

BLOCKCHAIN BASED SMART REAL ESTATE MANAGEMENT SYSTEM

PROJECT REPORT

Date	28 October 2023
Team ID	NM2023TMID03827

TEAM MEMBERS :

SATHIYA BABU .R
JAYA SURYA S
SATHIYA BALAN S
VIDHYA SAGAR V

INDEX

1. INTRODUCTION

Project Overview

Purpose

2. LITERATURE SURVEY

Existing problem

References

Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

Empathy Map Canvas

Ideation & Brainstorming

4. REQUIREMENT ANALYSIS

Functional requirement

Non-Functional requirements

5. PROJECT DESIGN

Data Flow Diagrams & User Stories

Solution Architecture

6. PROJECT PLANNING & SCHEDULING

Technical Architecture

Sprint Planning & Estimation

Sprint Delivery Schedule

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

Feature 1

Feature 2

Database Schema (if Applicable)

8. PERFORMANCE TESTING

Performance Metrics

9. RESULTS

Output Screenshots

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1.INTRODUCTION

1.1 PROJECT OVERVIEW :

Project Description:

The Blockchain-Based Smart Real Estate Management System is an innovative platform designed to transform the way real estate assets are bought, sold, managed, and maintained. Leveraging blockchain technology and smart contracts, this system offers a secure, transparent, and automated solution for real estate transactions and management.

Project Objectives:

Enhanced Transparency:

Implement a transparent ledger system that records all real estate transactions and ownership details on the blockchain, reducing the risk of fraud and disputes.

Efficient Property Transactions:

Facilitate the buying and selling of real estate through blockchain-based smart contracts, reducing paperwork, third-party intermediaries, and transaction time.

Streamlined Property Management:

Create a platform for property owners, tenants, and property managers to efficiently manage leases, maintenance, and payments using smart contracts and automated notifications.

Immutable Ownership Records:

Ensure that property ownership records are stored in an immutable manner on the blockchain, reducing the risk of fraudulent title changes.

Tokenization of Real Estate:

Explore the option of tokenizing real estate assets, allowing fractional ownership and easier liquidity for investors.

Integration with IoT:

Integrate Internet of Things (IoT) devices for real-time monitoring of property conditions, such as temperature, humidity, and security, to improve maintenance and security.

User-Friendly Interface:

Develop a user-friendly web or mobile application to make it easy for property owners, tenants, and real estate professionals to interact with the system.

Compliance:

Ensure the system complies with local real estate laws and regulations to maintain legal validity and trust.

Security:

Implement robust security measures to protect sensitive data and transactions, such as two-factor authentication and encryption.

Technology Stack:

Blockchain platform (e.g., Ethereum, Binance Smart Chain, Hyperledger Fabric)
Smart contract development (Solidity, Vyper, or similar) Web or mobile application development
IoT integration (sensors, smart locks, cameras) Database (for off-chain data storage)
Encryption and security protocols User authentication and authorization systems

Project Phases:

Research and Planning:

Define project scope, requirements, and objectives. Conduct a feasibility study and select the appropriate blockchain platform.

Blockchain Development:

Develop the core blockchain infrastructure, including smart contracts for property transactions and ownership management.

User Interface Development:

Create a user-friendly front-end application for property owners, tenants, and real estate professionals.

IoT Integration:

Integrate IoT devices and sensors to collect real-time data from properties.

Testing and Quality Assurance:

Thoroughly test the system for security, functionality, and performance.

Deployment and Launch:

Deploy the system to a production environment and launch it for users. Maintenance and Updates: Provide ongoing support, updates, and maintenance to ensure the system's reliability and security.

1.2 PURPOSE :

A blockchain-based smart real estate management system serves several important purposes in the real estate industry, leveraging the unique capabilities of blockchain technology and smart contracts. Some of the key purposes include:

1. Transparency and Trust:

- Blockchain technology provides a transparent and immutable ledger that records all transactions and changes in real estate ownership and management. This transparency builds trust among stakeholders, as it ensures that the data cannot be altered or manipulated without consensus.

2. Security:

- Blockchain's cryptographic security features make it difficult for unauthorized parties to tamper with or hack into the system, reducing the risk of fraud and unauthorized access to sensitive real estate data.

3. Reduced Intermediaries:

- By utilizing smart contracts on a blockchain, real estate transactions can be

automated and executed without the need for intermediaries like real estate agents, title companies, or banks. This reduces costs and speeds up the process.

4. Ownership Verification:

- Blockchain can be used to establish and verify ownership of real estate properties, eliminating disputes and fraud related to property titles. This is particularly beneficial in regions where land ownership records are often unreliable.

5. Fractional Ownership:

- Blockchain allows for the tokenization of real estate, enabling fractional ownership. This opens up real estate investments to a wider range of individuals and allows for easier diversification of real estate portfolios.

6. Improved Due Diligence:

- Smart real estate management systems can provide a comprehensive and accessible history of a property, including records of maintenance, repairs, and renovations. This can simplify the due diligence process for potential buyers or investors.

7. Efficiency and Automation:

- Smart contracts can automate various aspects of property management, including rent collection, maintenance requests, and lease agreements, reducing administrative overhead and improving overall efficiency.

8. Enhanced Access:

- Blockchain-based systems can make real estate investments more accessible to a global audience, facilitating cross-border transactions and investment opportunities.

9. Decentralization:

- A blockchain-based system can operate without a central authority, reducing the concentration of power and potentially leading to a more equitable and decentralized real estate market.

10. Data Integrity:

- Blockchain ensures that historical data about real estate properties and transactions remains secure and unaltered, making it an invaluable resource for property valuation, historical research, and regulatory compliance.

11. Regulatory Compliance:

- Smart contracts can be programmed to automatically enforce regulatory compliance, such as rent control laws, property tax payments, and environmental regulations.

12. Cost Reduction:

- By eliminating intermediaries, reducing the potential for fraud, and automating various processes, blockchain-based real estate management systems can significantly

reduce costs associated with real estate transactions and ownership.

Overall, a blockchain-based smart real estate management system aims to make the real estate industry more efficient, secure, and accessible while enhancing trust and transparency among all parties involved in real estate transactions and property management.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM :

While blockchain-based real estate management systems offer numerous benefits, they also face several challenges and problems, some of which include:

1. Adoption and Integration:

- The adoption of blockchain technology in the real estate industry can be slow and challenging due to the need for industry-wide consensus, changes in existing processes, and the integration of legacy systems with new blockchain-based platforms.

2. Scalability:

- As the number of real estate transactions and properties on the blockchain increases, the network's scalability becomes a concern. Many blockchain networks struggle with high transaction volumes and slow processing times.

3. Regulatory Hurdles:

- The real estate industry is heavily regulated, and blockchain may not always align with existing regulatory frameworks. Compliance with property laws, privacy regulations, and tax codes can be complex and may vary from one jurisdiction to another.

4. Security Concerns:

- While blockchain is known for its security, it is not entirely immune to attacks or vulnerabilities. Smart contracts can have bugs that are exploited, and private keys can be lost or stolen. Ensuring the security of digital assets and transactions is paramount.

5. Interoperability:

- Different blockchain platforms and projects may not be compatible with each other, leading to interoperability issues. This can make it difficult for various stakeholders to collaborate and share data seamlessly.

6. Data Privacy:

- Blockchain's transparency can be a double-edged sword when it comes to data privacy. While it's essential for transparency, it can also expose sensitive real estate information to unauthorized parties if not managed properly.

7. Lack of Legal Clarity:

- Legal frameworks for blockchain-based real estate transactions and property ownership are still evolving. This can lead to uncertainty and legal disputes, particularly in cases where smart contracts are used to automate complex agreements.

8. User-Friendliness:

- Blockchain technology can be complex for non-technical users. User-friendly interfaces and intuitive applications are needed to make these systems accessible to a wider range of participants.

9. Tokenization Risks:

- Tokenizing real estate assets can make them more liquid and tradable, but it also introduces new risks related to market volatility and speculative trading. It may not always align with the long-term, illiquid nature of real estate investments.

10. Oracles and Data Sources:

- Smart contracts rely on external data sources (oracles) to execute actions based on real-world events. The accuracy and reliability of these oracles can be a significant challenge.

11. Reversibility:

- The immutability of blockchain transactions can be problematic when errors occur or disputes arise. Unlike traditional systems, blockchain transactions are generally irreversible, which can make dispute resolution more challenging.

12. Limited Access:

- Blockchain technology requires internet access and digital literacy, which may exclude individuals who do not have access to these resources. This could lead to a digital divide in real estate transactions.

13. High Energy Consumption:

- Some blockchain networks, such as Bitcoin and Ethereum, consume a significant amount of energy, which has raised environmental concerns. This may become a problem for real estate projects aiming to maintain sustainability.

2.2 REFERENCES :

- *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008.
- *Eden Hall : Digitized documents/a digital system must be in place before blockchain can be utilized*(2008
- *Ioannis Karamitsos, Maria Papadaki and Nedaa Baker Al Barghuthi, "Design of the Blockchain Smart Contract: A Use Case for Real Estate", 2018*
- *Konashevych O. General Concept of Real Estate Tokenization on Blockchain. Eur Prop Law J (Internet)-2020*
- *Spielman A. Blockchain: digitally rebuilding the real estate industry [Internet] [Thesis]. Massachusetts Institute of Technology; 2016 [cited 2020 Jun 17]*
- *Tokenisation of corporate real estate on the blockchain: new strategies for corporate ownership and financial management*
- *Corp. Real. Estate J.,(2020)*
- *Barriers to the digitalisation and innovation of Australian Smart Real Estate: a managerial perspective on the technology non-adoption - 2021*

2.3 PROBLEM STATEMENT :

The problem statement for a blockchain-based real estate management system can be framed as follows:

"Traditional real estate management systems suffer from inefficiencies, lack of transparency, and susceptibility to fraud and disputes. Property transactions and management processes are often slow, expensive, and reliant on intermediaries. Additionally, property ownership records can be prone to errors and tampering, leading to legal disputes and title-related issues. To address these challenges, there is a need for a blockchain-based real estate management system that can provide transparency, security, automation, and cost-effectiveness. Such a system should also ensure compliance with legal and regulatory requirements while making real estate transactions and property management more accessible, efficient, and trustworthy."

