BLOCKCHAIN-BASED SMART REAL ESTATE MANAGEMENT

NAAN MUDHALVAN

(2023 – 2024)

PROJECT REPORT

Submitted By

TEAM ID : NM2023TMID09171

ALAGU SATHISH M (950520106001)

BALAMURUGAN G (950520106003)

JEGADHEES M (950520106007)

MAHARAJAN M (950520106010)

LINGARAJ L(950520106031)

In Partial Fulfilment for the Award of the Degree

Of

BACHELOR OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION ENGINEERING

Dr . SIVANTHI ADITANAR COLLEGE OF ENGINEERING,

TIRUCHENDUR - 628 215.

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW:

In the era of smart cities and smart houses, the integration of blockchain technology offers a promising solution to enhance security and efficiency in the real estate industry. This project aims to design and implement a comprehensive system that leverages the Ethereum blockchain for managing real estate-related contracts. The primary objectives are to enable the secure addition of property details to the blockchain, provide the capability to query property information from the blockchain, and facilitate seamless and appropriate ownership changes for real properties..

This project aims to create a comprehensive system that harnesses the capabilities of the Ethereum blockchain to manage real estate-related contracts. The primary objectives include the development of smart contracts that enable the secure addition of property details to the blockchain, the creation of a user-friendly interface for querying property information from the blockchain, and the implementation of mechanisms for changing property ownership securely and appropriately.

1.2 PURPOSE:

The primary purpose of the project is to utilize blockchain technology, specifically Ethereum, to enhance security and efficiency in real estate-related transactions..

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM:

The existing problems within the real estate industry are multifaceted and have long been a concern for both buyers and sellers. First and foremost, there is a significant lack of transparency in traditional real estate transactions. This lack of transparency makes it challenging for individuals to obtain a complete and accurate history of a property, leading to misunderstandings and disputes during transactions. Moreover, the real estate process is laden with lengthy and complex paperwork, involving contracts, deeds, and various legal documentation. This manual and paper-based approach is not only time-consuming but can also introduce errors into the process, causing unnecessary delays and complications.

In addition, traditional real estate transactions often rely on multiple intermediaries, such as real estate agents and escrow services, resulting in significant fees that can be a financial barrier for many buyers. The data in real estate is often scattered across various sources, leading to inconsistencies and inaccuracies in property listings, which can be frustrating for both buyers and sellers seeking reliable information. Furthermore, the centralized nature of traditional real estate databases exposes them to security breaches and fraudulent activities, potentially compromising the integrity of property records.

2.2 REFERENCES:

1. Warburton, D. The Role of Technology in the Real Estate Industry. Ph.D. Thesis, University of Cape Town Cape Town, South Africa, 2016.
2. Zhang, Z.; Qiang, M.; Jiang, H. Finding Academic Concerns on Real Estate of US and China: A Topic Modeling Based Exploration. In Proceedings of the 21st International Symposium on Advancement of Construction Management and Real Estate; Springer: Singapore, 2018; pp. 807–817.
3. Winson-Geideman, K.; Krause, A. Transformations in Real Estate Research: The Big Data Revolution In Proceedings of the 22nd Annual Pacific-Rim Real Estate Society Conference, Queensland, Australia, 17–20 January 2016; pp. 17–20
4. Allameh, E.; Jozam, M.H.; de Vries, B.; Timmermans, H.; Beetz, J. Smart Home as a smart real estate: A state of the art review. In Proceedings of the 18th Annual European Real Estate Society Conference, Eindhoven, The Netherlands, 16–18 June 2011

2.3 PROBLEM STATEMENT DEFINITION:

The real estate industry faces a myriad of challenges rooted in inefficiencies, lack of transparency, and high transaction costs. Traditional real estate transactions often lack transparency, leading to misunderstandings and disputes. Lengthy and complex paperwork introduces delays and potential errors. The involvement of intermediaries results in significant fees, making

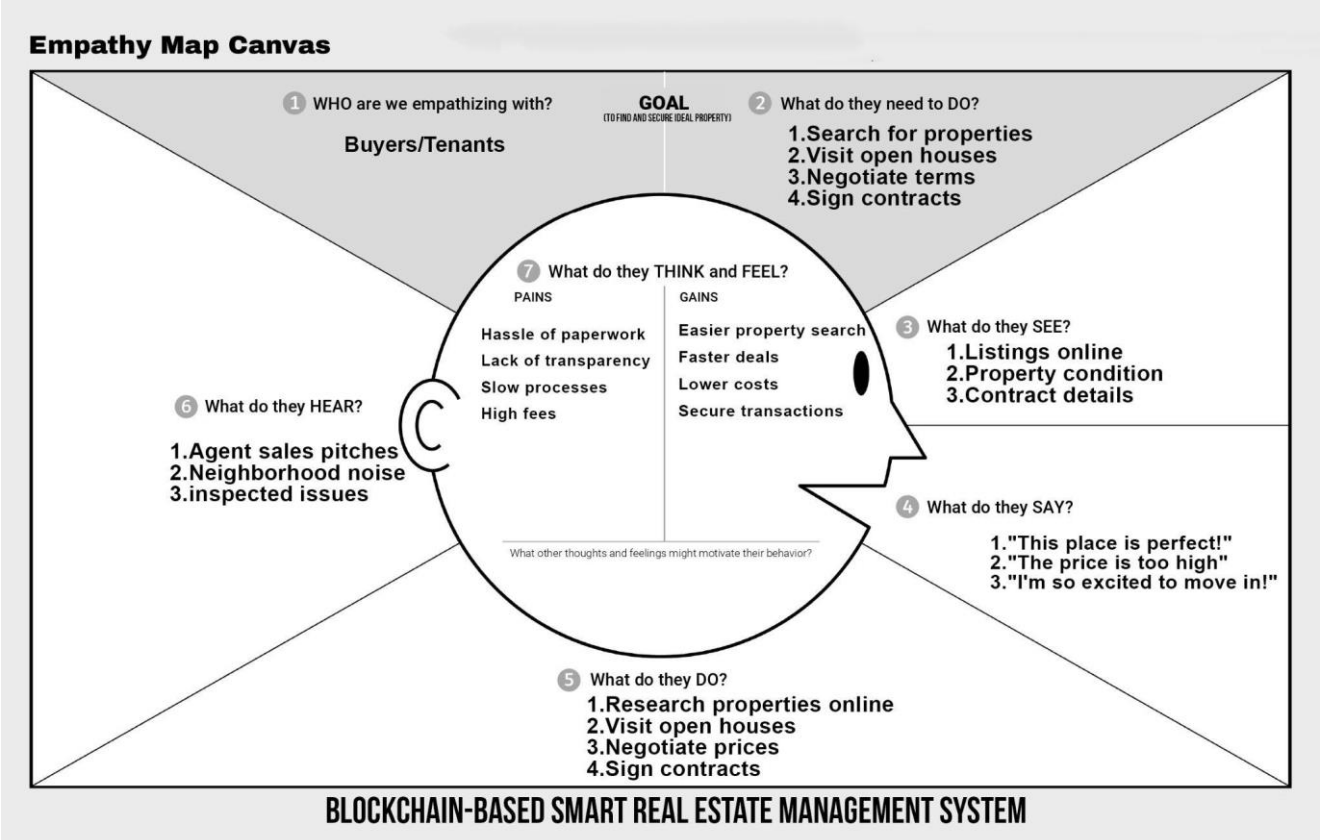
real estate transactions costly. Inaccuracies and inconsistencies in property data can cause frustration and hinder trust.

Security vulnerabilities and fraudulent activities pose a threat to property records, and property ownership verification is often cumbersome. Furthermore, accessibility to real estate opportunities is constrained for some individuals, creating disparities. High transaction costs, including legal fees and taxes, can be prohibitive.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING

- **QR Code Labeling:**
  Implement QR code labeling on real estate properties to enable quick access to property details, transaction history, ownership records, and legal documents. Users, including buyers, sellers, and real estate agents, can scan these QR codes to gain comprehensive insights into the property, fostering transparency and streamlining the decision-making process.
- **IoT Sensors:**

  Deploy IoT sensors to monitor and record real-time property data, including temperature, humidity, security breaches, and property access. These sensors will continuously collect and transmit data to the blockchain, ensuring the property's integrity, security, and safety.

