VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

THE INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Research on Non-Fungible Token (NFT) and application in Real Estate

Advisor: Dr. TRAN THANH TUNG

Student name: VO HO NHAT QUANG

A thesis submitted to the School of Computer Science and Engineering

in partial fulfilment of the requirements for the degree of

Bachelor of Computer Science

Ho Chi Minh City, Vietnam

2021

A research on Non-Fungible Token (NFT) and application in Real Estate

APPROVED BY:

_____

Tran Thanh Tung, Ph.D.

_____

_____

_____

_____

THESIS COMMITTEE

ACKNOWLEDGMENTS

I would love to convey my profound gratitude to Dr. Tran Thanh Tung for his insightful and valuable guidance during the planning and researching of this thesis. His unwavering encouragement and support aided me in achieving my goal. Moreover, I would like to express my sincere appreciation to those helped me during the accomplishment of this thesis essay.

LIST OF ABBREVIATIONS

ABI	Application Binary Interface
NFT	Non-Fungible Token
ICO	Initial Coin Offering
IPFS	Interplanetary File System
API	Application Programming Interface
CLI	Command Line Interface
DOM	Document Object Model
EOA	Externally Owned Account
ERD	Entity Relationship Diagram
EVM	Ethereum Virtual Machine
I/O	Input / Output
NPM	Node Package Management
P2P	Peer-to-Peer
RPC	Remote Procedure Call
UI	User Interface
APY	Annual Percentage Yield

ABSTRACT

Real estate is one of the vital economies of any country. However, there are still issues in the current system from ownership verification, lease agreements, sale process, transactions, third-party dependence and so on. The blockchain and real estate can work together. The real estate procurement process adopts the blockchain model, to strongly empower land industry and put a premise for blockchain future. For the most part, the downside of digitized data is facilitated on different frameworks, which causes the absence of transparency and a higher occurrence of potential fraudulence. Blockchain can resolve it. The blockchain tokenization for estates methods is a new approach for organizations that are consistently uncovering a part of these realities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1.1 Background

The continuous development of industry 4.0 has brought outstanding inventions in the field of digitization, and blockchain is the representative. Since its first appearance, it has changed everything that we have known about the traditional operation of services by offering an open ecosystem where transactions can be read and made publicly. The very first application of Blockchain has been exploited in digital currency and finance, and then it has constantly expanded to health care, supply chain monitoring and internet of things. The concept of Initial Coin Offerings (ICOs) emerges and poses challenges on stock market, start-up loans and venture capital. By leveraging the potential of Blockchain, many enterprises have reduced the workload in their internal business process and significantly improved performance. Apparently, real estate can't miss out the blockchain disturbance either.

1.2 Problem Statement

Real estate exchange through digital platforms has unprecedentedly been the norm. Traditionally, transactions will be conducted offline involving face-to-face engagements between stakeholders. However, this can be changed when Blockchain comes into the play. It enables everyone to make transactions remotely and comprehensively through an online marketplace which helps buyers can purchase out of state more conveniently.

Brokers or third-party property aggregators have long been part of the real estate ecosystem. It will take an additional high fee charge for both buyers and sellers to pay for the intermediaries before their transactions conclude. New platforms with Blockchain applied now can cut out the role of intermediaries and save on commissions. This also improves the document process compact and faster.

Scamming in exchanging real estate has always been a pain point for traders. It happens at many different stages of transaction including appraisal, closing or even foreclosure proceedings. Crime can be committed in a variety of forms such as identity fraud or performing multiple transactions on a property. With the transparency and consistency in the data storage that

Blockchain has brought to the system, investors now can rest assured before making their decision.

Furthermore, as we have known that it usually takes quite some time for the documents and paperwork be prepared before the buyer can actually own the property, which may cause inconvenience for all the relevant parties and make the market illiquid. But as tokens, real estate can be readily traded.

1.3 Objective

The thesis's ideal is about giving a thorough and overall views of Blockchain Ethereum technology, how Blockchain takes part and adapt in the current real estate system and what kind of achievement that it brings to the process.

This thesis will walk us through all the fundamental information of the system from the system requirements, system architecture and design, to the very last part where users can actually interact in the system through the implementation.

Specifically, the output of this project is the very first version of the real estate management and exchange application by using Blockchain including:

- Real Estate Tokenization
- Create a marketplace for trading, negotiating NFT
- Ownership Verification

1.4 Assumption

The context of this thesis proposes that all the required documents of the property has been legally digitalized by the government and the usage of the system is approved.

CHAPTER 2. BACKGROUND AND REALTED WORK

2.1 Background

2.1.1 Blockchain Fundamentals

At the most basic level, blockchain is a system of information management and storing in such a way that no entity can tamper or corrupt data in it. We can consider it as a digital ledger of transactions which is distributed to a specific network of systems. Each block of the chain holds a number of transactions, and whenever a new transaction occurs, a record of that transaction will be added to every participant's ledger [1].

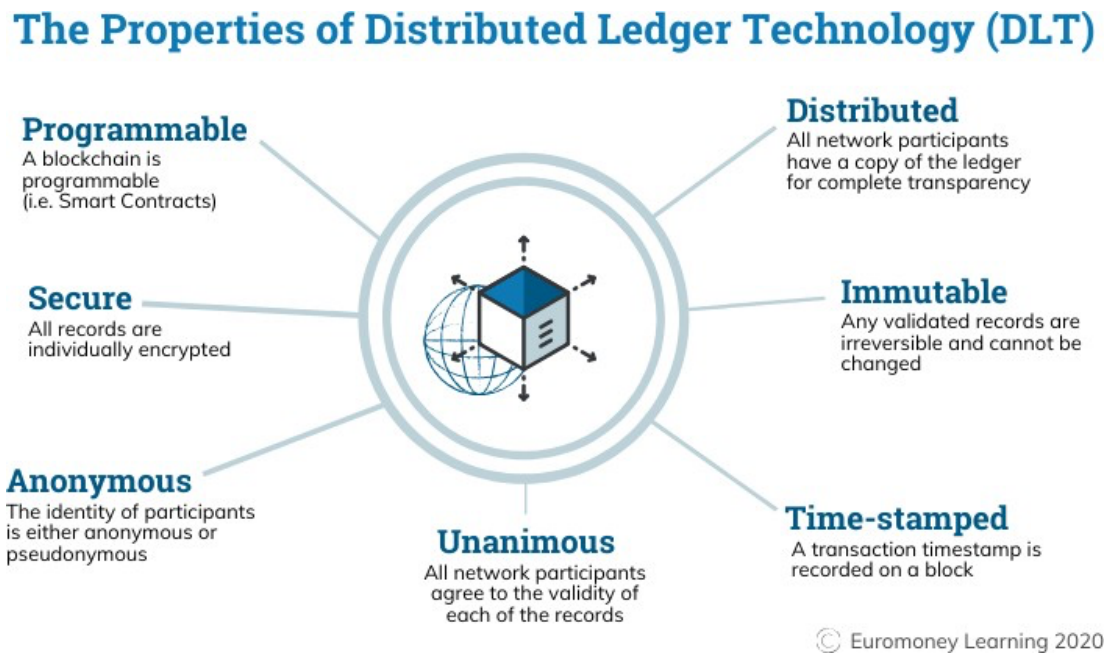


Figure 2 - 1. Distributed Ledger Technology (DLT) Properties

The goal of the blockchain is to create a distributed, decentralized, and trusted record of the history of the system. The emergence of Bitcoin is a huge achievement for Blockchain. Since Bitcoin first's launch, Blockchain technology has been thriving rapidly. Currently, the term

“Blockchain” no longer applies only to Bitcoin in particular or to cryptocurrencies in general. In fact, there are certain blockchain technology support scripting through smart contract, allowing for specific rules to be applied to the logic of system, but we will get to it later. Before learning the meaning of Blockchain in real estate, there are some basic terminologies about Blockchain that we need to understand.

a. Hash Function

A hash function is a mathematical process that takes input data of any size, performs an operation on it, and returns output data of a fixed size. The hash value is unique and irreversible, which means that no matter how many times we pass the input to hash function, the same input will have one and only one hash value, and it cannot be traced back to original content from the hash.

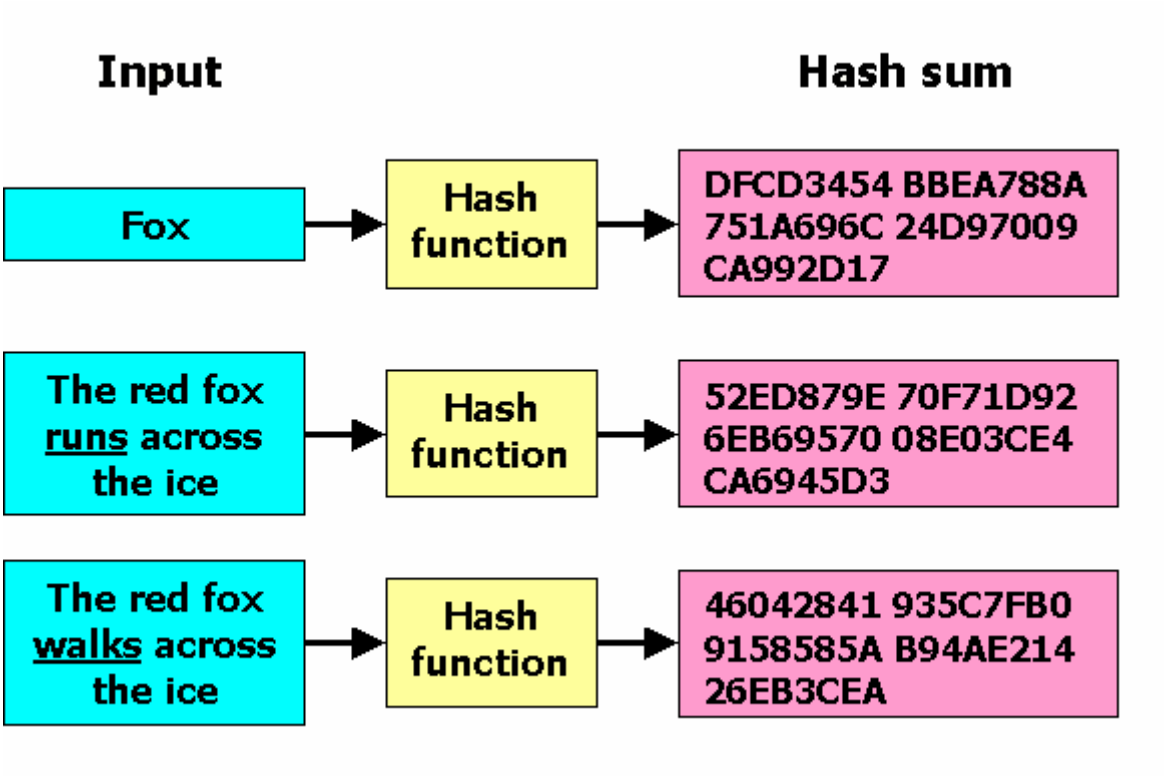


Figure 2 – 2. Hash Function

There are some properties of cryptographic hash function:

- Collision free: It's infeasible to find two inputs which can produce the same outputs.
- Hiding: Even when output is revealed, the input is still untraceable.
- Puzzle-friendly: The difficulty of brute-forcing through all possible outcomes of input.

b. Hash Pointer

Hash pointer is part of a block. It holds the hash of the previous block. The meaning of hash pointer is to make sure that data from the previous block is immutable. To put it simpler, any change of data in previous block would change the hash value of that block, which subsequently cause the change of hash pointer in current block.

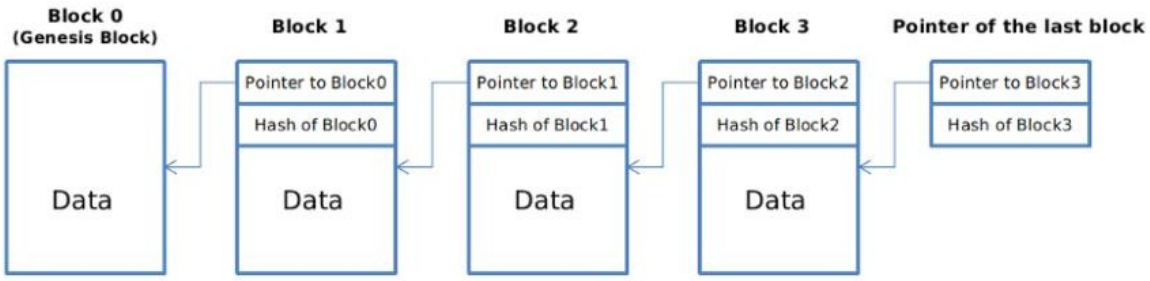


Figure 2- 3. Linked List

Hash pointer helps us check the integrity of each block and keep track of the tamper evident of the blockchain. In Figure 2 – 3, assume that an attacker A tampers content of Block 1, and because of the "collision free", Block 1 now generates a new hash. To avoid the inconsistency, A must modify the pointer of Block 1 in Block 2 also, which causes the change of generated hash from Block 2. If A keeps modifying each block and bumps into the final block, it will be a dead end for A because we already had the hash pointer of the last block.

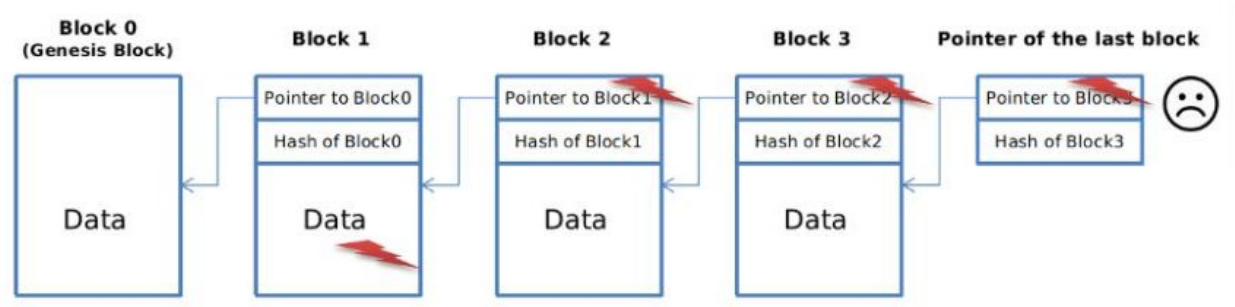


Figure 2 - 4. Tamper Evidence Property

c. Merkle Tree

Merkle tree is a binary tree building with hash pointers. The leaves are data blocks, nodes further up in the tree are the hashes of their respective children. With Merkle Tree, we only need to know the hash pointer of the root (top level node), then we can traverse down to any data block and check its integrity. Merkle Tree also helps improving the efficiency while linked list blockchain traversing with complexity O(n), now we only take O (log n). Moreover, if Merkle Tree is sorted, we can easily check whether a given data existed in the tree.