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS :

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

It is a useful tool to help teams better understand their users.

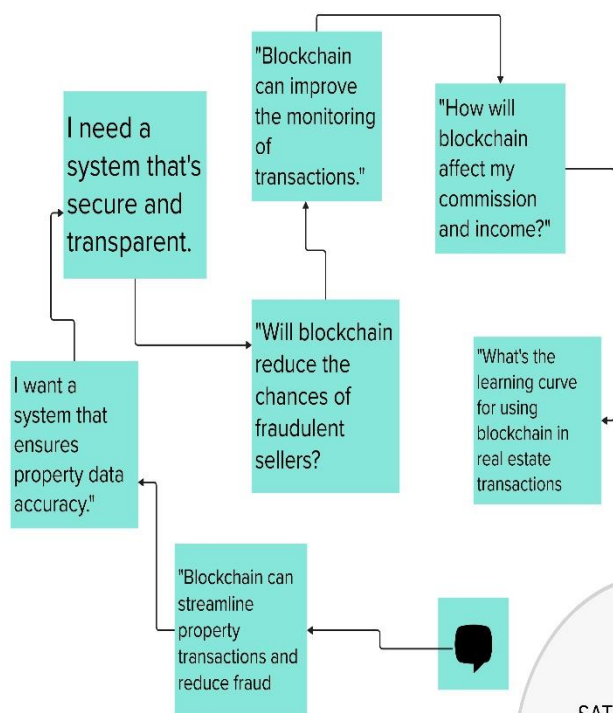
Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Reference: <https://www.mural.co/templates/empathy-map-canvas>



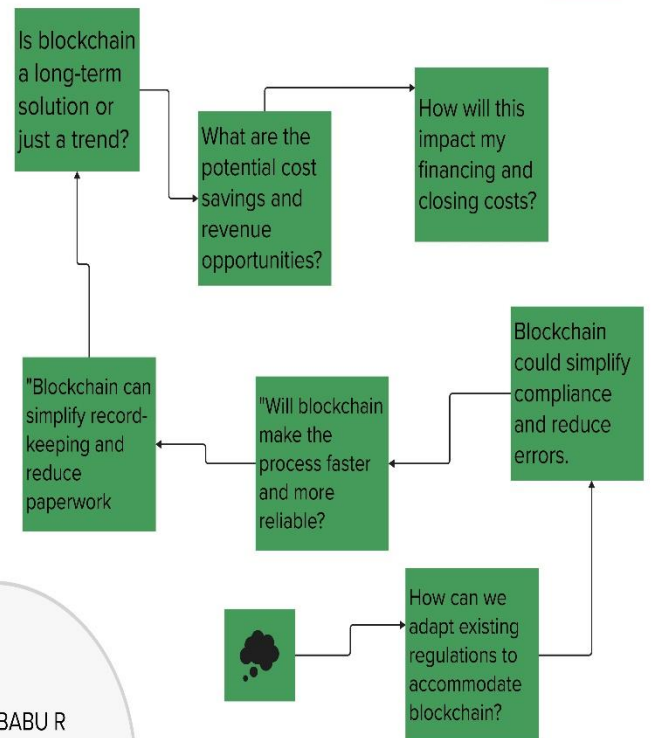
Says

What have we heard them say?
What can we imagine them saying?



Thinks

What are their wants, needs, hopes, and dreams?
What other thoughts might influence their behavior?

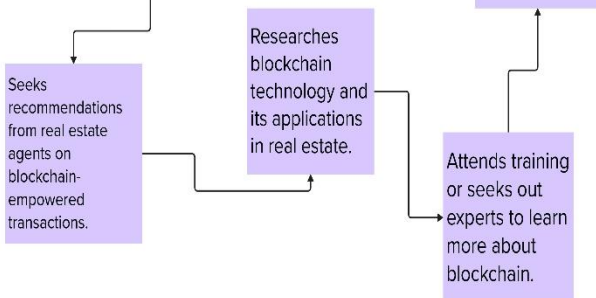


SATHIYA BABU R
VIDHYA SAGAR V
JAYA SURYA S
SATHIYA BALAN S



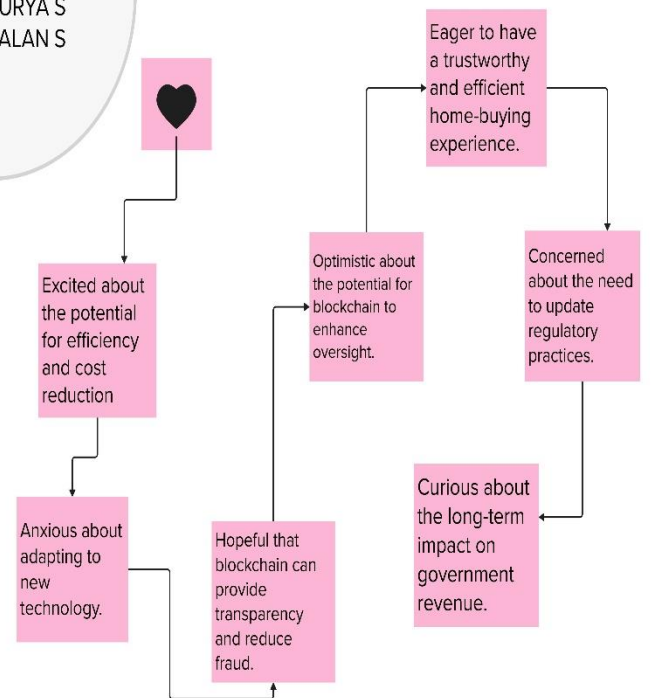
Does

What behavior have we observed?
What can we imagine them doing?



Feels

What are their fears, frustrations, and anxieties?
What other feelings might influence their behavior?



[See an example](#)

3.2 IDEATION & BRAINSTROMING :

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Reference: <https://www.mural.co/templates/empathy-map-canvas>



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 **10 minutes** to prepare

🕒 **1 hour** to collaborate

👤 **2-8 people** recommended



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 **10 minutes**

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

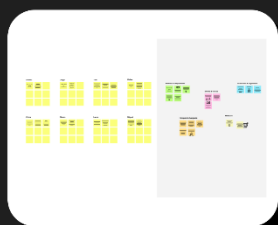
Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#)



Need some inspiration?

See a finished version of this template to kickstart your work.

[Open example](#)



1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

 5 minutes

PROBLEM

How might we [your problem statement]?

Blockchains like Bitcoin and Ethereum have faced scalability issues, causing slow transaction processing times and high fees during periods of high demand. Scaling blockchain technology to handle the volume of real estate transactions is a challenge.



Key rules of brainstorming

To run a smooth and productive session



Stay in topic.



Encourage wild ideas.



Defer judgment.



Listen to others.



Go for volume.



If possible, be visual.

2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Person 1

To leverage blockchain technology for more efficient, secure, and transparent real estate management.

Enable investors to easily track and manage their investments

Tokenize real estate properties, allowing for fractional ownership through blockchain-based tokens.

Person 2

Eliminate the risk of fraudulent property transactions by linking titles to immutable blockchain records.

Create user-friendly mobile apps and web platforms that make it easy for users to interact with blockchain-based real estate services.

Launch awareness campaigns and educational initiatives to inform stakeholders

Person 3

Facilitate peer-to-peer property sales

Enable seamless KYC (Know Your Customer) and AML (Anti-Money Laundering) checks

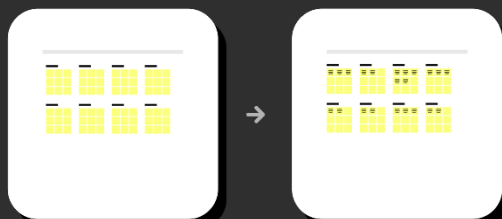
Automate maintenance requests and payments for repairs through smart contracts

Person 4

Implement a blockchain-based system for property valuation and appraisal data.

Enhance data security by storing sensitive real estate information on a distributed and encrypted blockchain.

Enable transparent procurement processes for developers and builders.



3

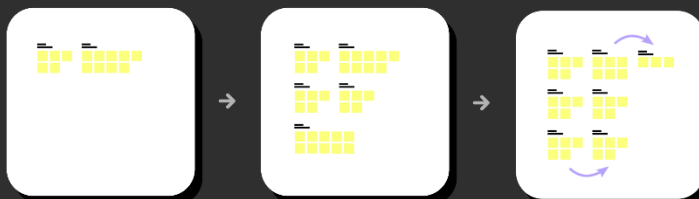
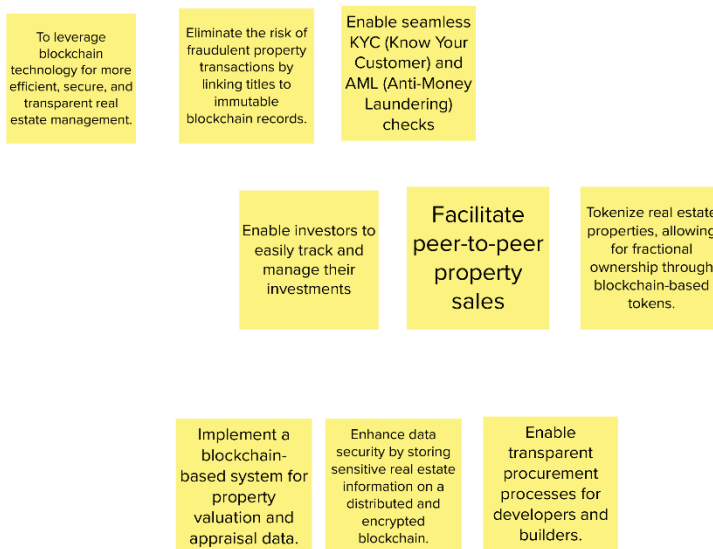
Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.



