DECENTRALIZED REAL ESTATE USING BLOCKCHAIN

A PROJECT REPORT

Submitted by

VIGNASH M [211419104300]

VIJAY M P [211419104301]

SUNDARAPANDIYAN G [211419104273]

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

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

APRIL 2023

PANIMALAR ENGINEERING COLLEGE

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

BONAFIDE CERTIFICATE

Certified that this project report **"DECENTRALIZED REAL ESTATE USING BLOCKCHAIN"** is the bonafide work of "**VIGNASH M (211419104300), VIJAY M P (211419104301), SUNDARAPANDIYAN G (211419104273)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.L.JABASHEELA,M.E.,Ph.D., **PROFESSOR** **HEAD OF THE DEPARTMENT**	**Mr. M.MAHENDRAN, M.Tech.,(Ph.D.),** **SUPERVISOR** **ASSISTANT PROFESSOR**
DEPARTMENT OF CSE, PANIMALAR ENGINEERING COLLEGE, NASARATHPETTAI, POONAMALLEE, CHENNAI-600 123.	DEPARTMENT OF CSE, PANIMALAR ENGINEERING COLLEGE, NASARATHPETTAI, POONAMALLEE, CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on..........................

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **VIGNASH M (211419104300), VIJAY M P (211419104301), SUNDARAPANDIYAN G (211419104273)** hereby declare that this project report titled "**DECENTRALIZED REAL ESTATE USING BLOCKCHAIN**" , under the guidance of **Mr.M.MAHENDRAN, M.Tech.,(Ph.D.),** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

VIGNASH M

VIJAY M P

SUNDARAPANDIYAN G

ACKNOWLEDGEMENT

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

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

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

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

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

VIGNASH M

VIJAY M P

SUNDARAPANDIYAN G

ABSTRACT

Decentralized Real Estate using blockchain is a system that leverages blockchain technology to address issues in the traditional real estate industry such as fraud, lack of transparency, and inefficiency. This system provides a secure and transparent platform that enables buyers and sellers to directly interact and transact with each other without the need for intermediaries. The system is built on a decentralized network that stores data in a tamper-proof manner, which ensures that all parties involved can access and verify the information without the need for a central authority. The use of smart contracts in this system automates many of the processes involved in real estate transactions, reducing the time and costs associated with these processes. The system also allows for fractional ownership, which enables investors to buy and sell portions of a property. This feature promotes liquidity in the real estate market, making it more accessible to investors. Additionally, the system facilitates crowdfunding for real estate projects, enabling smaller investors to pool their resources and invest in high-value real estate projects that were previously only available to larger investors.

Overall, the Decentralized Real Estate system offers a range of benefits over traditional real estate systems, including increased transparency, security, efficiency, liquidity, and accessibility. The potential for this system to transform the real estate industry is significant, and it is already gaining traction in many parts of the world.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Decentralized Real Estate using blockchain is a revolutionary approach that aims to transform the traditional real estate industry by leveraging the benefits of blockchain technology. Blockchain-based real estate systems are characterized by enhanced transparency, immutability, and security, which is expected to mitigate many of the challenges currently faced by the real estate industry, such as fraud, inefficiency, and lack of transparency. The use of smart contracts, which are self-executing and self-enforcing, enables the automation of various real estate processes, including property listings, purchase agreements, and title transfers. Moreover, the decentralized nature of blockchain eliminates the need for intermediaries, such as banks, lawyers, and brokers, thus reducing transaction costs and increasing accessibility to the market. The integration of blockchain in the real estate industry is expected to enable faster, more secure, and more transparent transactions, leading to increased efficiency and trust in the market. In this paper, we provide an overview of the current state of decentralized real estate using blockchain, as well as its potential benefits and challenges. We also explore various use cases and discuss future developments in this rapidly evolving field.

1.2 Real estate market and its challenges

The real estate market is a highly lucrative and dynamic market, but it is also fraught with challenges such as high transaction costs, lack of transparency, fraudulent activities, and inefficient processes. These challenges often lead to mistrust among stakeholders and hinder the growth of the market.

1.3 Blockchain technology

Blockchain technology is a distributed ledger technology that allows for secure and transparent transactions without the need for intermediaries. Its decentralized nature and immutable record-keeping capabilities make it an ideal solution for various industries, including real estate.

1.4 Decentralized real estate using blockchain

The decentralized real estate using blockchain concept involves leveraging blockchain technology to create a more transparent, secure, and efficient real estate market. This concept aims to eliminate the need for intermediaries, reduce transaction costs, and increase trust among stakeholders.

1.5 Smart contracts

Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller being directly written into code. They enable the automation of various processes, including property transfers, rental agreements, and mortgage contracts.

1.6 Tokenization

Tokenization involves converting real estate assets into digital tokens on a blockchain network. These tokens can then be easily traded, providing increased liquidity and accessibility to the real estate market.

1.7 Advantages of decentralized real estate

The advantages of decentralized real estate include reduced transaction costs, increased transparency, improved efficiency, and increased access to the market.

1.8 Case studies

Several projects have already started implementing decentralized real estate using blockchain technology. These include Propy, BitProperty, and Ubitquity.

1.9 Challenges and limitations

The implementation of decentralized real estate using blockchain technology faces several challenges and limitations, including regulatory barriers, scalability issues, and the need for interoperability between different blockchain networks.

1.10 Future potential

Despite the challenges and limitations, the potential for decentralized real estate using blockchain technology is significant. Its potential to transform the real estate market by increasing transparency, reducing transaction costs, and increasing accessibility cannot be overstated.

CHAPTER 2

LITERATURE SURVEY

LITERATURE SURVEY

Sobhan Latifi, Yunpeng Zhang, Liang-Chieh Cheng [1] et al "Blockchain-Based Real Estate Market: One Method for Applying Blockchain Technology in Commaercial Real Estate Market", IEEE International Conference on Blockchain (Blockchain), 2020

Global real estate (RE) investments exceed the size of the stock market. Despite this, the number of RE investors is much lower due to liquidity and global access. The current system is barely satisfactory for tenants, owners, and investors. The goal of this paper is to test the use of blockchain in the real estate market and represent the benefits it can provide. So far, the research has led to the following conclusions: Blockchain technology and smart contracts can solve the traditional problems that RE is facing, and they provide far more useful tools for a game-theoretic stable-priced market.

Jack Laurie Tilbury, Ed de la Rey, Karl van der Schyff [2] et al "Business Process Models of Blockchain and South African Real Estate Transactions", International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD), 2019

This research looks at two approaches to real estate transaction execution: the South African case and an international blockchain technology use case. Using Business Process Modelling and Notation, two conceptual models are presented. Document review was used to provide adequate information on the real estate transactions. According to the findings, the South African real estate transaction process is inefficient because it is manual, involves paper-based documents, and is heavily reliant on third parties, resulting in numerous bottlenecks. According to the study, blockchain-based transactions are more efficient and reduce the need for third parties and manual processes.

Disha Shinde, Snehal Padekar, Siddharth Raut, Abdul Wasay, S. S. Sambhare [3] et al "Land Registry Using Blockchain - A Survey of existing systems and proposing a feasible solution", 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA), 2020

When a person purchases land under this system, the government authority will provide the person with a hard copy of the property papers, and the system will store the documents in the Inter Planetary File System (IPFS), a decentralized database. The hash of the document will be generated by the IPFS network. After the smart contracts' conditions are met, this hash will be securely stored in the Ethereum blockchain. The documents from the government authorities will be validated and verified by the smart contracts. This will create a decentralized, tamper-proof ledger from which we will be able to easily retrieve the stored data.

Toqeer Ali, Adnan Nadeem, Ali Alzahrani, Salman Jan [4] et al "A Transparent and Trusted Property Registration System on Permissioned Blockchain", International Conference on Advances in the Emerging Computing Technologies (AECT), 2020

In this proposal, lands are registered on the Blockchain network using a smart contract. The proposed study can provide several benefits to stakeholders, such as efficiency, transparency, trustworthiness, and integrity for various entities and processes involved in buying and selling real estate. Essentially, the framework provides services that provide a detailed history and unaltered information about a property to ensure that the record is not tampered with. Restful provides an external link to traditional property dealing apps, allowing them to extract real-time records of the land, such as dimension, location, and price. The proposed system will ultimately ensure trust in conducting real estate transactions over the Internet.

Ankit Mittal, Bhavyansh Sharma, Pinku Ranjan [5] et al "Real Estate Management System based on Blockchain", IEEE 7th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), 2021

This paper describes a real estate management system powered by blockchain technology that will provide a transparent, secure, and efficient real estate management system. This system will include all real estate management departments. It will store all transactions on a distributed permission blockchain, which will be very secure and resistant to hacking, and it will be highly automated. The system will be centralized in terms of connecting all departments and decentralized in terms of data storage. It is a practical solution to the problem of real estate management.

VO Khoa Tan, Thu Nguyen [6] et al "The Real Estate Transaction Trace System Model Based on Ethereum Blockchain Platform", 14th International Conference on Computer and Automation Engineering (ICCAE), 2022

VO Khoa Tan created the Blockchain network architecture as well as the distributed ledger and smart contracts. As a result, this method is capable of digitizing assets on the Blockchain, storing decentralized transaction history, enabling encryption, and facilitating transactions between sellers and buyers. Furthermore, this system model can reduce data explosion, perform multiple transactions at the same time, and prevent data tampering and sensitive information disclosure. Based on the Ethereum Blockchain platform, the researchers created a prototype of this system model. As a result, they have demonstrated the efficacy of the RETT system model and its practical applicability through experimental transactions. In real estate transactions, this method improves transparency, eliminates intermediaries, saves money, and increases mutual trust.

Andrey Averin, Pavel Rukhlov, Elnur Musaev [7] et al “Review of Existing Solutions in the Field of Real Estate and Cadastral Accounting Based on Blockchain Technology”, International Conference on Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS), 2021

For more than a decade, the global real estate market has been rebuilding in response to new realities. Because of mass digitalization and instant access to the Internet, network users were able to obtain information about the object they required as quickly as possible. To remain relevant and competitive, the market is attempting to implement all new technologies. Many of the third-party factors that make real estate and cadastre work impossible or difficult are eliminated by the use of blockchain technology.

Ishan Yapa, Samitha Heanthenna, Nadun Bandara, Isuru Prasad, Yashas Mallawarachchi [8] et al “Decentralized Ledger for Land and Property Transactions in Sri Lanka Acresense”, IEEE Region 10 Humanitarian Technology Conference (R10-HTC), 2019

The proposed solution includes a decentralized ledger that runs on top of blockchain technology to record land and property transactions. Smart contracts will validate all new transaction entries to ensure the transactions' validity. The transaction records will be stored in cryptographically secured blocks after a proper consensus mechanism has been performed. As a result, the data integrity could not be compromised. Furthermore, before purchasing or investing in land, investors and land buyers can obtain information about current market trends and predictions. Furthermore, the proposed system allows users to view land information graphically and to view the desired land parcels without physically visiting the real location via the geographical information system.

P. Shamili, B. Muruganantham [9] et al "Blockchain based Application: Decentralized Financial Technologies for Exchanging Crypto Currency", International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), 2022

The main contributions of this work are based on the uniswap protocol, which allows users to exchange crypto coins. The automated liquidity protocol provision on the blockchain Ethereum platform is defined as the Uniswap protocol. A user can use the uniswap to buy or sell ERC20 tokens in the decentralized distributed network using an Ethereum smart contract. This paper will assist blockchain network users in using the uniswap protocol as a trustworthy and efficient way of exchanging crypto coins or tokens in the blockchain network.

Dipika Bhanushali, Akshara Koul, Sainiranjan Sharma, Bushra Shaikh [10] et al "BlockChain to Prevent Fraudulent Activities: Buying and Selling Property Using BlockChain", International Conference on Inventive Computation Technologies (ICICT), 2020

The deed or an agreement signed by the government that is physically present on a piece of paper and the records kept by the government in the ledger is proof that you own the property. You must hope that you do not lose the deed or that the government ledger is not damaged or misplaced. That is the type of assurance you have. The researchers propose a system that will assist us in solving this problem. If you want to buy or sell property, you can fill out the smart contract form and receive a digital deed that is uploaded as a new block in the chain. Each node's copy will be present on multiple servers, allowing us to maintain integrity in the event of an attack or system failure.

CHAPTER 3

SYSTEM SPECIFICATIONS

3.1 EXISTING SYSTEM

Customer walks into the developers office asks Developer for required property.Developer searches for named properties/location/flats among cluster/group of books.After finding out it serves to the customer.If customer is interested in buying that flats, Developer sales the property/flat bycompleting all formalities, and makes entry into the Customer Details Registermanually.All the permanent data of the order such as the name of the property / flat,Estate ID, as well as name of buyer/customer, Member_ID and so on aremaintained in a master file for future reference.The customer has the option of paying by cash or check or credit card.At the time customer pays the price he is issued a receipt acknowledging thesame.After carefully studying requirements and working of the organization it has beendecided to develop an Online Real Estate Website.An Online Real Estate Website is nothing but a Website that is supported by a(relational or non-relational) database.It consists of data-entry of forms, in which the customers can fill-up the data.After validation, the data can be added to the database. the Owner/Admin can search, view, and records. Maintaining details of the customers in registers is a time-consuming process and error-prone.If a particular record is to be searched then one has to search through a lot of physicalrecords.No Security has been provided to the data.As the work is manual it consumes lot of time and energy.There is no facility provided for online booking of property. At present everything is taken care-off manually. The admin does not have access to the information that is generated by the Developer.And because the work is done manually there is area possibility of the information being error-prone.Thus the admin is not in a position to make quick decisions.