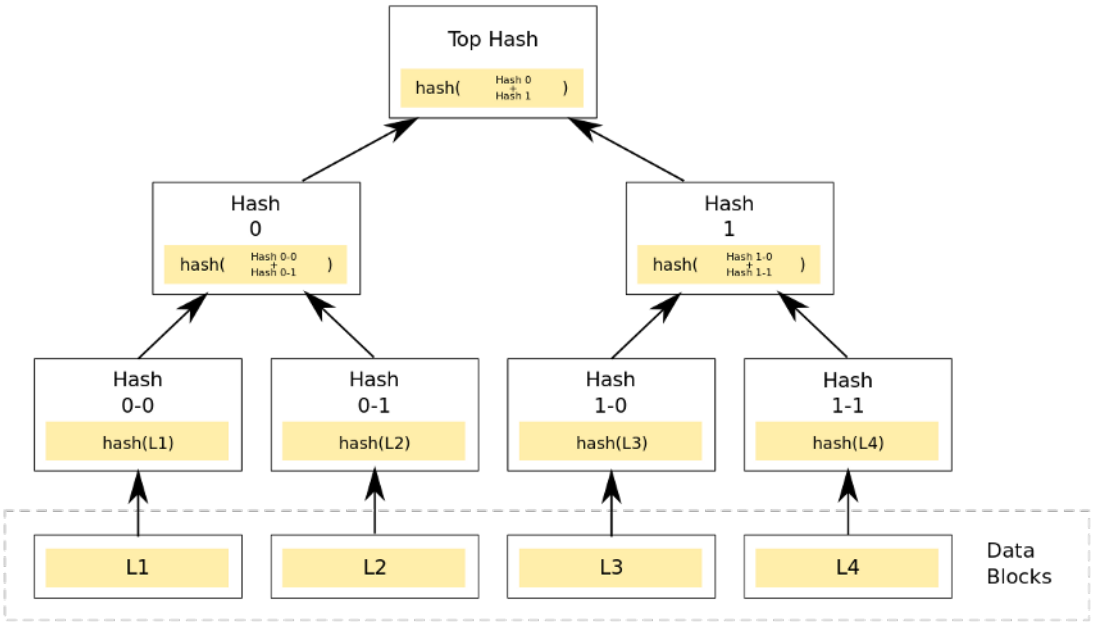


Figure 2 – 5. Semantic Merkle Tree

Merkle tree totals all transactions in a block and generates a digital fingerprint of the entire set of operations, allowing the user to verify whether it includes a transaction in the block. It is built up from the bottom by using transaction's IDs, which are the unique hashes of transactions, and then every adjacent pair of hashes will be hashed again to generate a new hash called non-leaf node. It will be executed repeatedly until only one hash remains, which is the Merkle Root.

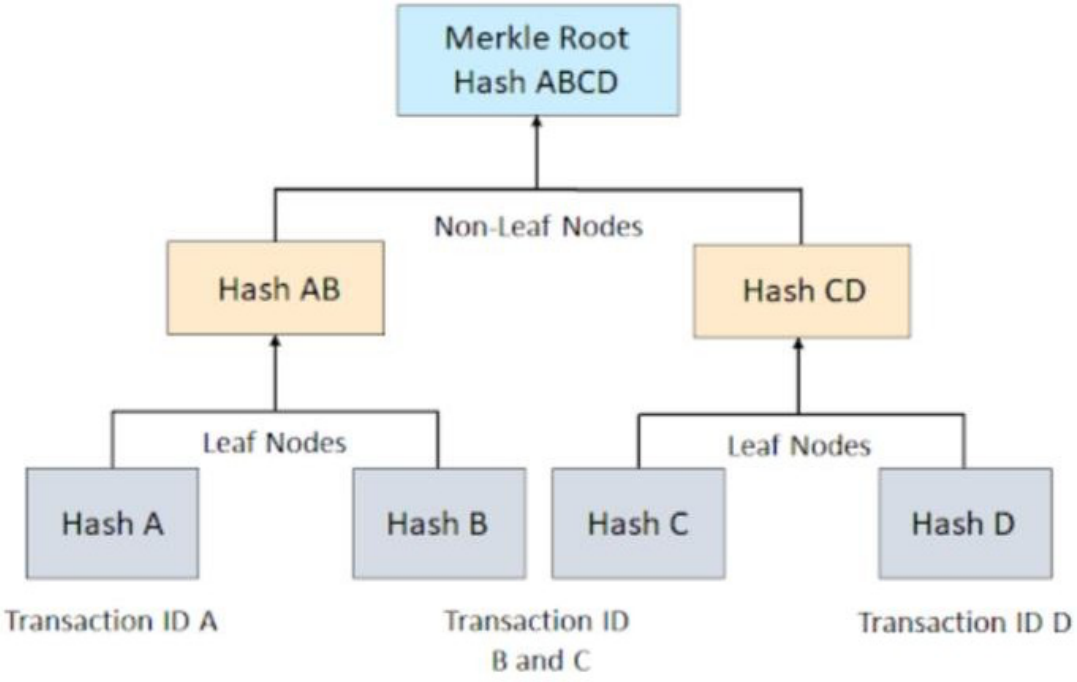


Figure 2 – 6. Merkle Tree Hashing Process

d. Mining Mechanism

Mining involves Blockchain miners who add transaction data to the global public ledger of past transactions. In the ledgers, blocks are secured by Blockchain miners and are connected to each other forming a chain [2].

A miner is a node in the network who collects transactions and organizes them into blocks. Whenever transactions are made, all network nodes receive them and verify their validity. Miner then collects all the transactions pending from the memory pool and begin assembling them into a block, which is called candidate block. In a little more detail, candidate block is a block that a

mining node is trying to mine in order to receive the block reward. Hence, this block may be either validated or discarded. Here is the process of mining:

1. Miner initializes the very first transaction of the block where they send themselves the mining reward, which is called Coinbase transaction.
2. Miners hash each transaction token in the memory pool.
3. Hashes value are organized into a Merkle Tree, where a pair of transactions is put together and hashed again. The outputs are then put together in pairs and repeatedly hashed until the top of the tree reached. The top tree is called root hash which represents all the previous hashes. As mentioned above, this can be used to trace back any leaf in the tree.
4. The root hash, previous block hash, and a nonce are put into block header.
5. The block header is then hashed producing an output which is the identifier for candidate block.
6. To be validated, the output must be less than a certain target value which is predefined by protocol.
7. Miners constantly change the nonce value in block header and hash the whole block header repeatedly until a valid block hash found.
8. Eventually, the successful miner who found the block will broadcast it to the entire network.
9. All other miners then check the validity of that block, if it is valid, then they will add it to their copy of the blockchain and move on to the next block.

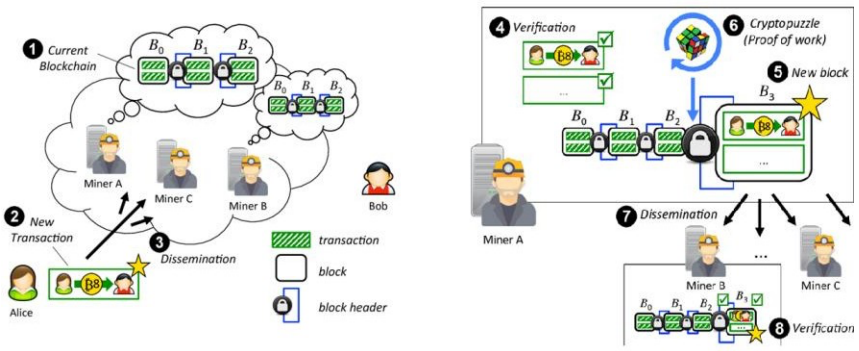


Figure 2 – 7. Mining Mechanism

e. Consensus Mechanism

As we have known so far, in blockchain ecosystem, transactions will be grouped into blocks and broadcasted through the network. Still, this only happens because the network participants have come to an agreement, or consensus on the validity of transactions. Because of the decentralized nature of Blockchain, consensus mechanism is indispensable. However, to achieve the consensus, it usually takes quite some time to process. One of the most well-known mechanisms is Proof of Work (PoW).

In PoW method, miners compete to solve the computational puzzle to generate a hash, which takes all unconfirmed ledger transactions as input. The hash puzzle will be easy to verify, but extremely hard to solve. Nodes in the network will be the judge, they verify the hash puzzle broadcasted and if it passes, the hash owner will receive a reward. Despite the high security of this method, it is environmentally taxing and highly inefficient. It takes around ten minutes to generate a hash puzzle and 4-5 transactions per second. Comparing to Visa network which can process up to 1700 per second, it will be a huge problem on large scale. Moreover, by consuming electricity for computational hardware, it is significantly harmful to the environment.

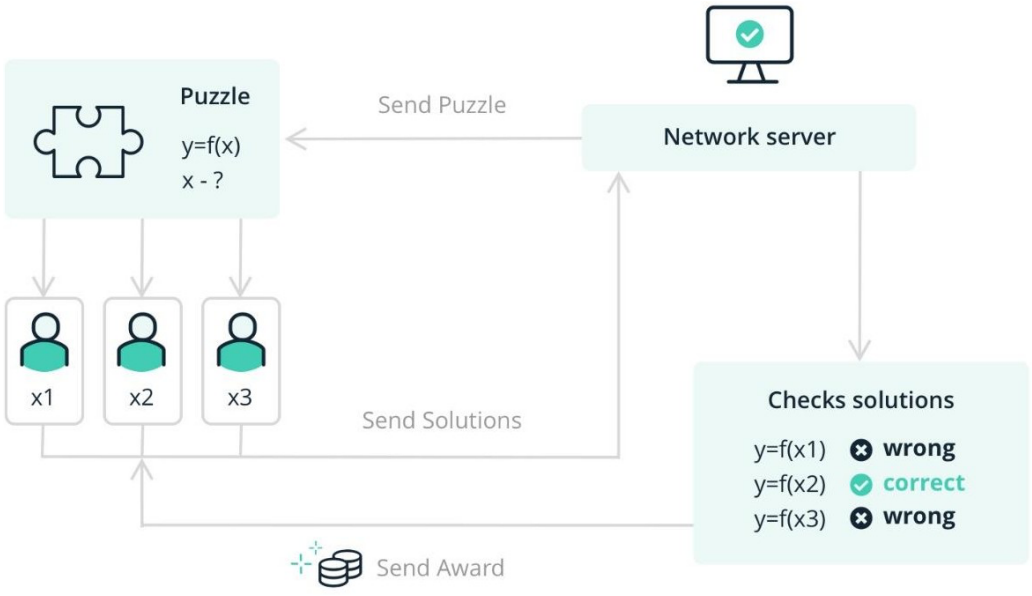


Figure 2 – 8. Proof of Work Mechanism

f. Block Structure

Every block in a chain shares exact same structure including:

- Block Size: the size of the block in bytes
- Block Header:
    - Version: used to track software protocol upgrades
    - Previous Block Hash: a reference to the parent block in chain
    - Merkle Root: hash value of the root of Merkle tree of current block transactions
    - Timestamp: created time of block
    - Difficulty Target: Proof of Work (PoW) algorithm difficulty target for current block
    - Nonce: a counter for PoW algorithm
- Transaction Counter: how many transactions are in current block
- Transactions: the list of transactions recorded in current block

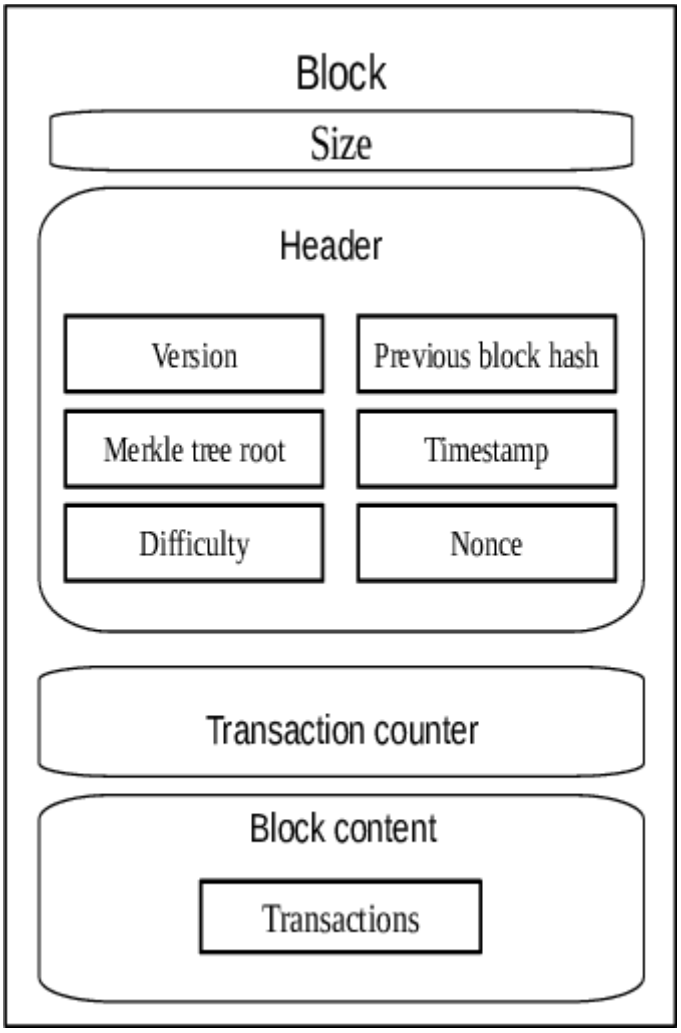


Figure 2 - 9. Block Structure

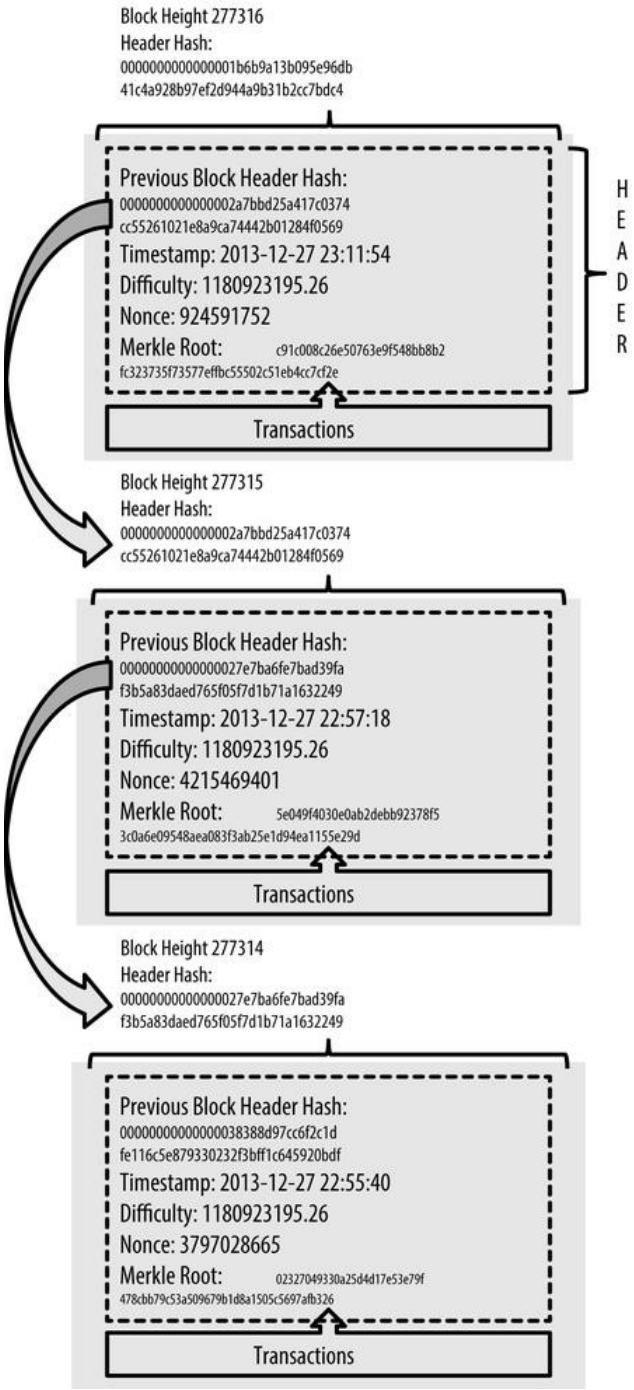


Figure 2 – 10. Blocks Structure in Chain

g. Digital Signature

A digital signature is a cryptographic mechanism used to verify the authenticity and integrity of digital data. It is basically a digital version of the ordinary handwritten signatures, but with higher levels of complexity and security. It is a unique code that attached to a message or document. After the message was sent, it will act as the proof that the message has not been tampered. There are two kinds of keys: public key and private key.