4

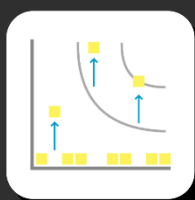
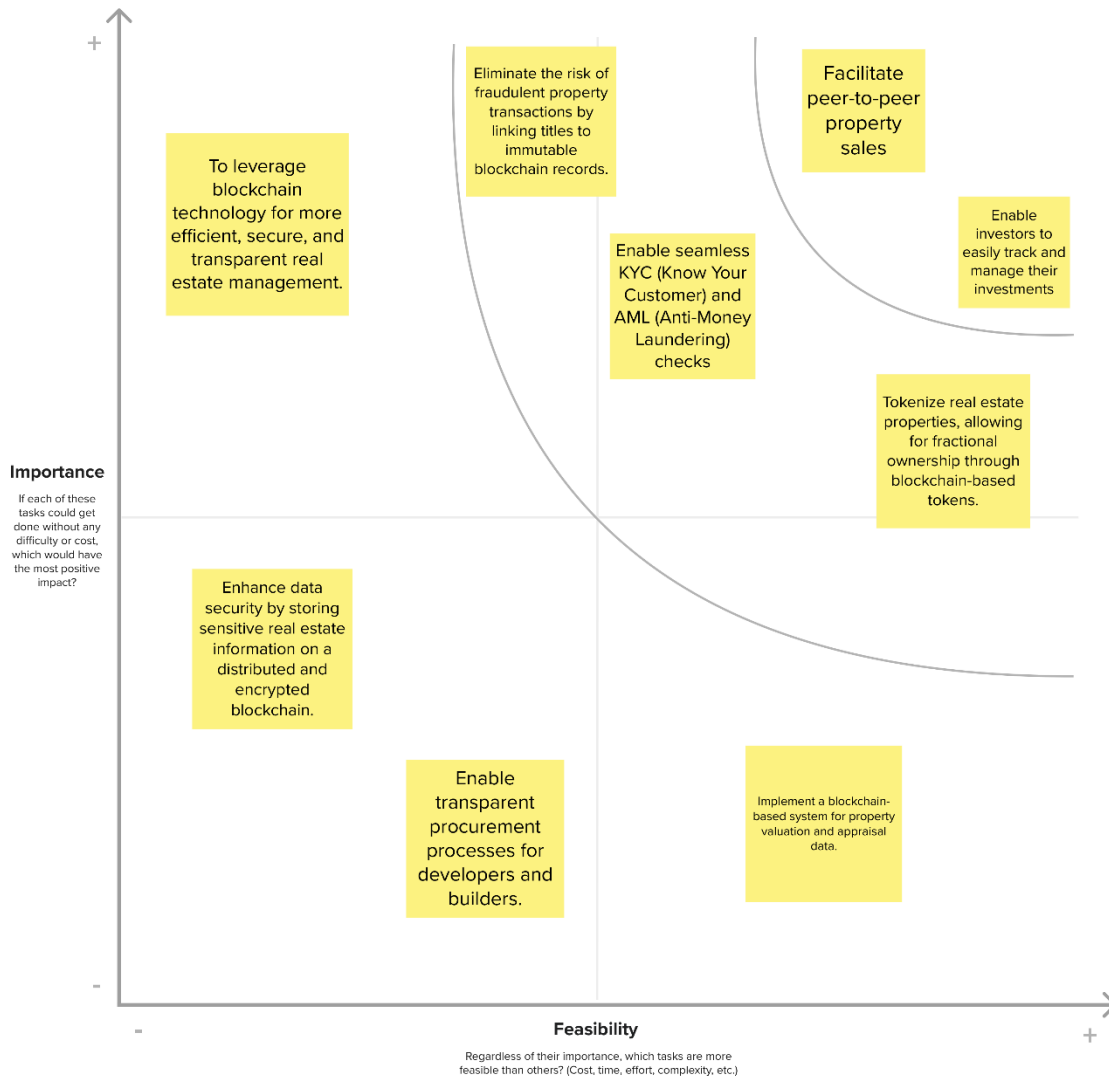
Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H** key on the keyboard.



4. REQUIREMENT ANALYSIS

Project Design Phase-II

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration and Authentication	<ul style="list-style-type: none">• Users should be able to register and authenticate themselves securely.• Different user roles, such as property owners, buyers, agents, and administrators, should be supported.
FR-2	Property Listings	Property owners should be able to create and manage property listings, including details like location, size, price, and media (photos, videos). Listings should include unique property identifiers and be timestamped on the blockchain.
FR-3	Property Search and Discovery:	<ul style="list-style-type: none">• . Users should be able to search for properties based on various criteria, including location, price, and property type.
FR-4	Payment Processing:	. Facilitate secure and transparent payment processing for property transactions, including support for various payment methods and cryptocurrencies
FR-5	Document Management:	Allow users to upload and store important documents related to the property, such as titles, deeds, inspections, and permits. Ensure document authenticity and immutability through blockchain technology.
FR-6	Notifications and Alerts:	Send notifications and alerts to users for important events in the real estate transaction process, such as offer acceptance, payment verification, and ownership transfer.

FR-7	User Feedback and Ratings:	Allow users to leave feedback and ratings for property listings, agents, and other participants in the system.
FR-8	Integration with External Services:& Reporting and Analytics:	<p>Provide reporting and analytics tools to help users track their real estate activities and investments.</p> <p>Integrate with external services like mapping and geolocation, property data providers, legal services, and financial institutions for a comprehensive real estate ecosystem.</p>
FR-9	Maintenance and Support	Define requirements for ongoing system maintenance, updates, and user support
FR-10	User Experience (UX) and Accessibility:	<p>Ensure the system has an intuitive and user-friendly interface.</p> <p>Make the system accessible to all users, including those with disabilities.</p>

4.2 Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

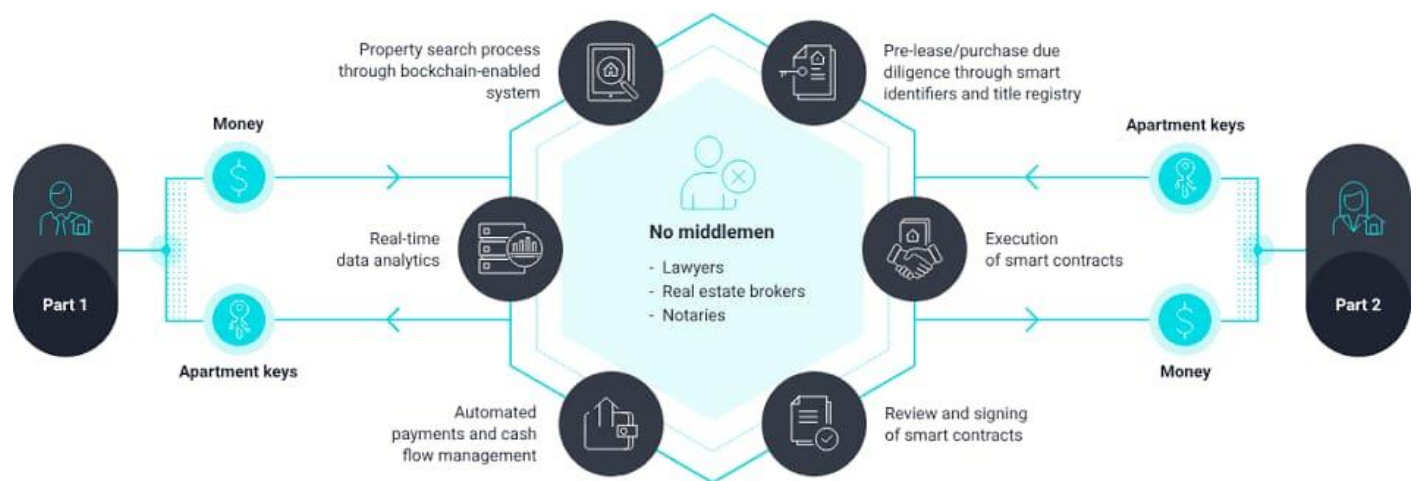
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	User Experience: Ensure the user interface is intuitive and user-friendly. Accessibility: Ensure the system is accessible to users with disabilities.
NFR-2	Security	Data Security: Ensure that all data on the blockchain is securely stored and tamper-resistant. Access Control: Implement strict access controls to ensure that only authorized users can interact with the system. Identity Management: Provide robust identity management to verify the authenticity of users and entities involved in real estate transactions.
NFR-3	Reliability	High Availability: Ensure the system is available 24/7, with minimal downtime. Backup and Recovery: Implement regular backups and a reliable recovery plan in case of system failures or data loss. .
NFR-4	Performance	Transaction Throughput: Define the required transactions per second (TPS) the system should handle. Latency: Specify acceptable response times for various system operations. Scalability: The system should be able to scale to accommodate growing numbers of users and real estate transactions..
NFR-5	Cost and Energy Efficiency	Assess the energy consumption and overall operational costs of the blockchain network and aim for efficiency improvements
NFR-6	Scalability	Scalability and Performance Testing: Define a testing strategy to assess how the system performs under various loads and stress conditions.