3.2 PROPOSED SYSTEM

Blockchain-Based SaaS Platform That Transforms Documents and Contracts into Fully Digital, Intelligent Assets. It allows sellers to tokenize assets, essentially handling it like a stock sale, and liquidating that asset through a token sale using the platform. The collected tokens can be exchanged for fiat currency, with buyers owning a percentage stake of the property. You can use tokenization to divide a large number of real estate assets into parcels that are more easily individually purchased. Instead of worrying about a chain of title, the house would be represented by a token whose ownership can be verified on the blockchain. Tokenization provides another side benefit, particularly for small investors. Smart contracts could allow for lower-cost real estate transactions because there's not as much work involved. If you don't have to go through the effort to get a property on blockchain as an agent and then taking it down, that's a significant time savings. Additionally, title work won't be as hard if there's a clear ownership history on the blockchain. This is going to translate into lower costs for everyone involved.

The Proposed system is built in a secured by storing all the data in a blockchain so there won't be any frauds and scams can possibly happen. In the proposed system, fractional ownership feature is also added so if many users can have a small fraction of ownership in a big land

Also the proposed system will allocate NFT (Non – fungible token) to all the buyers who are buying , so there avoid online and fake document scams

3.3 MINIMUM HARDWARE REQUIREMENTS

Processor	: Pentium Dual Core 2.00GHZ
Hard disk	: 120 GB
RAM	: 2GB (minimum)
Keyboard	: 110 keys enhanced

3.4 SOFTWARE REQUIREMENTS

Operating system	: Windows 7 ,8,9,10,11
Language	: Solidity

3.4.1 LANGUAGE SPECIFICATION– Solidity

Solidity is a high-level programming language that is used for creating smart contracts on the Ethereum blockchain. It is an object-oriented language, which is designed to provide safety, security, and reliability to the code. Solidity programming language is similar to JavaScript and has a syntax that is easy to learn for developers with experience in programming.However, creating and testing Ethereum applications requires more than just knowledge of Solidity. Several tools and frameworks are available for development and testing Ethereum applications, which can help developers to build better and more reliable applications. Two such tools are Hardhat and Chai, and a framework is Mocha.

Hardhat

Hardhat is a development environment for building and testing Ethereum applications. It includes a task runner, a local blockchain network, and a suite of plugins that make it easy for developers to create and test Ethereum applications. The task runner allows developers to automate repetitive tasks such as compiling

code, deploying contracts, and running tests. The local blockchain network provides a local Ethereum environment for developers to test their applications without the need for a real Ethereum network.
Hardhat has several plugins, including the Solidity compiler, which compiles Solidity code into bytecode that can be executed on the Ethereum network. It also has plugins for deploying contracts, testing, and debugging. The testing plugin includes support for testing smart contracts using Mocha and Chai, which makes it easy for developers to write and run tests for their applications.

Chai

Chai is an assertion library for testing JavaScript code, which provides a range of BDD/TDD-style interfaces for writing tests. It makes it easy for developers to write readable and expressive tests that can be used to verify the correctness of their code. Chai supports several assertion styles, including should, expect, and assert, which provide different ways of expressing the same assertion.

Chai can be used with any testing framework, but it is commonly used with Mocha. The Chai library provides a set of functions that can be used to assert the correctness of the code. For example, the expect function can be used to check the value of a variable, while the should function can be used to check the type of a variable.

Mocha

Mocha is a JavaScript testing framework for running tests and generating test reports. It supports a variety of testing styles and assertion libraries and can be used to test both front-end and back-end code. Mocha provides a simple and flexible API for writing tests, which makes it easy for developers to create tests that are readable and maintainable.

Mocha can be used with any assertion library, but it is commonly used with Chai. Mocha provides a suite of functions that can be used to describe the tests and specify the behavior of the code. For example, the describe function can be used to group tests together, while the it function can be used to specify individual tests.

Advantages

The use of Hardhat, Chai, and Mocha in Ethereum application development and testing provides several advantages for developers. In this section, we will discuss some of the key advantages of using these tools and frameworks.

1. Efficiency and productivity

Hardhat, Chai, and Mocha can significantly improve the efficiency and productivity of developers. Hardhat provides a local blockchain network and a suite of plugins that make it easy for developers to build and test Ethereum applications. This means that developers can test their code in a local environment without needing to connect to a real Ethereum network. Additionally, the task runner allows developers to automate repetitive tasks, which saves time and effort.

Chai and Mocha make it easy for developers to write and run tests for their applications. The libraries provide a range of BDD/TDD-style interfaces that make tests more readable and expressive. This means that developers can write tests faster and with less effort, which improves their productivity.

2. Safety and reliability

Solidity is a high-level programming language that is designed to provide safety, security, and reliability to the code. Hardhat, Chai, and Mocha can help to further improve the safety and reliability of Ethereum applications.

Hardhat provides a suite of plugins that can be used for testing and debugging Ethereum applications. This means that developers can catch bugs and issues early in the development process, which reduces the likelihood of errors occurring in the deployed application.
Chai and Mocha provide a range of assertion styles that can be used to verify the correctness of the code. This means that developers can test their code more thoroughly and ensure that it behaves as expected in different scenarios. By using these tools, developers can create applications that are more reliable and less prone to errors.

3. Flexibility and customization

Hardhat, Chai, and Mocha are highly customizable tools that can be adapted to suit the specific needs of different projects. Hardhat provides a range of plugins that can be used for different purposes, such as deploying contracts, testing, and debugging. Developers can choose which plugins to use and customize their development environment accordingly.
Chai and Mocha provide a range of assertion styles that can be used to express different types of tests. Developers can choose the assertion style that best suits their needs and write tests that are more readable and expressive. Additionally, Mocha provides a simple and flexible API for writing tests, which allows developers to create tests that are tailored to their specific requirements.

4. Community support

Hardhat, Chai, and Mocha are widely used tools and frameworks in the Ethereum development community. This means that developers can benefit from a large and active community of users who can provide support and assistance. There are numerous online resources, such as documentation, tutorials, and forums, where developers can find help and advice on using these tools and frameworks.

5. Future-proofing

Hardhat, Chai, and Mocha are constantly evolving tools and frameworks that are updated regularly to keep up with the latest developments in Ethereum technology. By using these tools, developers can future-proof their applications and ensure that they are compatible with the latest Ethereum standards and protocols.

CHAPTER 4

SYSTEM DESIGN

4.1 PROPOSED SYSTEM

4.1.1 SYSTEM ARCHITECTURE

This graphic provides a concise and understandable description of all the entities currently integrated into the system. The diagram shows how the many actions and choices are linked together. You might say that the whole process and how it was carried out is a picture. The figure below shows the functional connections between various entities.

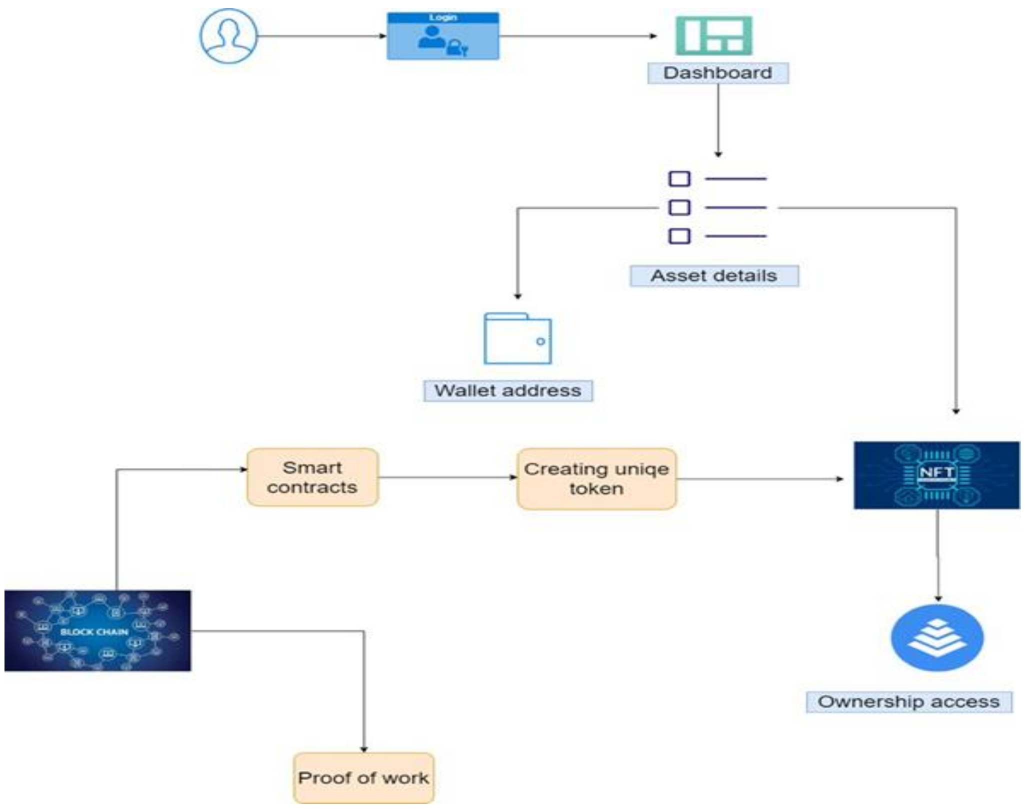


Fig 4.1 – Architecture Diagram

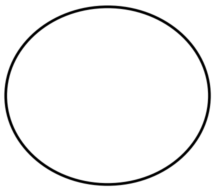
4.1.2 DATA FLOW DIAGRAM

To illustrate the movement of information throughout a procedure or system, one might use a Data-Flow Diagram (DFD). A data-flow diagram does not include any decision rules or loops, as the flow of information is entirely one-way. A flowchart can be used to illustrate the steps used to accomplish a certain data-driven task. Several different notations exist for representing data-flow graphs. Each data flow must have a process that acts as either the source or the target of the information exchange. Rather than utilizing a data-flow diagram, users of UML often substitute an activity diagram. In the realm of data-flow plans, site-oriented data-flow plans are a subset. Identical nodes in a data-flow diagram and a Petri net can be thought of as inverted counterparts since the semantics of data memory are represented by the locations in the network. Structured data modeling (DFM) includes processes, flows, storage, and terminators.

Data Flow Diagram Symbols

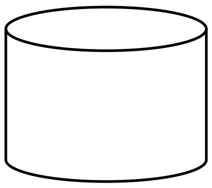
Process

A process is one that takes in data as input and returns results as output.



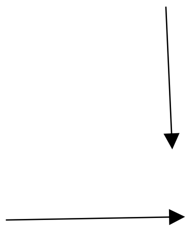
Data Store

In the context of a computer system, the term "data stores" is used to describe the various memory regions where data can be found. In other cases, "files" might stand in for data.



Data Flow

Data flows are the pathways that information takes to get from one place to another. Please describe the nature of the data being conveyed by each arrow.



External Entity

In this context, "external entity" refers to anything outside the system with which the system has some kind of interaction. These are the starting and finishing positions for inputs and outputs, respectively.

DATA FLOW DIAGRAM

The whole system is shown as a single process in a level DFD. Each step in the system's assembly process, including all intermediate steps, are recorded here. The "basic system model" consists of this and 2-level data flow diagrams.

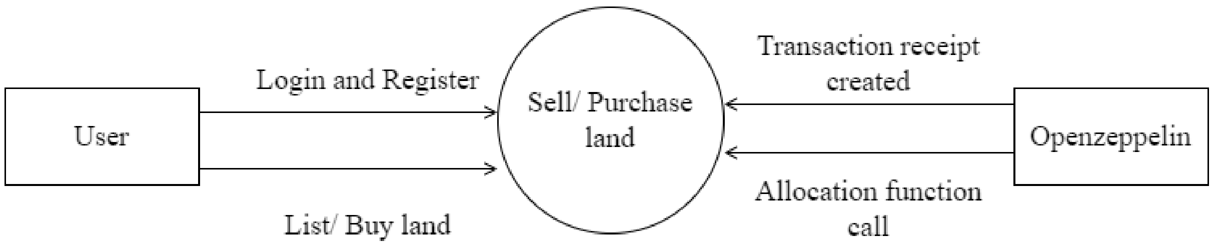


Fig 4.2 – Data Flow Diagram Level 0

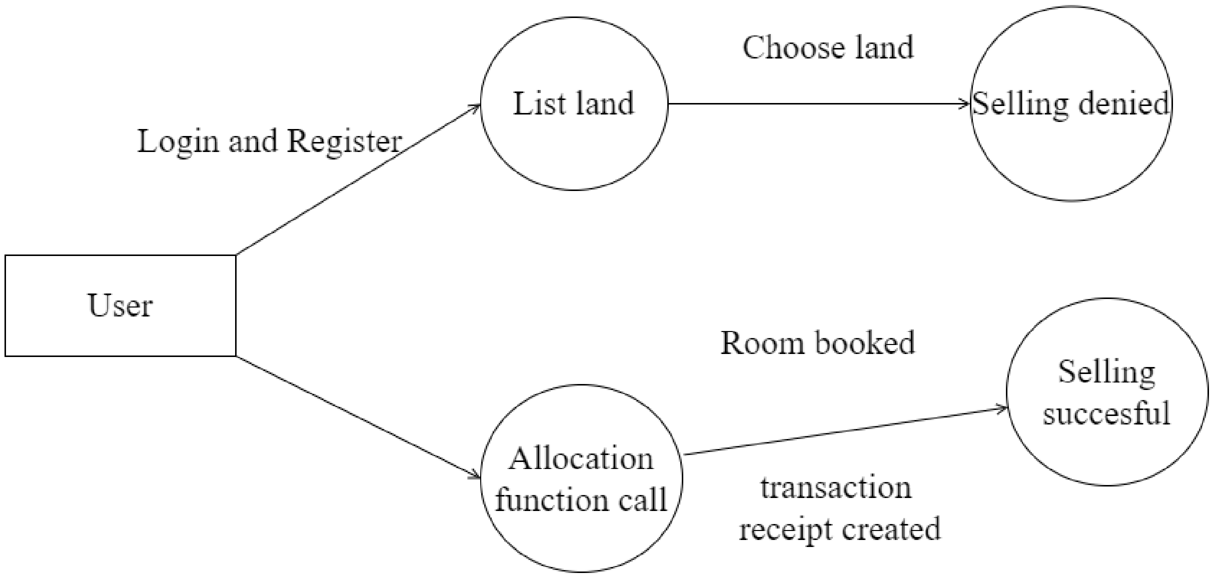


Fig 4.3 – Data Flow Diagram Level 1

4.1.3 ENTITY RELATIONSHIP DIAGRAM

- **Definition**

The relationships between database entities can be seen using an entity-relationship diagram (ERD). The entities and relationships depicted in an ERD can have further detail added to them via data object descriptions. In software engineering, conceptual and abstract data descriptions are represented via entity-relationship models (ERMs). Entity-relationship diagrams (ERDs), entity-relationship diagrams (ER), or simply entity diagrams are the terms used to describe the resulting visual representations of data structures that contain relationships between entities. As such, a data flow diagram can serve dual purposes. To demonstrate how data is transformed across the system. To provide an example of the procedures that affect the data flow.