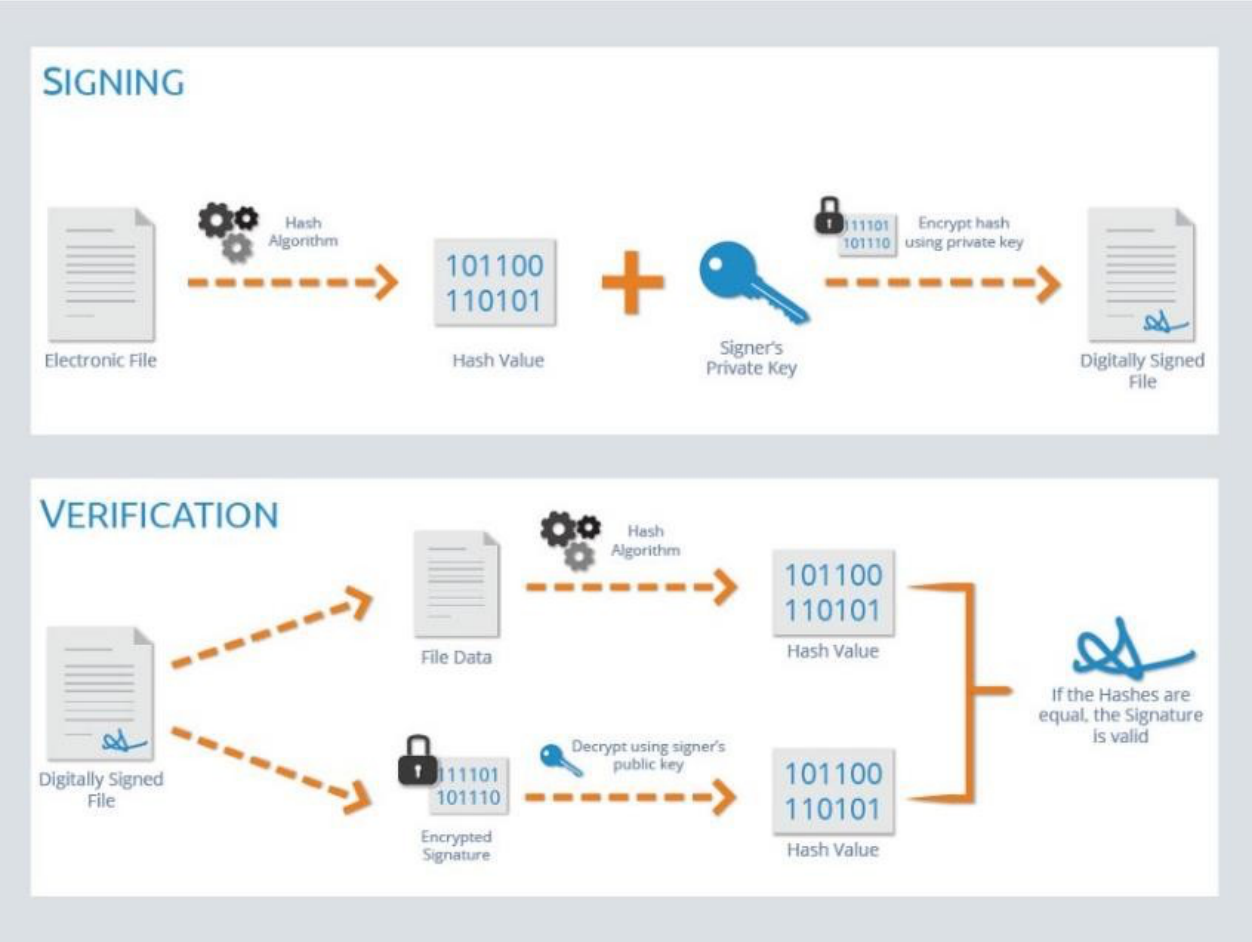


Figure 2 – 11. How keys be used in Digital Signature context

2.1.2 Non-Fungible Token (NFT)

NFTs started out as a craze, but have been all the rage in recent years. In March 2021, Beeple, a digital artist had his work of art called Everydays: The First 5000 Days sold for $69 million at Christie's Auction House which positioned him to top three most valuable living artists, according to the auction house [3]. What sets this auction apart is that it was sold as a non-fungible token. Since then, NFTs have been constantly thriving and becoming part of the pop culture. In the late 2021, Collins Dictionary claimed that NFT is the word of the year. However, many people still doubt that NFTs are just a marketplace for digital art and consider them as gambling or even a criminal act, money laundry to be specific.

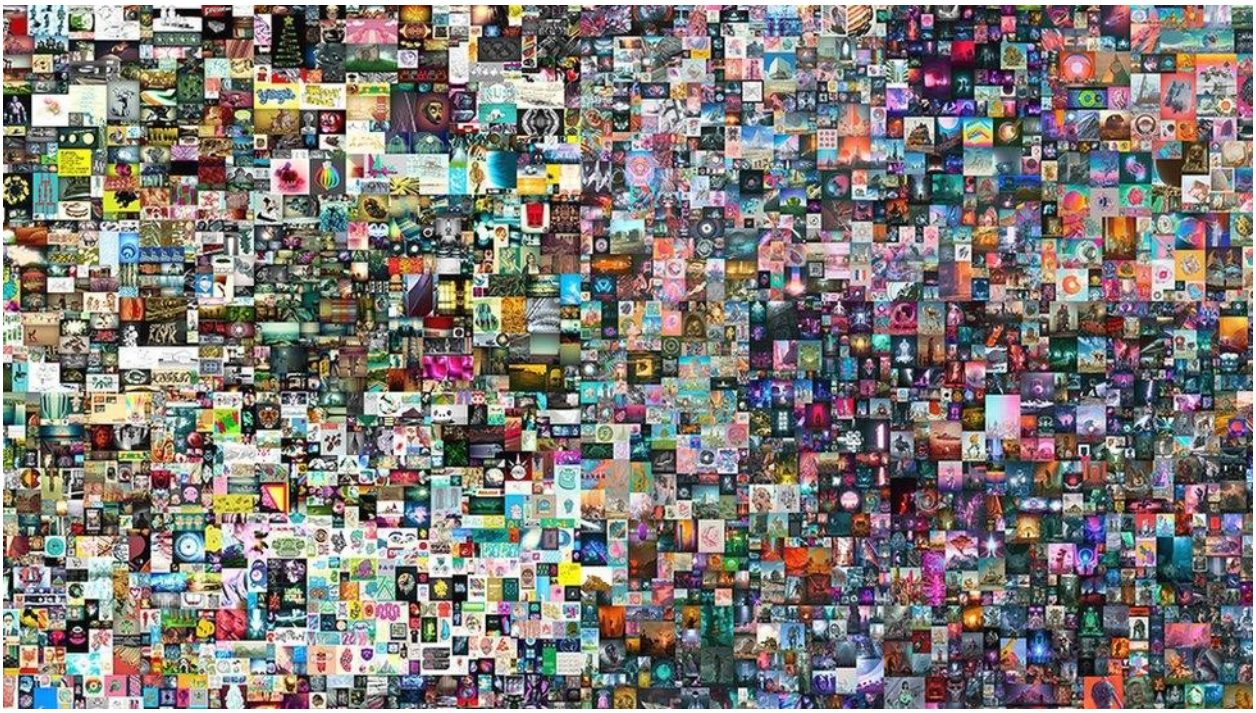


Figure 2 – 12. Everydays: The First 5000 Days

In short, Non-Fungible Token is one of a kind and cannot be replaced with. It exists in a wide variety of forms such as art, gifs, game items, tweets or even this thesis paper. NFT has its own value, just like how physical art works, these assets can be bought, sold and traded by using cryptocurrency. What makes NFT stand out is the ability to provide proof of ownership and non-fungibility. Let's say that Alice has a 10$ bill, and Bob wants to trade it with two separate bills of

5$, it is totally fine. However, it is not applicable to non-fungible token. We cannot trade evenly a signed limited-edition book with a standard one.

Blockchain and NFT aims to bring content creators and artists a chance to monetize for their work. Artists no longer have to depend on the intermediaries to sell their artwork. Instead, they will interact directly with the buyer, which helps them have full control over the profits. Moreover, with the royalties program, they will be able to earn extra credits whenever the artwork is sold to a new consumer. This is a promising feature that makes NFT exchange different from traditional process.

What is the meaning of owning an NFT when people can easily copy and save it is frequently asked question. In fact, it is indeed accessible by anyone. However, that does not give them commercial rights, nor does make them the owner. Since NFTs are stored on public chain, the ownership record will be public as well. Hence, the copies will hold no value than just an image.

2.1.3 Ethereum and Smart Contract

Ethereum is basically Bitcoin's digital cousin. However, Ethereum is not merely a cryptocurrency like Bitcoin, it is also decentralized computing platform. We can deploy our code on Ethereum, and interact with applications created by other users. The ideal of Ethereum is that it creates a playground for developers creating and launching their code which run across a distributed network instead of existing on a centralized server.

a. Accounts

There 2 two types of accounts in Ethereum called Externally Owned Account (EOA) and Contract Account. To put it simply, EOA will be owned by a tangible user with their private keys for authority and Contract Account is a smart contract deployed to the network which controlled by code. Both kinds of accounts share the same structure when they all have balance, can send and receive ETH and tokens, interact with deployed contracts. However, there are some main differences:

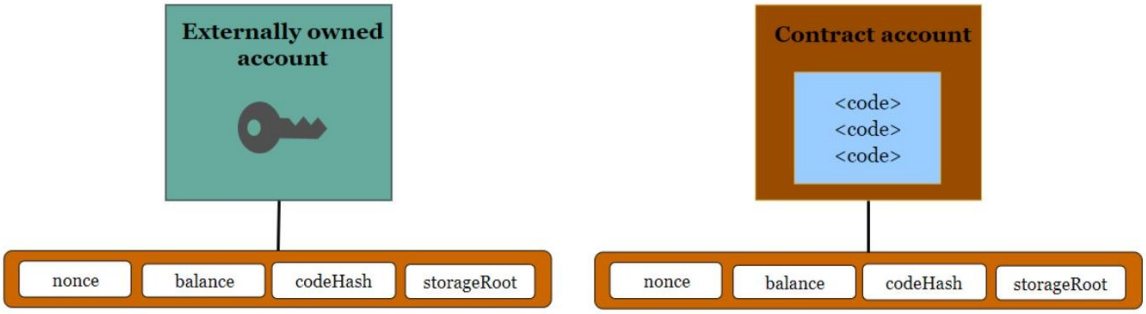


Figure 2 – 13. Accounts Structure

- Externally Owned Account: creating account costs nothing and transactions between EOA can only be ETH.
- Contract Account: cost a small amount of ETH when initiating, an only send transactions in response to receiving a transaction and transactions can be transferring tokens or creating a new contract.

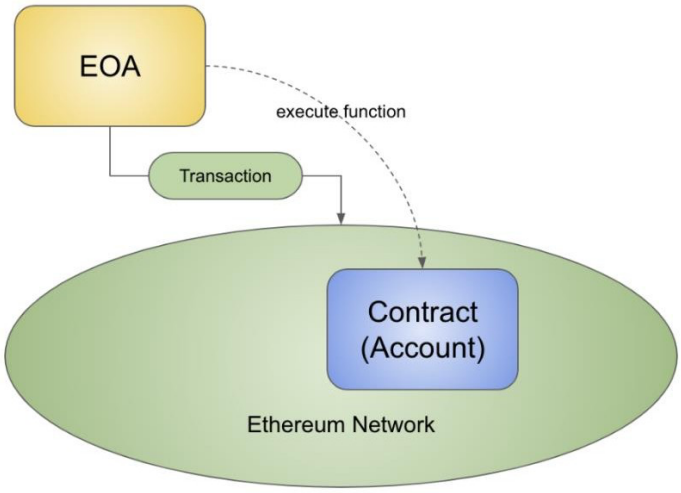


Figure 2 - 14. Interaction between Accounts

b. Ethereum Virtual Machine (EVM)

Ethereum Virtual Machine (EVM) is a powerful, sandboxed virtual stack embedded within each full Ethereum node, responsible for executing contract bytecode [4]. The EVM is essential to the Ethereum Protocol and is instrumental to the consensus engine of the Ethereum system. It allows anyone to execute code in a trust less ecosystem in which the outcome of an execution can be guaranteed and is fully deterministic. To put it simply, Ethereum has one and only one general state, and the EVM is what defines the rules for computing a new valid state from block to block.

c. Ether

The unit of currency in the Ethereum network is called Ether (ETH). Ether exists as an incentive for miners who solved the block puzzle and helps driving up the crypto market.

d. Gas

Gas refers to the unit that measures the amount of computational effort required to execute specific operations on the Ethereum network. We use a smaller unit (gwei) to denote it where **0.000000001 ETH = 1 Gwei**. We have the formula

The total cost of transaction = Gas Limit x Gas Price

Where gas limit is the number of units of gas you are willing to offer per transaction and gas price describes how fast your block will be mined in the next block. To put it simply, if you offer gas limit less than the required amount for transaction, your transaction will fail and the more gas price, the more impressive to miners to mine.

e. Smart Contracts

In simple terms, Smart Contract (SM) is just plain computer code. A persistent piece of code on the Ethereum Blockchain that has a set of data and executable functions. These functions execute when Ethereum transactions are made to them with certain input parameters. Based on the input parameters, the functions will execute and interact with data within and outside of the contract. When we said earlier about Contract Account, Smart Contract is the one we implied,

which means Smart Contract can hold balance and send transactions over the network. EOA can interact with Smart Contract by submitting transactions that execute a function predefined on the deployed Smart Contract.

Smart Contract code is initially written in high-level language that needs to be compiled into EVM bytecode to run. Once the bytecode is pushed into Ethereum storage and the contract gets its own address, an Application Binary Interface (ABI) comes into play as a bridge for interaction. To be more specific, if we have an application written in JavaScript language, for example, in order to interact with deployed contract, we must implement ABI [5].

Smart Contract can be implemented by many developer-friendly languages such as Solidity or Vyper. In this thesis paper, we will implement the system by using Solidity. Remix will be used for deploying and testing purpose.

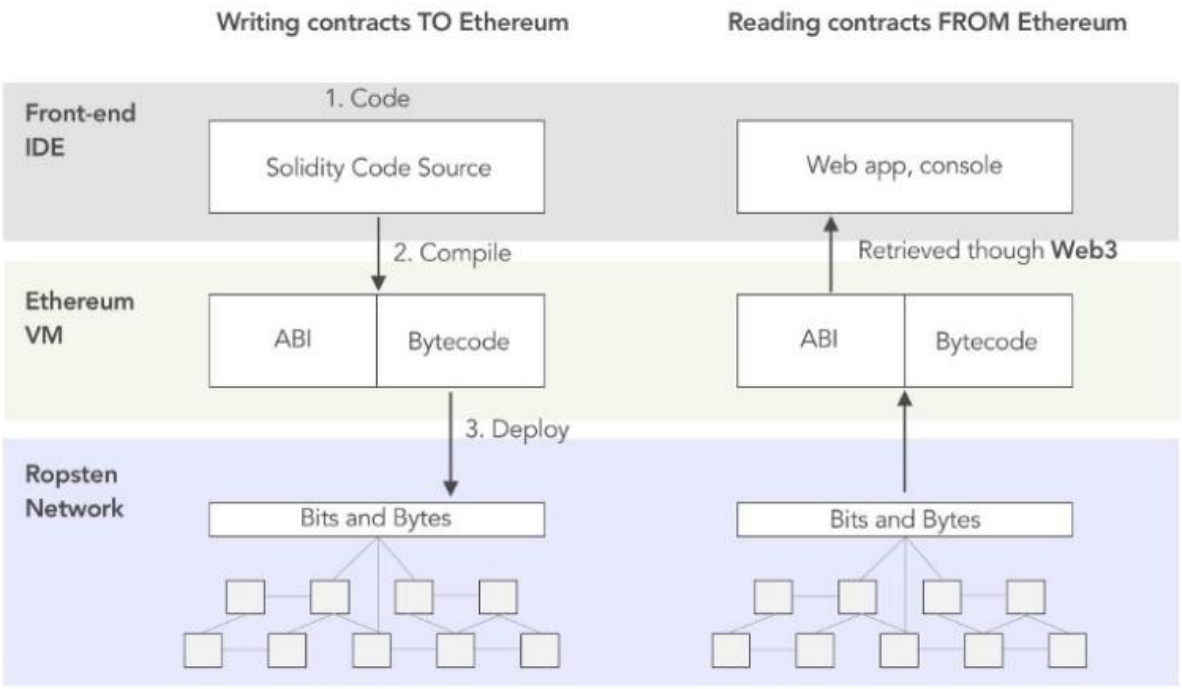


Figure 2 – 15. ABI and Bytecode's interaction

f. Token Standards

As we have mentioned above, by following predefined standards, smart contract is able to operate some basic functions such as creation of tokens, performing transactions, spending and so on. Smart contracts standards are essential since they define rules of utilizing the blockchain network and optimize the interaction between other contracts among the network. Token standards are the subsidiaries of the smart contract standard. For Blockchain that supports the smart contract, token standards are often included to guide everyone how to create, issue and deploy new token based on the underlying blockchain. There are some common standards including:

- ERC-20

  Fungible token standards that provide basic functionality to transfer tokens, as well as allows tokens to be approved. To put it in a nutshell, as fiat currency, ERC-20 tokens are interchangeable with equal values. For instance, one BTC equals all other BTCs. There examples of ERC-20 tokens include LINK, USDT, SHIB, ...
- ERC-721

  In contrast to ERC-20 for fungible token standard, ERC-721 is a token standard that defines non-fungible tokens (NFTs). As we have known so far, NFTs are tokens but interchangeable. They have different traits that set them apart from one another.

