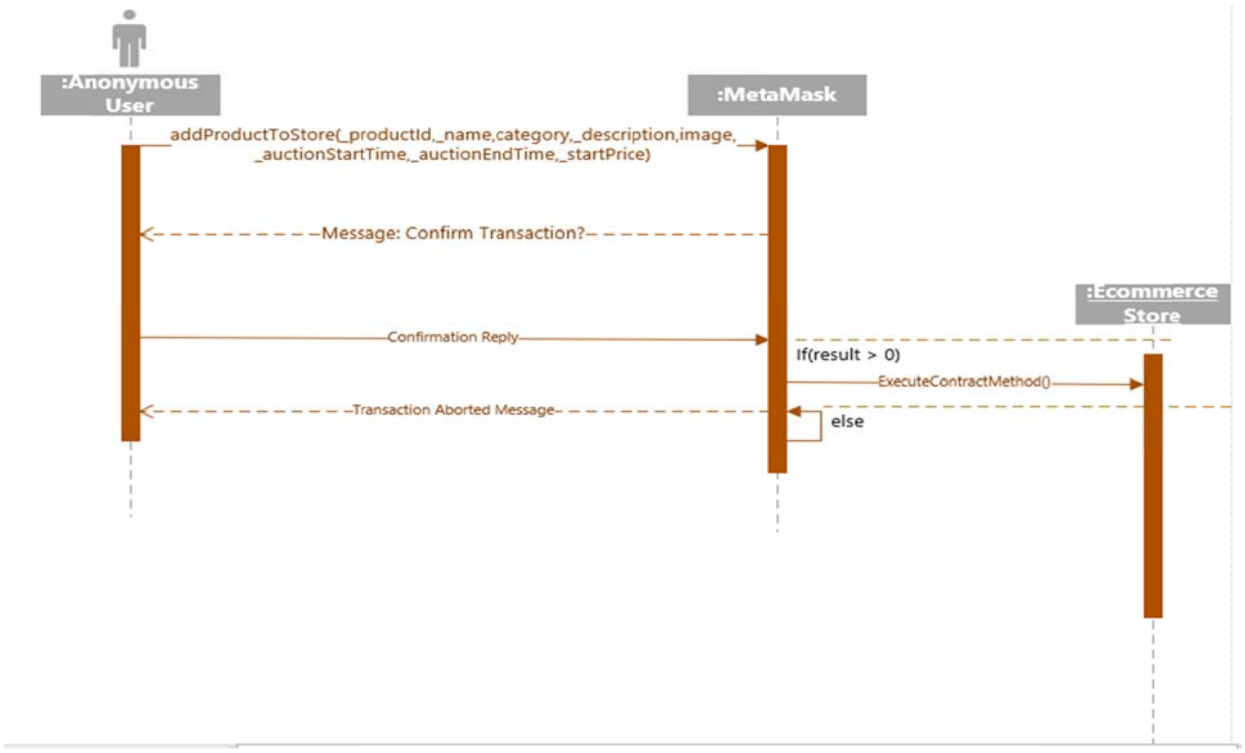

:Anonymous User
:MetaMask
addProductToStore(_productId,_name,category,_description,image,_auctionStartTime,_auctionEndTime,_startPrice)
Message: Confirm Transaction?
:Ecommerce Store
Confirmation Reply
If(result > 0)
ExecuteContractMethod()
Transaction Aborted Message
else

Figure 4.8 Sequence Diagram Create Auction

4.4.2 Place Bid

Bid is placed by the person who wants to buy the item. Meta-Mask will ask for the transaction confirmation. Sequence Diagram is shown in figure 4.9

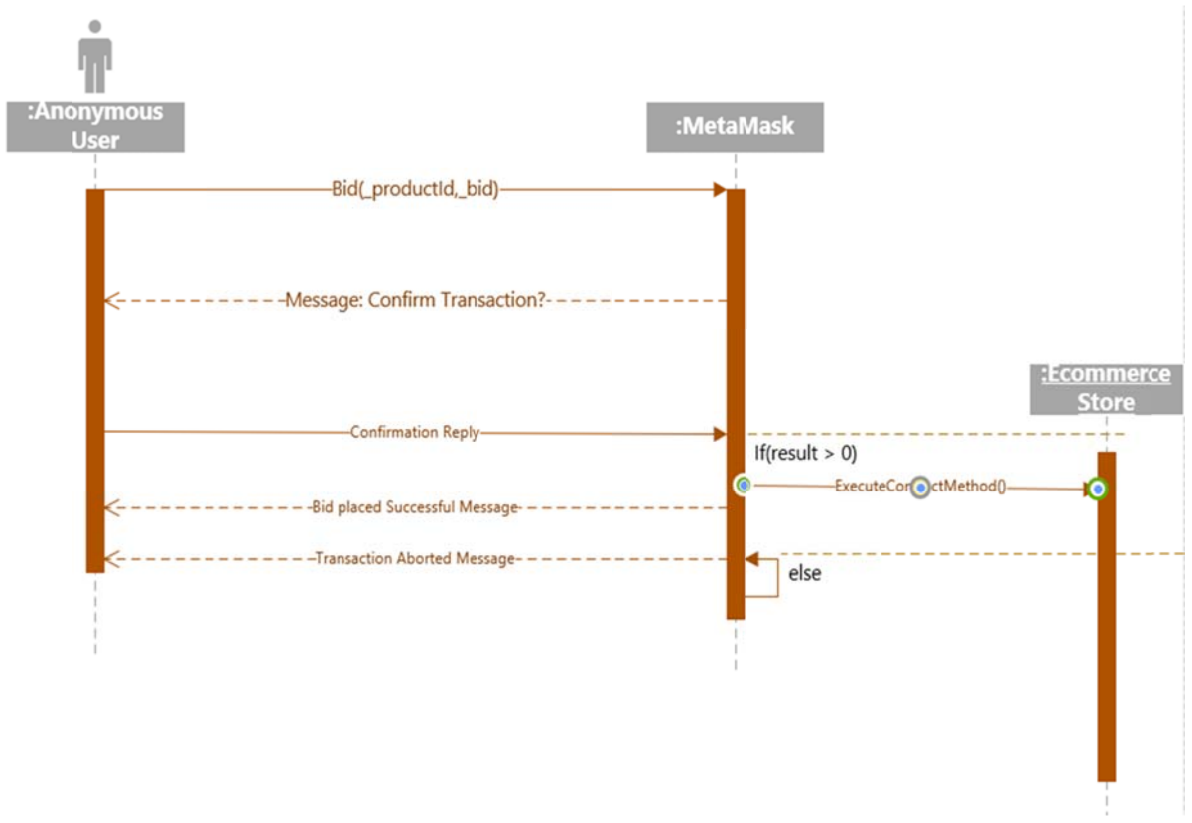


Figure 4.9 Sequence Diagram Place Bid

4.4.3 Reveal Bid

Users who placed bid will reveal their bids after auction end time. Sequence diagram is shown in figure 4.10.