- **Mobile Apps for Consumers:**
  Create user-friendly mobile apps tailored for real estate stakeholders, allowing them to scan property QR codes, access detailed property information, track ownership changes, and receive real-time notifications regarding property transactions and ownership updates.
- **Supplier Verification:**

  Implement a system for supplier verification and certification. Suppliers must meet certain standards for their products to enter the supply chain.

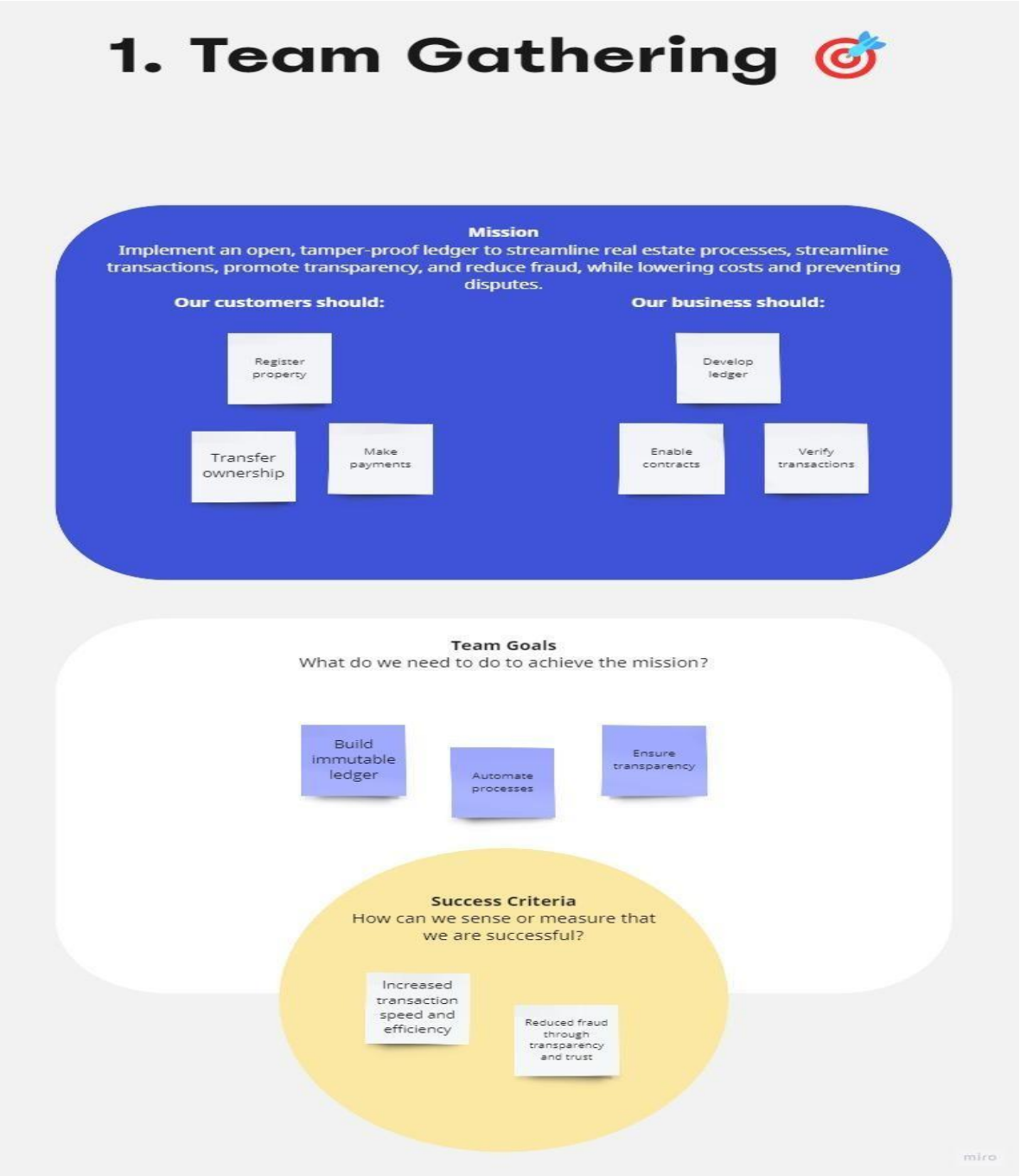
- **Property Ownership Verification:**

Establish a comprehensive property ownership verification system that requires rigorous authentication and validation for property transfers. To be included in the blockchain-based ownership ledger, all parties involved in property transactions, including buyers, sellers, and agents, must meet predefined criteria and adhere to legal standards.

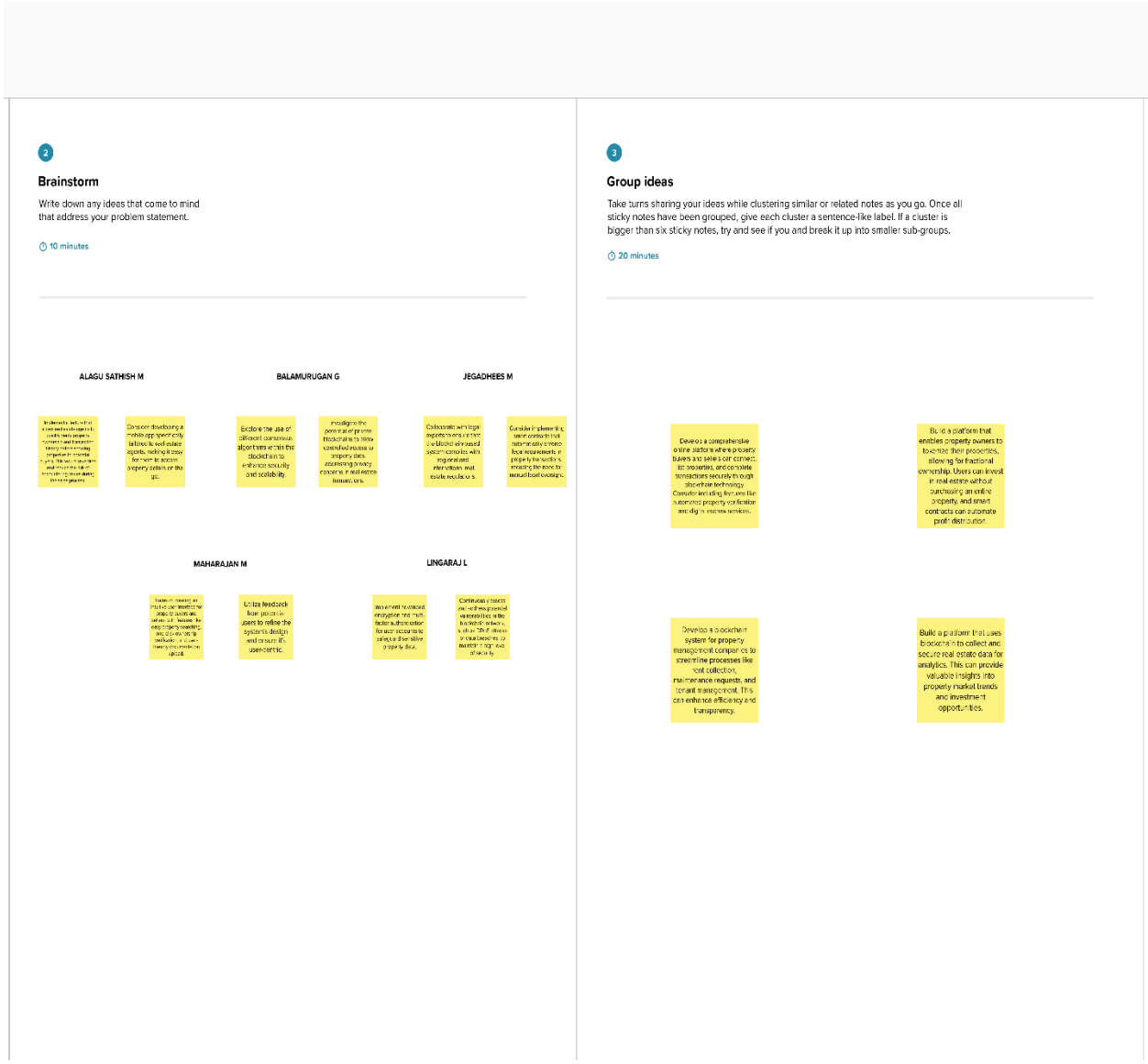
- **AI-Based Quality Control:**

  Incorporate artificial intelligence (AI) into the real estate ecosystem to perform property inspections, utilizing image recognition and data analysis.