Figure 2 – 16. Fungible vs Non-Fungible Tokens

2.2 Related Work

In recent years, despite the fact that there have been many proposals for real estate tokenization as well as blockchain application in procurement process, none of them has been issued by legislature yet. However, there are some prominent projects in the cryptocurrency community that expose the potential for real estate industry.

- Upland

  The mission of Upland is to create a sound economy that blurs the boundaries with the real world, providing a decentralized market where stakeholders interact directly with each other. In Upland, users are able to buy, sell and trade virtual properties based on real-world addresses. Unfortunately, the digital properties collected in Upland don’t have any association or rights to the correlating properties in the real world, however, the addresses do mirror real-life locations. Upland recently launched Upland Legits, which is a new suite of NFT offerings designed to enable brands, personalities, and organizations, such as sports clubs, athletes, artists, entertainment brands, to extend their real-life presence into the Upland metaverse and beyond. "We know that we bring a unique proposition to the dynamic field of NFTs. What sets us

apart is Upland’s connection to the real world in combination with engaging game mechanics driving an almost endless number of compelling use cases for NFTs,” said Dirk Lueth, co-founder of Upland

- Decentraland

  Decentraland was founded in 2017 with the ambition to create a metaverse that gives users full ownership over their virtual assets. In Decentraland, users can buy and sell digital real estate by using MANA, a cryptocurrency that facilitates purchases of LAND, a non-fungible token (NFT) used to define the ownership of land parcels. LAND's owner can build structures then sell or lease it for profit. There are many contents that can be constructed on land such as casinos, art galleries, clubs or even gaming applications. What set Decentraland apart from any other VR platforms is the significance of owning an estate. Since Facebook's name change event and the announcement that they will start developing the metaverse, Decentraland has achieved an unstoppable growth. Recently, a plot of Decentraland has been sold for more than 2 million dollars [5].

CHAPTER 3. METHODOLOGY

3.1 System Description

ERC-721 will be the token standard for tokens in the system. In this thesis paper, system will be first launched on Ropsten Network which is an Ethereum test network for development and testing purpose before being deployed to Mainnet. To exchange NFT, user can store their cryptocurrency in a crypto wallet such as Metamask. All the metadata of tokens will be stored on a decentralized storage called Pinata. User Interface must be approachable and intuitive for user. Hence, Single Page Application would be perfect for user experience. In order to reduce the workload for smart contract, server and centralized database will be in charge of storing and retrieving data off-chain. However, for the purpose of introducing NFT and Blockchain in Real Estate Management and Exchange, server will be only used for uploading metadata to Pinata and all records will be retrieved directly on chain.All the transactions of the system will be recorded on Ropsten Blockchain.

3.2 Use Cases

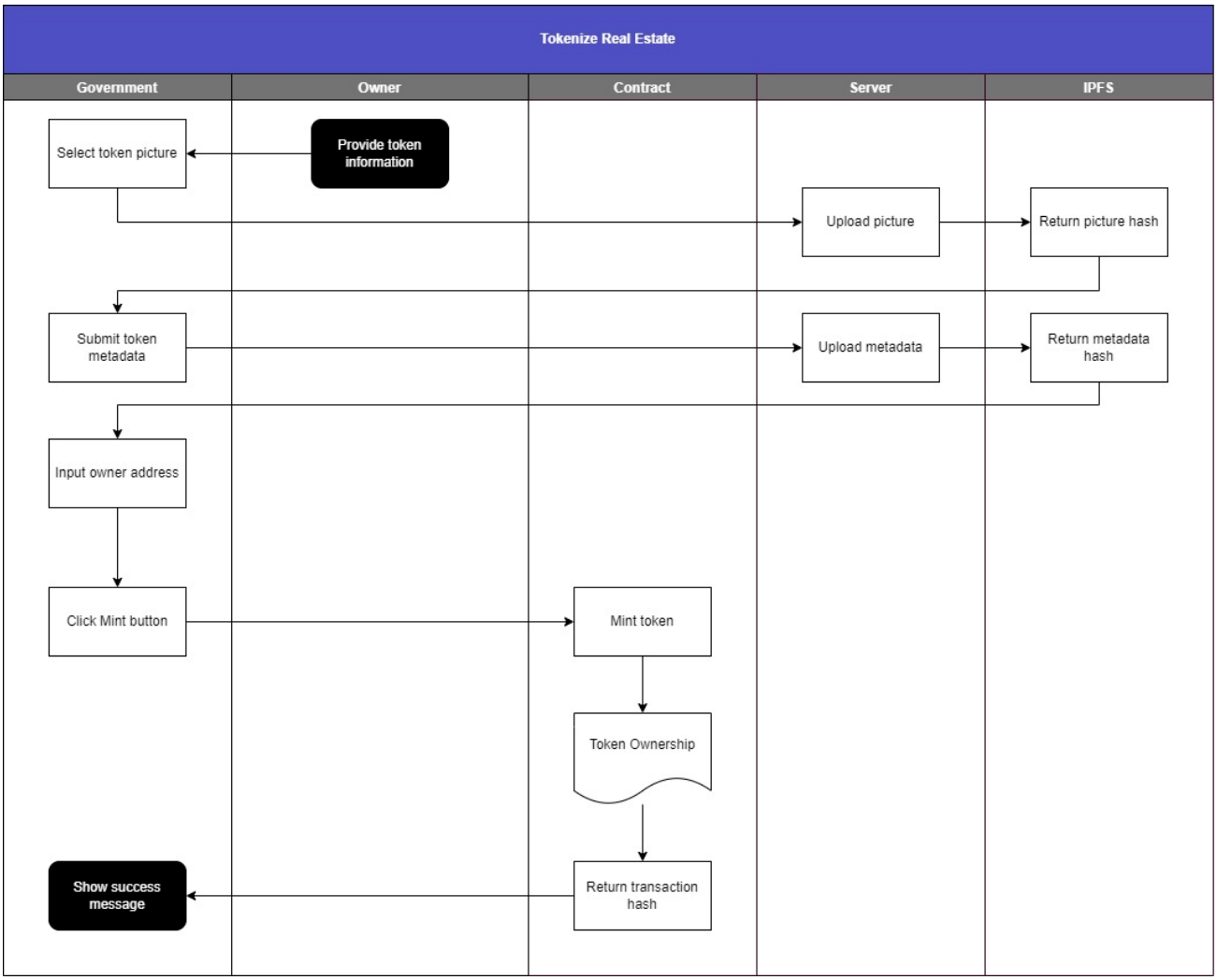


Figure 3 – 1. Tokenize Real Estate

Name	Mint Token
Description	Create new token based on real-world information of real estate
Actors	System owner
Pre-conditions	1. User sign in wallet 2. User's address is valid (system owner) 3. Real estate information is validated
Main course	1. Upload real estate information in image format (owner identification, interior design, outdoor design, household registration book, ..) 2. Fill in real estate information (address, rooms, type, ..) 3. Fill in owner's address 4. Confirm 5. Cotract mint token with submitted metadata 6. Success notification

Table 3 – 1. Use case description – Mint token

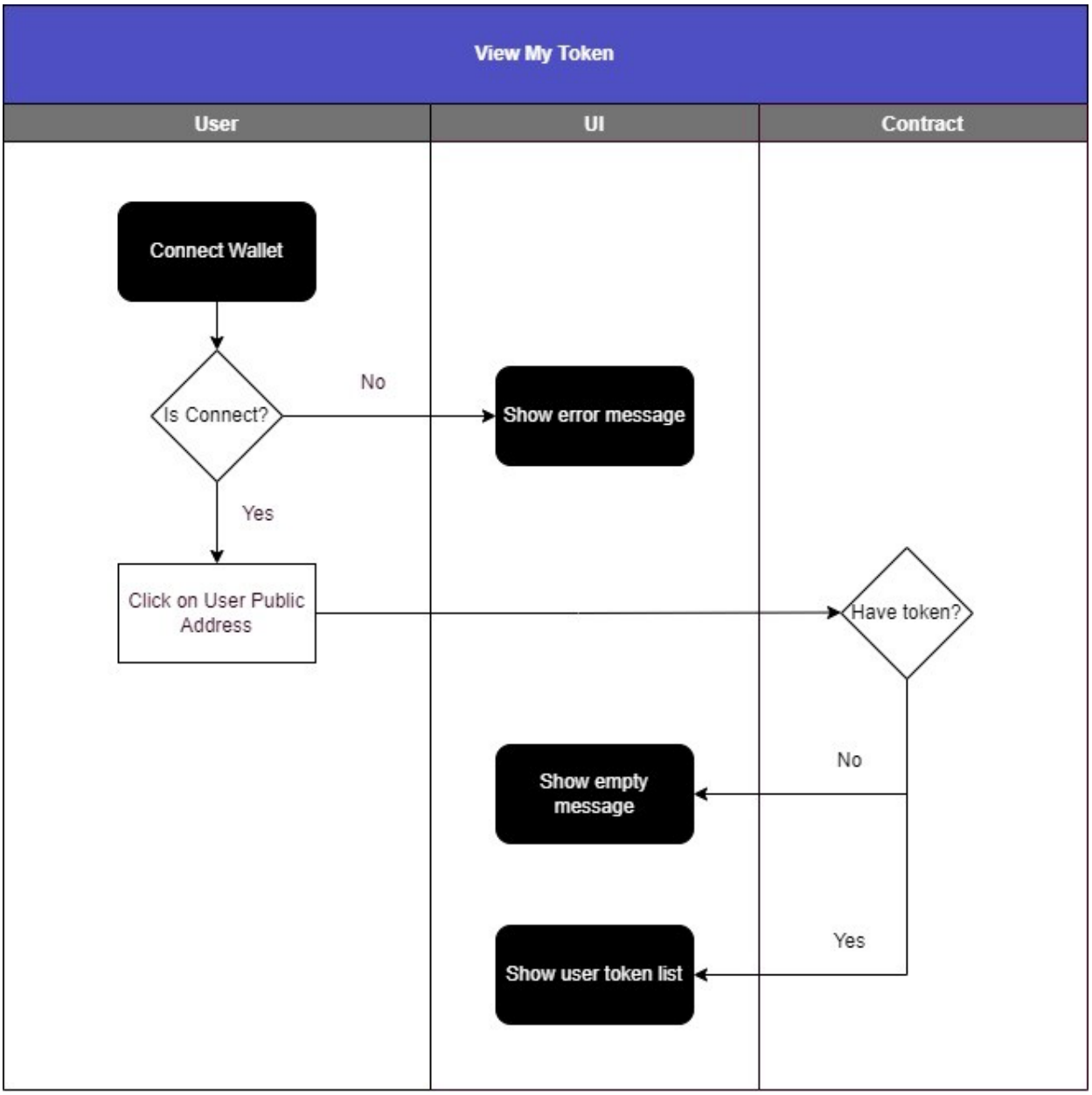


Figure 3 – 2. View My Tokens

Name	View My Tokens
Description	Owner view their tokens
Actors	Owner
Pre-conditions	1. User sign in wallet
Main course	1. User click on their address 2. System query from smart contract 3. Show owner's tokens
Alternative	Show empty message

Table 3 – 2. Use case description – View my tokens

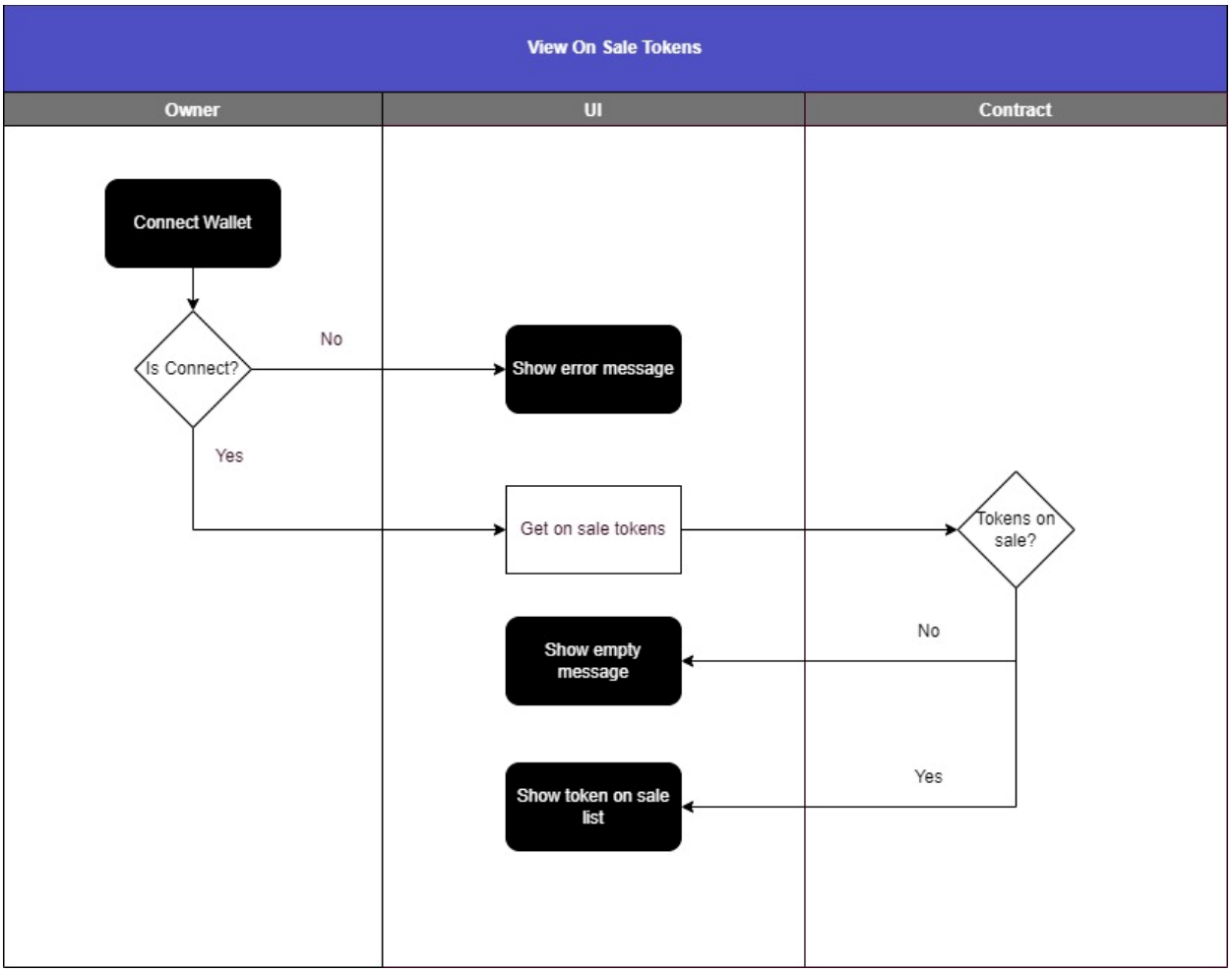


Figure 3 – 3. View On Sale Tokens

Name	View On Sale Tokens
Description	Users view for-sale tokens
Actors	Users
Pre-conditions	1. User sign in wallet
Main course	1. User click on home screen 2. System query from smart contract tokens that on sale 3. Show on sale tokens
Alternative	Show empty message