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS & USER STORIES:

Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



The benefits of blockchain for the real estate industry



Easier property search



Improves pre-purchase due diligence



Reduces the need for intermediaries



Provides smart property contracts



Safer property transactions



Lowers the entrance to the real estate investing



Turns real estate into a liquid asset

USER STORIES :

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Farmer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Desika
		USN-2	As a user, I will receive confirmation email once I have registered for the application.	I can receive confirmation email & click confirm	High	Anuja
		USN-3	As a user, I can register for the application through Gmail.	I can get notification	Low	Gandhimathi
	Login	USN-4	As a user, I can log into the application through my email & password.	The login page must provide fields for entering a username and password	High	Noorul Rashana
		USN-5	The login page should have a "Remember Me" checkbox	Allows the user to remain logged in between sessions.	High	Desika
	Dashboard	USN-6	I want to have a dashboard that provides an overview of relevant information and features to help me manage my account or access important data.	The dashboard should include widgets or components that offer at-a glance information, such as recent activity, account status etc	Medium	Anuja
Customer (Web user)		USN-7	Users should have the ability to perform common tasks directly from the dashboard.	Tasks such as creating new records, viewing recent notifications	High	Noorul Rashana
Supply Chain developer	Username	USN-8	As a user of the application, I want to have the ability to set or change my username, ensuring it's unique and reflective of my	Users should have the option to change their username after	High	Gandhimathi

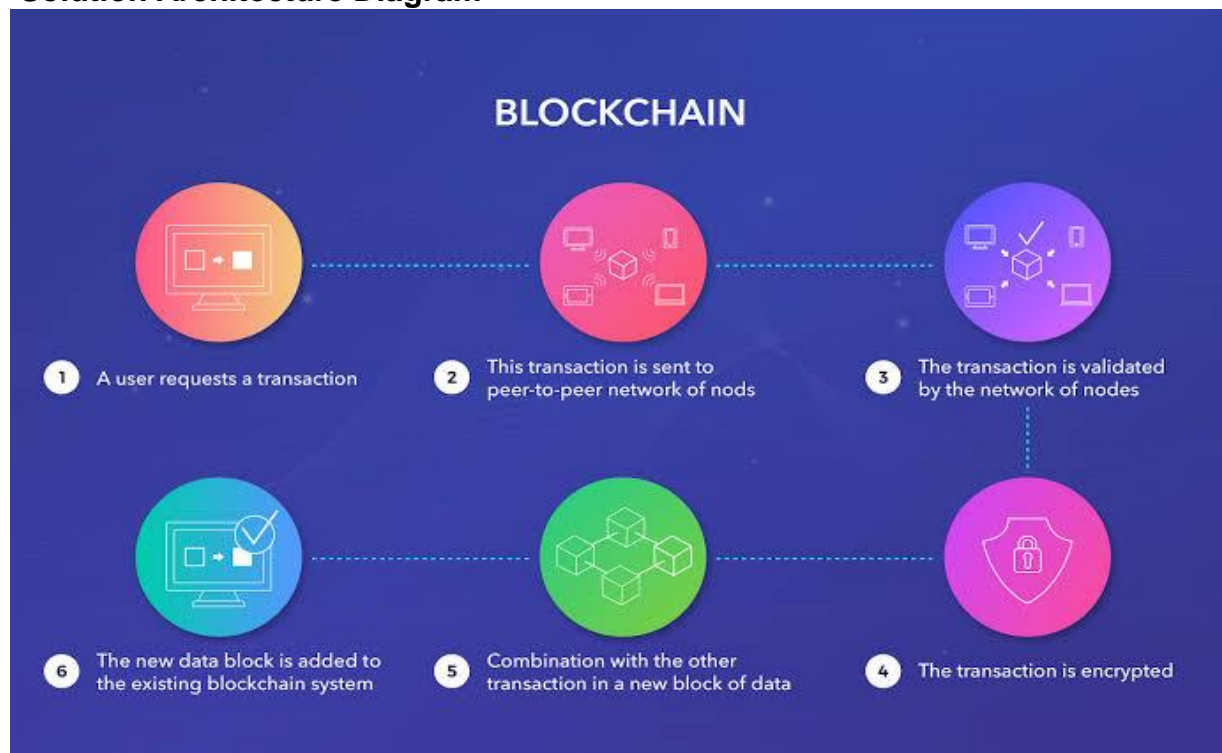
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
			identity within the system.	registration.		
		USN-9	Users should receive immediate feedback during the username creation or change process.	Indicating whether their chosen username is valid and available.	High	Anuja
Administrator	Membership	USN-10	I want to manage the membership records of our farmer members on Agridocchain.	It facilitate information sharing among them	Medium	Gandhimathi

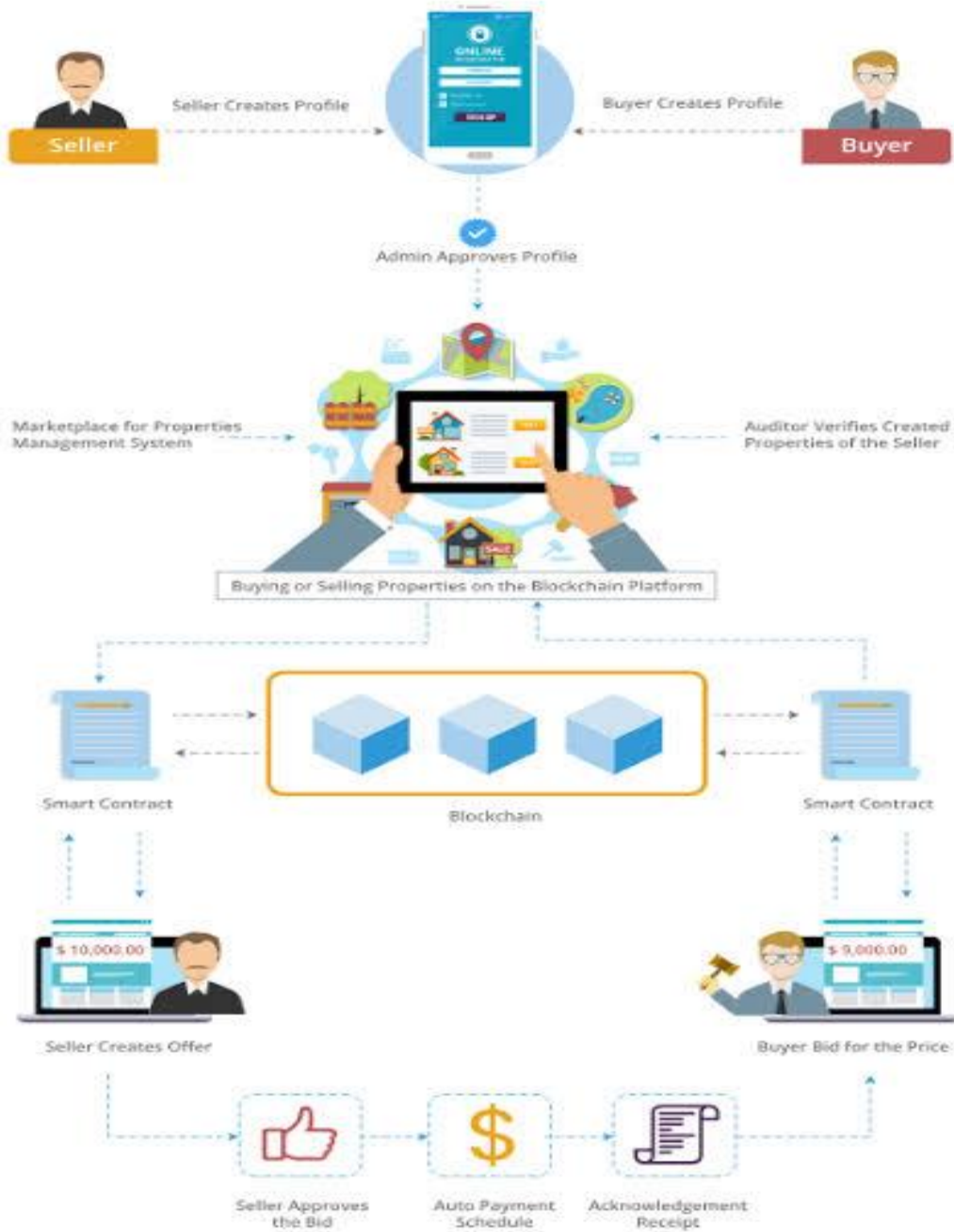
5.2 SOLUTION ARCHITECHTURE:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Solution Architecture Diagram