1. **One-to-One**

   Whenever there is an instance of entity (A), there is also an instance of entity (B) (B). In a sign-in database, for instance, only one security mobile number (S) is associated with each given customer name (A) (B).

2. One-to-Many

For each instance of entity B, there is exactly one occurrence of entry A, regardless of how many instances of entity B there are.

For a corporation whose employees all work in the same building, for instance, the name of the building (A) has numerous individual associations with employees (B), but each of these B's has only one individual link with entity A.

3. Many-to-Many

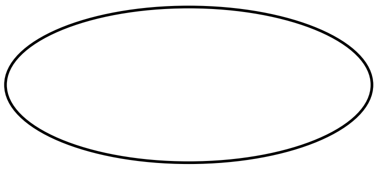
For each instance of entity B, there is exactly one occurrence of entry A, regardless of how many instances of entity B there are.

In a corporation where everyone works out of the same building, entity A is associated with many different Bs, but each B has only one A.

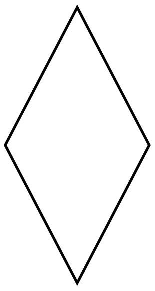
SYMBOLS USED

External Entity

Attribute



Relationship



Data Flow

Fig 4.4 – Entity Relationship Diagram

4.1.4 USE-CASE DIAGRAM

The possible interactions between the user, the dataset, and the algorithm are often depicted in a use case diagram. It's created at the start of the procedure.

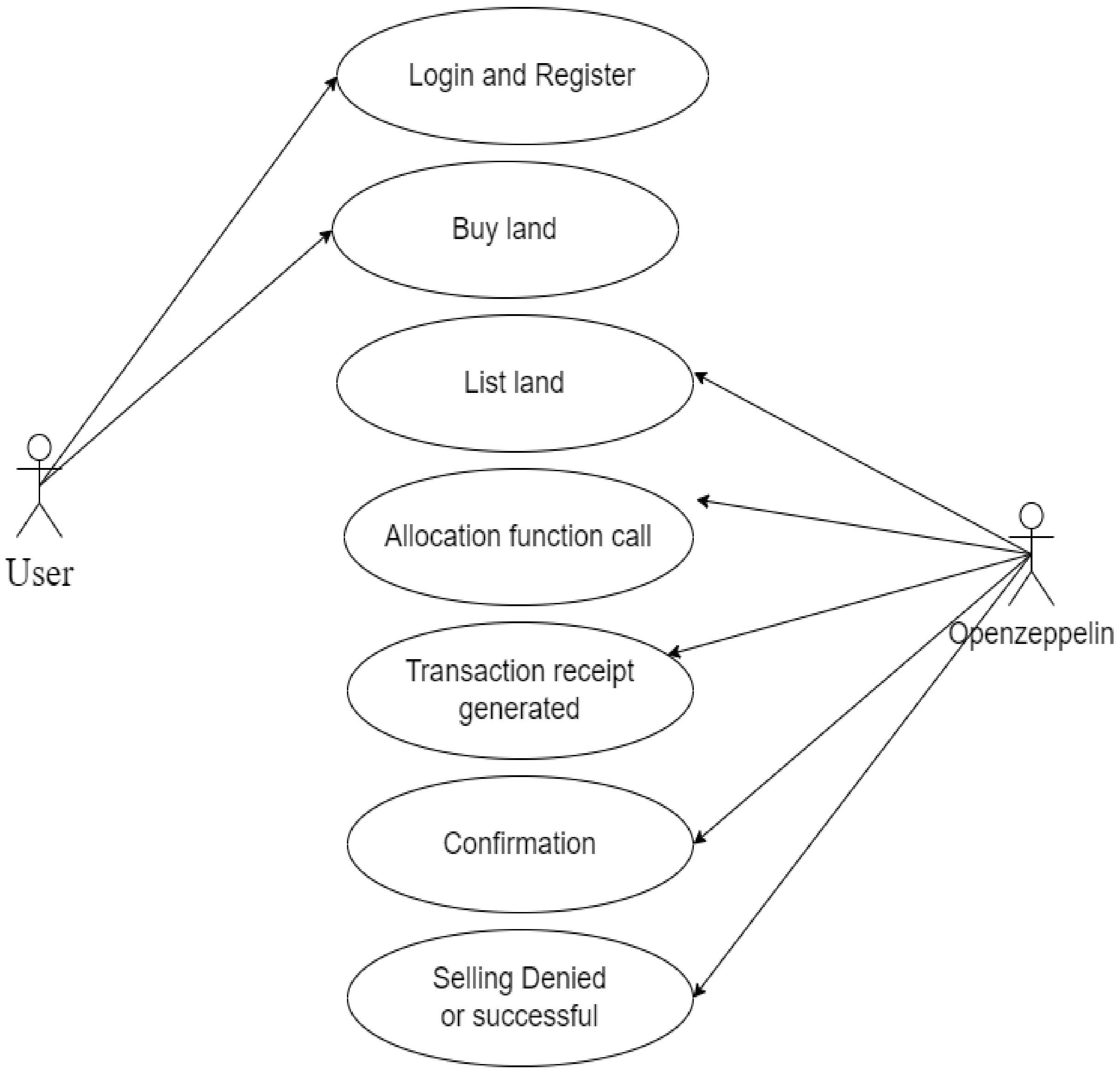


Fig 4.5 – Use-Case Diagram

4.1.5 SEQUENCE DIAGRAM

These are another type of interaction-based diagram used to display the workings of the system. They record the conditions under which objects and processes cooperate.

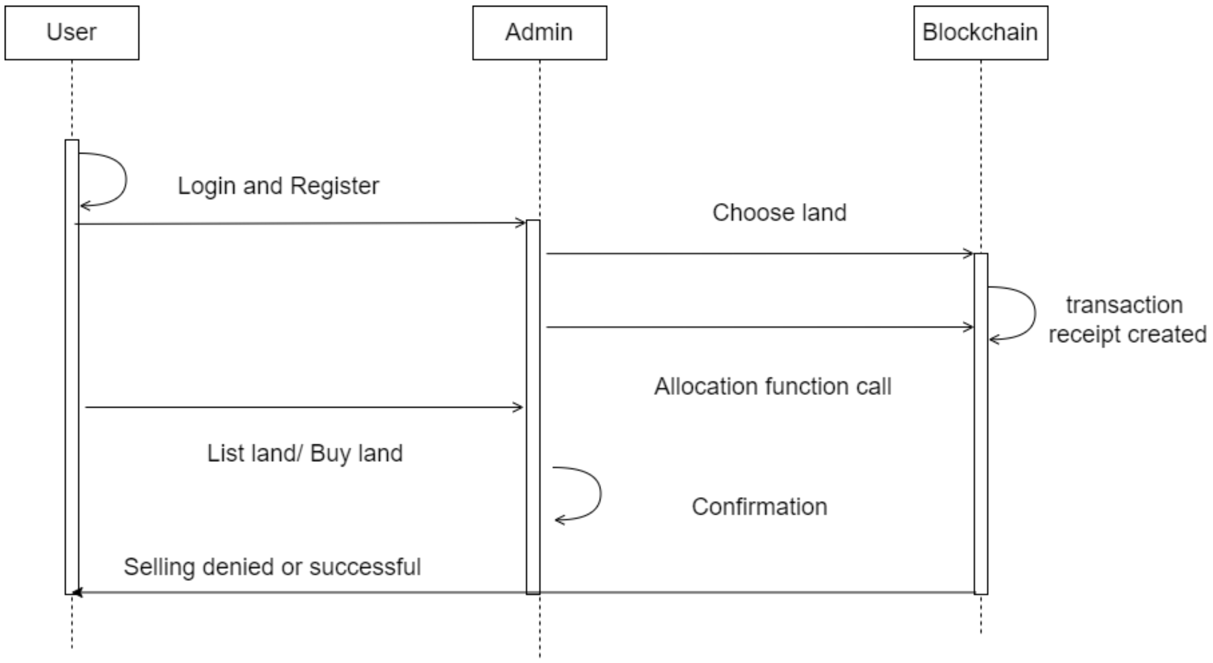


Fig 4.6 – Sequence Diagram

4.1.6 CLASS DIAGRAM

In essence, this is a "context diagram," another name for a contextual diagram. It simply stands for the very highest point, the 0 Level, of the procedure. As a whole, the system is shown as a single process, and the connection to externalities is shown in an abstract manner.

- A + indicates a publicly accessible characteristic or action.
- A - a privately accessible one.
- A # a protected one.
- A - denotes private attributes or operations.

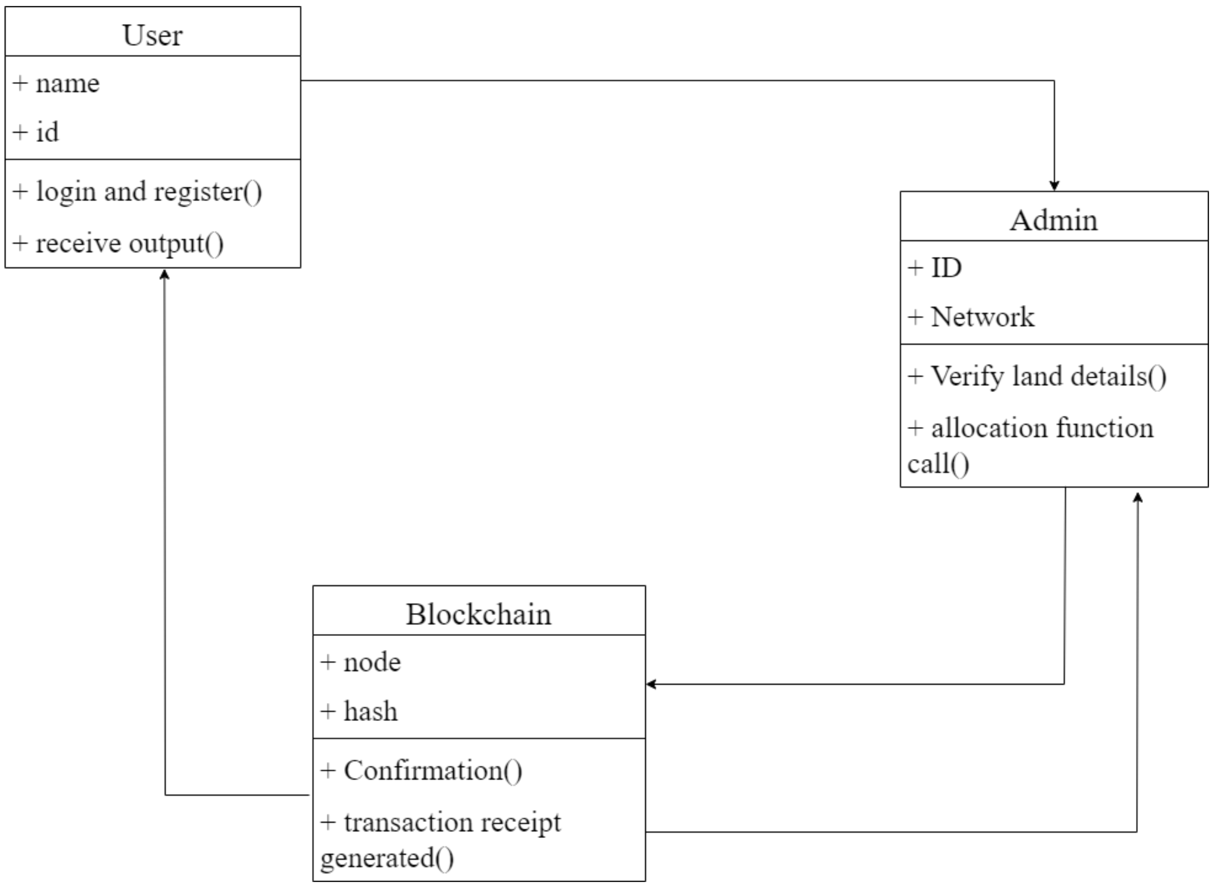


Fig 4.7 – Class Diagram

CHAPTER 5

SYSTEM ANALYSIS

5.1 FEASIBILITY STUDY

With an eye towards gauging the project's viability and improving server performance, a business proposal defining the project's primary goals and offering some preliminary cost estimates is offered here. Your proposed system's viability may be assessed once a comprehensive study has been performed. It is essential to have a thorough understanding of the core requirements of the system at hand before beginning the feasibility study. The feasibility research includes mostly three lines of thought:

- Economical feasibility
- Technical feasibility
- Operational feasibility
- Social feasibility

5.1.1 ECONOMICAL FEASIBILITY

The study's findings might help upper management estimate the potential cost savings from using this technology. The corporation can only devote so much resources to developing and analysing the system before running out of money. Every dollar spent must have a valid reason. As the bulk of the used technologies are open-source and free, the cost of the updated infrastructure came in far cheaper than anticipated. It was really crucial to only buy customizable products.