Table 3 – 3. Use case description – View on sale tokens

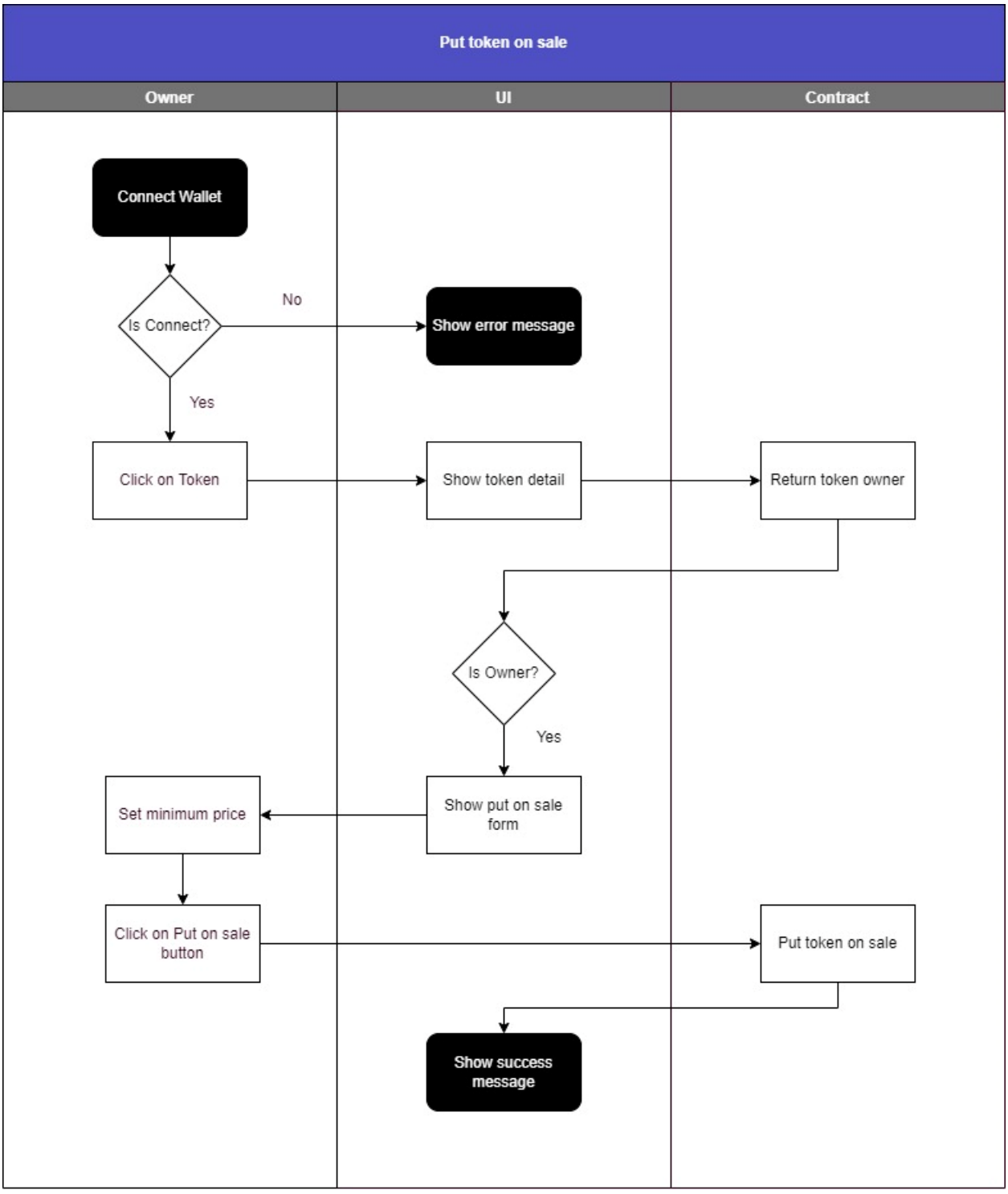


Figure 3 – 4. Put on Sale

Name	Put Token On Sale
Description	Owner put their token on sale
Actors	Owner
Pre-conditions	1. User sign in wallet 2. User is token's owner 3. Token is currently not on sale
Main course	1. Owner click to view token detail 2. Set minimum price for buyers 3. Click put on sale 4. Smart contract change token status for sale

Table 3 – 4. Use case description – Put on sale

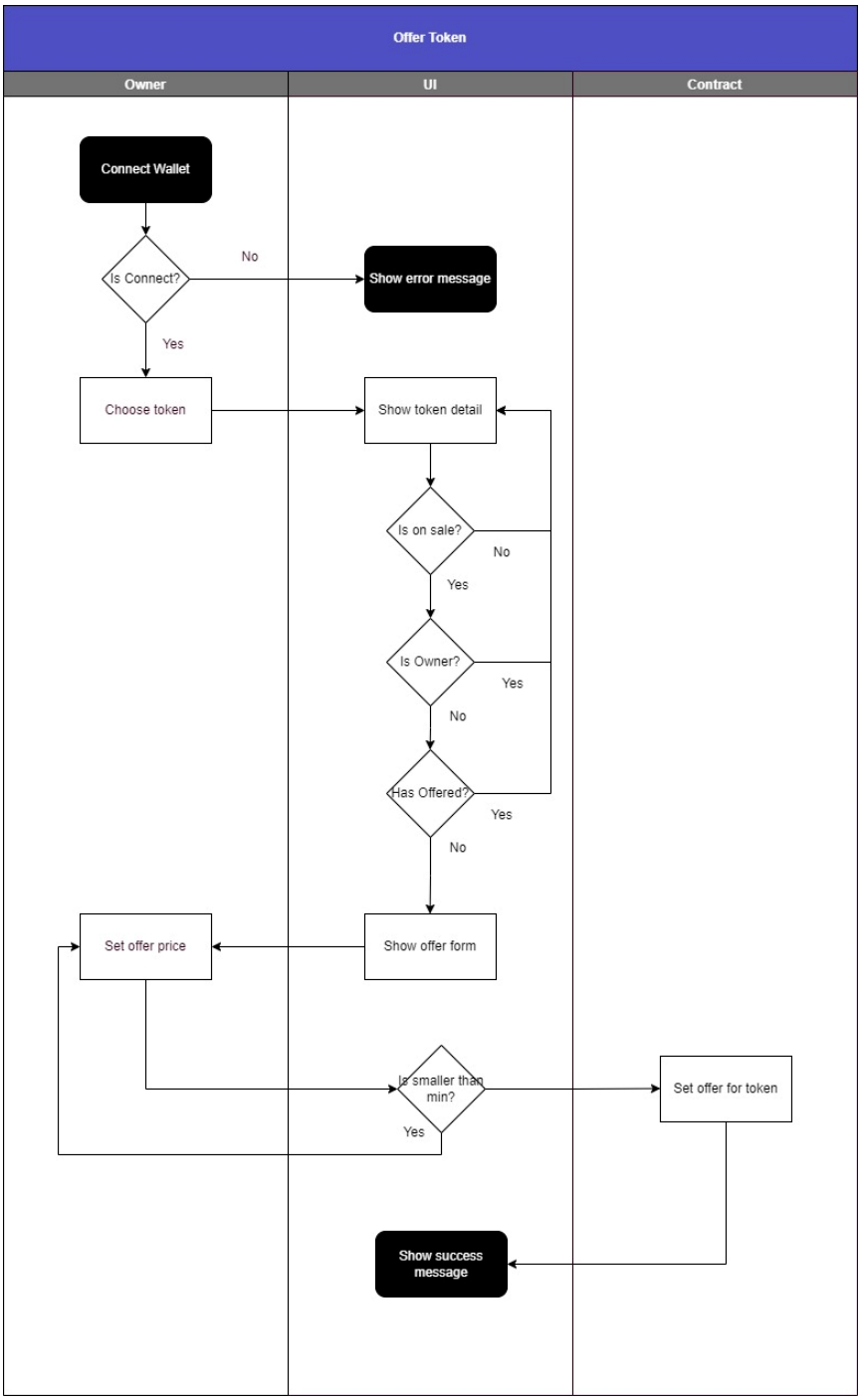


Figure 3 – 5. Offer Token

Name	Offer token
Description	Users offer to buy a token that on sale
Actors	Users
Pre-conditions	1. User sign in wallet 2. User is not owner 3. User has not offered 4. Token is currently on sale
Main course	1. User click to view token details 2. User set offer price 3. System check if offer price is greater than minimum price 4. Click offer 5. Smart contract store offer money and add user to offer list

Table 3 – 5. Use case description – Offer token

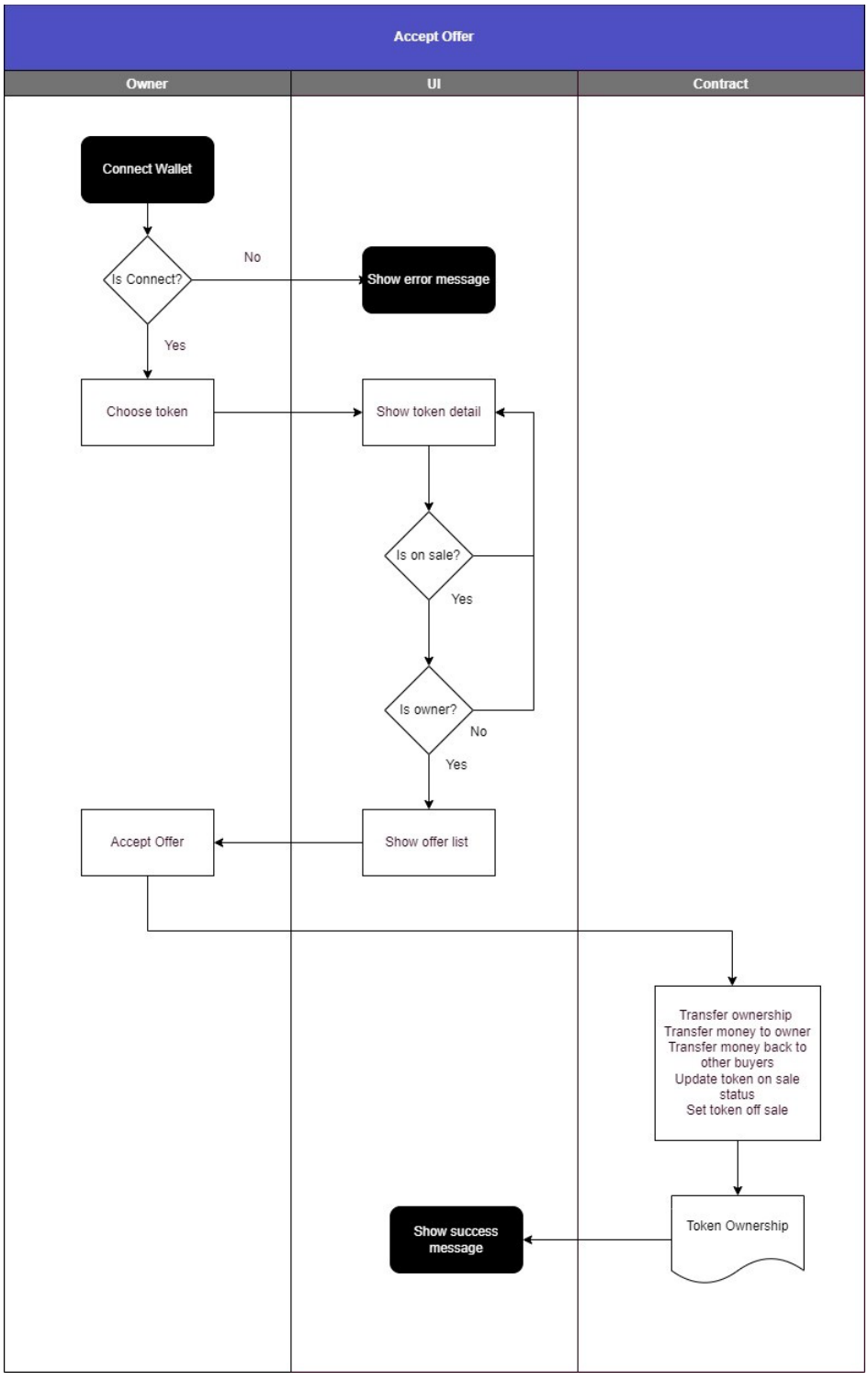


Figure 3 – 6. Accept Offer

Name	Accept offer
Description	Owner accepts offer to sell their token
Actors	Owner
Pre-conditions	1. User sign in wallet 2. User is owner 3. Token is on sale
Main course	1. Owner view token detail 2. Owner view offer list 3. Accept offer 4. Smart contract transfer ownership to offer address, revert all pending money to other buyers, and transfer money to owner.

Table 3 – 6. Use case description – Accept Offer

3.3 Functional Requirements

- Client Side

ID	Name	Constraint	Description
1	Verify Account Status	-	System check if user is logged in to the Metamask wallet and display the corresponding content
2	Mint Token	Whitelist	Only Accounts in Whitelist will be able to mint new token based on the information submitted by the owner. In this thesis paper, government will be the only Whitelist.
3	View My Tokens	-	User click on their address to redirect to their owned token list. Show empty message if address does not own any tokens.
4	View Token On Sale	-	Homepage will display all on sale tokens. Show empty message if there are no tokens on sale.
5	View Token Details	-	Show token details including: • Interior Design • Outdoor • Address • Room(s) • Status • Owner • History Transaction
6	Put On Sale	Owner	System check if token is on sale or not: • Is on sale: Display total offers • Is not on sale: Display put on sale form for owner to set minimum price
7	Offer	Not Owner	System check if user has offered or not: • Has offered: Display total offers • Has not offered: Display offer form for user to offer
8	Accept Offer	Is On Sale	Show offer list and allow owner to accept offer

		Owner	

Table 3 – 1. Front End Functional Requirement

- Server Side

ID	Name	Description
1	Pin File To IPFS	Upload image to IPFS
2	Pin JSON to IPFS	Upload metadata to IPFS
3	Retrieve Metadata	Retrieve token metadata from token hash

Table 3 – 2. Back End Functional Requirement

- Smart Contract

ID	Name	Constraint	Description
1	Mint	Whitelist	Create Token
2	Burn	Whitelist	Destroy Token
3	Put On Sale	Only owner	Check token status and put token on sale if it is valid
4	Offer	Not owner	Add user to offer list of tokens if user is valid
5	Accept Offer	Is on sale	Transfer money back to all users have offered for a specific token and transfer ownership to buyer.
6	Put Off Sale	Only owner	Transfer money back to all users have offered

			and remove token from marketplace.
7	Get On Sale Tokens	-	Get all tokens on sale
8	Get My Tokens	-	Get owned tokens of address