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step -2: Brainstorm , Idea Listing and Grouping



STEP 3: Idea Prioritization

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS:

FR NO	FUNCTIONAL REQUIREMENT	SUB REQUIREMENT
FR – 1	User Registration and Authentication	Implement user registration, allowing users to create accounts with unique usernames and secure password authentication. Provide the option for enhanced security through two-factor authentication methods, such as SMS codes or biometric verification.
FR – 2	User Profile Management	Develop user profile management features, enabling users to edit and update their personal information. This should include parameters like height, weight, dietary preferences, and fitness goals, ensuring that the system can provide personalized recommendations.

FR – 3	Property Details Input:	Enable users to input property details, including specifications, ownership records, and transaction history, allowing for accurate and comprehensive property information.
FR – 4	Property Query and Verification**:**	Create the capability for users to query and verify property details from the blockchain, ensuring transparency and trust in property transactions.
FR – 5	Ownership Transfer Mechanism:	Implement a secure mechanism for users to initiate and execute property ownership changes, adhering to legal standards and providing a seamless process for property transfers.
FR – 6	Security and Privacy Measures:	Establish robust security and privacy protocols to safeguard user data and protect sensitive property information, ensuring that only authorized users can access and modify data.
FR - 7	Decentralized Network Integration:	Ensure the system operates in a decentralized network to boost resilience.

FR - 8	User Interface Development:	Create a user-friendly interface for users to interact with the blockchain, allowing for easy access to property information and ownership management. This interface should be intuitive and accessible on various devices.
FR - 9	Notifications and Alerts:	Implement a notification system that notifies users about significant events related to property transactions, such as ownership changes or transaction history updates.
FR -10	Property Data Consistency:	Develop data consistency checks to ensure the accuracy and reliability of property information stored on the blockchain, reducing the risk of inaccuracies and disputes.
FR - 11	Accessibility and Inclusivity:	Ensure that the system is accessible to a diverse range of users, including those with various devices and connectivity options, making property information available to a wider audience.

4.2 NON-FUNCTIONAL REQUIREMENTS:

NFR NO	NON-FUNCTIONAL REQUIREMENT	DESCRIPTION
NFR – 1	Scalability	The system should be designed to easily scale to accommodate a growing user base and increasing data demands within the real estate ecosystem.
NFR – 2	Reliability	Ensure the system's availability and reliability, minimizing downtime or service interruptions, making it a dependable platform for property transactions.
NFR – 3	Security	Safeguard data by storing it securely with robust encryption measures to protect sensitive property and user information from unauthorized access or manipulation.
NFR – 4	Data Backup and Recovery	Implement regular data backup processes to prevent data loss in case of system failures, ensuring the integrity and continuity of property records.

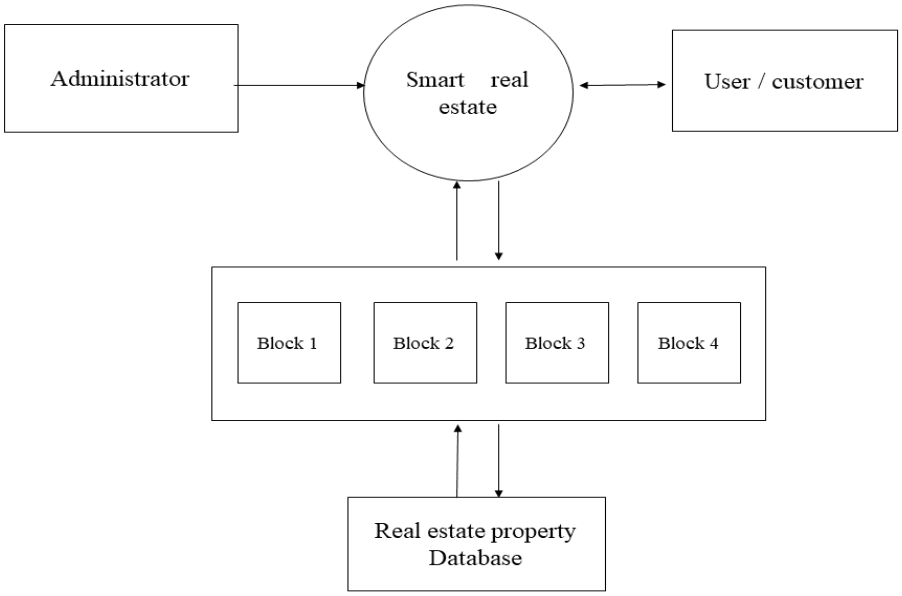
NFR – 5	Usability	Create a user-friendly system with an intuitive and easy-to-navigate interface, enhancing user satisfaction and adoption among real estate stakeholders.
NFR – 6	Accessibility	Ensure compliance with accessibility standards, such as WCAG, to make the system usable by individuals with disabilities, promoting inclusivity in the real estate ecosystem.
NFR – 7	Response Time	Define acceptable response times for various system operations, such as loading property details or processing ownership changes, ensuring optimal user experience and efficiency in real estate transactions.

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS & USER STORIES

DATA FLOW DIAGRAM:



BLOCKCHAIN –BASED SMART REAL ESTATE MANAGEMENT

USER STORIES:

User Story 1: As a Real Estate Buyer, I want to scan a QR code on a property to access its complete ownership history and transaction records, so that I can make informed and secure property purchase decisions.

Requirements:

- The system should generate a QR code for each real estate property listing.

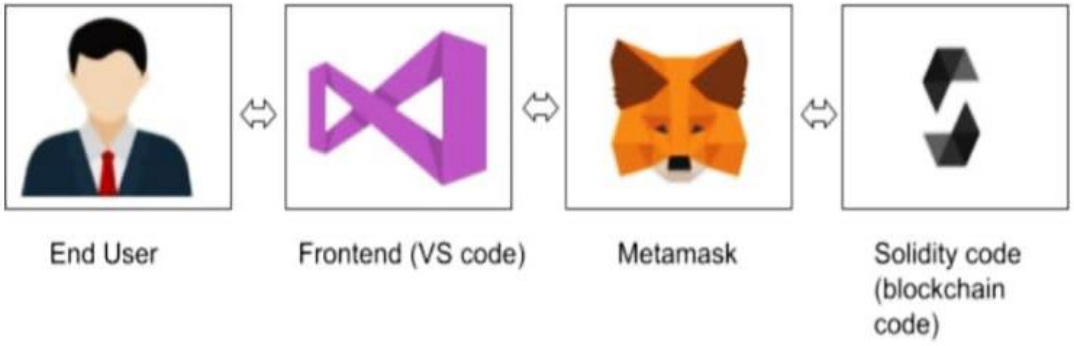
- When a user scans the property's QR code using the mobile app, it should display detailed information about the property, including ownership history, transaction records, property specifications, and any associated legal documentation.
- The information presented should be well-structured and easily comprehensible, ensuring that users can make well-informed property-related decisions.