5.1.2 TECHNICAL FEASIBILITY

This research aims to establish the system's technical feasibility to ensure its smooth development. Adding additional systems shouldn't put too much pressure on the IT staff.
Hence, the buyer will experience unnecessary anxiety. Due to the low likelihood of any adjustments being necessary during installation, it is critical that the system be as simple as possible in its design.

5.1.3 OPERATIONAL FEASIBILITY

An important aspect of our research is hearing from people who have actually used this technology. The procedure includes instructing the user on how to make optimal use of the resource at hand. The user shouldn't feel threatened by the system, but should instead see it as a necessary evil. Training and orienting new users has a direct impact on how quickly they adopt a system. Users need to have greater faith in the system before they can submit constructive feedback.

5.1.4 SOCIAL FEASIBILITY

The social feasibility of the platform would depend on the following factors:
User adoption: The success of the platform would depend on its user
adoption. It would need to be marketed effectively to attract users andgain their trust.
Customer support: The platform would require a dedicated customer
support team to provide assistance to users.
Regulatory compliance: The platform would need to complywithrelevant regulations in the jurisdictions

CHAPTER 6
SYSTEM IMPLEMENTATION

6.1 SYSTEM IMPLEMENTATION

6.1.1 MODULE 1: RealEstate.sol Contract

The RealEstate.sol contract is used to handle all the user storing & land registration logic.

Modifiers-

onlyAdmin()

It restricts function call to only Admin.The transaction will get reverted when if someone else tries to call the function.

onlyRegistrar()

It restricts the function call to only the Registrar.The transaction will get reverted if someone other than the registrar tries to call the function

Structs

Structs are custom data types.

ApprovedUser

This struct has id as unsigned integer,name of the user as string,district-code as unsigned integer,property count as unsigned integer,Acceptance status as Boolean.

Property

This struct has id as unsigned integer, Registrar's Name as string,Registrar's Address as address, District code as unsigned integer,Acceptance Status as boolean.

Functions

registerProperty()

This function is used to register property.At first it checks if the property is already registered or not.If the property is already registered then the transaction is reverted with a message saying that “Prop already registered”.After that the address of the msg.sender is converted to userID.Then the nextProp counter is incremented.Which signifies that a new property has been registered.

approveProperty()

onlyRegistrar modifier is used here.For that the calling access is restricted to only the Registrar.At first in the require statement ,it checks that the Acceptance of the prop is False or True.If it is false then that means the property is not Approved.If it is true , then the transaction is reverted with a message saying “already accepted”. Then a registrar ID is assigned.

initiateTransfer()

It takes property ID,part ID, New owner’s ID as parameter.It first checks if the part ID exists or not.If it exits then the transaction is reverted with a message saying that “Tranfer initiated alread”.It also checks if the acceptance status of the prop is true or not and if the user is verified or not.

acceptTransferRegistrar()

This function is used to accept the transferring of registrar.At first it checks the transfer is initiated or not.Then new owner address is assigned. Then registrar is created. Then it checks if the registrar is accepted and the new owner is accepted or not. Then the isUnderTransfer boolean value is assigned to false, registrarAccepted value, newOwnerAccepted value is assigned a false value.Then the new owner is assigned to the 0th address.Then new owner's address is pushed to the property owner.Then the property owners is addressed is finally assigned to the new owner's address.

userProposal()

This takes name, district-code, adhaar number as parameters. sThis at first checks if the user is already a user or not. Then it is checked if there is any pending user request or not. Then pendingUserReq is set to true. Finally the userID is incremented.

approveUser()

This has onlyRegistered modifier used in it .So the calling access is restricted to only Registrar.At first it checks if the user is already approved or not.Then the acceptance status of the user is set to true.Pending user request is set to false.

registrarProposal()

It takes aadhar number,name,district code as parameters. It checks if the person is already registrar or not. Then the pending transaction is set to true. Then the next Registrar id is incremented.

Libraries and Framworks Used

The right blockchain framework to run our project is one of the most basic and important things to think about.

Libraries Used For Smart Contracts

Hardhat

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

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

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

To install it, let’s go to an empty folder, run npm init, and follow the instructions. We could have used a different package manager, like yarn, but we thought it is better to use npm 7 or later because it makes it easier to install Hardhat plugins.

Once the project is ready, it is necessary to run

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

Connecting a wallet or Dapp to Hardhat Network

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

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

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

Chai

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

Mocha

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

We installed Mocha by the following codes:

Install with npm globally:

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

or as a development dependency for your project:

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

Working with Mocha

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

In editor:

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

Back in the terminal:

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

  Array
```

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

Setting up a test script in package.json:

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

6.1.2 MODULE 2: Front-end

The ABI & address of the deployed contract is used along with the help of web3.js library to connect the frontend with the smart-contract. The whole code for the front-end part is contained in App.js file.

At first it is checked if Metamask wallet is exists in the user’s browser or not.
If Metamask does not exist in the user’s wallet , then the
User is prompted to install a Wallet
Then the user is asked to connect to the wallet.
After the user signs the transaction the landing page is shown,where the user can fill the form to register their property and save the data on the tamper-proof, immutable Blockchain.

Libraries And Toola Used

Contract ABI

The standard way to interact with contracts in the Ethereum ecosystem, both from outside the blockchain and between contracts, is through the Contract Application

Binary Interface (ABI). Based on the type of data, as described in this specification, the data is encoded. The encoding does not describe itself, so it needs a schema to be decoded.

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

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

Ether.js Library

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

Metamask

Metamask acts as a bridge between websites and the Ethereum blockchain. The smart contracts in the blockchain check to see if the node in the network can access that data or not. When a local blockchain network is built, Metamask wallet can take care of the nodes. Metamask is a cryptocurrency wallet that can be added to Google Chrome. It is a way to communicate with the Ethereum blockchain. Metamask is used to store Ethereum accounts. It is safe and easy to use, and can also be used to deploy to the main Ethereum networks.

React.js

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. React can be used as a base in the development of single-page, mobile, or server-rendered applications with frameworks like Next.js.

Next.js

Next.js is a set of tools for making universal React.js apps that are rendered on the server (or statically pre-rendered). Next.js uses powerful tools like Webpack, Babel, and Uglify and gives the end user a very simple interface: next (to develop), next build (to get ready for production), next start (to serve), or next export (to pre-render to static files).

Installed next, react and react-dom in your project:

```
npm install next react react-dom
# or
yarn add next react react-dom
# or
pnpm add next react react-dom
```

Opened package.json and added the following scripts:

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start",
  "lint": "next lint"
}
```

These scripts refer to the different stages of developing an application:

dev - Runs next dev to start Next.js in development mode
build - Runs next build to build the application for production usage
start - Runs next start to start a Next.js production server
lint - Runs next lint to set up Next.js' built-in ESLint configuration

Created two directories pages and public at the root of your application:

pages - Associated with a route based on their file name. For example pages/about.js is mapped to /about
public - Stores static assets such as images, fonts, etc. Files inside public directory can then be referenced by your code starting from the base URL (/).

Pages are the main idea behind Next.js. A page is a React Component that is exported from a file in the pages directory that ends in.js,.jsx,.ts, or.tsx. You can even use the filename to add dynamic route parameters. We start by putting the index.js file in the pages directory. This is the page that gets shown when a user goes to your application's root.

Populate pages/index.js with the following contents:

```
function HomePage() {
  return <div>Welcome to Next.js!</div>
}
```

CHAPTER 7
METHODOLOGIES

7. Methodologies

The use of blockchain technology in real estate has been gaining popularity in recent years. This is because blockchain can help reduce fraud, increase transparency, and streamline the transaction process. Here are some potential methodologies for real estate in blockchain:

Smart contracts: Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller being directly written into code. This can help eliminate the need for intermediaries and speed up the transaction process.

Tokenization: Tokenization refers to the process of representing a physical asset, such as a property, with a digital token on a blockchain. This can help increase liquidity and enable fractional ownership of the property.

Decentralized databases: Decentralized databases can help increase transparency and reduce the risk of fraud by storing property records on a blockchain. This can help prevent double-spending and tampering with records.

Cryptocurrencies: Cryptocurrencies, such as Bitcoin or Ethereum, can be used to facilitate real estate transactions. This can help reduce transaction costs and increase the speed of transactions.

Identity verification: Blockchain can be used to securely verify the identity of buyers and sellers, which can help prevent fraud and increase the trust between parties.

Due diligence: Blockchain can be used to store important documents, such as title deeds and property surveys, which can help streamline the due diligence process.

Overall, the use of blockchain technology in real estate has the potential to significantly improve the efficiency, transparency, and security of real estate transactions. However, there are also challenges to be addressed, such as regulatory issues, technical implementation, and adoption by industry stakeholders.

CHAPTER 8
SYSTEM TESTING

8.1 SYSTEM TESTING

8.1.1 Unit Testing

The term "unit testing" refers to a specific kind of software testing in which discrete elements of a program are investigated. The purpose of this testing is to ensure that the software operates as expected.

Test Case 1: Test the creation of a new property listing: Verify that a new property can be successfully created and added to the blockchain, including all relevant details such as address, price, and description.

Input: Property information (address, owner, price, etc.)

Output: Property information is stored on the blockchain

Property can be retrieved from the blockchain using its unique ID

Test Case 2:Test the retrieval of property listings: Verify that property listings can be successfully retrieved from the blockchain and displayed to users, including all relevant details such as address, price, and description.

Input: Property ID

Output: List of all transactions associated with the property

Transaction details (date , new owner, etc.) are included

Test Case 3: Test the updating of property listings: Verify that property listings can be successfully updated on the blockchain, including changing details such as price or description.

Input: Property ID

Updated property information

Output: Property information is updated on the blockchain

Updated property information can be retrieved from the blockchain using its unique ID

8.1.2 Security Testing

This type of testing is performed to ensure that the systemis secureandcannot be compromised by attackers. Security testing for the escrowplatform would involve testing for vulnerabilities such as smart contract
bugs, denial-of-service attacks, and other security threats.

Test case 1: SQL injection attack

Input: Malicious code entered into a text field

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

Test case 2: Cross-site scripting attack

Input: Malicious code entered into a text field

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

Test case 3: Brute force attack

Input: Multiple incorrect login attempts

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

8.1.3 Functional Testing

One kind of software testing is called functional testing, and it involves comparing the system to the functional requirements and specifications. In order to test functions, their input must first be provided, and then the output must be examined. Functional testing verifies that an application successfully satisfies all of its requirements in the correct manner. This particular kind of testing is not concerned with the manner in which processing takes place; rather, it focuses on the outcomes of processing. Therefore, it endeavours to carry out the test cases, compare the outcomes, and validate the correctness of the results.

Test Case 1: Smart contract and blockchain integration test: To verify that the smart contract is integrated correctly with the blockchain and is functioning as expected.

Input: Property information (address, owner, price, etc.)

Output: Property information is stored on the blockchain

Property can be retrieved from the blockchain using its unique ID

2. Front-end and back-end integration test: To ensure that the front-end and back-end of the application are integrated seamlessly and the system is working as expected.

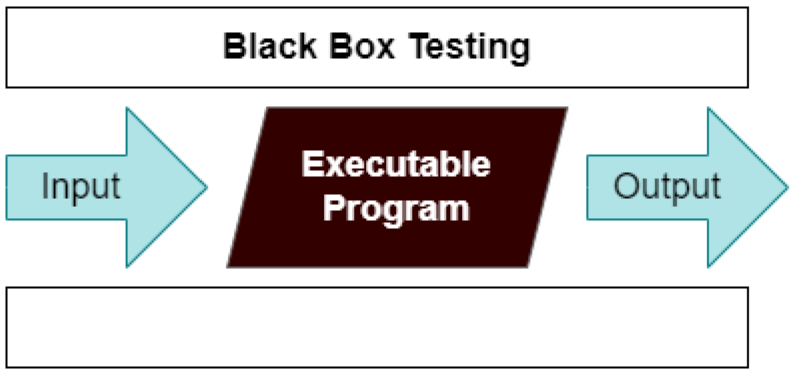
3. Wallet integration test: To ensure that the system can interact with the user's wallet and perform the required transactions smoothly.

8.1.4 TESTING TECHNIQUES

There are many different techniques or methods for testing the software, including the following:

BLACK BOX TESTING

During this kind of testing, the user does not have access to or knowledge of the internal structure or specifics of the data item being tested. In this method, test cases are generated or designed only based on the input and output values, and prior knowledge of either the design or the code is not necessary. The testers are just conscious of knowing about what is thought to be able to do, but they do not know how it is able to do it.



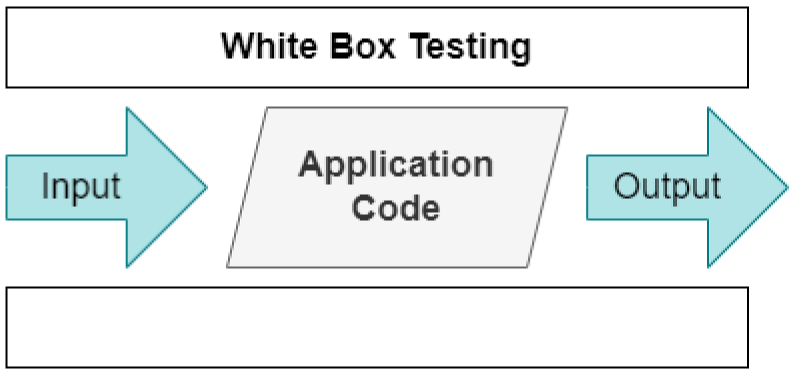
For example, without having any knowledge of the inner workings of the website, we test the web pages by using a browser, then we authorise the input, and last, we test and validate the outputs against the intended result.