Table 3 – 3. Smart Contract Functional Requirement

3.4 System Architecture

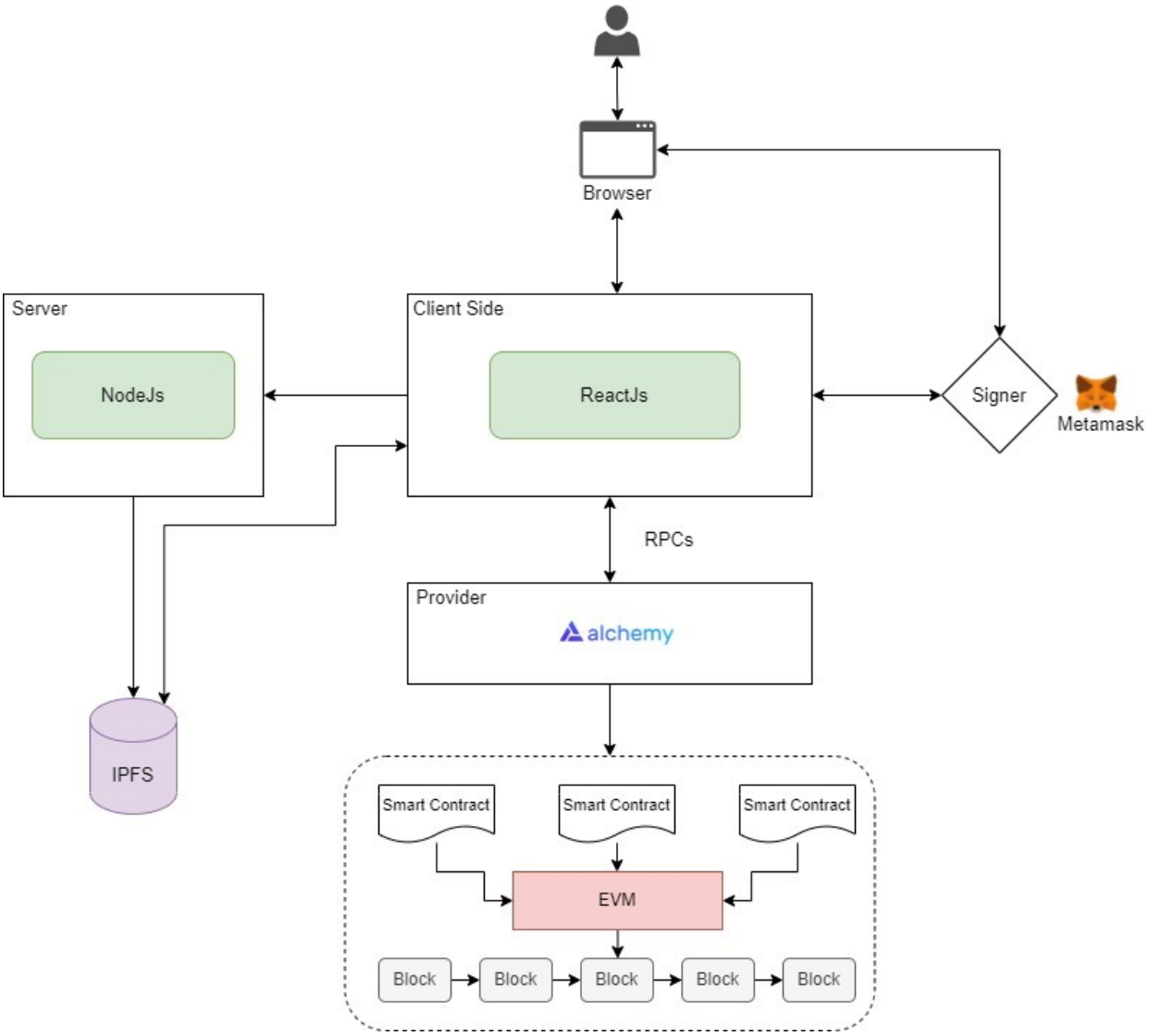


Figure 3 – 7. System Architecture

Decentralization is the strength of Web 3.0 infrastructure. Centralized database will now be replaced by IPFS and a centralized server is redundant. However, in this thesis paper, a one-single-task server will be built for uploading metadata to IPFS and leaving everything to the smart contract and Blockchain to handle. Server receives the request from the user and uses provided keys when signing up for the Pinata service to upload information to a public network. The server will receive the request from the user and will use the keys provided when signing up for the Pinata service to upload information to a public network. After processing, Pinata will then send

back a corresponding hash to submitted content and transmit it back to the front end to send to the smart contract. As we have known, the person who has access to a centralized database will easily change the data. The purpose of using IPFS in Web 3.0 ensures that data is intact and cannot be modified, even by the owner of the system.

In order to communicate with deployed smart contracts, we can either set up own node to run the blockchain software or use services from third-party. Due to the complexity of setting up a full node and the high cost of maintenance, the latter would be appropriate for this project. Not to mention that, one out of sync block can cause massive damage to the system. Node providers such as Alchemy come into play to solve this problem. They provide free node to interact with chain on any network. Even the trickiest edge-cases as latest block problem can be resolved. By implementing a JSON-RPC specification, a set of predefined methods would be configured for client side to easily interact with data on blockchain. The system is now ready to read on chain state.

However, granting write permission is another issue. User must sign the transactions with their private key to change the global state on chain. To understand the importance of the signer, let's say we make transactions involving money transfer, user only needs to provide the wallet address and the backend will communicate with the network through that address. Deviants can take advantage of this if they have the API URL. In this case, the transaction must be signed by signer to prove that the person performing the transaction actually has access to the submitted address. Metamask is the most compatible tool for this scenario. Not only a safe storage for cryptocurrency, it also has connection to blockchain with its own nodes called Infura. In simple terms, users will be able to store their private keys, sign transactions and hold cryptocurrencies with Metamask. Web 3.0 generally let users add their data directly to blockchain, which also means that users must pay extra. Decentralized off-chain storage as IPFS will be a better way to cope with it. It ensures the decentralization and makes it easier to retrieve.

CHAPTER 4. IMPLEMENTATION

4.1 Configuration

- **NodeJs and NPM**

NodeJs is the most commonly used platform to develop applications. It supports asynchronous programming, single-threaded which can handle huge traffic and high scalability. NPM (Node Package Manager) is part of NodeJS that makes it extraordinary. Both NPM and NodeJs can be installed on local machine via https://nodejs.org/en/download/. To check the installed version, run these following commands:

<node -v>

<npm-v>

After successfully installing, all packages in package.json file now can be easily downloaded and we can start running scripts.

- **Metamask**

Metamask is an well-known browser extension which acts as a crypto wallet. With metamask, user can interact directly with the entire Ethereum ecosystem which is the foundation of many decentralized applications, without local installation. It is compatible with mostly nowadays browsers such as Chrome, Edge, Firefox. It also supports cross-platform storage which is both on mobile and browser.

To set up Metamask in browser, let's first navigate to https://metamask.io/. Then select Download -> Chrome -> Install Metamask for Chrome. After installing, Metamask icon will appear on the top right corner of browser.

Figure 4 – 1. Metmask Icon In Browser

Click on Metamask icon and keep following user guide of Metamask to create an account. After reaching the main screen as below

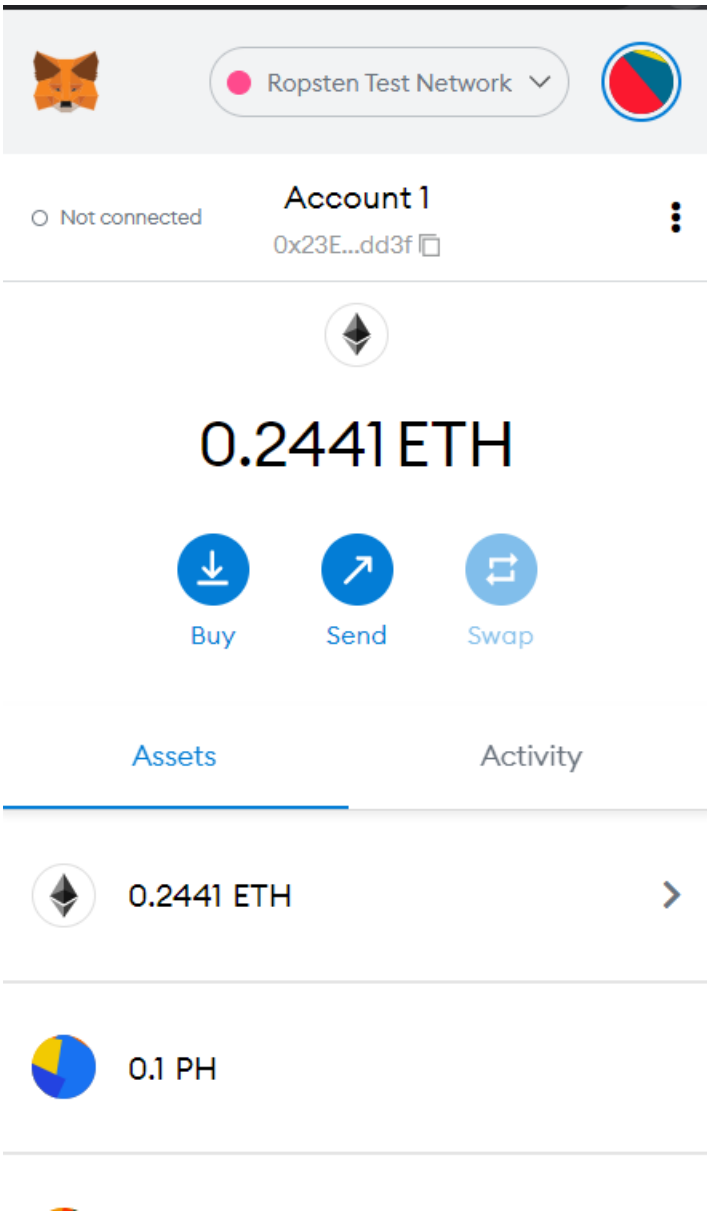


Figure 4- 2. Metamask Home Screen

Let's switch to Ropsten network and the Metamask setup is done.

- **Hardhat**

Hardhat is a handy development tool for developers to compile, debug, test and deploy smart contracts. The built-in Hardhat network, which is a local Ethereum keeps all the processes done easily. The config file hardhat.config.js is normally placed in the root folder of the project. Even an empty config file is good enough for Hardhat to function.

Figure 4 – 3. Hardhat Configuration

In this thesis paper, the config file will be subtle enough for a quick deploy to Ropsten network. After successfully running command

<npx hardhat compile>

The compiled contracts will be generated in the Artifact folder in JSON format. Now everything is ready for deploying.

- **Alchemy**

As mentioned above, by using a third-party service, we will reduce a large amount of work that comes from running a full node. Alchemy is a free service that has all the core

products to eliminate that pain point with user-friendly documentation. We can sign up for free by following

Alchemy - Login (alchemyapi.io)

After signing up, we can start generating API key and creating app. Now, we let Alchemy do the hard work and freely make requests to Ropsten network.

4.2 Implementation

4.2.1 Server Side

As mentioned above, in this thesis paper, a lightweight server will be built with only purpose for uploading and retrieving metadata.

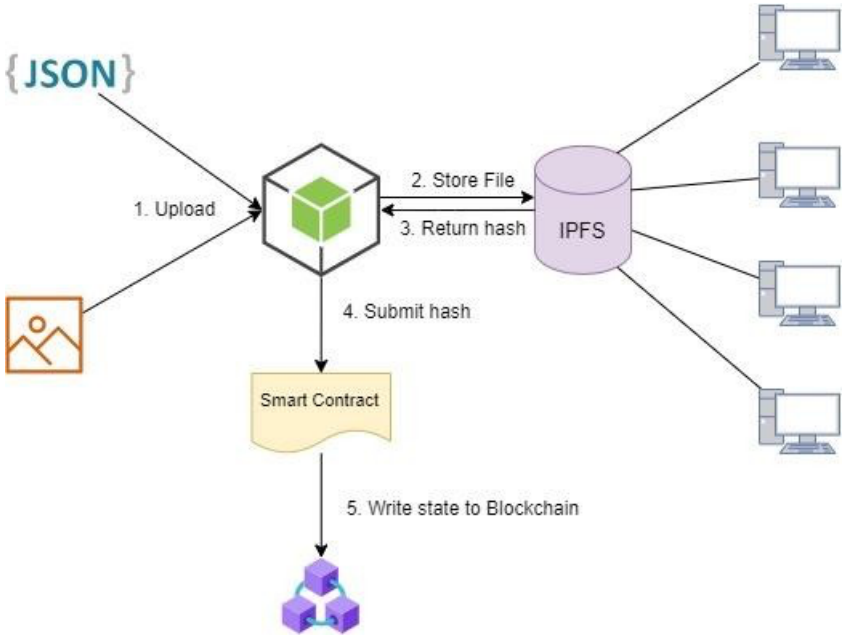


Figure 4 – 4. Pin File To IPFS

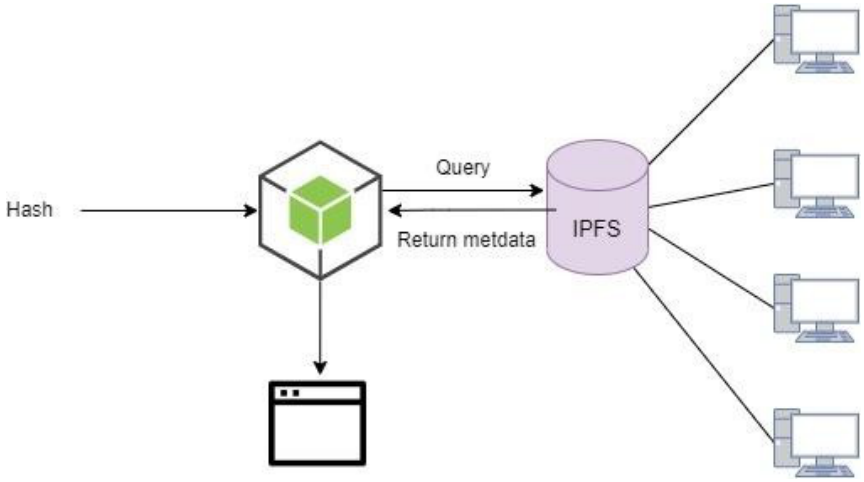


Figure 4 -5. Retrieve Metadata From IPFS

In order to make the implementation subtle, one single file server.js will be used as controller module, route module, to handle logic and start the server at the same time. The server will interact with Pinata service through 3 main APIs:

- Pin File: Take token's information in the form of images including design, architecture, household registration or legal documents and upload it through Pinata's API gateway to output the hash
- Pin JSON: Take token metadata such as address, residence, location, rooms in JSON format and submit through Pinata gateway to generate a hash.
- Retrieve metadata: Take token hash and call Pinata API to retrieve the submitted metadata.

4.2.2 Client Side

The implementation will have the usual directory structure of a React project. An index.html file is required to encapsulate React code and provide the context for React to render when the application starts. A index.js will be responsible for storing the rendered data from ReactDOM.

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";
import {Provider} from 'react-redux'
import store from "./store";
import "./styles/bootstrap.min.css";
import "./styles/index.css";


ReactDOM.render(
    <Provider store={store}>
        <App />
    </Provider>
    , document.querySelector("#root"));
```

Figure 4 - 6. Index File

All the screens and its route for mapping will be specified in the core component App.js file. In order to optimize state management and gain an intuitive idea of how the implementation works, React Hooks will be applied to all the components. A central state management such as Redux is no need in this project because states will be queried directly from Blockchain and be used for single page at a time. To interact with deployed contract, a local instance must be created from contract address and ABI.

```
import { createAlchemyWeb3 } from "@alch/alchemy-web3";
import contract from "../../../artifacts/contracts/House.sol/House.json";
const alchemyKey = process.env.REACT_APP_API_URL;
const contractAddress = process.env.REACT_APP_DEPLOYED_CONTRACT;
export const web3 = createAlchemyWeb3(alchemyKey);
export const tokenContract = new web3.eth.Contract(contract.abi, contractAddress);
```

Figure 4 – 7. Setup Contract Local Instance

4.2.3 Smart Contract

OpenZeppelin library's ERC721 will be applied on contract implementation. It helps developers minimize risks by defining a common standard which supports the contract development and integration. To install the library, in the command line, run

<npm install @openzeppelin/contracts>

We now have some predefined classes which our contract can inherit from

- @openzeppelin/contracts/token/ERC721/ERC721.sol implements the backbone of smart contract logic. A valid ERC721 Token must implement all the methods in this interface.