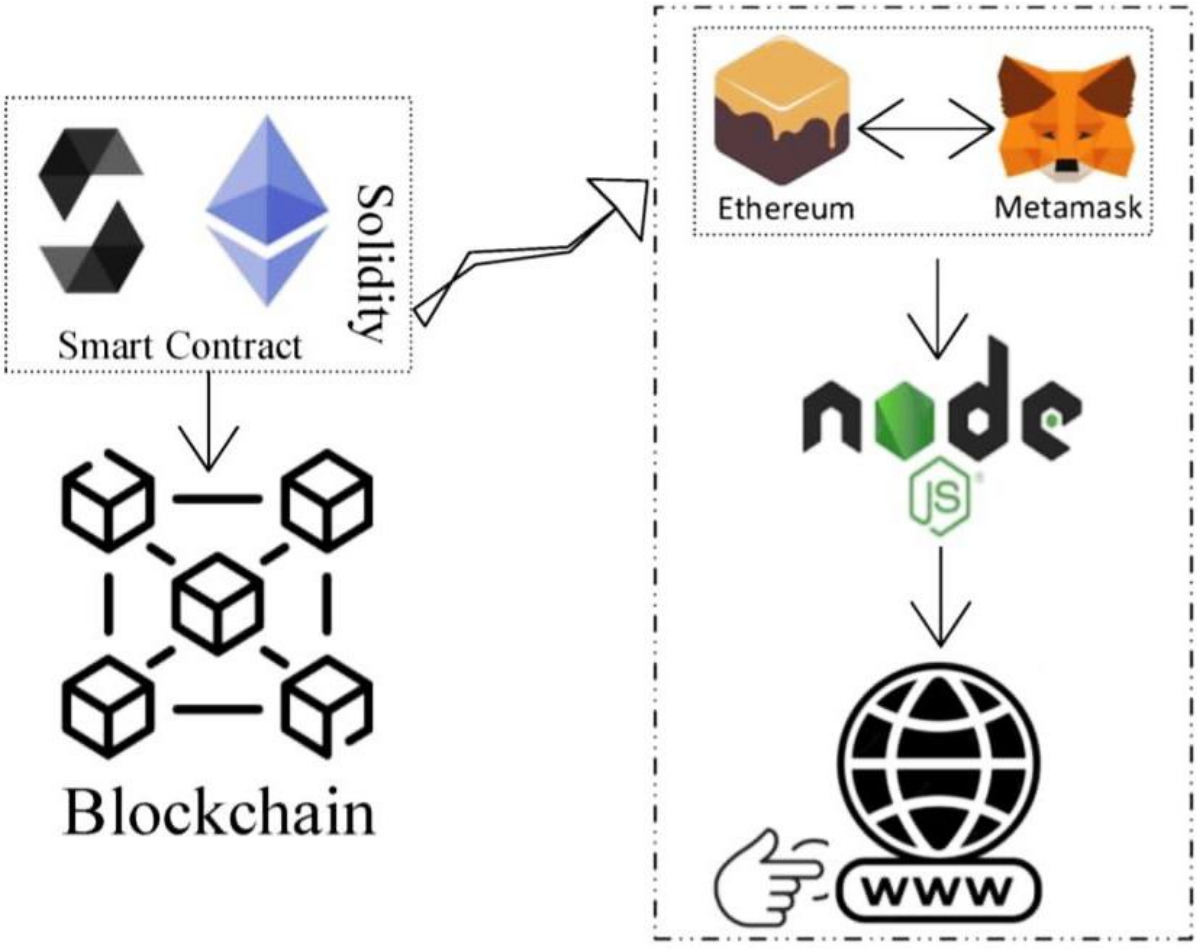
User Story 2: As a Real Estate Seller, I want to record and track the entire history of my property's ownership and transaction details on the blockchain, ensuring transparency and building trust among potential buyers throughout the property sale process.

Requirements:

- The system should provide a user-friendly interface for real estate sellers to input data about the property's ownership history, including transaction details, specifications, and any relevant legal documentation.
- The data should encompass information such as property transfer dates, previous owners, legal titles, property dimensions, and transaction values.
- Once entered, this data should be securely recorded on the blockchain, ensuring immutability and transparency, making the property's complete history accessible to relevant parties.
- Sellers and potential buyers should be able to access this information at any time to verify the property's authenticity, ownership history, and transaction records, enhancing trust and transparency in real estate transactions.

5.2 SOLUTION ARCHITECTURE



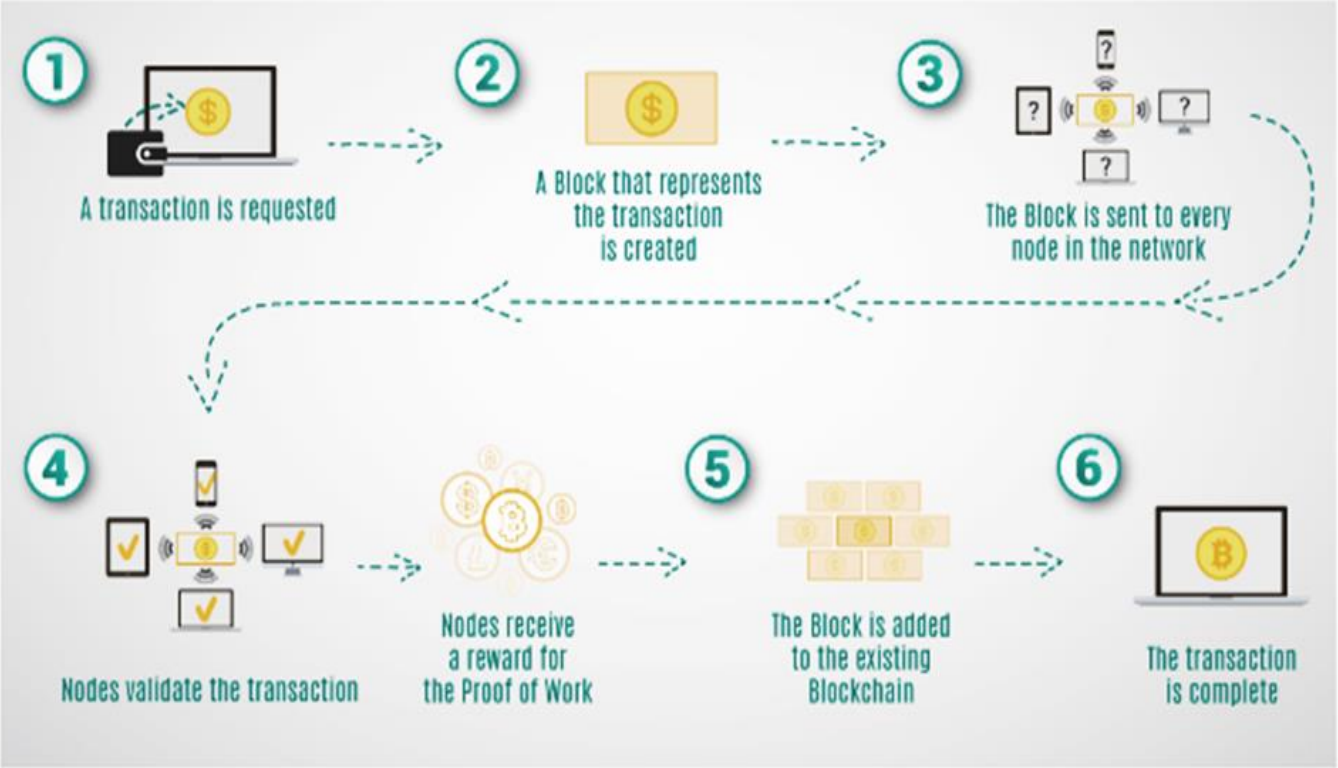


Interaction between the Web and the Contract

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 TECHNICAL ARCHITECTURE



6.2 SPRINT PLANNING & ESTIMATION

Sprint Planning:

Sprint planning in the real estate project involves selecting tasks and objectives from the project backlog and committing to completing them within the upcoming sprint cycle. This process ensures that the team's goals and priorities are aligned, and progress is made on key project milestones during each sprint.

Reviewing Product Backlog: Our real estate project team, comprising the Product Owner and development team, conducts regular reviews of the items in the project backlog. During these reviews, we assess user stories and technical tasks, considering the evolving needs and priorities of the project.

Setting Sprint Goals: Drawing from the real estate project backlog, our team defines explicit sprint goals that serve as guiding objectives for the upcoming sprint cycle. These goals provide a clear direction for the team's efforts, ensuring that their work aligns with the broader project objectives and contributes to achieving project milestones.

Breaking Down User Stories: In our real estate project, user stories and associated tasks are systematically broken down into smaller, actionable sub-tasks. This detailed decomposition process is essential for creating a comprehensive sprint plan, allowing the team to manage and execute individual components effectively to achieve the overarching sprint objectives.

Estimating Work: Our real estate development team utilizes agile estimation techniques, including methods like story points, to estimate the effort needed for each task within the sprint. These estimates are pivotal in helping the team comprehend the scope and complexity of the work, enabling better planning, allocation of resources, and managing sprint expectations.

Sprint Backlog: In our real estate project, the chosen user stories and tasks, along with their corresponding estimates, collectively form the sprint backlog. This backlog serves as the foundation for defining the specific work items that the team will focus on and complete during the sprint, providing a clear and actionable plan for the sprint cycle.

Estimation Techniques:

Story Points: In our real estate project, story points are employed as a relative measure to gauge the complexity and effort required for each task. Tasks are assigned story point values by comparing their complexity to reference tasks, helping the team to assess and plan work accurately during sprint activities.

Adjustment for Uncertainty: Acknowledging the inherent uncertainty in complex real estate projects, we incorporated adjustments into our estimates to accommodate potential deviations.