Test Cases

1. Verify that the system accurately records property ownership information and can trace the transaction history of a property.
2. Test the accuracy of the smart contracts used to execute property transactions, such as buying or selling property or rental agreements.
3. Ensure that the system can accurately verify the identity of users and prevent unauthorized access to the platform.

WHITE BOX TESTING

During this kind of testing, the user is aware of the internal structure and details of the data item, or they have access to such information. In this process, test cases are constructed by referring to the code. Programming is extremely knowledgeable of the manner in which the application of knowledge is significant. White Box Testing is so called because, as we all know, in the tester's eyes it appears to be a white box, and on the inside, everyone can see clearly. This is how the testing got its name.



As an instance, a tester and a developer examine the code that is implemented in each field of a website, determine which inputs are acceptable and which are not, and then check the output to ensure it produces the desired result. In addition, the decision is reached by analyzing the code that is really used.

Test Cases

1. Test the functionality of the smart contracts by simulating different scenarios of real estate transactions.
2. Test the accuracy and security of the hashing algorithm used to generate unique IDs for each property.
3. Test the validity of each transaction recorded on the blockchain by cross-checking with the real-world data.

8.1.5 Compatibility Testing

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

Test case 1: Compatibility with different browsers.

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

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

Test case 2: Compatibility with different operating systems.

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

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

Test case 3: Compatibility with different devices

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

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

Test case 4: Compatibility with different network environments.

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

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

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

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

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

CHAPTER 9

CONCLUSION

9.1 CONCLUSION

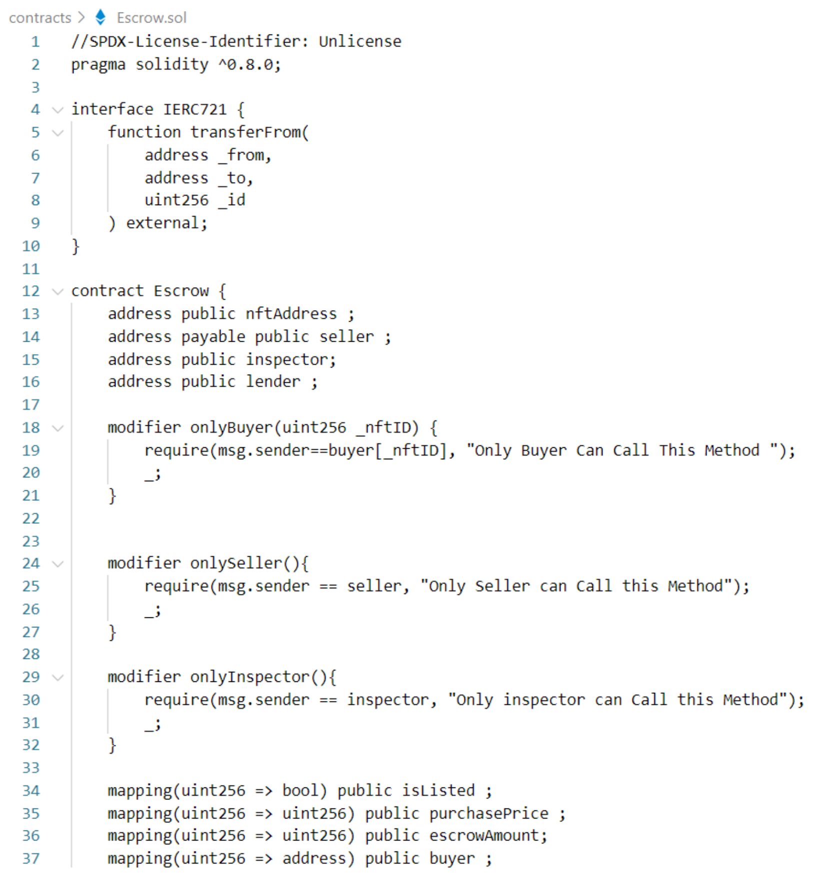
In conclusion, the implementation of a decentralized real estate system using blockchain technology has the potential to revolutionize the real estate industry by providing a secure, transparent, and efficient method of recording and transferring property ownership. The use of smart contracts enables automated and immutable transactions that remove the need for intermediaries, reducing costs and streamlining the buying and selling process. Additionally, blockchain's tamper-proof nature ensures that all information recorded on the ledger is accurate and trustworthy, mitigating the risk of fraud and improving the overall integrity of the system. Despite the promising potential of blockchain technology in the real estate industry, there are still some challenges that need to be addressed, such as regulatory and legal hurdles, standardization of data, and adoption by industry stakeholders. Moreover, the implementation of a decentralized system requires a significant investment of resources and infrastructure, which may deter some companies from adopting the technology. However, as the benefits of blockchain become more apparent and the technology becomes more widely adopted, it is likely that we will see an increase in decentralized real estate systems. The potential for increased transparency, reduced costs, and improved efficiency will drive the industry towards this new way of conducting real estate transactions. Ultimately, the successful implementation of a decentralized real estate system using blockchain technology could transform the industry, making it more accessible, efficient, and secure for all stakeholders involved.

CHAPTER 10

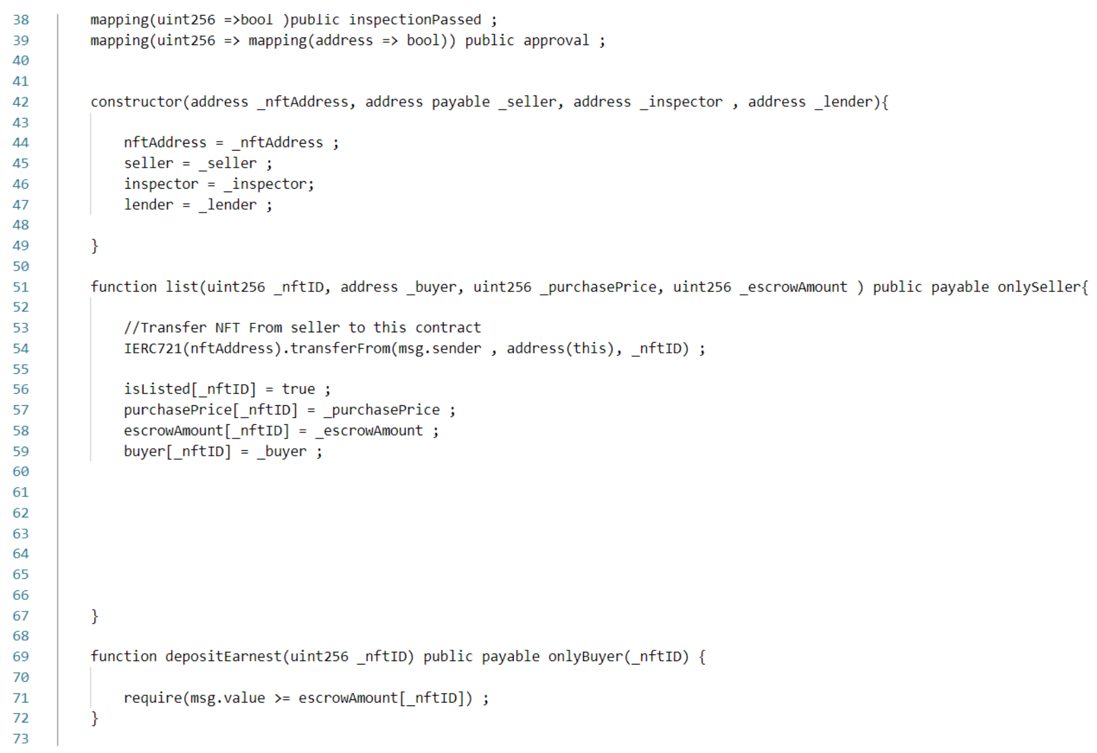
APPENDIX

10.1 CODING

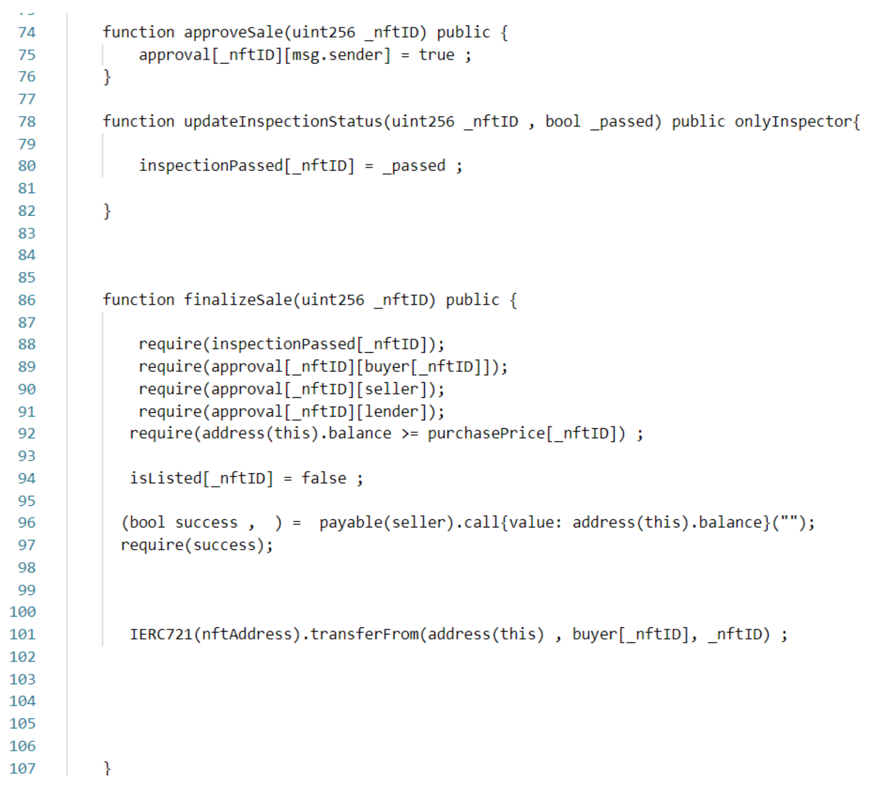
Escrow.sol



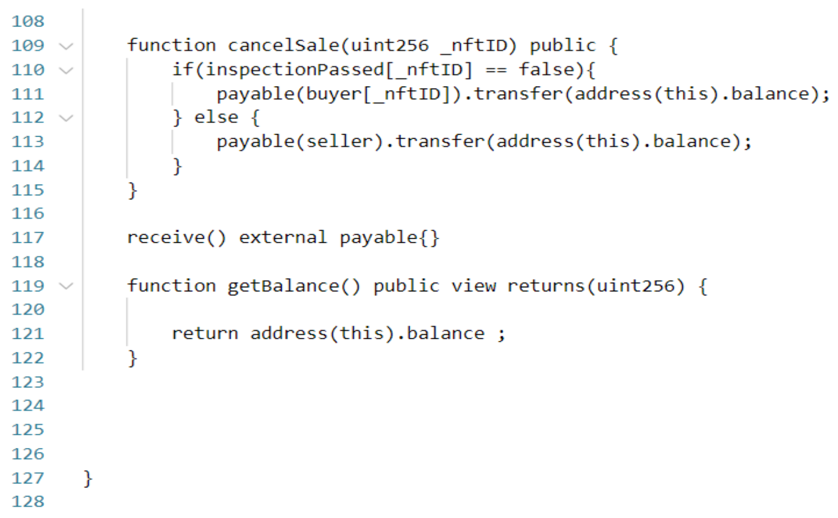
```
//SPDX-License-Identifier: Unlicense
pragma solidity ^0.8.0;

interface IERC721 {
    function transferFrom(
        address _from,
        address _to,
        uint256 _id
    ) external;
}

contract Escrow {
    address public nftAddress ;
    address payable public seller ;
    address public inspector;
    address public lender ;

    modifier onlyBuyer(uint256 _nftID) {
        require(msg.sender==buyer[_nftID], "Only Buyer Can Call This Method ");
        _;
    }


    modifier onlySeller(){
        require(msg.sender == seller, "Only Seller can Call this Method");
        _;
    }

    modifier onlyInspector(){
        require(msg.sender == inspector, "Only inspector can Call this Method");
        _;
    }

    mapping(uint256 => bool) public isListed ;
    mapping(uint256 => uint256) public purchasePrice ;
    mapping(uint256 => uint256) public escrowAmount;
    mapping(uint256 => address) public buyer ;
```



```
    mapping(uint256 =>bool )public inspectionPassed ;
    mapping(uint256 => mapping(address => bool)) public approval ;


    constructor(address _nftAddress, address payable _seller, address _inspector , address _lender){

        nftAddress = _nftAddress ;
        seller = _seller ;
        inspector = _inspector;
        lender = _lender ;

    }

    function list(uint256 _nftID, address _buyer, uint256 _purchasePrice, uint256 _escrowAmount ) public payable onlySeller{

        //Transfer NFT From seller to this contract
        IERC721(nftAddress).transferFrom(msg.sender , address(this), _nftID) ;

        isListed[_nftID] = true ;
        purchasePrice[_nftID] = _purchasePrice ;
        escrowAmount[_nftID] = _escrowAmount ;
        buyer[_nftID] = _buyer ;








    }

    function depositEarnest(uint256 _nftID) public payable onlyBuyer(_nftID) {

        require(msg.value >= escrowAmount[_nftID]) ;
    }