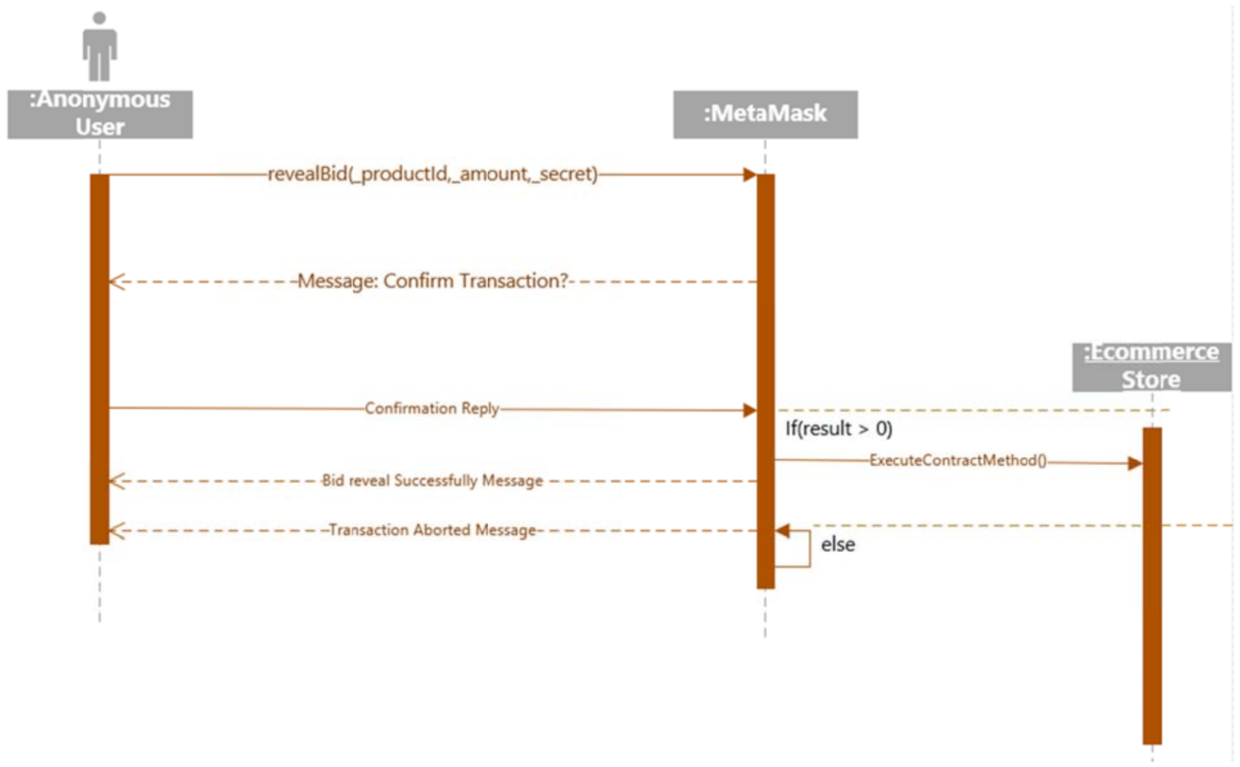

:Anonymous User
:MetaMask
:Ecommerce Store
revealBid(_productId,_amount,_secret)
Message: Confirm Transaction?
Confirmation Reply
If(result > 0)
ExecuteContractMethod()
Bid reveal Successfully Message
Transaction Aborted Message
else

Figure 4.10 Sequence Diagram Reveal Bid

4.4.4 Finalize Auction

After bid reveal time ends, auction is finalized by the user who is not buyer or seller. After finalizing winner is decided. Then escrow contract is created in which buyer, seller and the person who finalized the auction will vote to either release amount to seller or refund amount to buyer. Sequence diagram is shown in figure 4.11.

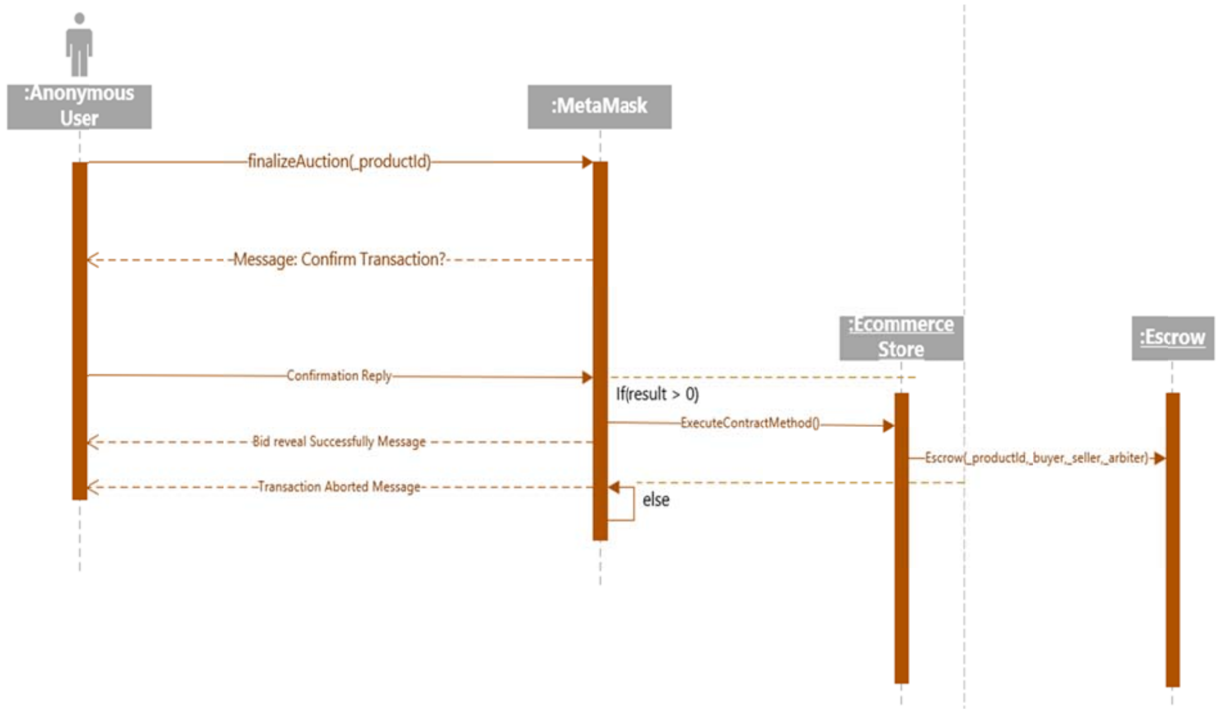

:Anonymous User
:MetaMask
:Ecommerce Store
:Escrow
finalizeAuction(_productId)
Message: Confirm Transaction?
Confirmation Reply
If(result > 0)
ExecuteContractMethod()
Escrow(_productId,_buyer,_seller,_arbiter)
Bid reveal Successfully Message
Transaction Aborted Message
else

Figure 4.11 Sequence Diagram Place Bid

4.4.5 Release Amount to Seller

At least two out of three persons will vote to release funds from escrow and send it to seller. Sequence diagram is shown in figure 4.12.