6.3 SPRINT DELIVERY SCHEDULE

Sprint Delivery Schedule for Smart Real Estate System Project

Sprint Delivery Schedule and its Objectives		
Sprint 1	Project Initiation	Establish the project's scope and define team roles in the real estate project.
Sprint 2	Data Modeling	Develop data models and prioritize user stories for property information management.
Sprint 3	Blockchain Integration	Commence the integration of blockchain technology for secure property data storage and management.
Sprint 4	IoT Sensor Integration	Incorporate IoT sensors to enable real-time data collection for property condition monitoring.

Sprint 5	User Interfaces and Mobile Apps	Create user interfaces and mobile apps to facilitate user interaction with the blockchain-based system.
Sprint 6	Quality Control and Testing	Implement quality control processes and initiate testing to ensure system reliability and accuracy.
Sprint 7	Compliance and Security	Ensure regulatory compliance and enhance security measures to protect property and user data.
Sprint 8	Deployment Preparation	Prepare the system for deployment, addressing any technical and operational requirements.
Sprint 9	Pilot Testing and Optimization	Conduct pilot tests to validate system functionality and optimize performance based on feedback.
Sprint 10	Full System Deployment	Launch the fully operational system to a broader audience, providing a comprehensive and secure platform for real estate transactions.

CHAPTER 7

CODING & SOLUTIONING

Smart Contract (Solidity)

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

contract PropertyDetail{
    address public owner;
```

```
struct Property {
    string propertyId;
    string name;
    string location;
    string discription;
    address currentOwner;
}

mapping(string => Property) public properties;
mapping(address => mapping(string => bool)) public hasAccess;

event PropertyAdded(
    string indexed propertyId,
    string name,
    string location,
    address indexed owner
);
event PropertyTransferred(
    string indexed propertyId,
    address indexed from,
    address indexed to
);

constructor() {
    owner = msg.sender;
```

```
}

modifier onlyOwner() {
    require(msg.sender == owner, "Only contract owner can call this");
    _;
}

modifier hasPropertyAccess(string memory propertyId) {
    require(
        hasAccess[msg.sender][propertyId],
        "You don't have access to this property"
    );
    _;
}

function addProperty(
    string memory propertyId,
    string memory name,
    string memory location,
    string memory _description
) external onlyOwner {
    require(
        bytes(properties[propertyId].propertyId).length == 0,
        "Property already exists"
    );
```

```
    properties[propertyId] = Property({
        propertyId: propertyId,
        name: name,
        location: location,
        discription : _description,
        currentOwner: owner
    });

    hasAccess[owner][propertyId] = true;

    emit PropertyAdded(propertyId, name, location, owner);
}

function transferProperty(
    string memory propertyId,
    address newOwner
) external hasPropertyAccess(propertyId) {
    require(newOwner != address(0), "Invalid new owner");

    address currentOwner = properties[propertyId].currentOwner;
    properties[propertyId].currentOwner = newOwner;

    hasAccess[currentOwner][propertyId] = false;
    hasAccess[newOwner][propertyId] = true;