```

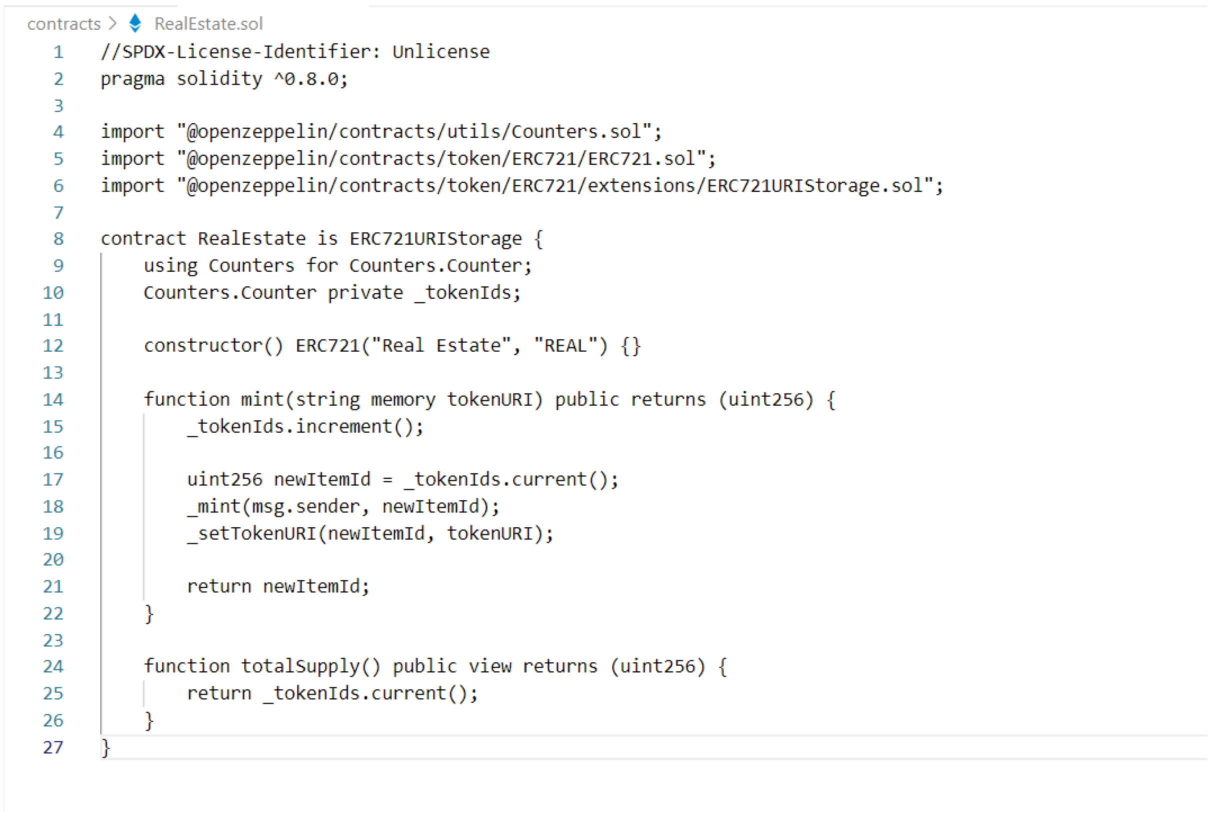


```
    function approveSale(uint256 _nftID) public {
        approval[_nftID][msg.sender] = true ;
    }

    function updateInspectionStatus(uint256 _nftID , bool _passed) public onlyInspector{

        inspectionPassed[_nftID] = _passed ;

    }



    function finalizeSale(uint256 _nftID) public {

        require(inspectionPassed[_nftID]);
        require(approval[_nftID][buyer[_nftID]]);
        require(approval[_nftID][seller]);
        require(approval[_nftID][lender]);
      require(address(this).balance >= purchasePrice[_nftID]) ;

      isListed[_nftID] = false ;

     (bool success ,  ) =  payable(seller).call{value: address(this).balance}("");
     require(success);



      IERC721(nftAddress).transferFrom(address(this) , buyer[_nftID], _nftID) ;





    }
```



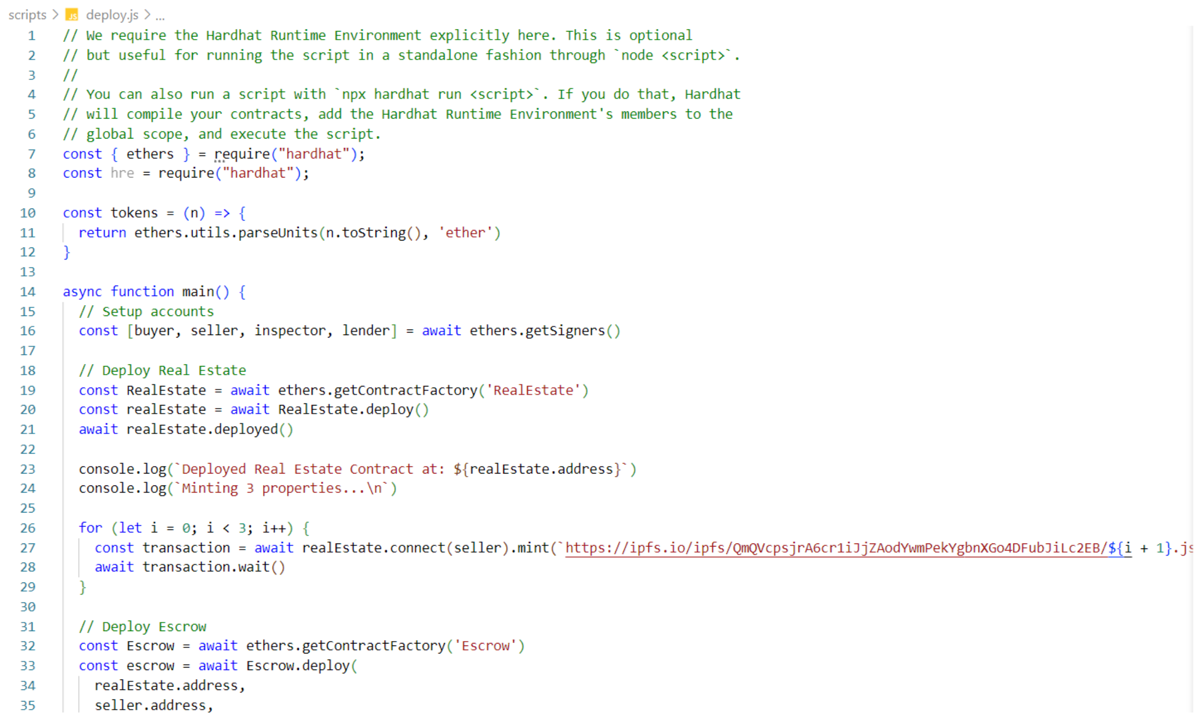
```
    function cancelSale(uint256 _nftID) public {
        if(inspectionPassed[_nftID] == false){
            payable(buyer[_nftID]).transfer(address(this).balance);
        } else {
            payable(seller).transfer(address(this).balance);
        }
    }

    receive() external payable{}

    function getBalance() public view returns(uint256) {

        return address(this).balance ;
    }




}
```

RealEsatate.sol

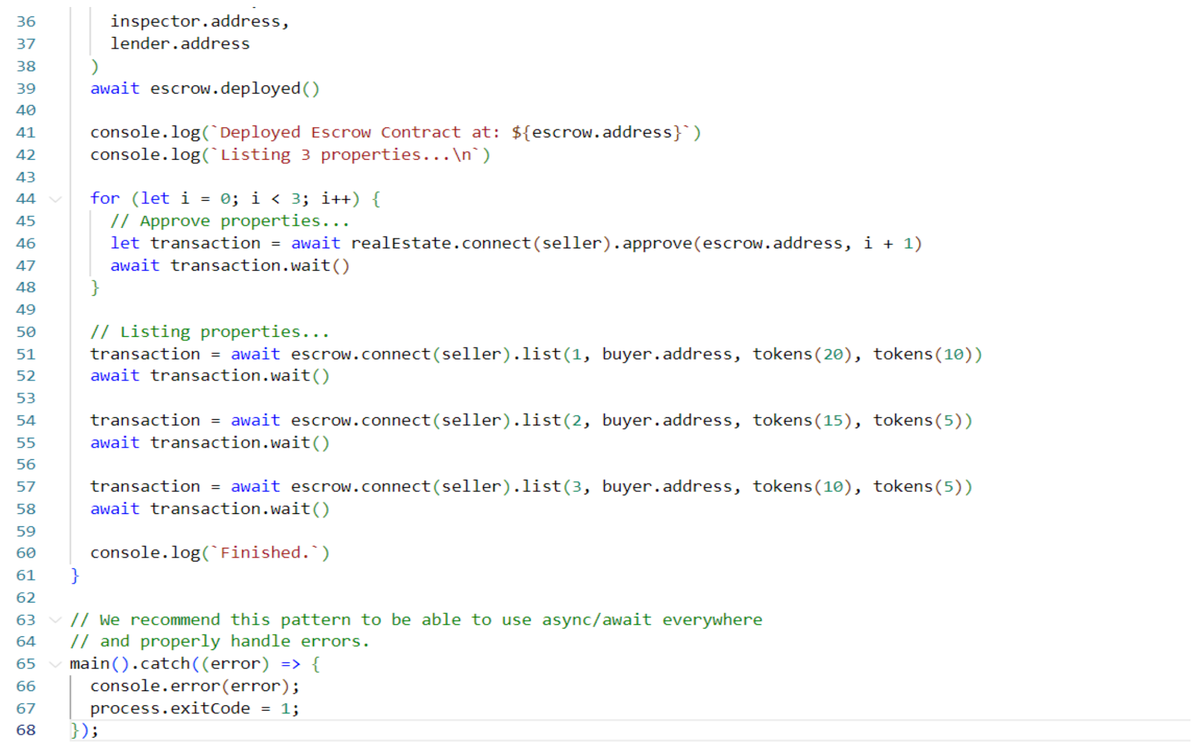


```
//SPDX-License-Identifier: Unlicense
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

contract RealEstate is ERC721URIStorage {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    constructor() ERC721("Real Estate", "REAL") {}

    function mint(string memory tokenURI) public returns (uint256) {
        _tokenIds.increment();

        uint256 newItemId = _tokenIds.current();
        _mint(msg.sender, newItemId);
        _setTokenURI(newItemId, tokenURI);

        return newItemId;
    }

    function totalSupply() public view returns (uint256) {
        return _tokenIds.current();
    }
}
```