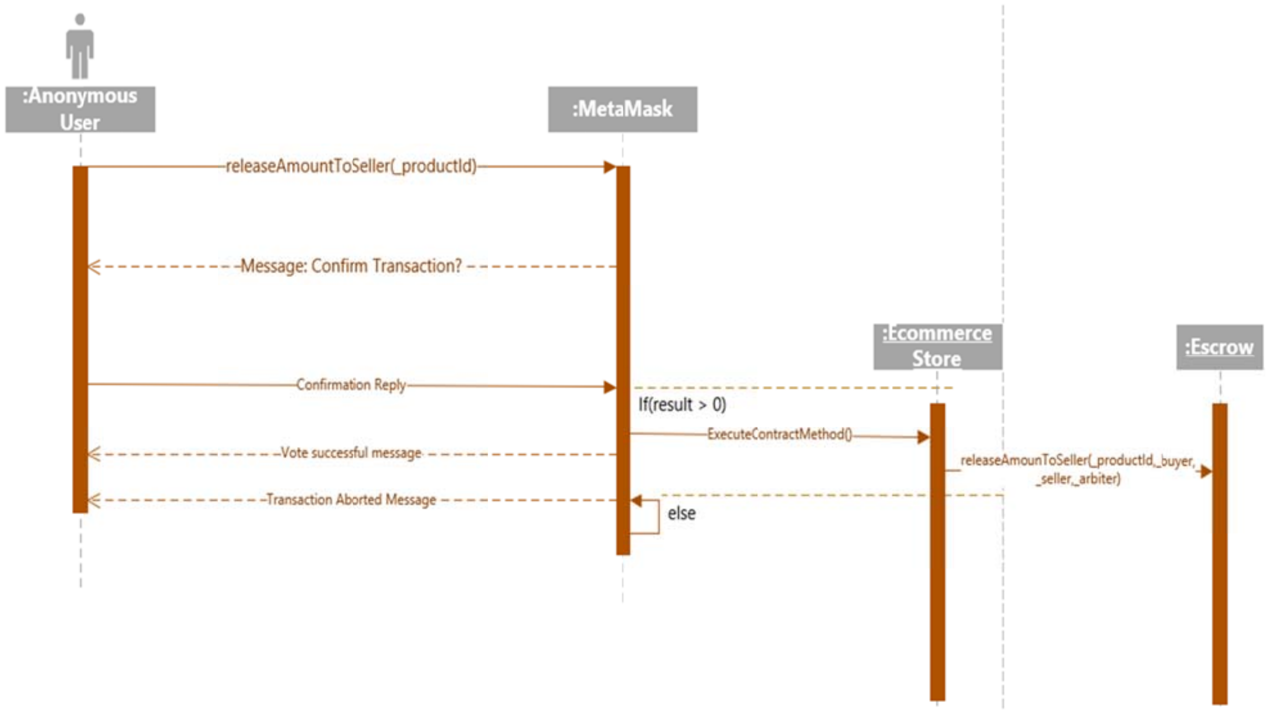

:Anonymous User
:MetaMask
:Ecommerce Store
:Escrow
releaseAmountToSeller(_productId)
Message: Confirm Transaction?
Confirmation Reply
If(result > 0)
ExecuteContractMethod()
releaseAmounToSeller(_productId,_buyer, _seller,_arbiter)
Vote successful message
Transaction Aborted Message
else

Figure 4.12 Sequence Diagram Cancel Auction

4.4.6 Refund Amount to Buyer

At least two out of three persons will vote to release funds from escrow and refund it back to buyer in case of any product related problem. Sequence diagram is shown in fig 4.13.

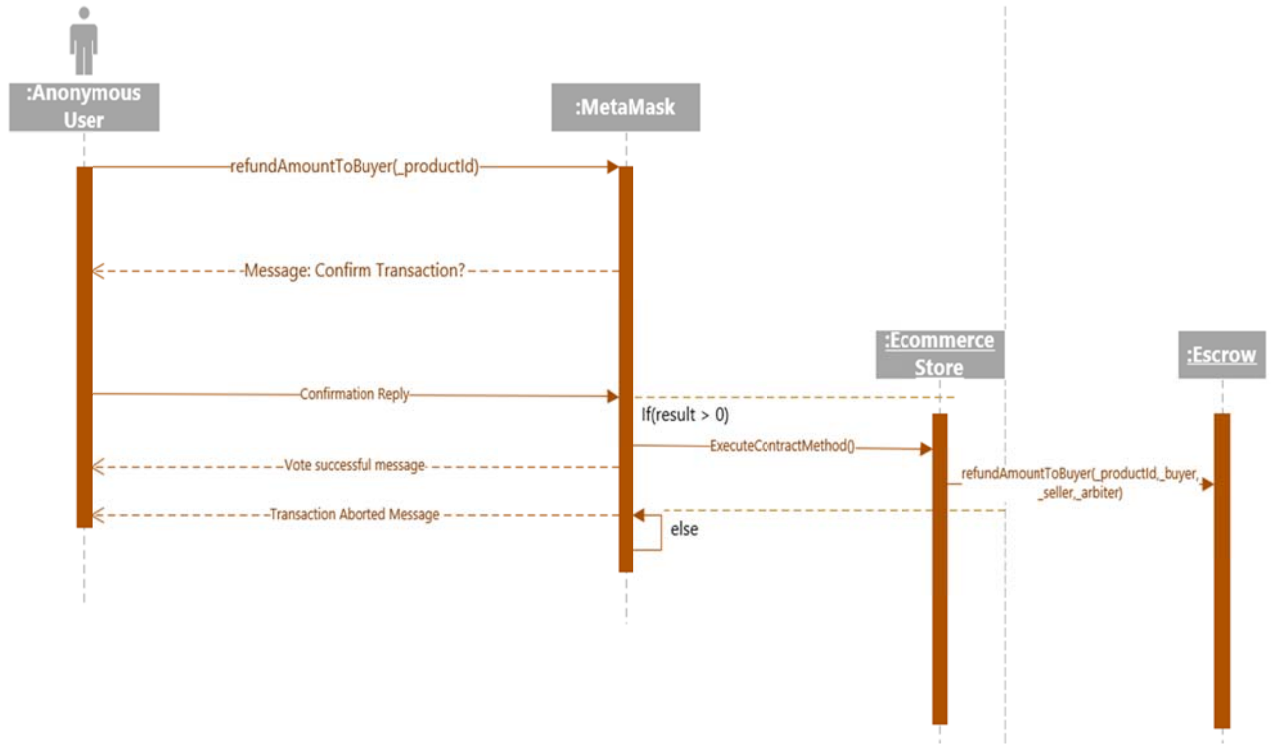

:Anonymous User
:MetaMask
:Ecommerce Store
:Escrow
refundAmountToBuyer(_productId)
Message: Confirm Transaction?
Confirmation Reply
If(result > 0)
ExecuteContractMethod()
refundAmountToBuyer(_productId,_buyer, _seller,_arbiter)
Vote successful message
Transaction Aborted Message
else

Figure 4.13 Sequence Diagram Refund Amount to Buyer

4.5 Class/Contract Diagram

Smart Contracts are used instead of classes for developing decentralized applications. Like classes, object-oriented concepts can be applied on smart contracts like inheritance. There is no official release on how to represent smart contracts on paper. In this case we will use class diagrams for showing smart contracts on paper and modelling the static view of application. Programming language used in this project is solidity which is a contract-oriented language. Concepts used in contract-oriented programming are almost the same as they are in object-oriented programming. User defined structures are also used in smart contracts. It is better not to create controller contracts because the more contracts to be executed, the more cost is put on the user. Executing Contracts costs ether.