    emit PropertyTransferred(propertyId, currentOwner, newOwner);
```

```
    }

    function getPropertyDetails(
        string memory propertyId
    ) external view returns (string memory, string memory, address) {
        Property memory prop = properties[propertyId];
        return (prop.name, prop.location, prop.currentOwner);
    }
}
```

The key features related to Smart real estate system in this contract are:

7.1 FEATURE 1

- **Verification of Property Details:**

The system facilitates the verification of property details through a specific contract function. When property information is added to the blockchain using the 'sendPropertyDetails' function, it initially exists in an "Unverified" state. This status changes to "Verified" once the property details are confirmed and validated through the 'verifyPropertyDetails' function. This feature is essential for assuring the accuracy and authenticity of property data within the blockchain, allowing for a transparent and accountable record of each property's status as it progresses through real estate transactions.

7.2 FEATURE 2

- **Property Transfer:**

The contract provides a mechanism for property transfers. The 'transferProperty' function allows a property to transition from its current

ownership to a new owner. This transition signifies the point at which ownership of the property changes hands. This feature ensures that there is a transparent and immutable record of property ownership transitions, helping to maintain a comprehensive and secure history of property transfers within the real estate system.

7.3 DATABASE SCHEMA

In the Ethereum-based smart contract provided, the data is organized and structured within the Ethereum blockchain, which utilizes a distributed ledger as its foundational data storage mechanism. The schema within this smart contract is primarily defined by the structure of the 'FoodItem' struct and how data is stored in the 'foodItems' mapping. Unlike traditional relational database schemas, the data within this blockchain-based smart contract is decentralized and immutable.

The contract itself functions as the data store, eliminating the necessity for a conventional centralized database schema. This decentralized and immutable data structure ensures that information related to food items and their verification status remains secure, transparent, and tamper-resistant within the Ethereum blockchain.

CHAPTER 8

PERFORMANCE TESTING

8.1 PERFORMANCE METRICS

Transaction Throughput: Measure the blockchain's capability to process property transactions per second. A higher throughput is vital for handling a substantial volume of real-time property data.

Data Accuracy: Evaluate the precision of property information recorded on the blockchain. Ensure that the system minimizes errors and discrepancies in real estate data.

Data Integrity: Assess the immutability of data stored on the blockchain. Verify that once property data is recorded, it remains unaltered and undeletable, preserving data integrity.

Security: Monitor the system's ability to secure sensitive property data and prevent unauthorized access or data manipulation.

Response Time: Evaluate the time it takes for the system to respond to inquiries or requests, such as querying property details or tracking ownership changes.

Supply Chain Efficiency: Assess the system's impact on the overall efficiency of real estate transactions, including streamlined processes, reduced costs, and minimized delays.

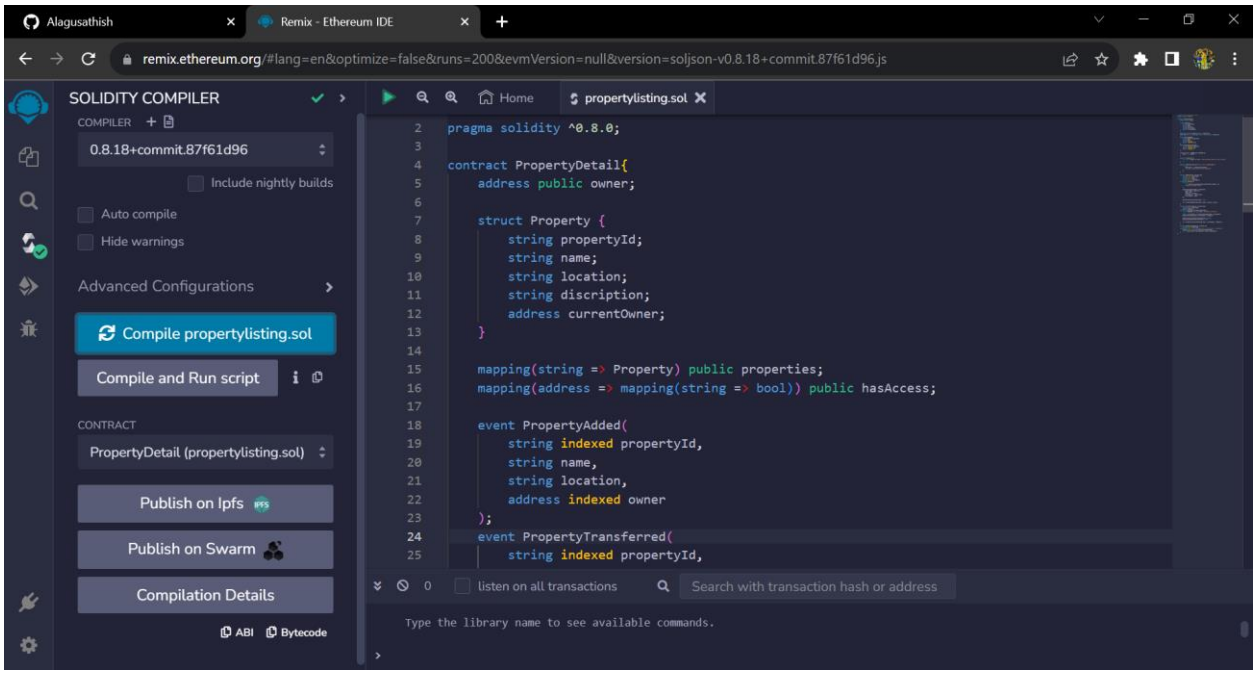
Feedback and Improvement: Collect feedback from users and stakeholders to iteratively enhance the system, including the user interface and features, ensuring continuous improvement.

Cost Efficiency: Evaluate the system's cost-effectiveness, considering expenses related to blockchain technology, IoT sensors, and maintenance in relation to the benefits it offers in enhancing real estate transactions.

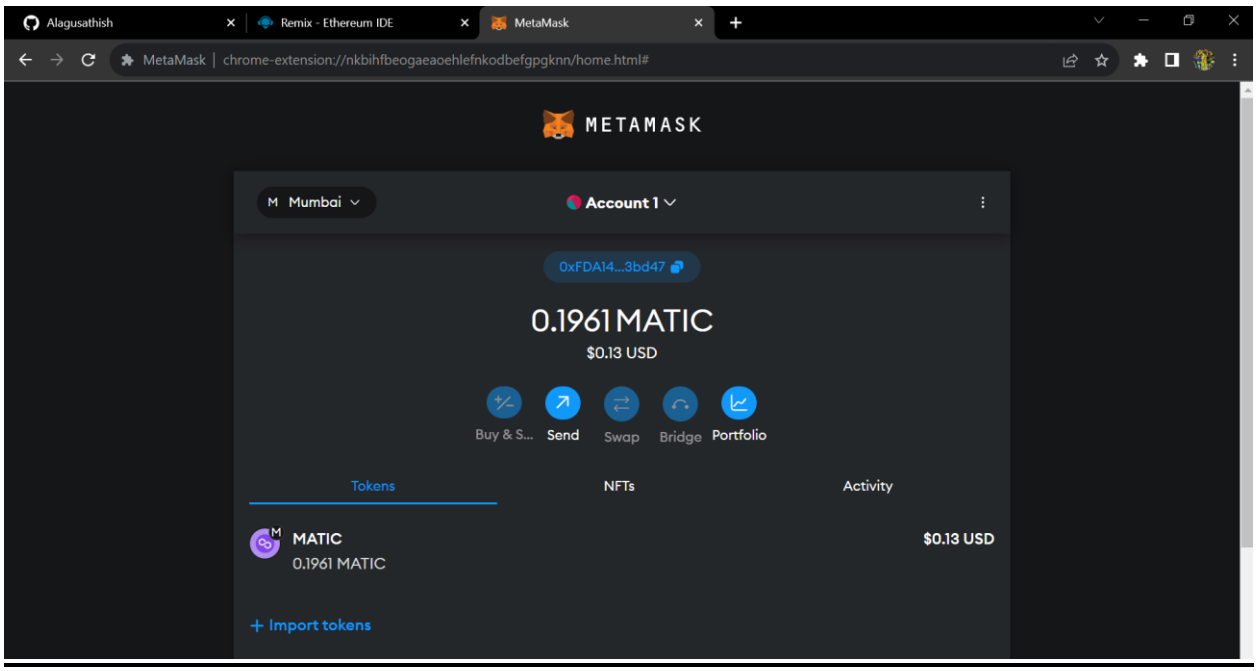
CHAPTER 9

RESULTS

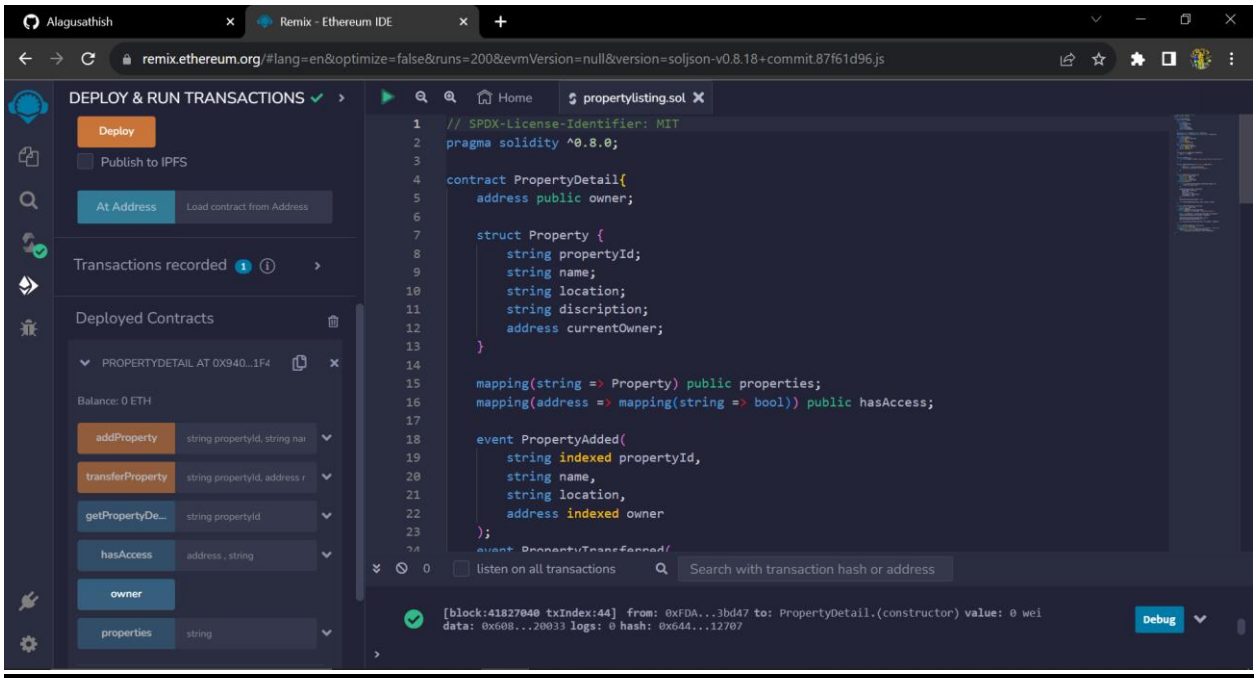
9.1 OUTPUT SCREENSHOTS



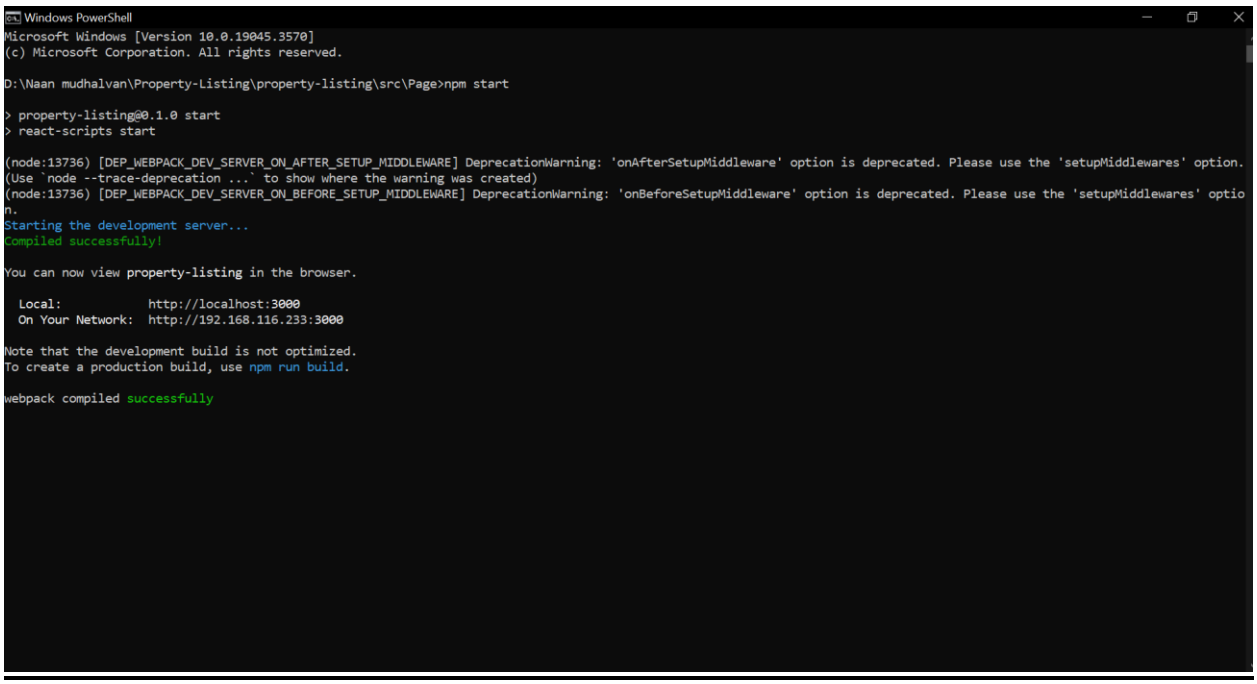
REMIX IDE – CREATING A SMART CONTRACT



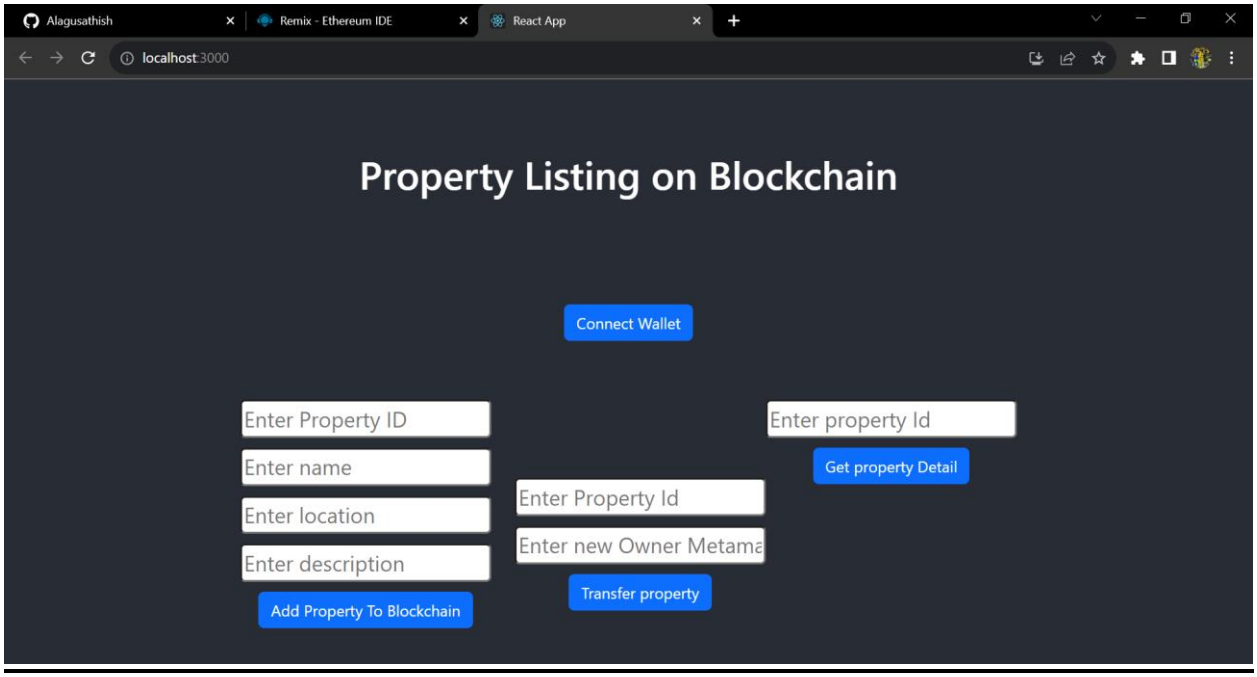
METAMASK ACCOUNT



REMIX IDE – DEPLOYED CONTRACT



TERMINAL OUTPUT SCREEN



FINAL FRONTEND OUTPUT SCREEN

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- Enables individuals to transparently monitor property transaction details, fostering trust and informed decision-making in real estate.
- Facilitates efficient and secure property ownership transfers, reducing complexities and transaction costs for buyers and sellers.
- Streamlines property information management, reducing errors and inconsistencies in property listings.
- Enhances accessibility to real estate opportunities by providing a decentralized platform, potentially increasing market participation.
- Offers a cost-effective and resilient solution by leveraging blockchain technology and IoT sensors for property data management.

DISADVANTAGES:

- Managing property transactions through the system may require a learning curve and adaptation for traditional real estate participants.
- Reliance on technology and blockchain infrastructure may introduce vulnerabilities to system interruptions or cyberattacks.
- The effectiveness of the system may be influenced by the availability and adoption of blockchain technology and IoT infrastructure in the real estate industry.

- Legal and regulatory challenges may arise as real estate transactions and ownership verification processes adapt to a blockchain-based ecosystem.
- Data privacy and security concerns must be addressed to safeguard sensitive property and user information.

CHAPTER 11

CONCLUSION

CONCLUSION:

In conclusion, the implementation of blockchain technology in the real estate industry offers a multitude of advantages for individuals, real estate businesses, and the overall real estate ecosystem. It empowers users to make well-informed decisions about property transactions, enhancing transparency, reducing transaction complexities, and lowering costs. These systems also provide an accessible, cost-effective, and secure solution, potentially increasing market participation and revolutionizing the real estate industry.

However, it's essential to acknowledge the potential disadvantages and challenges associated with blockchain-based real estate systems. These include adaptation difficulties, technical vulnerabilities, legal and regulatory adjustments, and data privacy concerns. Users and stakeholders should approach the adoption of blockchain technology in real estate with caution and work collaboratively to address these challenges.

Blockchain-based real estate systems are most effective when integrated into a comprehensive approach to real estate transactions. When combined with traditional real estate practices, legal compliance, and cybersecurity measures, these systems can contribute to a more efficient, transparent, and secure real estate industry.

CHAPTER 12

FUTURE SCOPE

FUTURE SCOPE:

- Future food tracking systems will increasingly leverage AI and machine learning algorithms to provide highly personalized recommendations and insights based on an individual's health goals, dietary preferences, and restrictions, further enhancing the user experience.
- Integration with wearable technology, such as smartwatches and fitness trackers, will enable real-time and seamless monitoring of food intake, making tracking more convenient, accurate, and integrated into users' daily lives.
- Augmented reality (AR) applications and visual recognition technology may revolutionize food tracking by allowing users to capture and identify food items using their smartphones, simplifying and enhancing the tracking process.
- Blockchain technology's potential extends to tracking the journey of food products from farm to table, ensuring transparency and traceability in the food supply chain. This could substantially improve food safety, quality control, and the management of food recalls.
- The integration of food tracking systems with electronic health records and telehealth platforms will empower healthcare providers to monitor and support patients more effectively, enabling a more holistic approach to health management and dietary guidance.

CHAPTER 13

APPENDIX

SOURCE CODE

Propertylisting.sol

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

contract PropertyDetail{
    address public owner;

    struct Property {
        string propertyId;
        string name;
        string location;
        string discription;
        address currentOwner;
    }

    mapping(string => Property) public properties;
    mapping(address => mapping(string => bool)) public hasAccess;
```

```
event PropertyAdded(
    string indexed propertyId,
    string name,
    string location,
    address indexed owner
);
event PropertyTransferred(
    string indexed propertyId,
    address indexed from,
    address indexed to
);

constructor() {
    owner = msg.sender;
}

modifier onlyOwner() {
    require(msg.sender == owner, "Only contract owner can call this");
    _;
}

modifier hasPropertyAccess(string memory propertyId) {
    require(
        hasAccess[msg.sender][propertyId],
        "You don't have access to this property"
    );
```

```
    _;
}

function addProperty(
    string memory propertyId,
    string memory name,
    string memory location,
    string memory _description
) external onlyOwner {
    require(
        bytes(properties[propertyId].propertyId).length == 0,
        "Property already exists"
    );

    properties[propertyId] = Property({
        propertyId: propertyId,
        name: name,
        location: location,
        discription : _description,
        currentOwner: owner
    });

    hasAccess[owner][propertyId] = true;

    emit PropertyAdded(propertyId, name, location, owner);
}
```

```
    function transferProperty(
        string memory propertyId,
        address newOwner
    ) external hasPropertyAccess(propertyId) {
        require(newOwner != address(0), "Invalid new owner");

        address currentOwner = properties[propertyId].currentOwner;
        properties[propertyId].currentOwner = newOwner;

        hasAccess[currentOwner][propertyId] = false;
        hasAccess[newOwner][propertyId] = true;

        emit PropertyTransferred(propertyId, currentOwner, newOwner);
    }

    function getPropertyDetails(
        string memory propertyId
    ) external view returns (string memory, string memory, address) {
        Property memory prop = properties[propertyId];
        return (prop.name, prop.location, prop.currentOwner);
    }
}
```