Deploy.js

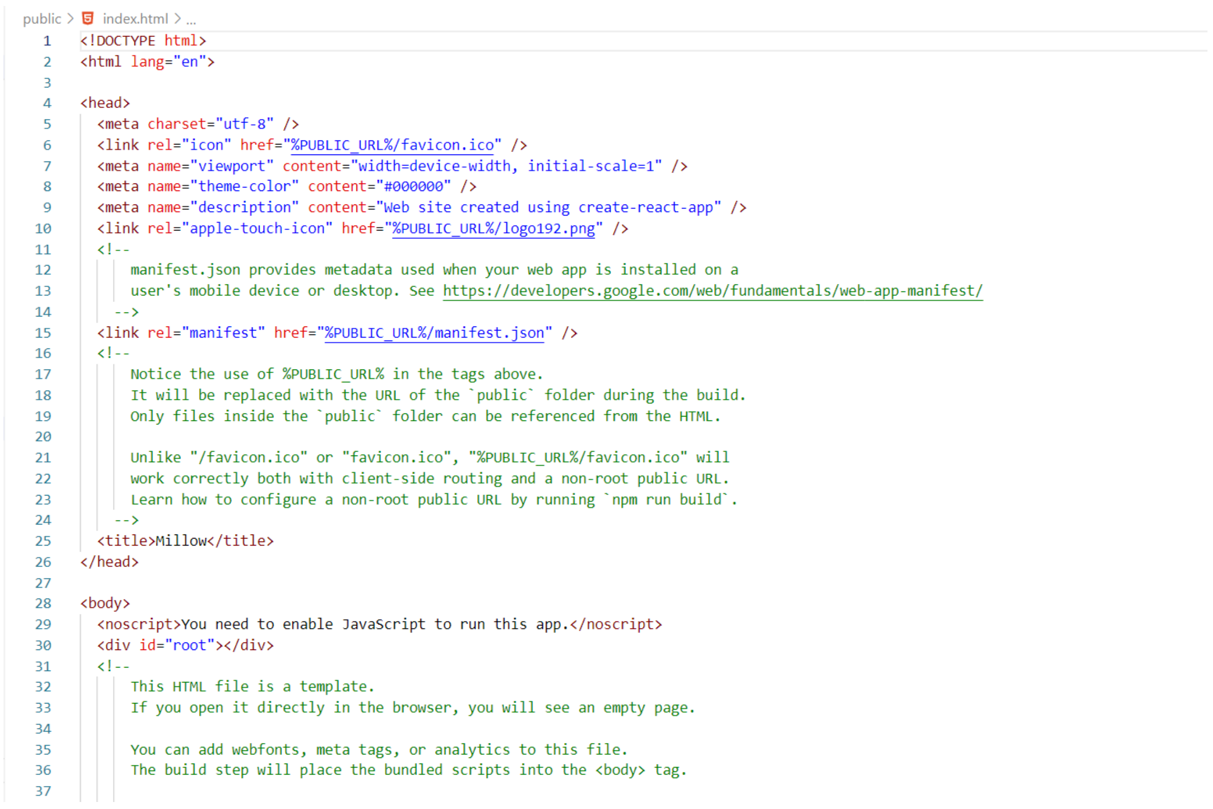


```javascript
// We require the Hardhat Runtime Environment explicitly here. This is optional
// but useful for running the script in a standalone fashion through `node <script>`.
//
// You can also run a script with `npx hardhat run <script>`. If you do that, Hardhat
// will compile your contracts, add the Hardhat Runtime Environment's members to the
// global scope, and execute the script.
const { ethers } = require("hardhat");
const hre = require("hardhat");

const tokens = (n) => {
  return ethers.utils.parseUnits(n.toString(), 'ether')
}

async function main() {
  // Setup accounts
  const [buyer, seller, inspector, lender] = await ethers.getSigners()

  // Deploy Real Estate
  const RealEstate = await ethers.getContractFactory('RealEstate')
  const realEstate = await RealEstate.deploy()
  await realEstate.deployed()

  console.log(`Deployed Real Estate Contract at: ${realEstate.address}`)
  console.log(`Minting 3 properties...\n`)

  for (let i = 0; i < 3; i++) {
    const transaction = await realEstate.connect(seller).mint(`https://ipfs.io/ipfs/QmQVcpsjrA6cr1iJjZAodYwmPekYgbnXGo4DFubJiLc2EB/${i + 1}.js
    await transaction.wait()
  }

  // Deploy Escrow
  const Escrow = await ethers.getContractFactory('Escrow')
  const escrow = await Escrow.deploy(
    realEstate.address,
    seller.address,
```



```javascript
    inspector.address,
    lender.address
  )
  await escrow.deployed()

  console.log(`Deployed Escrow Contract at: ${escrow.address}`)
  console.log(`Listing 3 properties...\n`)

  for (let i = 0; i < 3; i++) {
    // Approve properties...
    let transaction = await realEstate.connect(seller).approve(escrow.address, i + 1)
    await transaction.wait()
  }

  // Listing properties...
  transaction = await escrow.connect(seller).list(1, buyer.address, tokens(20), tokens(10))
  await transaction.wait()

  transaction = await escrow.connect(seller).list(2, buyer.address, tokens(15), tokens(5))
  await transaction.wait()

  transaction = await escrow.connect(seller).list(3, buyer.address, tokens(10), tokens(5))
  await transaction.wait()

  console.log(`Finished.`)
}

// We recommend this pattern to be able to use async/await everywhere
// and properly handle errors.
main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

Index.html

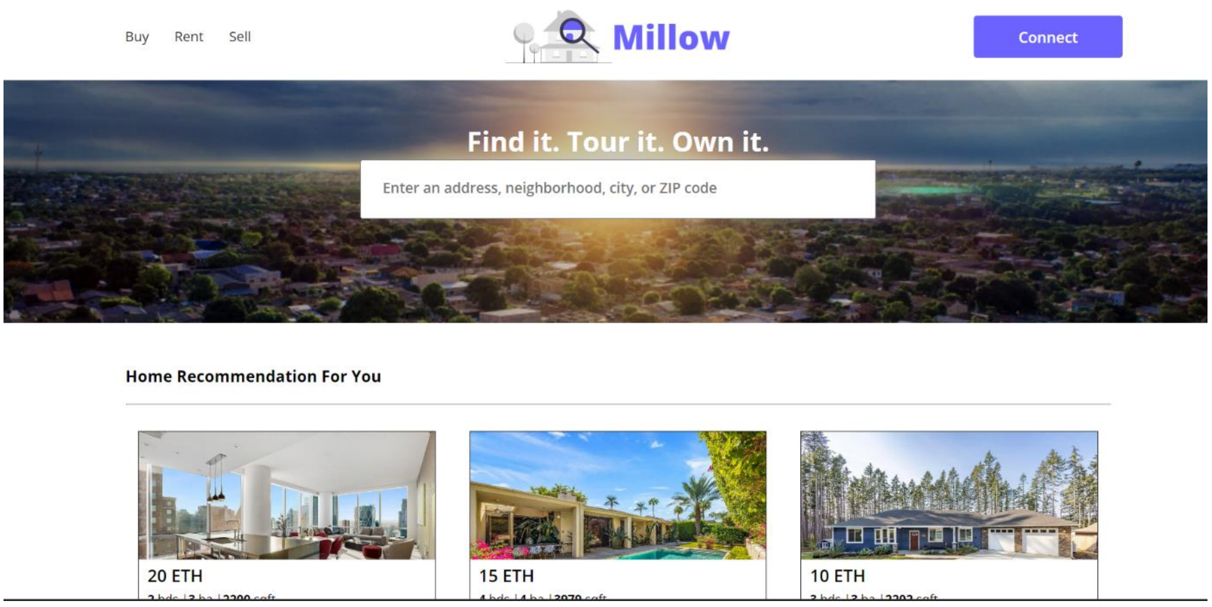


public > index.html > ...

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <meta name="description" content="Web site created using create-react-app" />
  <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
  <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
  <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
  <title>Millow</title>
</head>

<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

```

CHAPTER 10

10.2 OUTPUT



Find it. Tour it. Own it.

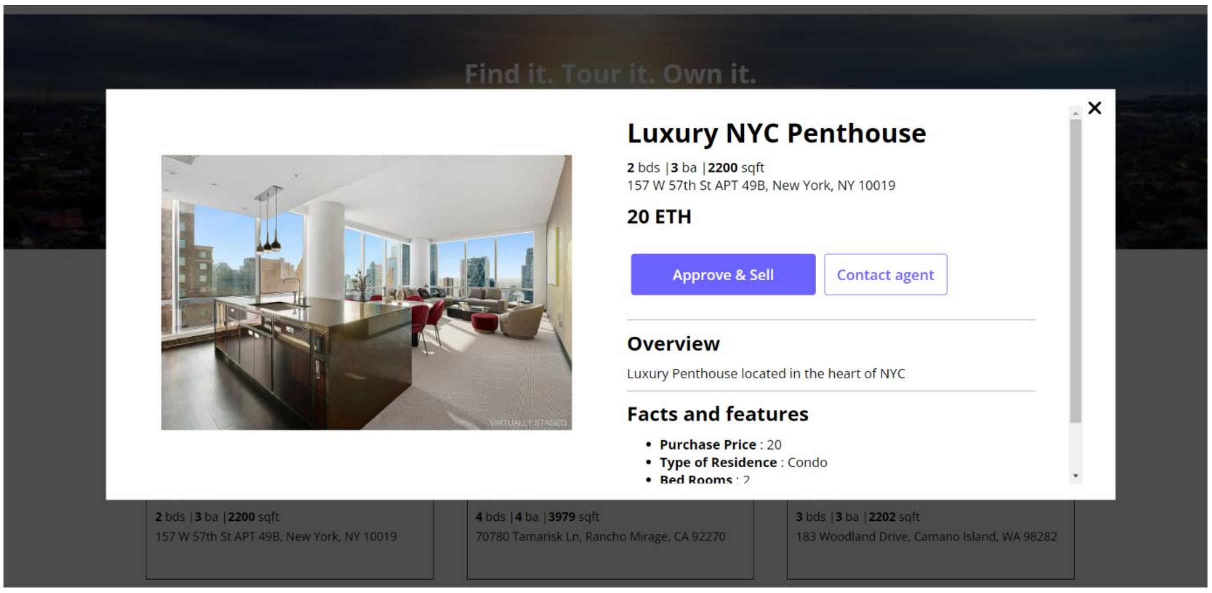
Enter an address, neighborhood, city, or ZIP code

Home Recommendation For You

20 ETH

15 ETH

10 ETH



Luxury NYC Penthouse

2 bds |**3** ba |**2200** sqft
157 W 57th St APT 49B, New York, NY 10019

20 ETH

Overview

Luxury Penthouse located in the heart of NYC

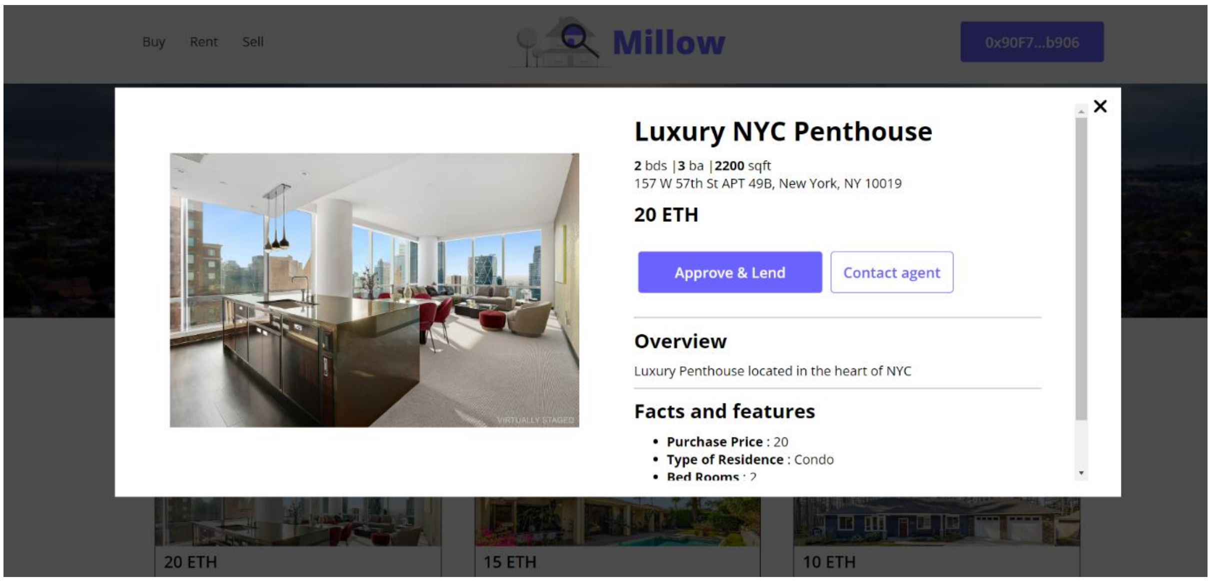
Facts and features

- **Purchase Price** : 20
- **Type of Residence** : Condo
- **Bed Rooms** : 2

2 bds |**3** ba |**2200** sqft
157 W 57th St APT 49B, New York, NY 10019

4 bds |**4** ba |**3979** sqft
70780 Tamarisk Ln, Rancho Mirage, CA 92270

3 bds |**3** ba |**2202** sqft
183 Woodland Drive, Camano Island, WA 98282



Luxury NYC Penthouse

2 bds | **3** ba | **2200** sqft
157 W 57th St APT 49B, New York, NY 10019

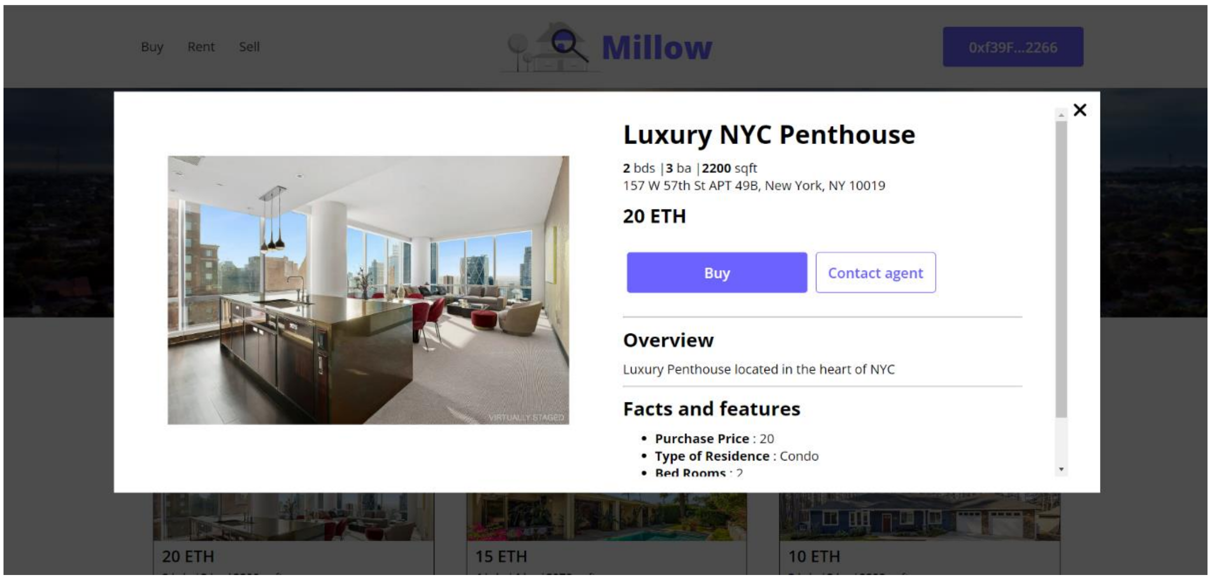
20 ETH

Overview

Luxury Penthouse located in the heart of NYC

Facts and features

- **Purchase Price** : 20
- **Type of Residence** : Condo
- **Bed Rooms** : 2



Luxury NYC Penthouse

2 bds | **3** ba | **2200** sqft
157 W 57th St APT 49B, New York, NY 10019

20 ETH

Overview

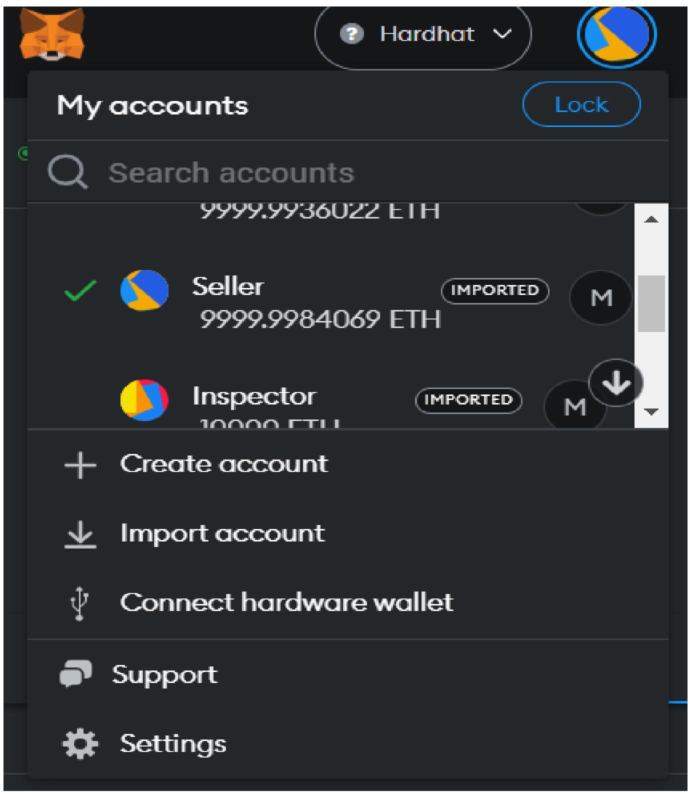
Luxury Penthouse located in the heart of NYC