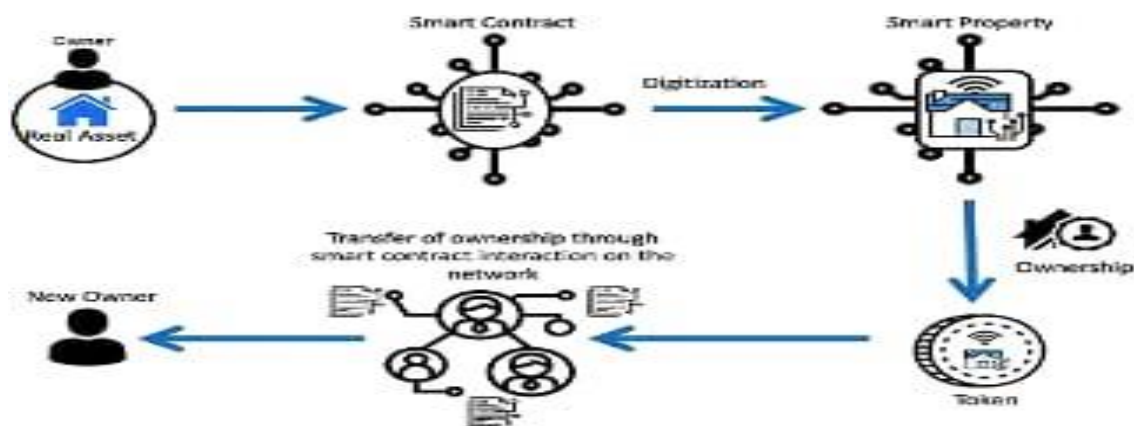




6. PROJECT PLANNING & SCHEDULING

6.1 TECHNICAL ARCHITECHTURE:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



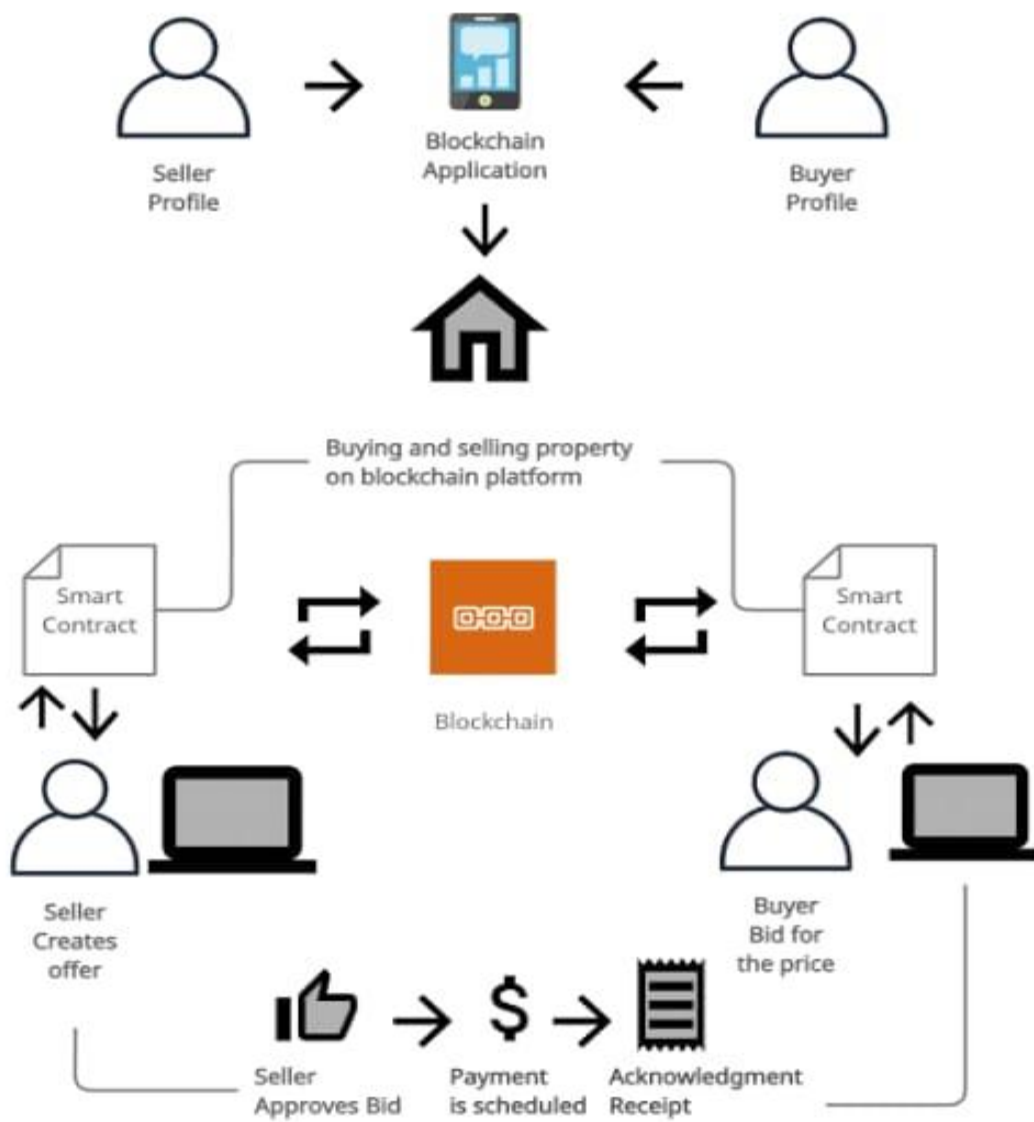


FIG: TECHNICAL ARCHITECTURE

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Develop user-friendly web interfaces	HTML, CSS, and JavaScript.
2.	Database	The core of the blockchain, it stores all transactions and data.	Distributed Ledger
3.	Cloud Services	Host your application on cloud platforms like for high availability	AWS, Azure, or Google Cloud
4.	File Storage	It will depend on the specific use cases, data volume, and regulatory constraints.	Centralized Storage, Cloud Storage, GlusterFS
5.	External API	For integrating with other systems and enabling data exchange.	REST or GraphQL
6.	Data Analytics Tools	Utilize data analytics tools for extracting insights from supply chain data.	Apache Spark or Elasticsearch
7.	Reporting Tools	Implement reporting frameworks for generating visual reports.	Tableau or Power BI
8.	Infrastructure	Involves a combination of hardware, software, and network components to support the application's functionality and performance	BC Nodes, Firewalls and Security, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Agridocschain should utilize open-source blockchain frameworks and this provides transparency, community support.	Ethereum, Hyperledger Fabric & Corda
2.	Security Implementations	Security is paramount, especially in a Doc chain management system. It should implement robust security measures.	SHA-256, Encryptions, Digital Certificates
3.	Scalable Architecture	This involves the ability to handle increased data and transaction loads without compromising performance or security.	Load Balancers, Docker and Kubernetes
4.	Availability	Implementing redundancy, failover mechanisms,	Cloud Service, Content Delivery

S.No	Characteristics	Description	Technology
		and 24/7 support is essential to ensure continuous availability.	Networks (CDNs)
5.	Performance	This includes efficient transaction processing, low latency, and the ability to handle a large number of concurrent users and data without slowdowns.	Caching Mechanism, Database Sharding & CDN

6.2 SPRINT PLANNING & EXECUTION:

Sprint planning and execution in blockchain based smart real estate management system Docs chain involves following steps to be taken to finish an overall quality project on time.

S.NO	SPRINTS	DESCRIPTION	EXECUTE BY
01	Ideation and proposed solution	Empathy map Canvas, Ideation & Brainstroming.	Vidhya sagar V
02	Requirements Analysis	Functional Requirements Non Functional Requirements	Sathiyabalan S
03	Project Design	Dataflow diagrams & User Stories Solution Architecture	Jayasurya S
04	Project planning	Technical Architecture	Vidhya sagar v Jaya surya s
05	Tools Requirements	Download VS code Node.js Metamask	Sathiya Babu R
06	Zip File Execution	Download zipfile & extract it & open it in VScode	Sathiya Babu R
07	Remix and Metamask	Execution of Solidity code and adding Metamask Extention to Remix.	Sathiya Babu R Vidhya sagar V
08	Implementation of Smart Contract &Interaction of frontend	Using file Connector.js Interaction with frontend .	Sathiya Babu R Jayasurya S
09	Performance testing	Testing of codes and project output	Sathiya Babu R Sathiya Balan S
10	Project Review	Review of output & results	Sathiya Babu R Jaya surya S Sathiya balan S Vidhya sagar V

6.3 SPRINT DELIVERY SCHEDULE:

S.NO	SPRINTS	START DATE	END DATE	STATUS
01	Ideation and proposed solution	20.10.2023	22.10.2023	Finished
02	Requirements Analysis	22.10.2023	23.10.2023	Finished
03	Project Design	24.10.2023	26.10.2023	Finished
04	Project planning	27.10.2023	28.10.2023	Finished
05	Tools Requirements &tools download	20.10.2023	22.10.2023	Finished
06	Zip File download & Extraction	22.10.2023	23.10.2023	Finished
07	Solidity code and adding Metamask Extention	24.10.2023	26.10.2023	Finished
08	Implementation of Smart Contract &Interaction of frontend	26.10.2023	28.10.2023	Finished
09	Performance testing Project Review	28.10.2023	29.10.2023	Review
10	Demo video creation	29.10.2023	29.10.2023	On Process

7. CODING AND SOLUTIONING

7.1 FEATURE 1:

Software – Remix Ethereum IDE

Remix is a software application, specifically an Integrated Development Environment (IDE), designed for Ethereum smart contract development. It provides a web-based platform for Ethereum developers to write, compile, deploy, and test smart contracts, making it easier to create decentralized applications (DApps) on the Ethereum blockchain.

It provides full support for the Solidity programming language, which is commonly used for Ethereum smart contract development. Remix can compile Solidity code into Ethereum bytecode, which is essential for deploying contracts on the Ethereum blockchain.

Remix provides a testing environment for writing and running unit tests for smart contracts, helping developers ensure their code functions as expected

```
// 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);
}
}

```

7.2 FEATURE 2:

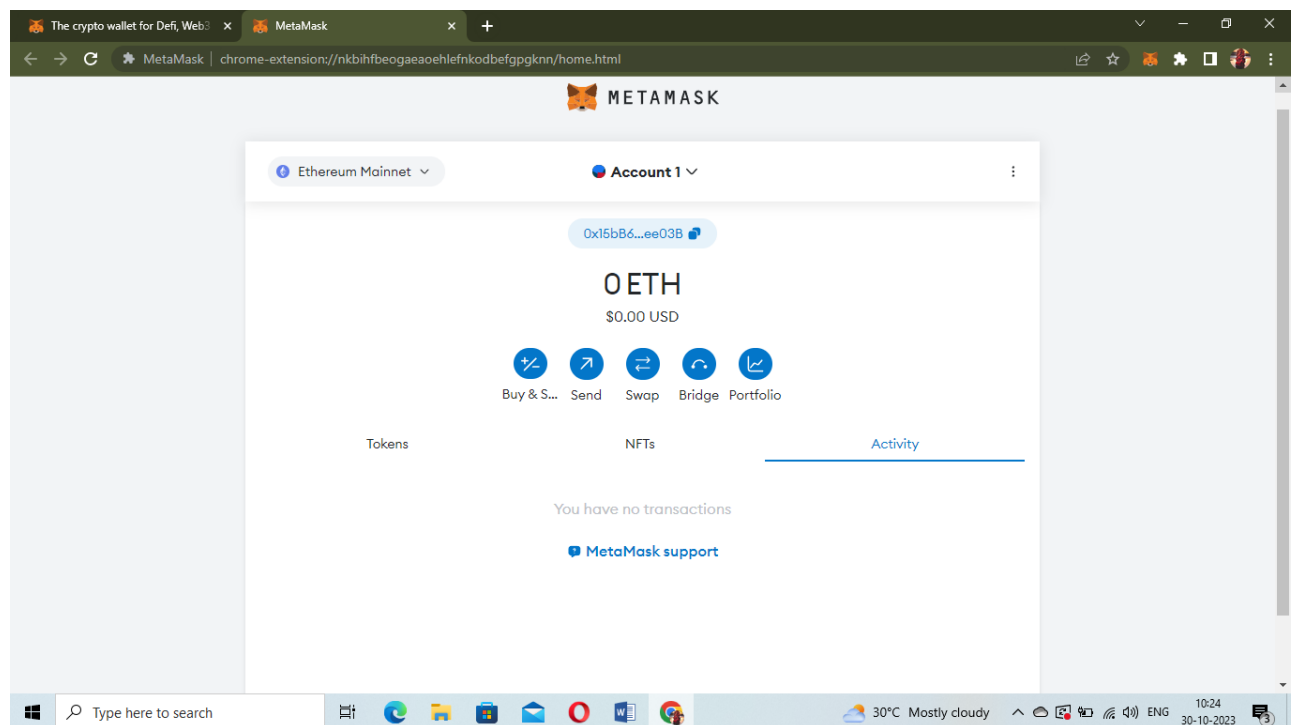
Metamask :

MetaMask is a popular cryptocurrency wallet and decentralized application (DApp) browser extension. It allows users to manage their Ethereum-based assets, interact with Ethereum-based DApps, and store private keys securely.