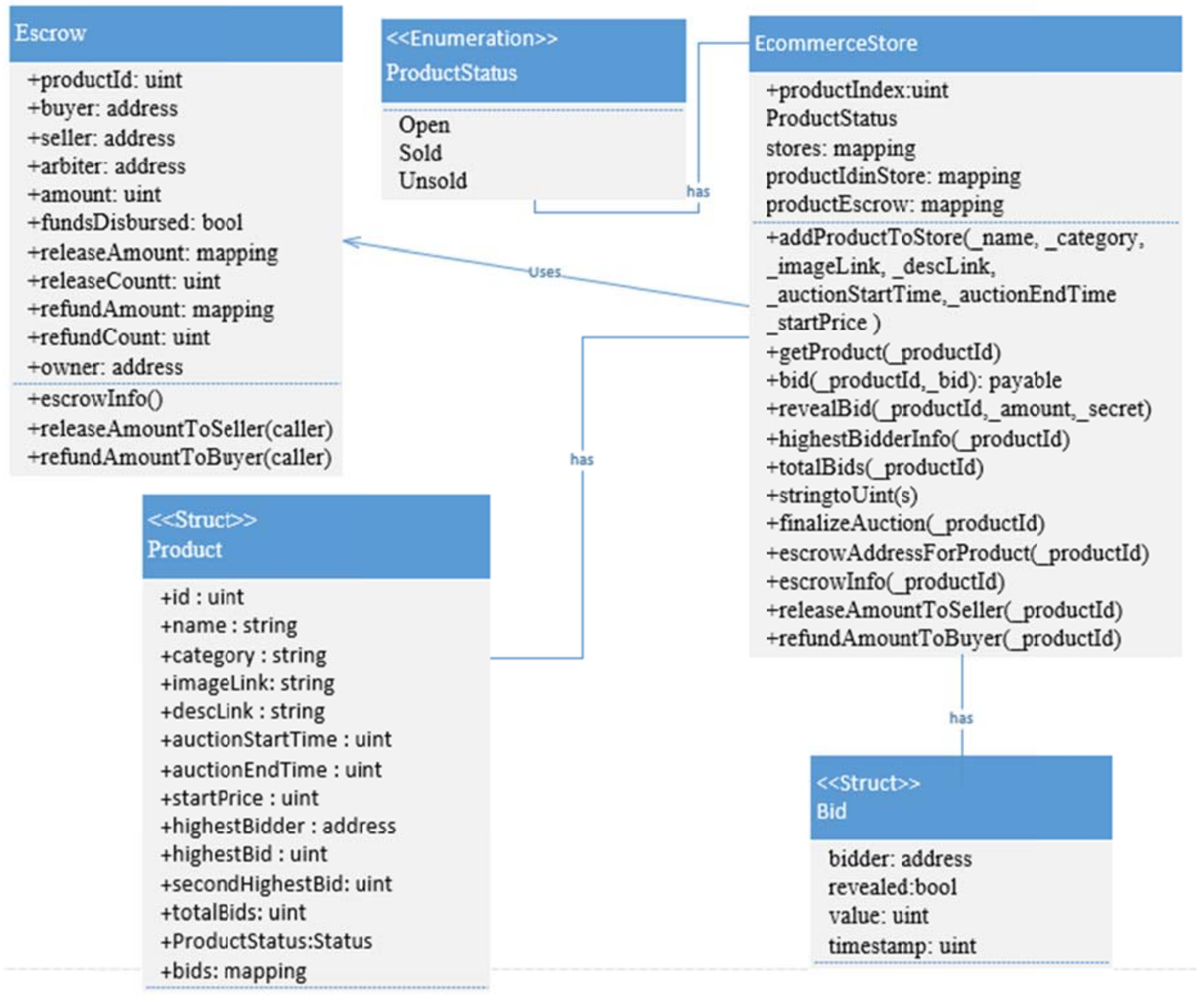


Figure 4.14 Contract/Class Diagram

This chapter covers the complete description of software design. It has provided a detailed description regarding architecture design, components of the system and user interface description. Finally, interaction between the object and human actor are shown by interaction diagram and relationship between the instances is shown by class diagram. System testing and test cases are to be discussed in the next chapter.

Chapter 5

Software Implementation

This document describes the project implementation for developing the project planner and scheduler.

5.1 Language Selection

- **Solidity**
  Programming language used for the development of smart contracts
- **Html/CSS**
  Used for designing web pages
- **JavaScript**
  Used for scripting and validation

5.2 Tools Selection

- Visual Studio Code
- Nodejs Server
- Mongo-Db
- Inter Planetary File System
- Web Browser
- Meta-Mask
- Ganache

5.3 Application Flow

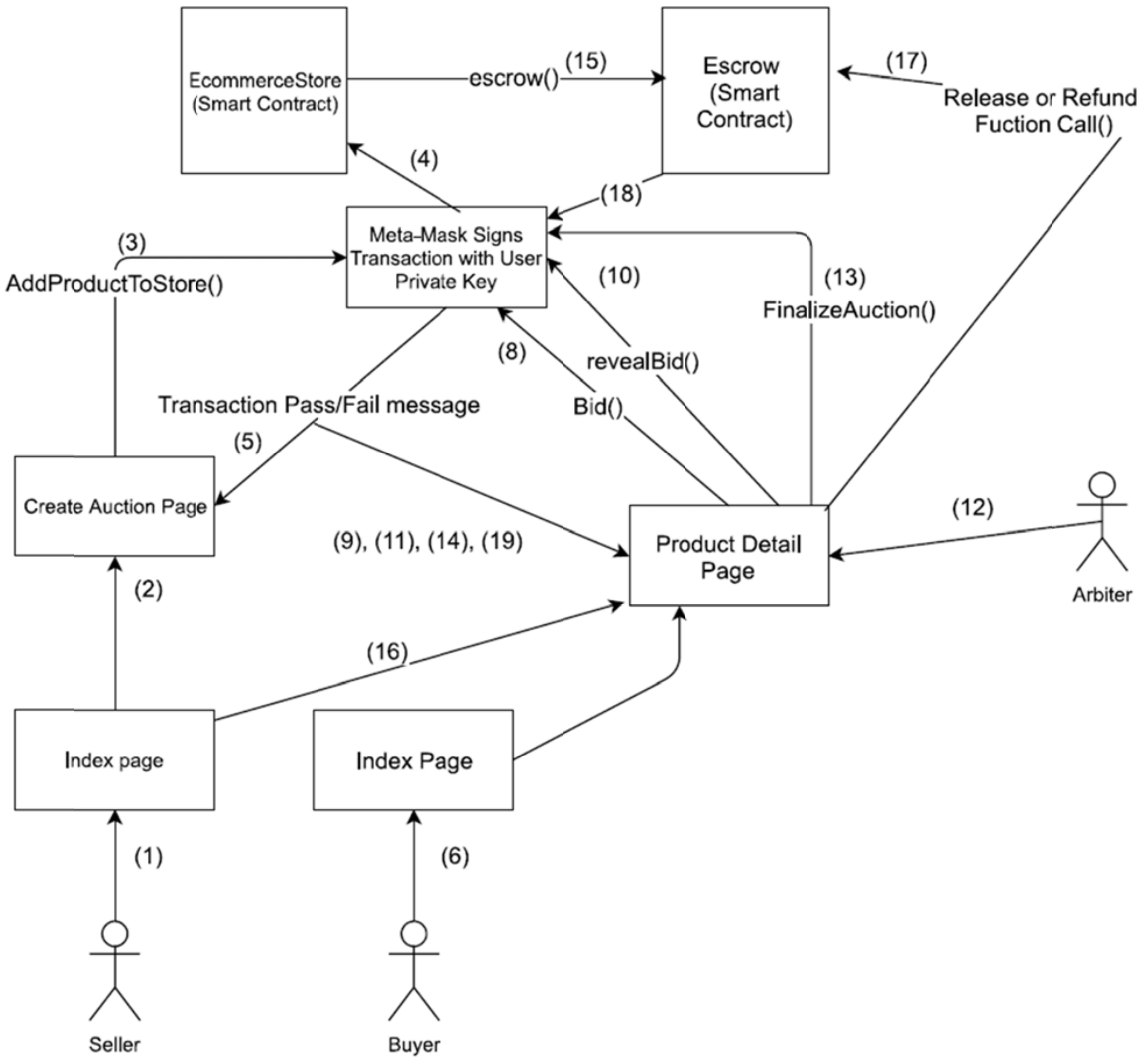
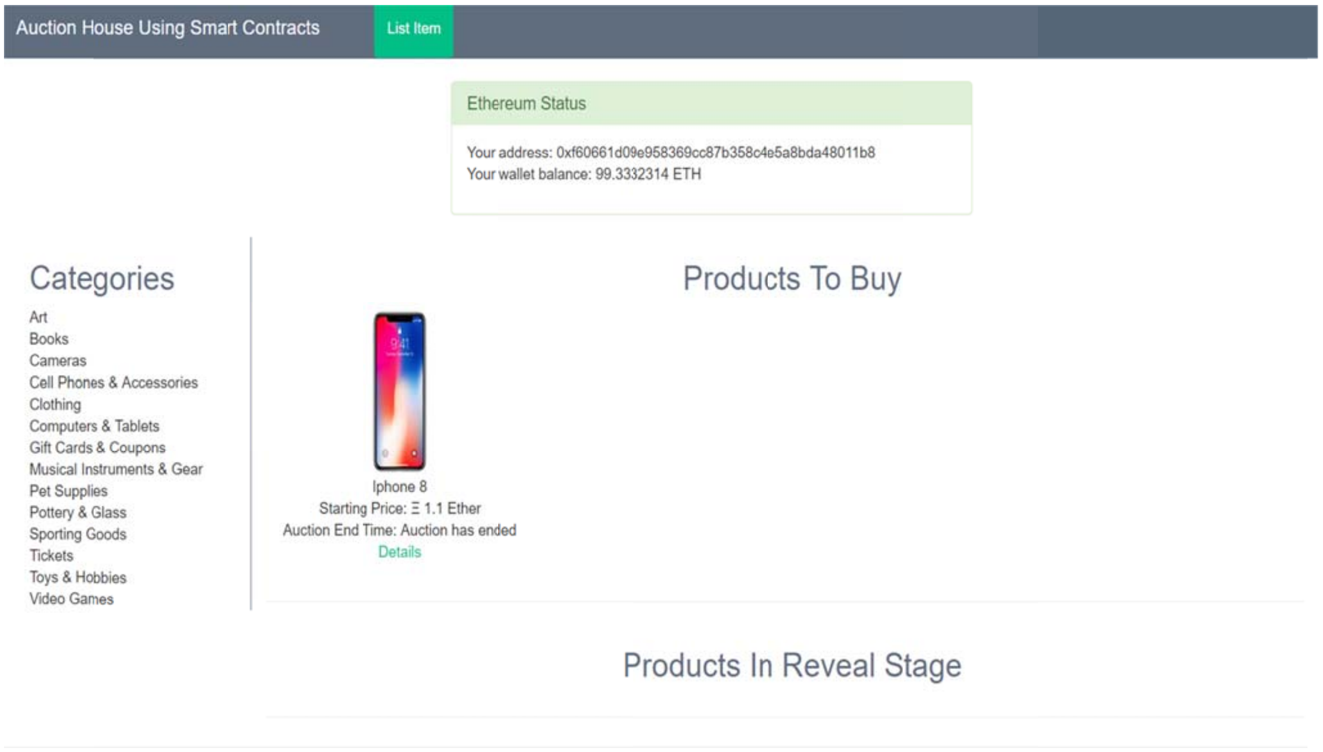