Connector.js

```
const { ethers } = require("ethers");
```

```
const abi = [
 {
  "inputs": [],
  "stateMutability": "nonpayable",
  "type": "constructor"
 },
 {
  "anonymous": false,
  "inputs": [
   {
    "indexed": true,
    "internalType": "string",
    "name": "propertyId",
    "type": "string"
   },
   {
    "indexed": false,
    "internalType": "string",
    "name": "name",
    "type": "string"
   },
   {
    "indexed": false,
    "internalType": "string",
    "name": "location",
```

```
    "type": "string"
   },
   {
    "indexed": true,
    "internalType": "address",
    "name": "owner",
    "type": "address"
   }
  ],
  "name": "PropertyAdded",
  "type": "event"
 },
 {
  "anonymous": false,
  "inputs": [
   {
    "indexed": true,
    "internalType": "string",
    "name": "propertyId",
    "type": "string"
   },
   {
    "indexed": true,
    "internalType": "address",
    "name": "from",
    "type": "address"
```

```
  },
  {
   "indexed": true,
   "internalType": "address",
   "name": "to",
   "type": "address"
  }
 ],
 "name": "PropertyTransferred",
 "type": "event"
},
{
 "inputs": [
  {
   "internalType": "string",
   "name": "propertyId",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "name",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "location",
```

```
    "type": "string"
   },
   {
    "internalType": "string",
    "name": "_description",
    "type": "string"
   }
  ],
  "name": "addProperty",
  "outputs": [],
  "stateMutability": "nonpayable",
  "type": "function"
 },
 {
  "inputs": [
   {
    "internalType": "string",
    "name": "propertyId",
    "type": "string"
   }
  ],
  "name": "getPropertyDetails",
  "outputs": [
   {
    "internalType": "string",
    "name": "",
```

```
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "",
   "type": "string"
  },
  {
   "internalType": "address",
   "name": "",
   "type": "address"
  }
 ],
 "stateMutability": "view",
 "type": "function"
},
{
 "inputs": [
  {
   "internalType": "address",
   "name": "",
   "type": "address"
  },
  {
   "internalType": "string",
   "name": "",
```

```
  "type": "string"
 }
],
"name": "hasAccess",
"outputs": [
 {
  "internalType": "bool",
  "name": "",
  "type": "bool"
 }
],
"stateMutability": "view",
"type": "function"
},
{
"inputs": [],
"name": "owner",
"outputs": [
 {
  "internalType": "address",
  "name": "",
  "type": "address"
 }
],
"stateMutability": "view",
"type": "function"
```

```
},
{
 "inputs": [
  {
   "internalType": "string",
   "name": "",
   "type": "string"
  }
 ],
 "name": "properties",
 "outputs": [
  {
   "internalType": "string",
   "name": "propertyId",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "name",
   "type": "string"
  },
  {
   "internalType": "string",
   "name": "location",
   "type": "string"
  },
```

```
 {
  "internalType": "string",
  "name": "discription",
  "type": "string"
 },
 {
  "internalType": "address",
  "name": "currentOwner",
  "type": "address"
 }
],
"stateMutability": "view",
"type": "function"
},
{
"inputs": [
 {
  "internalType": "string",
  "name": "propertyId",
  "type": "string"
 },
 {
  "internalType": "address",
  "name": "newOwner",
  "type": "address"
 }
```

```
  ],
  "name": "transferProperty",
  "outputs": [],
  "stateMutability": "nonpayable",
  "type": "function"
 }
]

if (!window.ethereum) {
 alert('Meta Mask Not Found')
 window.open("https://metamask.io/download/")
}

export const provider = new
ethers.providers.Web3Provider(window.ethereum);
export const signer = provider.getSigner();
export const address = "0xD397357A9c446da4a2eB1a7ff47884D2982CBec7"

export const contract = new ethers.Contract(address, abi, signer)
```