Users can access their wallet and interact with decentralized applications directly from their web browser.

Can be easily manage Ethereum-based tokens within MetaMask, including sending, receiving, and swapping tokens.

It's available as a browser extension for major browsers like Chrome, Firefox, and Brave, and as a mobile app for iOS



CODINGS:

Frontend Codings:

CSS code: (App.css)

```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
```



```

}
to {
  transform: rotate(360deg);
}
}

```

CSS code: (index.css)

```

body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}

```

JAVA SCRIPTS CODE: (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/")
}

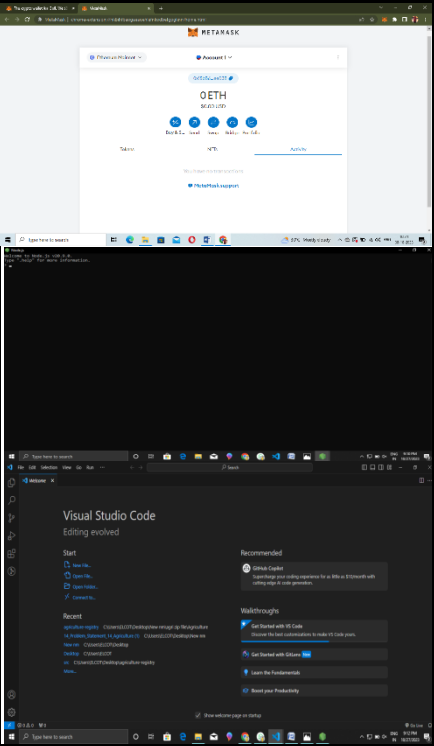
export const provider = new ethers.providers.Web3Provider(window.ethereum);
export const signer = provider.getSigner();
export const address = "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8"

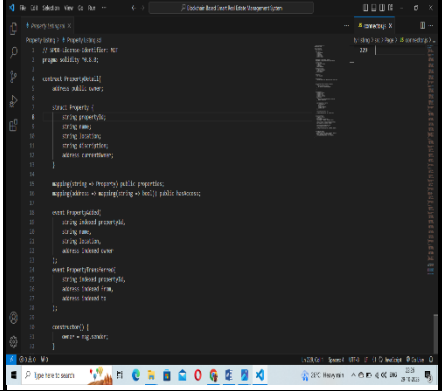
export const contract = new ethers.Contract(address, abi, signer)