Figure 5.1 Application Flow

5.4 Application Screenshots

Screenshot for Index Page



Auction House Using Smart Contracts

Ethereum Status

Your address: 0xf60661d09e958369cc87b358c4e5a8bda48011b8
Your wallet balance: 99.3332314 ETH

Categories

- Art
- Books
- Cameras
- Cell Phones & Accessories
- Clothing
- Computers & Tablets
- Gift Cards & Coupons
- Musical Instruments & Gear
- Pet Supplies
- Pottery & Glass
- Sporting Goods
- Tickets
- Toys & Hobbies
- Video Games

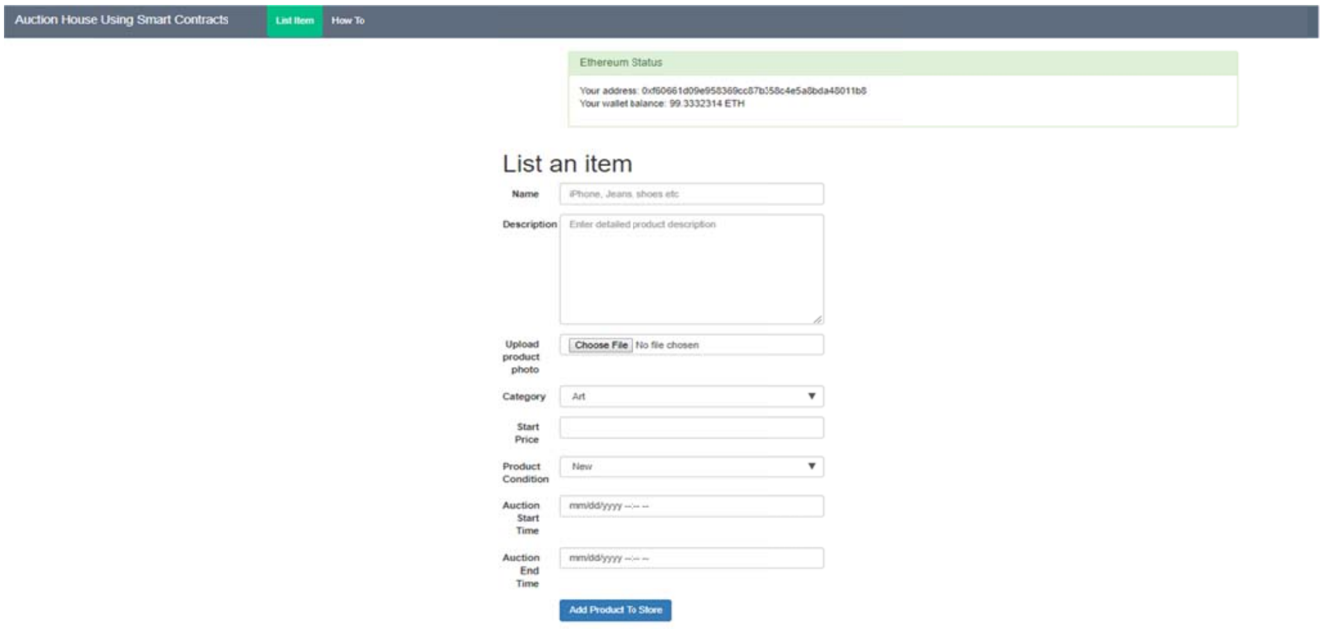
Products To Buy

Iphone 8
Starting Price: Ξ 1.1 Ether
Auction End Time: Auction has ended

Products In Reveal Stage

Figure 5.2 Index Page Screen

Screenshot for Create Auction



Auction House Using Smart Contracts

Ethereum Status

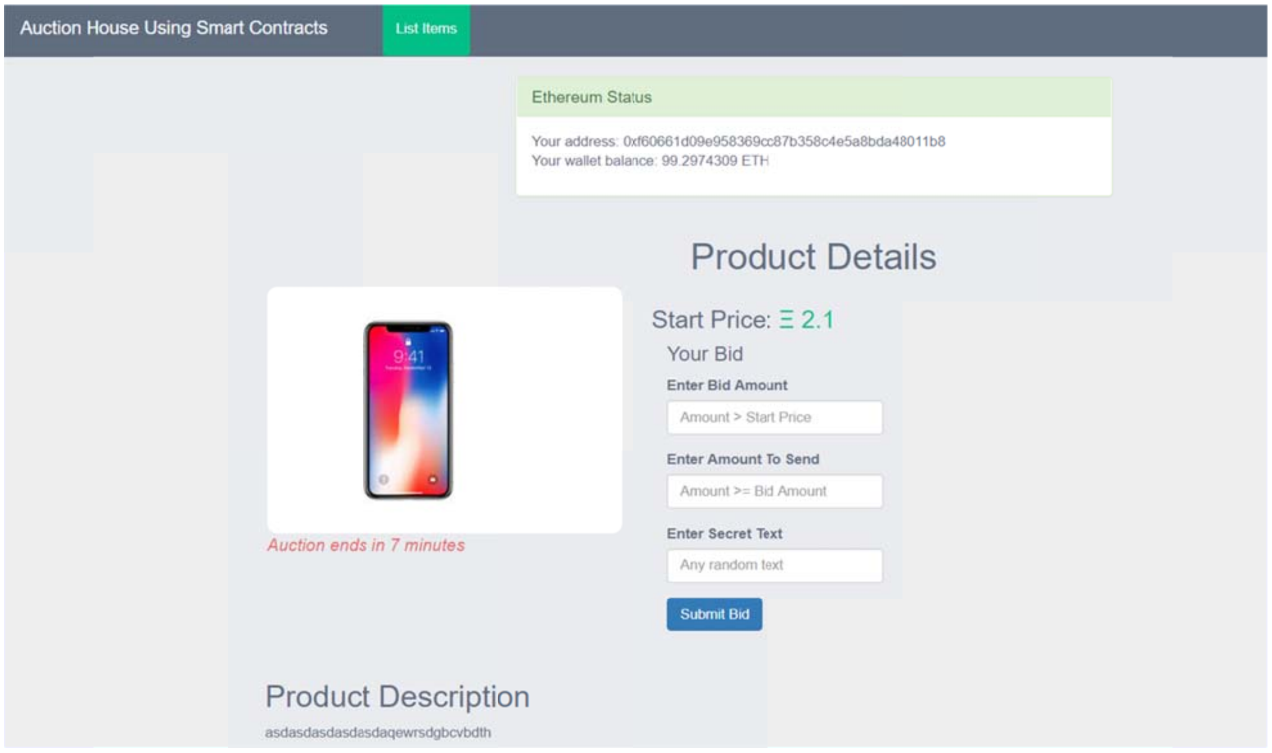
Your address: 0xf60661d09e958369cc87b358c4e5a8bda48011b8
Your wallet balance: 99.3332314 ETH