```
function _mint(address to, uint256 tokenId) internal virtual {
    require(to != address(0), "ERC721: mint to the zero address");
    require(!_exists(tokenId), "ERC721: token already minted");

    _beforeTokenTransfer(address(0), to, tokenId);

    _balances[to] += 1;
    _owners[tokenId] = to;

    emit Transfer(address(0), to, tokenId);
}

/**
 * @dev Destroys `tokenId`.
 * The approval is cleared when the token is burned.
 *
 * Requirements:
 *
 * - `tokenId` must exist.
 *
 * Emits a {Transfer} event.
 */
function _burn(uint256 tokenId) internal virtual {
    address owner = ERC721.ownerOf(tokenId);

    _beforeTokenTransfer(owner, address(0), tokenId);

    // Clear approvals
    _approve(address(0), tokenId);

    _balances[owner] -= 1;
    delete _owners[tokenId];

    emit Transfer(owner, address(0), tokenId);
}
```

Figure 4 - 8. Interface Functions

Figure 4 - illustrates some fundamental methods of a ERC721 Token. ERC721 standard claims that each token needs to be unique, so they must have their own Id. To create a new token, mint function must be called with the owner's address and a unique Id, which can be specified when we implement our contract. Tokens that are passed through the burn function will be transferred to an empty address, address(0), and be destroyed forever. Obviously, only owner has rights to do this and their restrictions can be assigned in the implementation phase.

```
//OWNER
//Burn a token
function burn(uint256 tokenId) public
{
    require(_exists(tokenId), "Token does not exist");
    require( _isApprovedOrOwner(_msgSender(), tokenId), "You are not authorized");

    string memory _hash = _tokenURIs[tokenId];
    _hashes[_hash] = false;
    delete _tokenURIs[tokenId];

    removeMyToken(msg.sender, tokenId);

    delete _primeOwners[tokenId];

    totalTokens -= 1;

    _burn(tokenId);
}
```

Figure 4 - 9. Burn Method Implementation

- @openzeppelin/contracts/utils/Counters.sol provides a global counter which can be used as token id. Our smart contract will have a tracking counter that can automatically increment whenever a new token minted.

```
using Counters for Counters.Counter;

Counters.Counter private _tokenIds;
```

Figure 4 - 10. Initiate Counter Instance

```
//PUBLIC
//Mint a new token
function mint(address _to, string memory _hash) public onlyOwner returns (uint256)
{
    require(_hashes[_hash] != true, "Your token already exists");
    _hashes[_hash] = true;

    _tokenIds.increment();
    uint256 newItemId = _tokenIds.current();

    _mint(_to, newItemId);
    _setTokenURI(newItemId, _hash);

    _myTokens[_to].push(newItemId);

    _primeOwners[newItemId] = _to;

    totalTokens += 1;

    return newItemId;
}
```

Figure 4 - 11. Mint Method Implementation

- @openzeppelin/contracts/access/Ownable.sol gives us the ability to control the restrictions. In the Figure 4 - , we can see that mint function implements onlyOwner restriction which means that only contract owner has rights to call it. In fact, we can create our own restriction and grants specific permission on any functions. In Figure 4 - , for every function that contains restrictOnMarket restriction, tokens must exist and be on marketplace.