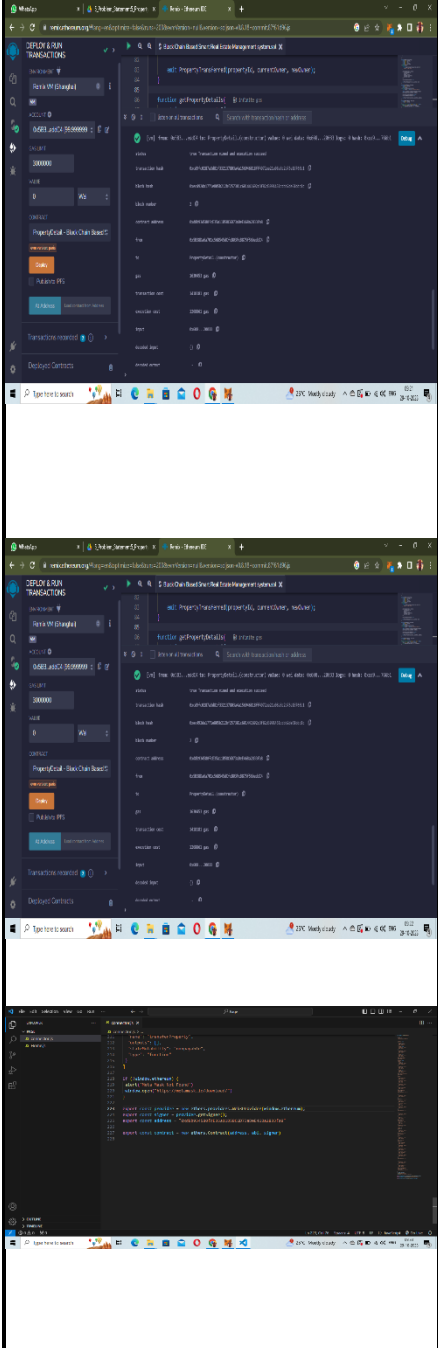
```

8. PERFORMANCE TESTING

8.1 PERFORMANCE METRICS :

S.No.	Parameter	Values	Screenshot
1.	Information gathering	Setup all the Prerequisite: 1.node.js 2.vs code 3.metamask	 The screenshot displays two overlapping windows. The top window is the Metamask browser extension interface, showing a 'Send ETH' transaction screen with a balance of 0.01 ETH. The bottom window is the Visual Studio Code editor, showing the 'Start' page with various recommended extensions and walkthroughs.

2.	Extract the zip files	Open to vs code	
----	-----------------------	-----------------	---

<p>3.</p>	<p>Remix Ide platform exploring</p>	<p>Deploy the smart contract code</p> <p>Deploy and run the transaction. By selecting the environment - inject the MetaMask.</p>	 <p>The image displays three screenshots of the Remix IDE interface. The top two screenshots show the 'DEPLOY & RUN TRANSACTIONS' panel, where the 'Injected MetaMask' environment is selected. The bottom screenshot shows the 'Compiler' panel, displaying Solidity code for a smart contract.</p>
-----------	---	--	--

4.	Open file explorer
----	--------------------

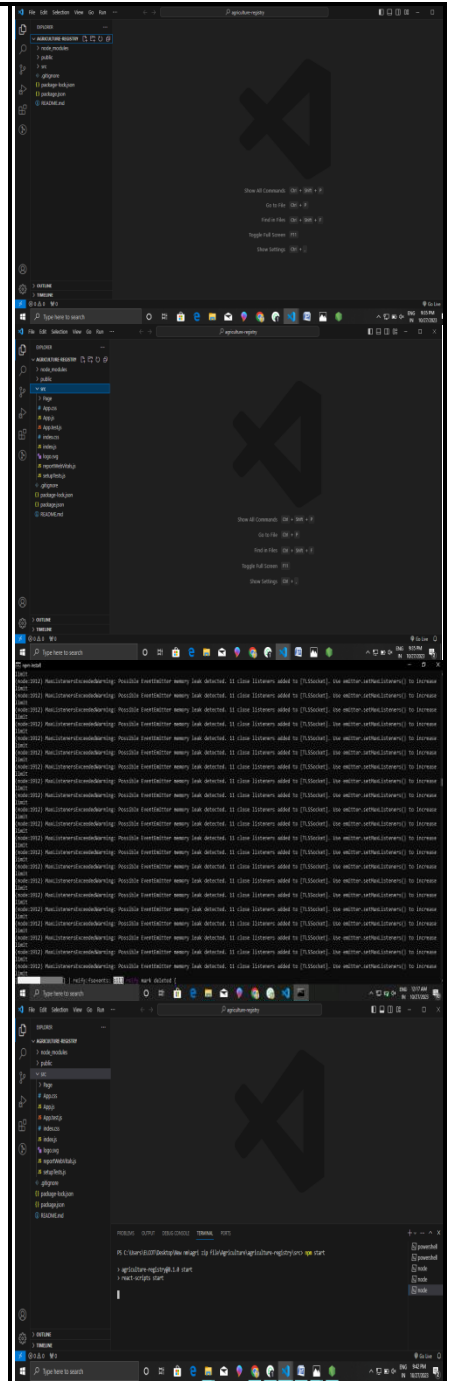
Open file explorer

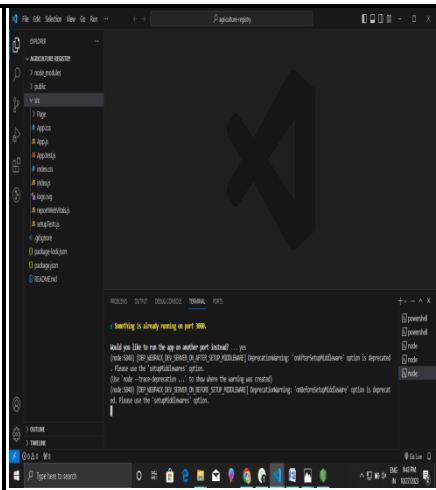
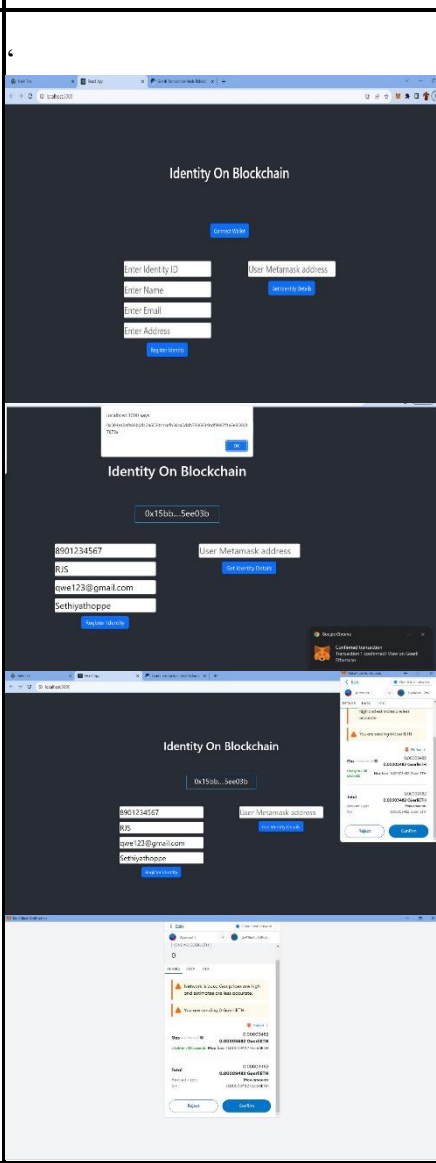
Open the extracted file and click on the folder.

Open src, and search for utiles.

Open cmd enter commands

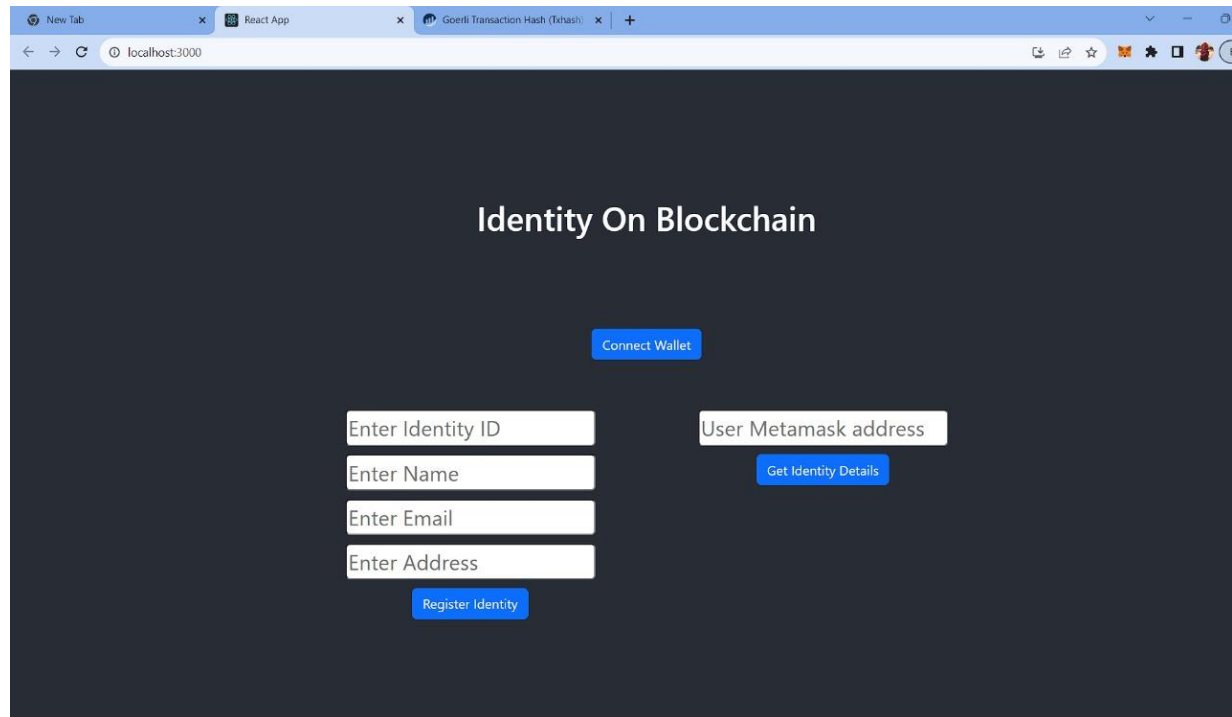
- 1.npm install
- 2.npm bootstrap
- 3.npm start

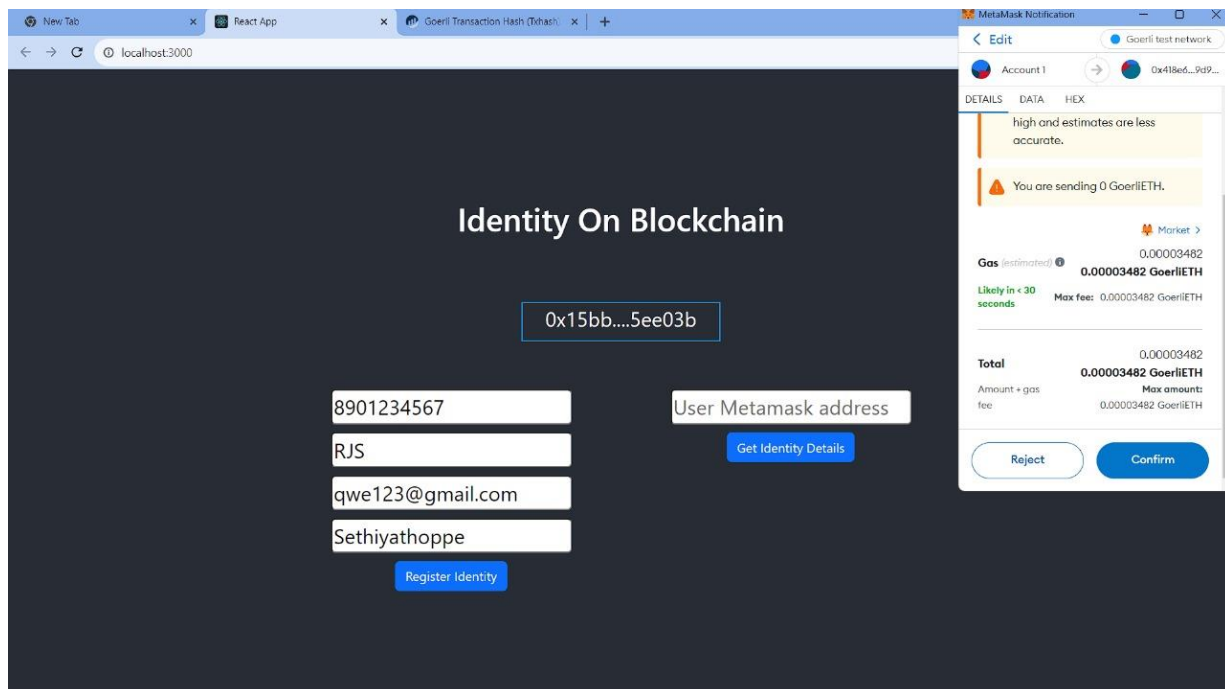
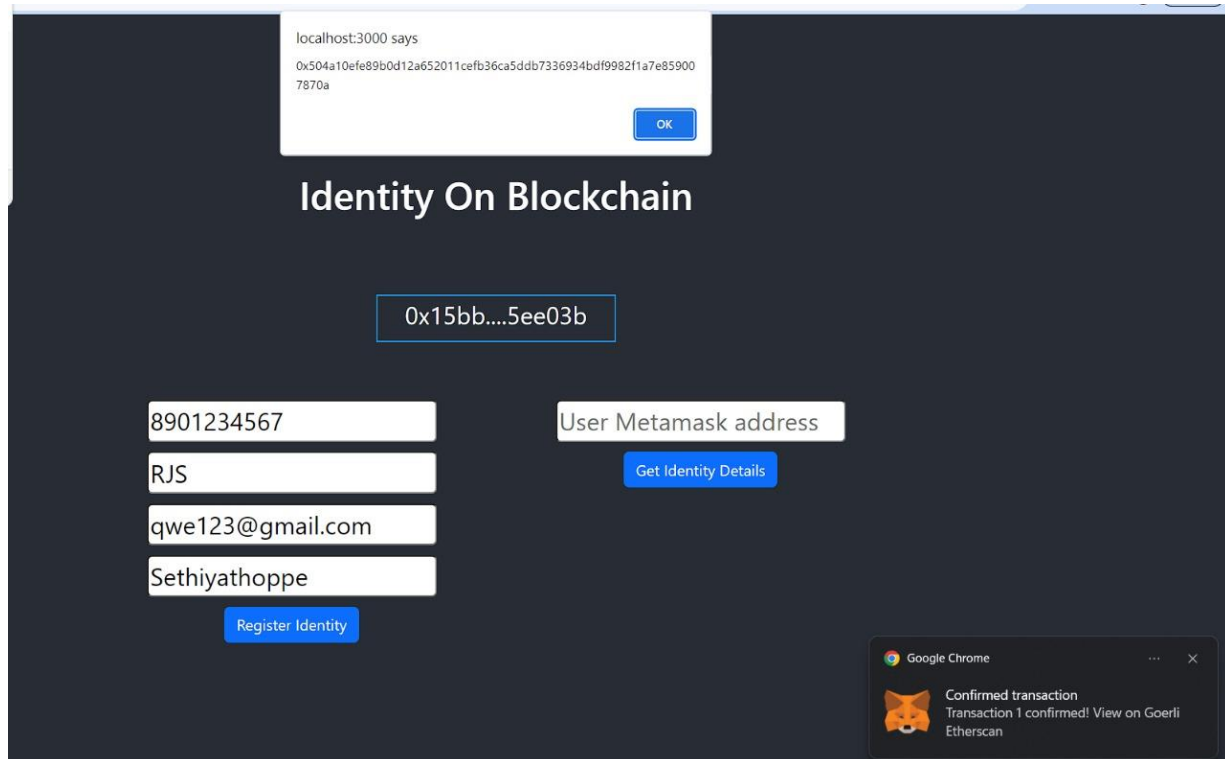


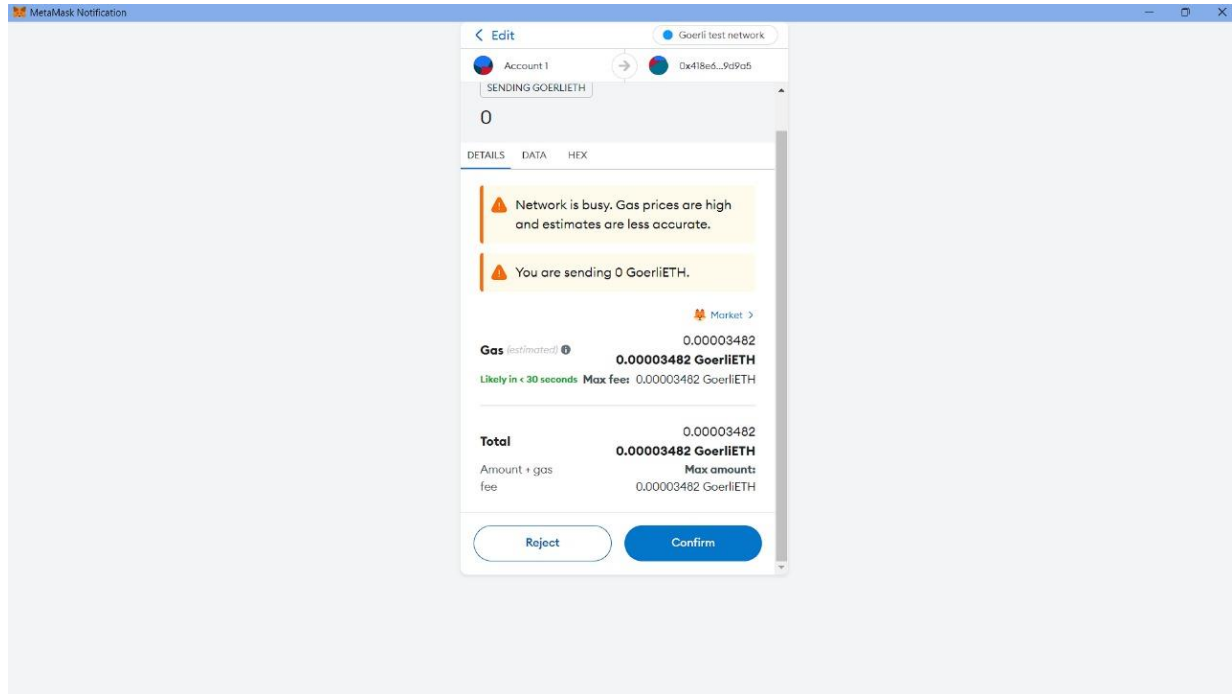
			
5.	{LOCALHOST IP ADDRESS	copy the address and open it to chrome so you can see the front end of your project.	

9. RESULTS

9.1 OUTPUT SCREENSHOTS







10. ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

A blockchain-based real estate management system offers several advantages, revolutionizing the way real estate transactions and property management are conducted. Here are some key advantages:

- Transparency and trust
- Security
- Reduced intermediaries
- Efficiency
- Global Accessibility
- Ownership Transparency
- Immutable Records

DIS ADVANTAGES :

A blockchain-based real estate management system offers several disadvantages, revolutionizing the way real estate transactions and property management are conducted. Here are some key disadvantages:

- Complexity
- Scalability issues
- Energy Consumption
- Market volatility
- Regularity Challenges
- Lack of standardization
- Data privacy
- Used adaption
- Smart contract vulnerabilities

11. CONCLUSION

A blockchain-based smart real estate management system offers a promising solution to address various challenges in the real estate industry. Here are some key conclusions about such a system:

1. **Transparency and Trust**: Blockchain technology enables transparent, tamper-proof records of real estate transactions, making it easier to establish trust among all parties involved, including buyers, sellers, real estate agents, and government authorities.
2. **Reduced Fraud**: By recording property ownership and transaction details on a blockchain, fraudulent activities such as double-selling or fraudulent titles can be significantly reduced, enhancing the overall security of real estate transactions.
3. **Efficiency**: Smart contracts within the blockchain can automate many aspects of real estate transactions, such as property transfers, escrow, and payments. This streamlines the process, reducing paperwork and intermediaries, and potentially decreasing costs.
4. **Accessibility**: Blockchain technology can make real estate investments more accessible to a wider range of people. Fractional ownership, enabled by tokenization of real estate assets, allows investors to buy smaller portions of properties, which is especially appealing to those with limited capital.
5. **Immutable Records**: Property records stored on a blockchain are immutable and resistant to alteration. This means that once information is added, it cannot be easily changed, ensuring a permanent and reliable history of property ownership.
6. **Global Transactions**: Blockchain technology allows for cross-border real estate transactions to occur more smoothly. It can handle different currencies and languages, making it easier for international investors to participate in the market.
7. **Challenges Remain**: Despite its potential, the adoption of blockchain in the real estate industry faces regulatory and technological challenges. Regulations around property rights and ownership need to adapt to accommodate blockchain technology. Scalability and energy consumption concerns are also important to address.
8. **Security Concerns**: While blockchain is generally considered secure, it's not immune to all threats. It's essential to implement robust security measures to protect the blockchain network from hacks and attacks.
9. **Education and Adoption**: Widespread adoption of blockchain in real estate management will require educating stakeholders about the technology and its

benefits. Additionally, industry-wide cooperation and standardization efforts will be crucial to realize the full potential of blockchain.

10. ****Potential for Disruption****: Blockchain-based real estate management has the potential to disrupt traditional real estate processes, leading to a more efficient and accessible real estate market. However, this disruption will take time and require collaboration among various stakeholders.