List an item

Name	iPhone, Jeans, shoes etc
Description	Enter detailed product description
Upload product photo	Choose File No file chosen
Category	Art
Start Price	
Product Condition	New
Auction Start Time	mm/dd/yyyy --:-- --
Auction End Time	mm/dd/yyyy --:-- --

Figure 5.3 Create Auction Screen

Screenshot for Place Bid



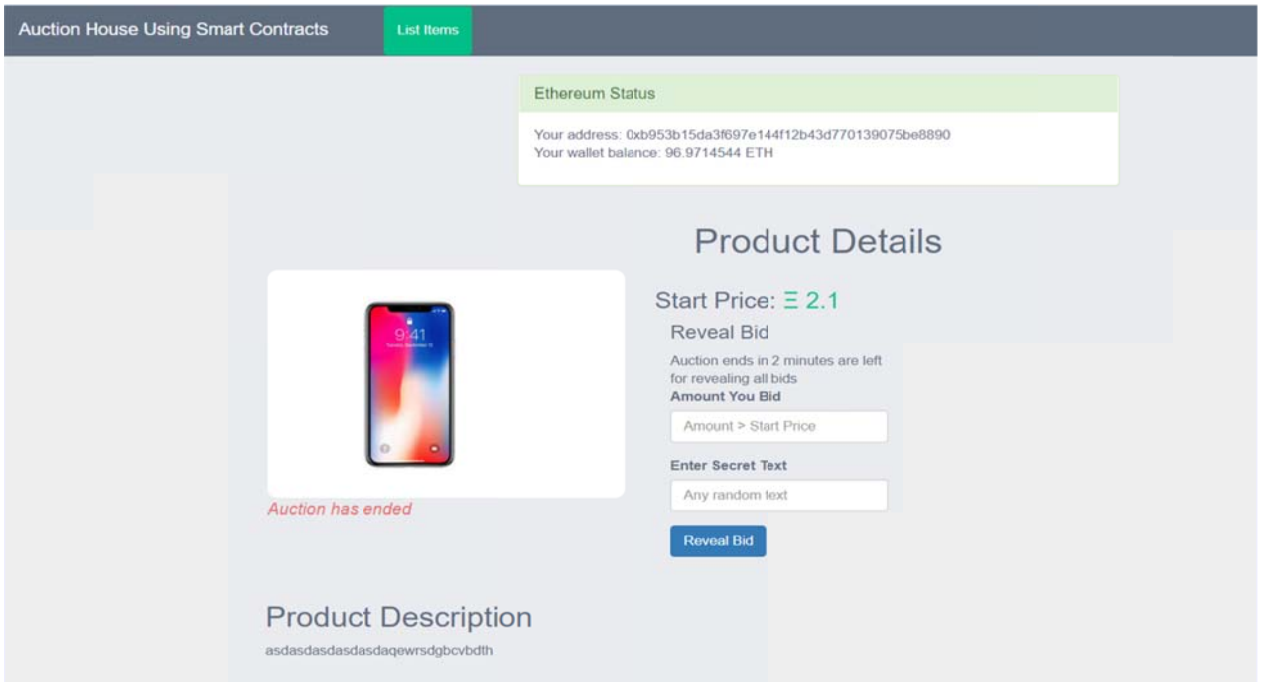
Your address: 0xf60661d09e958369cc87b358c4e5a8bda48011b8
Your wallet balance: 99.2974309 ETH

Auction ends in 7 minutes

asdasdasdasdasdaqewrsdgbcvbdth

Figure 5.4 Place Bid Screen

Screenshot for Reveal Bid



Your address: 0xb953b15da3f697e144f12b43d770139075be8890
Your wallet balance: 96.9714544 ETH

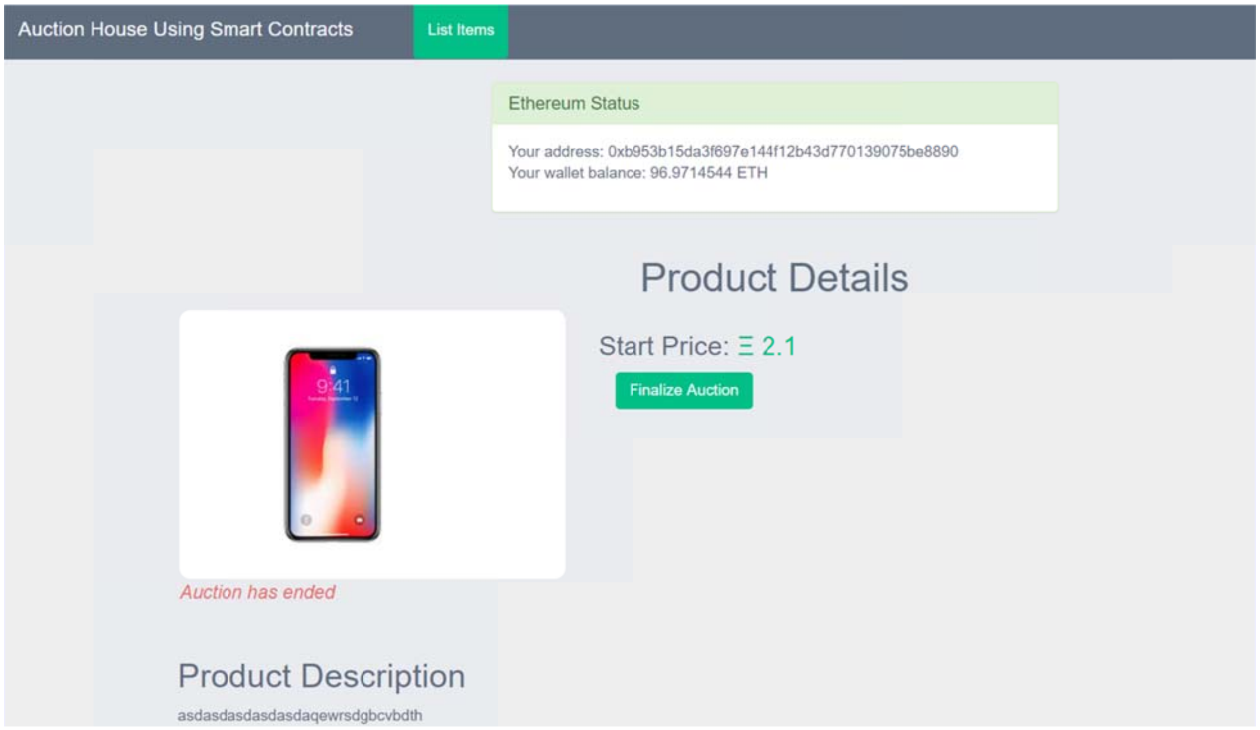
Auction ends in 2 minutes are left for revealing all bids

Auction has ended

asdasdasdasdasdaqewrsdgbcvbdth

Figure 5.5 Reveal Bid

Screenshot for Finalize Auction


Auction House Using Smart Contracts

Ethereum Status

Your address: 0xb953b15da3f697e144f12b43d770139075be8890
Your wallet balance: 96.9714544 ETH