```
modifier restrictOnMarket(uint256 tokenId)
{
    require(_exists(tokenId), "Token does not exist");
    require(_isOnSale[tokenId] != false, "Token is not on sale");
    _;
}
```

Figure 4 - 12. Only On Sale Restriction

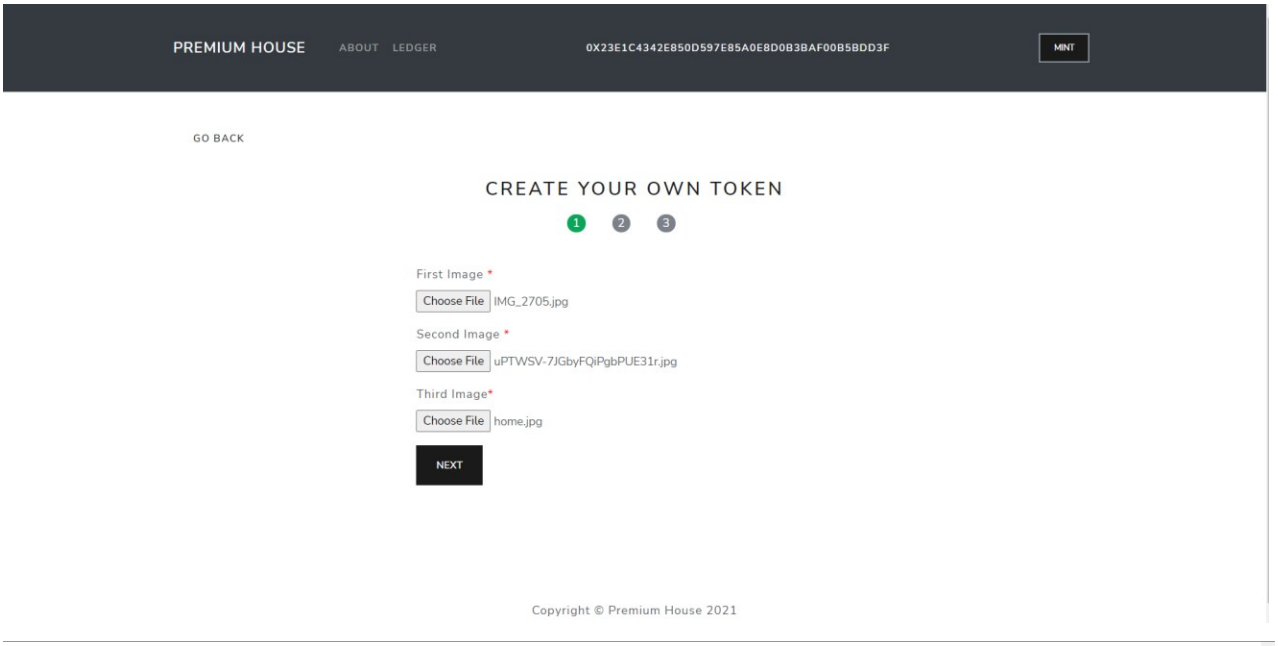
Each token will have a unique id, but what values them is the metadat. In the mint function, the hash parameter will be associated with corresponding token from the time it is

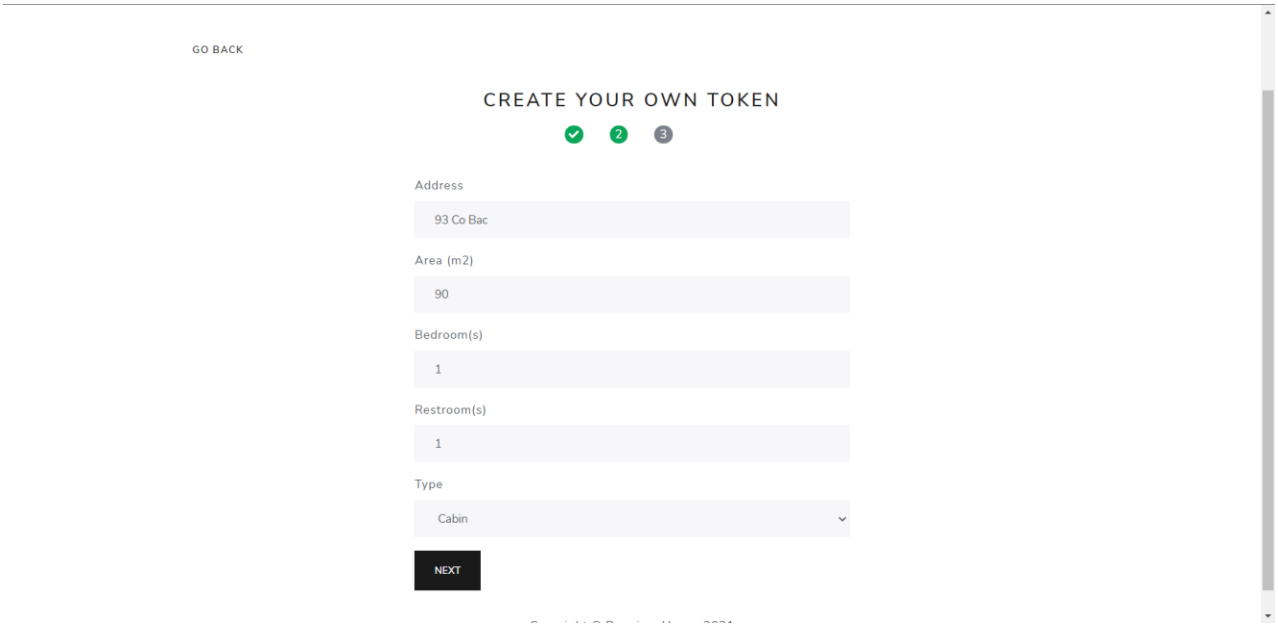
generated until it is burned. This is what makes the new token standards stand out from the previous smart contract implementation.

CHAPTER 5. RESULT AND EVALUATION

The output of the system meets the basic requirements for users to own and trade tokens. User interface is implemented intuitively and user-friendly as well as can grant permissions based on the wallet address to help increase the security of the system.Technically, the system has acquired all the functions that initially stated

5.1 Result





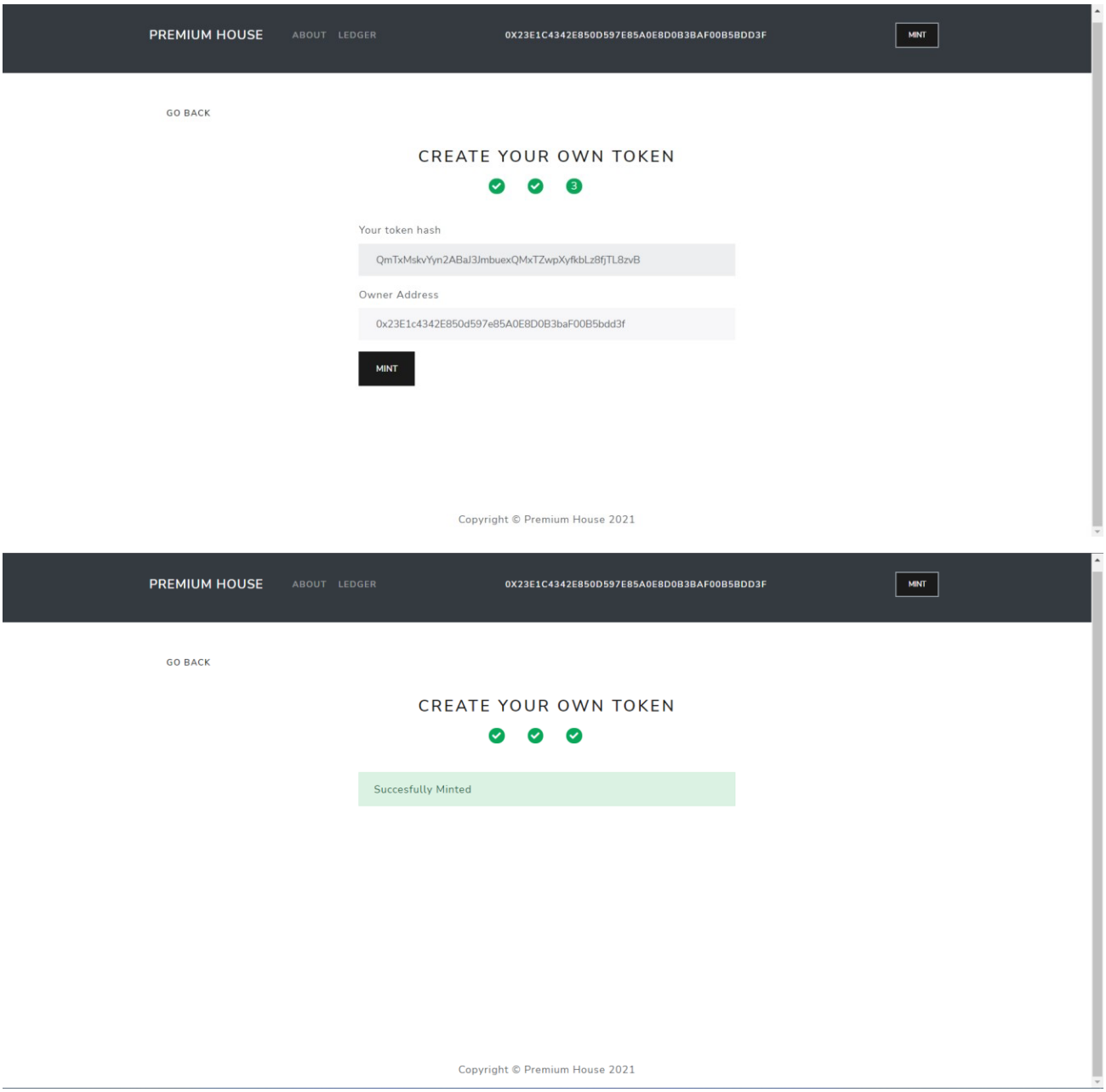


Figure 5 – 1. Mint Screen

The interface will allow the system owner to post information about real estate in the form of images in the first step. Image files will be uploaded directly to Pinata IPFS before heading to the next step. In the second step, information such as geographic location and room number will be filled in the form as provided by the owner, and will also be uploaded to IPFS as JSON. Finally, the owner's address will be filled in and sent to the contract to grant ownership of the token after it is minted.

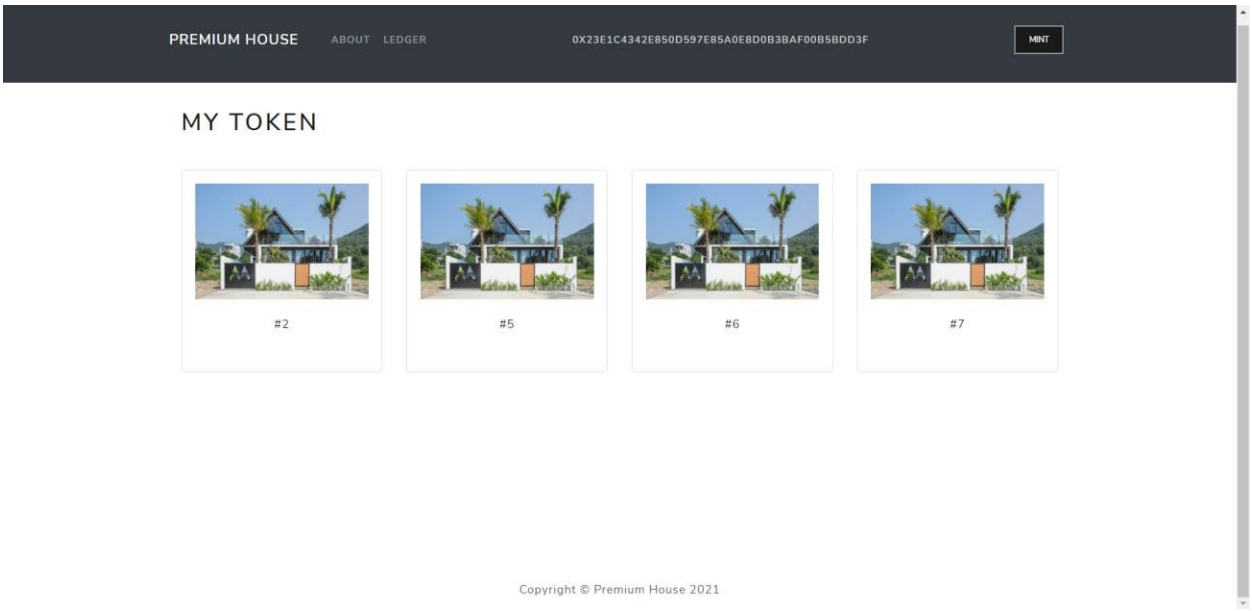


Figure 5 – 2. My Token Screen

The system calls the contract and retrieves the list of tokens owned by the user. User can click to go to detail page.

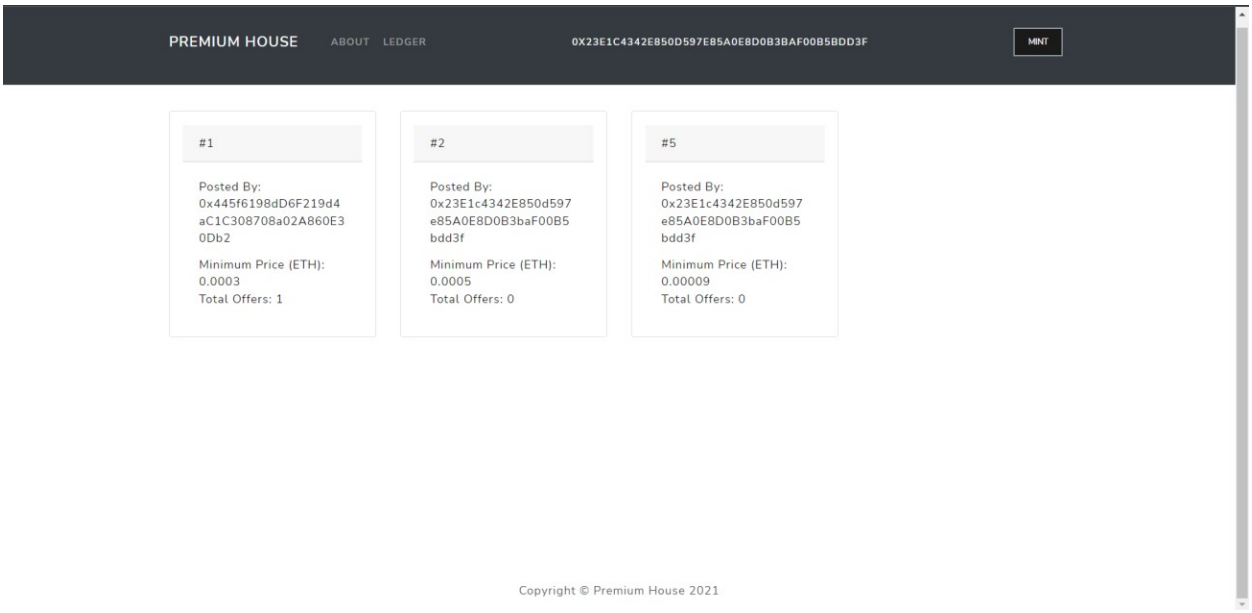


Figure 5 – 3. Home Screen

On the main screen, the list of tokens for sale will be displayed first. When accessing the system, users will be forced to log in to their wallet to see the list of tokens for sale.

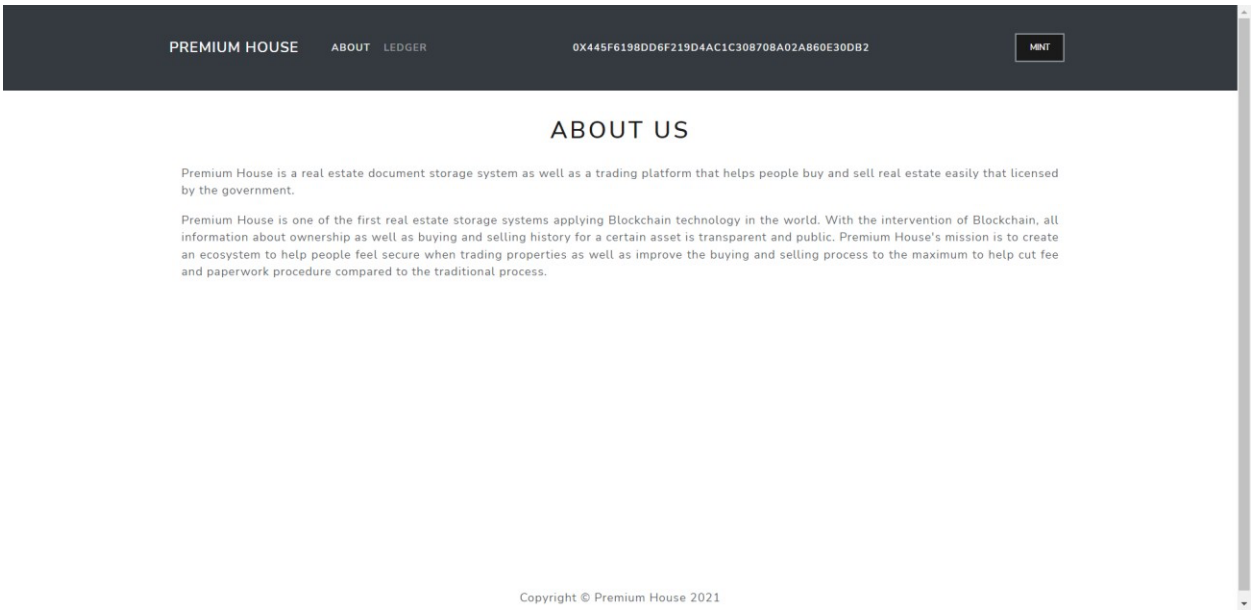


Figure 5 – 4. About Screen

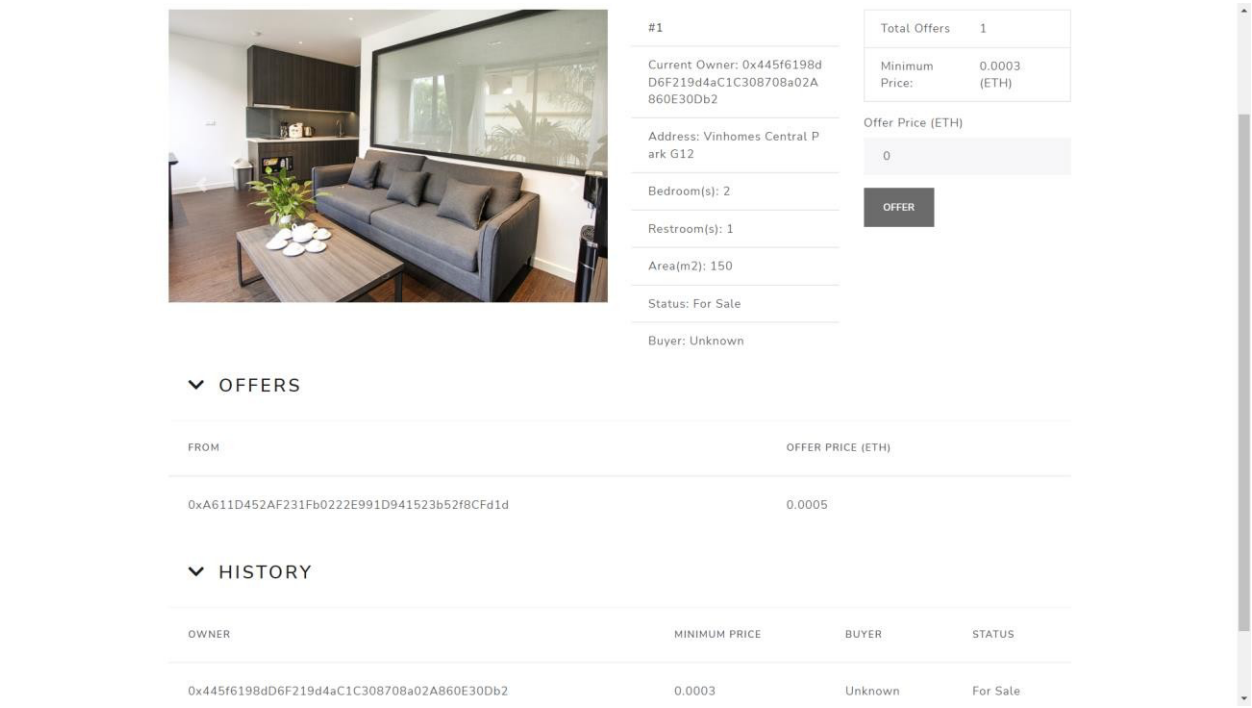


Figure 5 - 5. Token Detail Screen

In the details screen, the information of the token will be displayed by area. A carousel will be run to show an image of the token and the information the owner has previously provided.

In addition, the system also queries the transaction history of the token and the list of offers if the token is for sale.

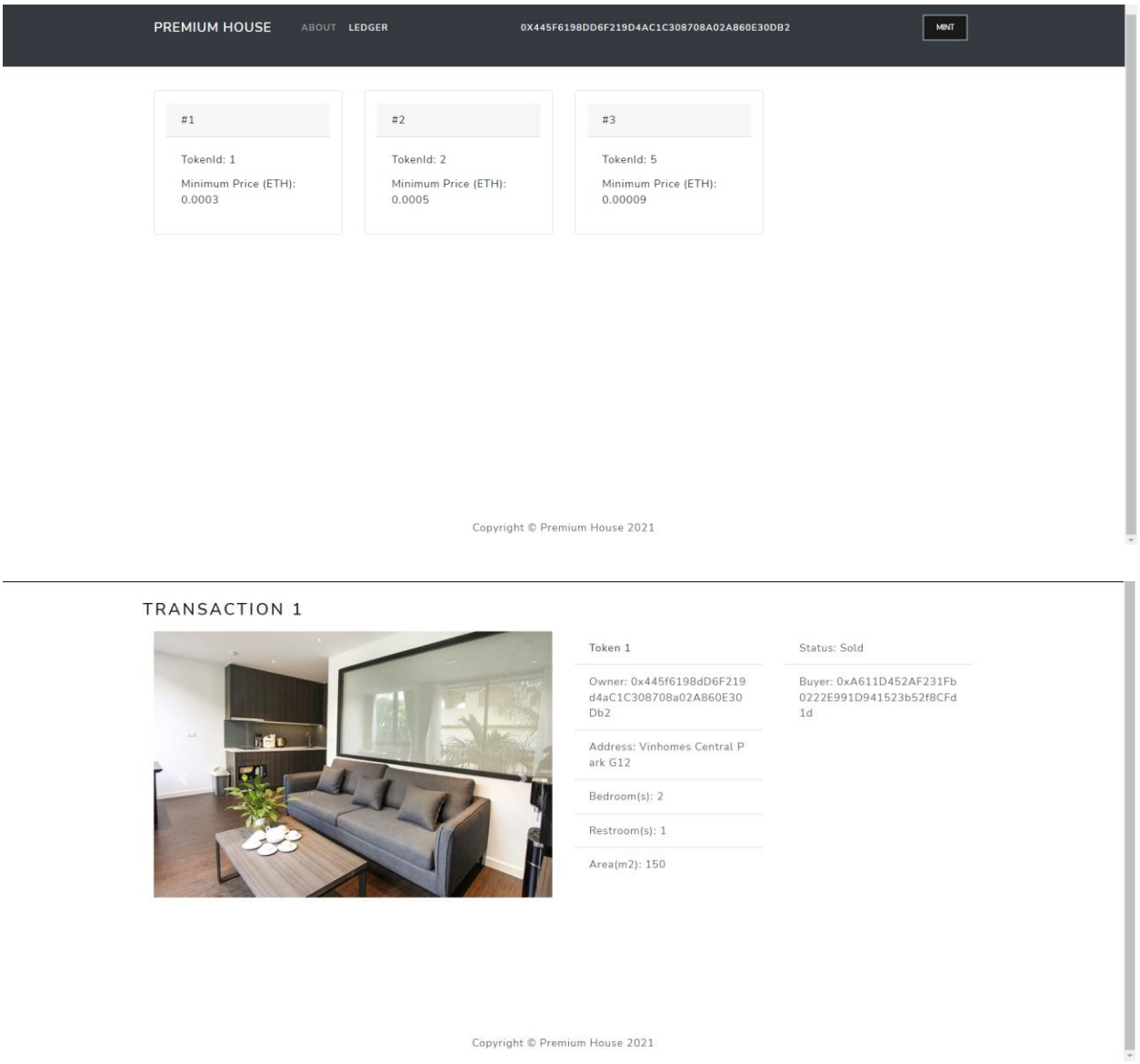


Figure 5 – 6. Public Ledger

The public ledger will record and query the token transactions that have occurred in the system so far. With the help of an open ledger that can record the transaction history of tokens, users can detect bad behaviors. For example, owner can create 2 accounts and trade back and forth to push up the price of their token, these transactions will be logged on Blockchain and people can spot the pattern.

5.2 Discussion

Although the system is already capable enough to go live, there are still performance limitations as well as features that need improvement. The blockchain trilemma states that is is almost impossible to achieve decentralization, security and scalability at the same time when building decentralized application. Currently, letting smart contracts process and save data directly on the Blockchain can ensure security for users, but performance and user experience are trade-offs. It will take quite some time to read and write data on chain when the web traffic is huge. Not only that, the large amount of traffic will also clog the network, causing high transaction fees. Hence, in order to optimize performance and make the system scalable, a centralized server and database management are still needed. For instance, only legal documents such as registration book and identification will be stored on IPFS, while real estate information such as location and area will be stored traditionally.

Transactions will be recorded on the network and anyone can see the history and details of the transactions on Blockchain explorer platforms such as BscScan or EtherScan. To be able to do so, users need to search using the transaction hash generated when they made a successful transaction before. The current system only stores the token's transaction history including basic information like buyer and offer price, but if users want to see it directly on chain explorers, they need a transaction hash. The downside of storing history on the contract is that it will not be possible to save the transaction hash. As mentioned above, the transaction hash is only generated when the transaction has completed successfully, which means the smart contract's function call has ended. Hence, it makes more sense to convert the transaction history to a centralized database to query the transaction hash and information in that transaction.

Beside the above technical requirement, for the project to be practical, the dissemination of knowledge and information to civilian is also indispensable. People need to know the existence of the system and understand the process of the system to avoid being scammed by unorthodox platforms. Because the fraud in the real estate buying and selling process in the traditional model has always been the norm.

CHAPTER 6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

This thesis paper has shown the advancements and innovations that Blockchain can bring to the real estate industry. Many new concepts have exposed such as Web 3.0, Smart Contract, Tokens that changed the entire architecture of an application. Documentation is automated by a predefined rule set of smart contract, legal information and transaction history become transparent and public, preferential policies for the prime owner, no boundaries transactions support. Blockchain is reshaping our lives and promises to bring tremendous improvements to every industry in the near future.

6.2 Future Work

Since the original purpose of the project was to create a mainstream platform that is widely used across a country, building a private network and issuing tokens is worth considering. More specifically, the native token will be used as a transaction fee for the network. A scalable system will attract more users. Therefore, new services and features should also be considered for the development of utility tokens. Simply put, it's like when we want to enter an amusement park, we need an entrance ticket, the native token will act as the ticket. And to be able to use the services in the game area, we need to exchange for coins that represent the turn, utility tokens have the same meaning. Building a private network and issuing coins not only brings benefits to users, but also shows the scale of the project.

Similar to renting, staking mechanism is a promising feature. Users can deposit their tokens into a pool with an corresponding annual percentage yield (APY) to specific option. That not only increases the liquidity of the system but also helps users to make profit. Users can even stake their non-fungible tokens and temporarily deposit ownership to the system for 9 months. Tokens after staking can be certified by the government to increase value or they can use it to rent and pay profits to owners.

REFERENCES

[1] Blockchain Explained: What Is Blockchain? | Euromoney Learning. //www.euromoney.com/learning/blockchain-explained/what-is-blockchain. Accessed 19 Jan. 2022.

[2] “What Is Blockchain Mining and Who Is a Blockchain Miner? - Intellipaat.” Intellipaat Blog, https://intellipaat.com/blog/tutorial/blockchain-tutorial/what-is-bitcoin-mining/. Accessed 19 Jan. 2022.

[3] Kastrenakes, Jacob. “Beeple Sold an NFT for $69 Million.” The Verge, 11 Mar. 2021, https://www.theverge.com/2021/3/11/22325054/beeple-christies-nft-sale-cost-everydays-69-million.

[4] “Ethereum Virtual Machine (EVM).” Ethereum.Org, https://ethereum.org. Accessed 19 Jan. 2022.

[5] Antonopoulos, Andreas M., and Gavin Wood. Mastering Ethereum: Building Smart Contracts and DApps. First edition, O’Reilly, 2019.

[6] Howcroft, Elizabeth. “Virtual Real Estate Plot Sells for Record $2.4 Million.” Reuters, 24 Nov. 2021. www.reuters.com, https://www.reuters.com/markets/currencies/virtual-real-estate-plot-sells-record-24-million-2021-11-23/.