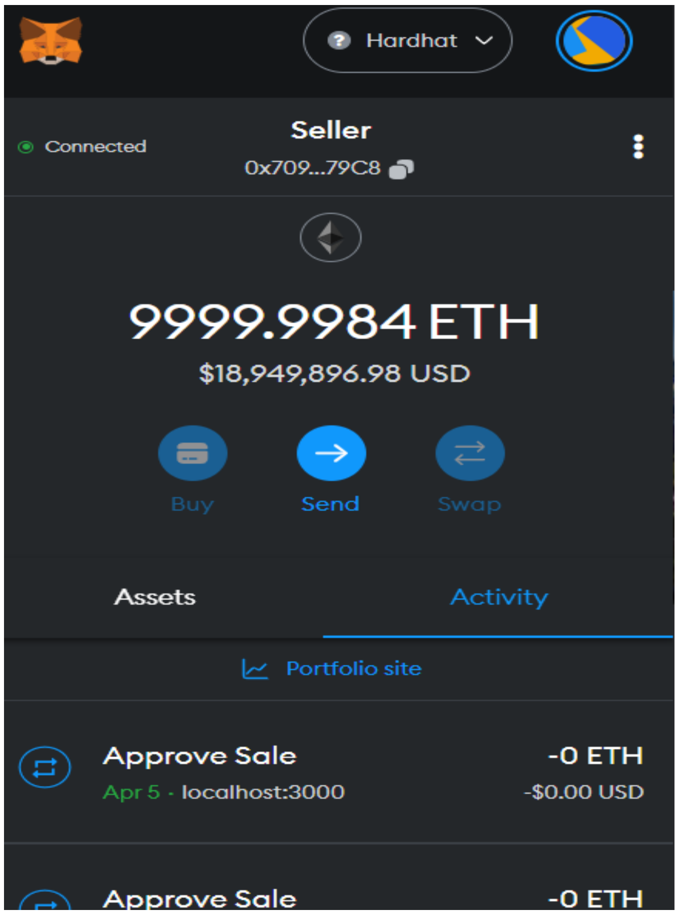
Facts and features

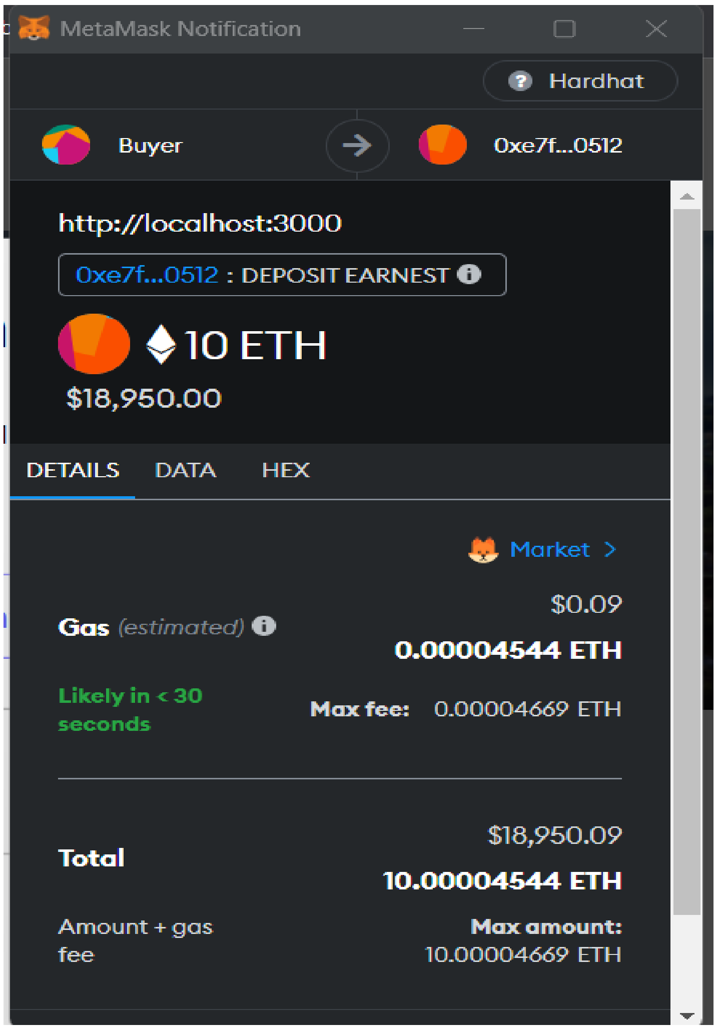
- **Purchase Price** : 20
- **Type of Residence** : Condo
- **Bed Rooms** : 2

```
Account #1: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8 (10000 ETH)
Private Key: 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d

Account #2: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC (10000 ETH)
Private Key: 0x5de4111afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a

Account #3: 0x90F79bf6EB2c4f870365E785982E1f101E93b906 (10000 ETH)
Private Key: 0x7c852118294e51e653712a81e05800f419141751be58f605c371e15141b007a6

Account #4: 0x15d34AAf54267DB7D7c367839AAf71A00a2C6A65 (10000 ETH)
Private Key: 0x47e179ec197488593b187f80a00eb0da91f1b9d0b13f8733639f19c30a34926a

Account #5: 0x9965507D1a55bcC2695C58ba16FB37d819B0A4dc (10000 ETH)
Private Key: 0x8b3a350cf5c34c9194ca85829a2df0ec3153be0318b5e2d3348e872092edffba

Account #6: 0x976EA74026E726554dB657fA54763abd0C3a0aa9 (10000 ETH)
Private Key: 0x92db14e403b83dfe3df233f83dfa3a0d7096f21ca9b0d6d6b8d88b2b4ec1564e

Account #7: 0x14dC79964da2C08b23698B3D3cc7Ca32193d9955 (10000 ETH)
Private Key: 0x4bbbf85ce3377467afe5d46f804f221813b2bb87f24d81f60f1fcdbf7cbf4356

Account #8: 0x23618e81E3f5cdF7f54C3d65f7FBc0aBf5B21E8f (10000 ETH)
Private Key: 0xdbda1821b80551c9d65939329250298aa3472ba22feea921c0cf5d620ea67b97

Account #9: 0xa0Ee7A142d267C1f36714E4a8F75612F20a79720 (10000 ETH)
Private Key: 0x2a871d0798f97d79848a013d4936a73bf4cc922c825d33c1cf7073dff6d409c6

Account #10: 0xBcd4042DE499D14e55001CcbB24a551F3b954096 (10000 ETH)
Private Key: 0xf214f2b2cd398c806f84e317254e0f0b801d0643303237d97a22a48e01628897

Account #11: 0x71bE63f3384f5fb98995898A86B02Fb2426c5788 (10000 ETH)
Private Key: 0x701b615bbdfb9de65240bc28bd21bbc0d996645a3dd57e7b12bc2bdf6f192c82

Account #12: 0xFABB0ac9d68B0B445fB7357272Ff202C5651694a (10000 ETH)
Private Key: 0xa267530f49f8280200edf313ee7af6b827f2a8bce2897751d06a843f644967b1
```

```
eth_estimateGas
eth_getBlockByNumber
eth_gasPrice
eth_sendTransaction
  Contract deployment: RealEstate
  Contract address:    0x5fbdb2315678afecb367f032d93f642f64180aa3
  Transaction:         0xe62b529fdb05781d00222c357de5ecffdb9dfa55a0a21bb4eeab10a571c7db92
  From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
  Value:               0 ETH
  Gas used:            2443701 of 2443701
  Block #1:            0x0afad3ee077ab11083c283b1b8e992d7740f7a710f0a49b3cd4a2c2a108a71dd

eth_chainId
eth_getTransactionByHash
eth_chainId
eth_getTransactionReceipt
eth_chainId
eth_getTransactionReceipt
eth_chainId
eth_estimateGas
eth_gasPrice
eth_sendTransaction
  Contract call:       RealEstate#mint
  Transaction:         0x9b116525e10de5eaa1346dec3772e063fd584c57b120800b8457228f0f72db1b
  From:                0x70997970c51812dc3a010c7d01b50e0d17dc79c8
  To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
  Value:               0 ETH
  Gas used:            182884 of 182884
  Block #2:            0x066ca34d74607447a495864a09e780400a2cefa90e07b650201bb61b32336700

eth_chainId
eth_getTransactionByHash
eth_chainId
eth_getTransactionReceipt
eth_chainId
eth_getTransactionReceipt
eth_chainId
eth_estimateGas
eth_gasPrice
```


Hardhat
My accounts
Lock
Search accounts
9999.9936022 ETH
Seller IMPORTED M
9999.9984069 ETH
Inspector IMPORTED M
10000 ETH
Create account
Import account
Connect hardware wallet
Support
Settings


Hardhat
Connected
Seller
0x709...79C8
9999.9984 ETH
$18,949,896.98 USD
Buy Send Swap
Assets Activity
Portfolio site
Approve Sale -0 ETH
Apr 5 · localhost:3000 -$0.00 USD
Approve Sale -0 ETH



MetaMask Notification

Hardhat

Buyer → 0xe7f...0512

http://localhost:3000

0xe7f...0512 : DEPOSIT EARNEST

10 ETH

$18,950.00

DETAILS DATA HEX

Market >

Gas *(estimated)* $0.09

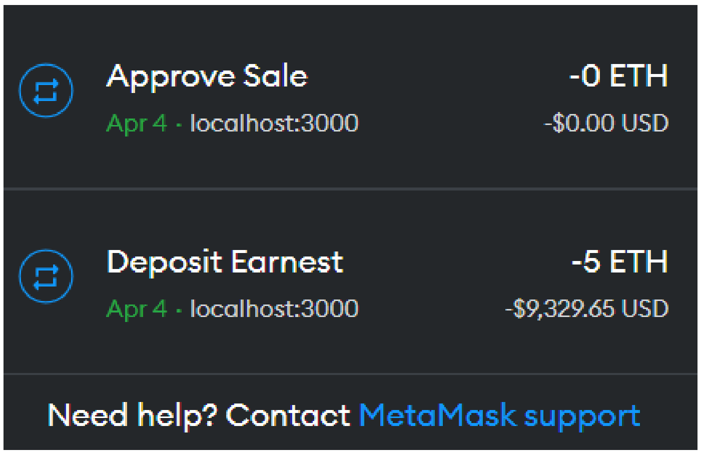
0.00004544 ETH

Likely in < 30 seconds **Max fee:** 0.00004669 ETH

Total $18,950.09

10.00004544 ETH

Amount + gas fee **Max amount:** 10.00004669 ETH



Approve Sale -0 ETH

Apr 4 · localhost:3000 -$0.00 USD

Deposit Earnest -5 ETH

Apr 4 · localhost:3000 -$9,329.65 USD

Need help? Contact MetaMask support

CHAPTER 11
BIBLIOGRAPHY

[1] Sobhan Latifi, Yunpeng Zhang, Liang-Chieh Cheng, “Blockchain-Based Real Estate Market: One Method for Applying Blockchain Technology in Commercial Real Estate Market”, IEEE International Conference on Blockchain (Blockchain), 2020

[2] Jack Laurie Tilbury, Ed de la Rey, Karl van der Schyff, “Business Process Models of Blockchain and South African Real Estate Transactions”, International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD), 2019

[3] Disha Shinde, Snehal Padekar, Siddharth Raut, Abdul Wasay, S. S. Sambhare, “Land Registry Using Blockchain - A Survey of existing systems and proposing a feasible solution”, 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA), 2020

[4] Toqeer Ali, Adnan Nadeem, Ali Alzahrani, Salman Jan, “A Transparent and Trusted Property Registration System on Permissioned Blockchain”, International Conference on Advances in the Emerging Computing Technologies (AECT), 2020

[5] Ankit Mittal, Bhavyansh Sharma, Pinku Ranjan, “Real Estate Management System based on Blockchain”, IEEE 7th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), 2021

[6] VO Khoa Tan, Thu Nguyen, “The Real Estate Transaction Trace System Model Based on Ethereum Blockchain Platform”, 14th International Conference

on Computer and Automation Engineering (ICCAE), 2022

[7] Andrey Averin, Pavel Rukhlov, Elnur Musaev, “Review of Existing Solutions in the Field of Real Estate and Cadastral Accounting Based on Blockchain Technology”, International Conference on Quality Management, Transport and Information Security, Information Technologies (IT&QM&IS), 2021

[8] Ishan Yapa, Samitha Heanthenna, Nadun Bandara, Isuru Prasad, Yashas Mallawarachchi, “Decentralized Ledger for Land and Property Transactions in Sri Lanka Acresense”, IEEE Region 10 Humanitarian Technology Conference (R10-HTC), 2019

[9] P. Shamili, B. Muruganantham, “Blockchain based Application: Decentralized Financial Technologies for Exchanging Crypto Currency”, International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), 2022

[10] Dipika Bhanushali, Akshara Koul, Sainiranjan Sharma, Bushra Shaikh, “BlockChain to Prevent Fraudulent Activities: Buying and Selling Property Using BlockChain”, International Conference on Inventive Computation Technologies (ICICT), 2020