Product Details

Start Price: Ξ 2.1

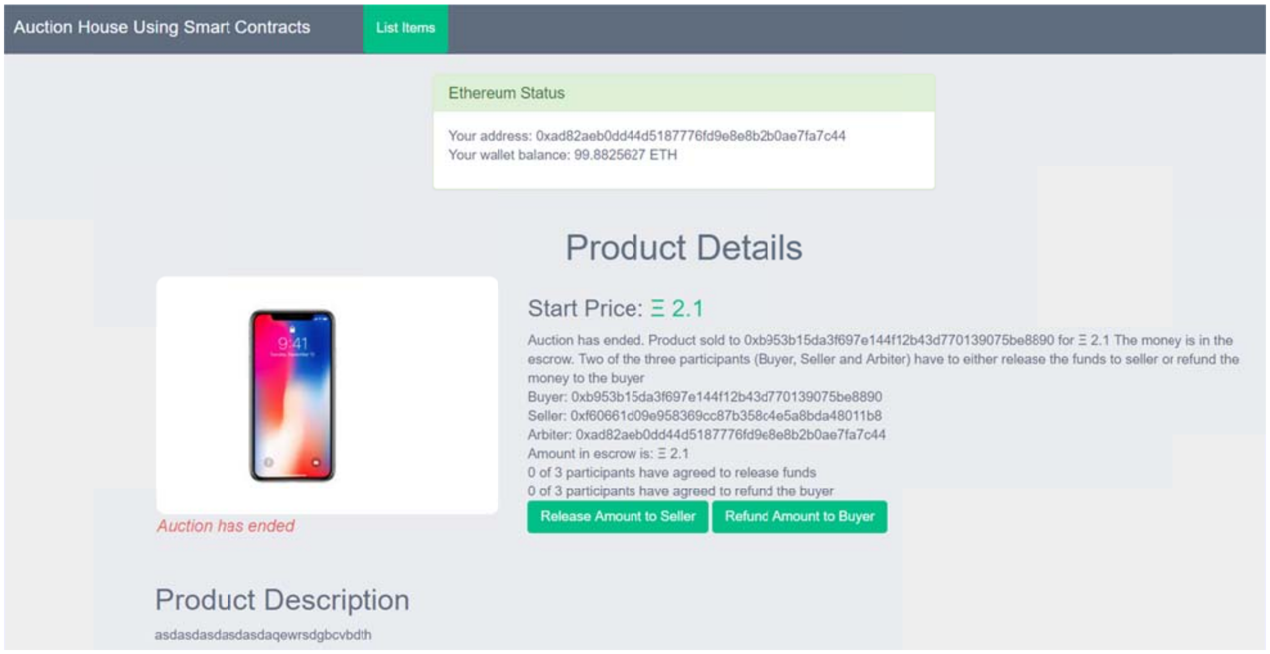
Auction has ended

Product Description

asdasdasdasdasdaqewrsdgbcvbdth

Figure 5.6 Finalize Auction Screenshot

Screenshot for Refund and Release Amount


Auction House Using Smart Contracts

Ethereum Status

Your address: 0xad82aeb0dd44d5187776fd9e8e8b2b0ae7fa7c44
Your wallet balance: 99.8825627 ETH

Product Details

Start Price: Ξ 2.1

Auction has ended. Product sold to 0xb953b15da3f697e144f12b43d770139075be8890 for Ξ 2.1 The money is in the escrow. Two of the three participants (Buyer, Seller and Arbiter) have to either release the funds to seller or refund the money to the buyer
Buyer: 0xb953b15da3f697e144f12b43d770139075be8890
Seller: 0xf60661d09e958369cc87b358c4e5a8bda48011b8
Arbiter: 0xad82aeb0dd44d5187776fd9e8e8b2b0ae7fa7c44
Amount in escrow is: Ξ 2.1
0 of 3 participants have agreed to release funds
0 of 3 participants have agreed to refund the buyer

Auction has ended

Product Description

asdasdasdasdasdaqewrsdgbcvbdth

Figure 5.7 Refund and Release Amount Screenshot

Victory belongs to the most preserving." Will Durant (1769-1821)

Chapter 6

Software Testing

This chapter describes software testing and software testing processes. This chapter further elaborates the acceptance test cases which are used to test the functional and non-functional requirements after coding of software

6.1 Introduction

Software test document involves the documentation of artefacts that should be developed before or during the testing of software. Software testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

6.1.1 Test Approach

Manual testing includes testing a software manually without using any automated tool or any script. The tester takes over the role of an end-user and tests the software to identify any unexpected behaviour or bug. There are different stages for manual testing such as unit testing, system testing, and user acceptance testing. Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it. Unit testing is the process of testing program components, such as methods or object and classes. Individual method or function are simplest type of the component.

Test-first development is an approach to development where tests are written before the code to be tested. Small code changes are made and the code is refactored until all tests execute successfully.

6.2 Test Plan

Test planning is an activity that ensures that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. Test plan determines the scope and the risk that need to be tested and are not to be tested. Deciding fail and pass criteria.

6.2.1 Testing Tools and Environment

This setup consists of the physical setup which includes hardware, and logical setup that includes operating system, client operating system, blockchain database, front end running environment, browser (if web application), or any other software components required to run this software product.

6.3 Test Cases

6.3.1 Create Auction

Table 6.1 Create Test Case

ID	T1
Description	Random Item details will be added to the system, and transaction will be executed
Tester	User
Setup	Login through MetaMask
Instructions:	1. Select create new auction 2. Enter item name Iphone X 3. Enter item description efg 4. Add item image (iphone.png) 5. Enter price (1 ether) 6. Add auction start time (current time) 7. Add auction end time (20 mins after current time) 8. Press Create Auction button
Expected Results	Message will be generated that you have successfully added the item
Actual Result	As expected
Status	Pass

6.3.2 Place Bid

Table 6.2 Place Bid Test Case

ID	T2
Description	Random Bid will be placed for the item.
Tester	User
Setup	Login through MetaMask, item is already added to the system
Instructions:	1. Enter Bid Amount (2 ether) 2. Enter Amount to Send (3 ether) 3. Enter Secret Text (abcdefgh) 4. Press Place Bid
Expected Results	Message will be generated that you have successfully placed bid
Actual Result	As expected
Status	Pass

6.3.3 Reveal Bid

Table 6.3 Reveal Bid Test Case

ID	T3
Description	The amount that was placed as a bid on item will be revealed
Tester	User
Setup	Login through MetaMask, User has added bid for the item before auction time ends
Instructions:	1. Enter Bid Amount (2 ether) 2. Enter Secret Text (abcdefgh)
Expected Results	Message will be generated that your bid is successfully revealed
Actual Result	As expected
Status	Pass

6.3.4 Finalize Bid

Table 6.4 Finalize Bid Test Case

ID	T4
Description	A user except buyer or seller will finalize the auction
Tester	User
Setup	Login through MetaMask, Bid reveal time is over
Instructions:	1. Press Finalize Auction Button
Expected Results	Message will be generated that auction is finalized successfully and winner is decided
Actual Result	As expected
Status	Pass

6.3.5 Release Amount to Seller

Table 6.5 Release Amount to Seller Test Case

ID	T5
Description	User will vote for release amount to seller
Tester	User
Setup	Login through MetaMask and use three accounts of buyer, seller and arbiter
Instructions:	1. Select one of three accounts and press Release amount to seller 2. Repeat step 1
Expected Results	Message will be generated that amount has been released from escrow and sent to its rightful honour
Actual Result	As expected
Status	Pass

6.3.6 Refund Amount to Buyer

Table 6.6 Refund Amount to Buyer Test Case

ID	T6
Description	User will vote for refund amount to buyer
Tester	User
Setup	Login through MetaMask and use three accounts of buyer, seller and arbiter
Instructions:	1. Select one of three accounts and press Refund amount to buyer 2. Repeat step 1
Expected Results	Message will be generated that amount has been released from escrow and sent to its rightful honour
Actual Result	As expected
Status	Pass

6.3.7 Search Auction

Table 6.7 Search Auction Test Case

ID	T7
Description	User will search the item being auctioned by using category
Tester	User
Setup	Login through MetaMask
Instructions:	1. Select category Cell Phones and Accessories
Expected Result	Items related to that category will be displayed
Actual Result	As expected
Status	Pass

This chapter has given the complete description of a software testing of the system. It has elaborated the test case approach, test plan and test case of the features that need to be tested. This chapter further explained the testing approaches, tools and environment for testing a system, and finally contains the detailed description of the test cases of the product.