home.js

```
import React, { useState } from "react";
import { Button, Container, Row, Col } from 'react-bootstrap';
import '../../node_modules/bootstrap/dist/css/bootstrap.min.css';
import { contract } from "./connector";
```

```
function Home() {
 const [Id, setId] = useState("");
 const [name, setname] = useState("");
 const [location, setLocation] = useState("");
 const [des, setDes] = useState("");
 const [TransferPropertId, setTransferPropertId] = useState("");
 const [MeatAddr, setMeatAddr] = useState("");
 const [PropDetailId, setPropDetailId] = useState("");
 const [PropDetails, setPropDetails] = useState("");
 const [Wallet, setWallet] = useState("");



 const handleId = (e) => {
  setId(e.target.value)
 }

 const handleName = (e) => {
  setname(e.target.value)
 }

 const handleLocation = (e) => {
  setLocation(e.target.value)
 }

 const handleDes = (e) => {
```

```
 setDes(e.target.value)
}

const handleAddProperty = async() => {
 try {
  let tx  = await contract.addProperty(Id.toString(),name,location,des)
  let wait = await tx.wait()
  console.log(wait);
  alert(wait.transactionHash)
 } catch (error) {
  alert(error)
 }
}

const handleTransferPropertyId = (e) => {
 setTransferPropertId(e.target.value)
}

const handleMetaAddr = (e) => {
 setMeatAddr(e.target.value)
}

const handletransferProperty = async () => {
 try {
  let tx = await contract.transferProperty(TransferPropertId.toString(),
MeatAddr)
```

```
  let wait = await tx.wait()
  console.log(wait);
  alert(wait.transactionHash)

 } catch (error) {
  alert(error)
 }
}

const handlePropId = (e) => {
 setPropDetailId(e.target.value)
}

const handlePorpDetails = async() => {
 try {
  let tx = await contract.getPropertyDetails(PropDetailId.toString())
  let arr = []

  tx.map(e => arr.push(e))

  setPropDetails(arr)
  console.log(tx);
  // alert(tx)
 } catch (error) {
  alert(error)
 }
```

```
}

const handleWallet = async () => {
 if (!window.ethereum) {
  return alert('please install metamask');
 }

 const addr = await window.ethereum.request({
  method: 'eth_requestAccounts',
 });

 setWallet(addr[0])

}

return (
 <div>
  <h1 style={{ marginTop: "30px", marginBottom: "80px" }}>Property
Listing on Blockchain</h1>
  {!Wallet ?

   <Button onClick={handleWallet} style={{ marginTop: "30px",
marginBottom: "50px" }}>Connect Wallet </Button>
   :
   <p style={{ width: "250px", height: "50px", margin: "auto", marginBottom:
"50px", border: '2px solid #2096f3' }}>{Wallet.slice(0, 6)}....{Wallet.slice(-
6)}</p>
```

```
  }
  <Container style={{ display: "flex" }}>
   <Row >
    <Col>
     <div>


      <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleId} type="number" placeholder="Enter Property ID"
value={Id} /> <br />


      <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleName} type="string" placeholder="Enter name"
value={name} /> <br />


      <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleLocation} type="string" placeholder="Enter location"
value={location} /><br />


      <input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleDes} type="string" placeholder="Enter description"
value={des} /><br />


      <Button onClick={handleAddProperty} style={{ marginTop: "10px" }}
variant="primary">Add Property To Blockchain</Button>
     </div>
    </Col>
    <Col>
```

```
<div style={{marginTop:"80px"}}>

<input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleTransferPropertyId} type="string" placeholder="Enter
Property Id" value={TransferPropertId} /><br />

<input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handleMetaAddr} type="string" placeholder="Enter new Owner
Metamask Address" value={MeatAddr} /><br />

<Button onClick={handletransferProperty} style={{ marginTop: "10px"
}} variant="primary">Transfer property</Button>

</div>
</Col>
</Row>
<Row>
<Col>
<div>

<input style={{ marginTop: "10px", borderRadius: "5px" }}
onChange={handlePropId} type="string" placeholder="Enter property Id"
value={PropDetailId} /><br />

<Button onClick={handlePorpDetails} style={{ marginTop: "10px" }}
variant="primary">Get property Detail</Button>
```

```
        {PropDetails ? PropDetails?.map(e => {
         return <p>{e}</p>
        }) : <p></p>}
       </div>
      </Col>
     </Row>

    </Container>

   </div>
  )
}

export default Home;
```

App.js

```
import './App.css';
import Home from './Page/Home'

function App() {
  return (
    <div className="App">
      <header className="App-header">
```

```
        <Home />
      </header>
    </div>
  );
}

export default App;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

```
// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

reportWebVitals.js

```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) =>
{
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;
```

setupTests.js

```
import '@testing-library/jest-dom';
```

GITHUB & PROJECT DEMO LINK

- GitHub Link: https://github.com/Alagusathish/NM2023TMID09171.git
- Demo Link: https://drive.google.com/file/d/1XWNEtJhpQejTJ6pYGEzN_5r65gvyxhoT/view?usp=sharing