Victory belongs to the most preserving.”   Will Durant (1769-1821)

Chapter 7

Conclusion and Future Enhancements

This document describes the project conclusions and future enhancements i.e. what type of new features can be added with time.

7.1 Summary

This application has removed the need of 3rd party to run web application. This application is completely decentralized using the power of blockchain technology. People can buy and sell products without depending upon any third-party service like eBay. Smart Contracts allow this application to run in trust-less and decentralized environment.

7.2 Conclusion

- People no longer need to provide their private information like name, password, email address to access features of website.
- People can buy and sell products without giving away their private information to anyone.
- Large data like product images and description are not stored in blockchain. IPFS is used to store them and their link in the form of hash is stored in the blockchain. As a result, there will be no burden on blockchain to store large amounts of data.
- Blockchain is not used to query products and display them on web. Mongo-Db is used instead. There may be thousands of products on the website, so querying them from blockchain again and again will be very much hectic. The problem of handling large amount of data is solved
- Multisignature Escrow service in the form of Smart Contract is implemented to avoid any fraud from seller or product related problems
- Need for third party to run a website is removed.

7.3 Future Enhancements

This technology is still under development and research. Future enhancements can be

- Escrow service should be created in a different way in future
- Blockchain application for android are not yet possible. But in future it will be, so this application can be developed for iOS and android
- Products tracking and shipping service can also be implemented to help user track his package

References

[1] P. Mohapatra. “Chapter 10 Software Requirement Specifications” in *Software Engineering lifecycle approach,* 1st edition, New Age International (P) Ltd., Ansari Road, New Delhi: New age international (P) limited, publishers, 2010, pp. 211-228.

[2] P. Mohapatra. “Chapter 14 Object-oriented Design” in *Software Engineering A lifecycle approach,* 1st edition, New Age International (P) Ltd., Ansari Road, New Delhi: New age international (P) limited, publishers, 2010, pp. 295-321.

[3] Craig Larman “Chapter 6 Use cases and Functionl Requirement” in *Applying UML and Patterns,* 3rd edition, Don O’Hagan, 75 Arlington Street Suite 300 Boston MA 02116 USA: Addison Wesley Publishing, 2004, pp. 115-174.

[4] Yogesh Singh, Ruchika Malhotra “Chapter 6 Object-Oriented Design” in *Object-Oriented Software Engineering,* 3rd edition, Asoke K. Ghosh, Rajkaml Electric Press, Plot no. 2, phase 4, HSIDC, Sonepat, Haryana: PHI Learning Private Limited, 2012, pp. 203-259.

[5] Satoshi Nakamoto, “Bitcoin White paper”. Internet: https://bitcoin.org/bitcoin.pdf, Oct. 31, 2008.

[6] Eric Petroff, “Ethereum White Paper”. Internet: https://www.ethereum.org, July 30, 2013.

[7] John V. Duca, “Subprime Mortgage crisis”. Internet:

https://www.federalreservehistory.org/essays/subprime_mortgage_crisis, July 10, 2013

[8] Investopedia Staff, “The Collapse of Lehman Brothers” Internet:

https://www.investopedia.com/articles/economics/09/lehman-brothers-collapse.asp, December 11, 2017

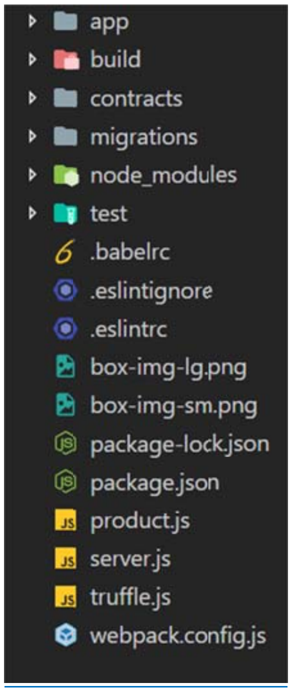
[9] Ethereum Project Solidity team, “Solidity Documentation” Internet:

https://solidity.readthedocs.io/en/develop/, Feb 1, 2017

User Guide

Environment Setup

1. Install Ganache Blockchain GUI software from https://truffleframework.com/ganache
2. Download Meta-Mask Plugin from https://metamask.io. It is only available for Google Chrome and Mozilla FireFox.
3. Create Truffle Project using windows Command line using these commands:
    - mkdir any name (give name to folder)
    - cd any name
    - truffle unbox webpack
    - rm contracts/ConvertLib.sol constracts/Metacoin.sol
    - npm install
4. Download Inter Planetary File System from https://dist.ipfs.io/#go-ipfs
    - Enter these commands in cmd
    - Ipfs init
    - Ipfs daemon (to start ipfs at localhost)
5. Download Mongo-Db community edition from: https://www.mongodb.com/download-center. For installation follow tutorial on this link: https://docs.mongodb.com/master/tutorial/install-mongodb-on-windows/
6. User any editor for coding. I recommend Visual Studio Code. Download from here: https://code.visualstudio.com/
7. Copy Private key of any account from Ganache and import it in to Meta-Mask to create account.
8. Your project directory should look like this:



```
app
build
contracts
migrations
node_modules
test
.babelrc
.eslintignore
.eslintrc
box-img-lg.png
box-img-sm.png
package-lock.json
package.json
product.js
server.js
truffle.js
webpack.config.js
```

Figure 7.1 Project Directory

9. Go to package.json file and enter this lines:
    - “ethereumjs-util”: “5.1.2”
    - “nodemon”: “^1.11.0”
    - “mongoose”: “4.11.7”
    - “ipfs-api”: “18.1.1”
    - Go to cmd and enter npm install
10. Your environment will be ready. For integration of mongo-Db and blockchain create a server.js file and use these commands:

```
var ecommerce_store_artifacts = require('./build/contracts/EcommerceStore.json')
var contract = require('truffle-contract')
var Web3 = require('web3')
var provider = new Web3.providers.HttpProvider("http://localhost:7545")
var EcommerceStore = contract(ecommerce_store_artifacts)
EcommerceStore.setProvider(provider)

//Mongoose setup to interact with the mongodb database
var mongoose = require('mongoose')
mongoose.Promise = global.Promise
var ProductModel = require('./product')
mongoose.connect('mongodb://localhost:27017/Auction_dapp')
var db = mongoose.connection
db.on('error', console.error.bind(console, 'MongoDB connection error:'));

// Express server which the frontend with interact with
var express = require('express');
var app = express()

app.use(function (req, res, next) {
  res.header('Access-Control-Allow-Origin', '*')
  res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type, Accept')
  next()
})

app.listen(3000, function() {
  console.log('Auction House Ethereum server listening on port 3000!');
})
```

- No need to use same names like Ecommerce or auction. You can use your own names according to your project

- Open 3 cmds
- In first cmd give command “mongod”
- In second cmd give command “mongo”
- In third cmd give command “node server.js”
- Mongo-Db server will start

11. Run front end using command in cmd: “npm run dev”
12. It is recommended to install Git bash for running commands instead of using cmd.
13. For coding smart Contracts use this online compiler: https://remix.ethereum.org/ and then save the file in your project contracts directory
14. After coding your smart contracts, start ganache blockchain using command: “truffle migrate --compile-all --reset --network development”