In conclusion, a blockchain-based smart real estate management system has the potential to revolutionize the industry by increasing transparency, reducing fraud, improving efficiency, and making real estate investments more accessible. However, its success will depend on overcoming regulatory and security challenges, as well as the willingness of the industry to embrace this innovative technology.

12. FUTURE SCOPE

The future scope of blockchain-based smart real estate management is full of potential for transforming the real estate industry in numerous ways. Here are some key aspects of its future scope:

1. ****Tokenization of Real Estate****: Blockchain enables the fractional ownership of real estate through tokens. This means that investors can buy and trade tokens representing a portion of a property, making it easier for individuals to invest in real estate, thereby increasing liquidity in the market.
2. ****Smart Contracts****: Smart contracts can automate and facilitate various aspects of real estate transactions. They can handle everything from property purchases and lease agreements to rent payments and maintenance requests, reducing the need for intermediaries and streamlining the process.
3. ****Transparency and Trust****: Blockchain's transparency and immutability can be used to create a secure and tamper-proof record of property transactions and ownership, reducing the risk of fraud and disputes.
4. ****Cross-Border Transactions****: Blockchain can facilitate international real estate transactions by simplifying the transfer of funds and ensuring that property ownership records are accurate and easily accessible across borders.
5. ****Efficient Property Management****: IoT devices can be integrated into the blockchain network to provide real-time data on property conditions, energy usage, and maintenance needs. Property managers and owners can make more informed decisions to improve efficiency and reduce operational costs.
6. ****Enhanced Due Diligence****: Property history, including ownership records, maintenance history, and inspection reports, can be stored on the blockchain. This can simplify due diligence for buyers and investors.
7. ****Real Estate Financing****: Blockchain can open up new opportunities for real estate financing. DeFi (Decentralized Finance)

platforms can provide real estate loans and investment opportunities, reducing the reliance on traditional banks.

8. ****Sustainability and ESG Compliance****: Blockchain can be used to track and verify sustainability efforts and ESG (Environmental, Social, Governance) compliance in real estate, which is becoming increasingly important for investors and regulators.

9. ****Decentralized Property Listings****: Blockchain can decentralize property listing platforms, giving more control to property owners and reducing fees associated with traditional listing services.

10. ****Cost Reduction****: By automating various real estate processes, blockchain can significantly reduce administrative costs associated with property management, transactions, and compliance.

11. ****Interoperability****: The development of standards and protocols for blockchain-based real estate management can enhance interoperability, ensuring that different systems and platforms can work together seamlessly.

12. ****Improved Security****: Blockchain's cryptographic security features can enhance the protection of sensitive real estate data, reducing the risk of data breaches and cyberattacks.

While the potential for blockchain in real estate is substantial, it's essential to address challenges such as regulatory compliance, scalability, and widespread industry adoption. As blockchain technology continues to mature and gain acceptance, it has the potential to revolutionize how real estate is transacted, managed, and invested in, ultimately making the industry more efficient, transparent, and accessible.

13. APPENDIX

1. Technical Specifications:

Detailed technical specifications including programming languages, frameworks, and libraries used. Database schema diagrams. Blockchain integration details, such as the chosen blockchain platform (Ethereum, Hyperledger, etc.) and smart contract code snippets.

2. User Guides:

Comprehensive user guides for healthcare providers, patients, and administrators, explaining system functionalities, authentication processes, and data management procedures.

Metamask setup instructions for users unfamiliar with blockchain interactions.

3. Code Samples:

Code snippets demonstrating key features like user authentication, data encryption, smart contract interactions, and API integrations. Examples of error handling and edge cases in the codebase.

4. Data Security Measures:

Detailed information about encryption algorithms used for data security. Explanation of access control mechanisms implemented in smart contracts. Protocols and policies ensuring data confidentiality and integrity during transmission and storage.

5. Performance Testing Reports:

Performance testing methodologies, including tools used and test scenarios. Performance test results, including response times, throughput, and system scalability under various loads.

6. Regulatory Compliance Documentation:

Documentation showcasing how the project complies with healthcare data regulations (HIPAA, GDPR, etc.). Details about user consent management and data deletion policies.

7. User Feedback and Improvement Reports:

Summaries of user feedback collected

during usability testing. Reports on system improvements and enhancements made based on user suggestions.

8. Future Enhancements:

Detailed plans for future enhancements, including new features, integrations, and technology upgrades. Roadmap outlining the project's evolution over the next few years.

9. References:

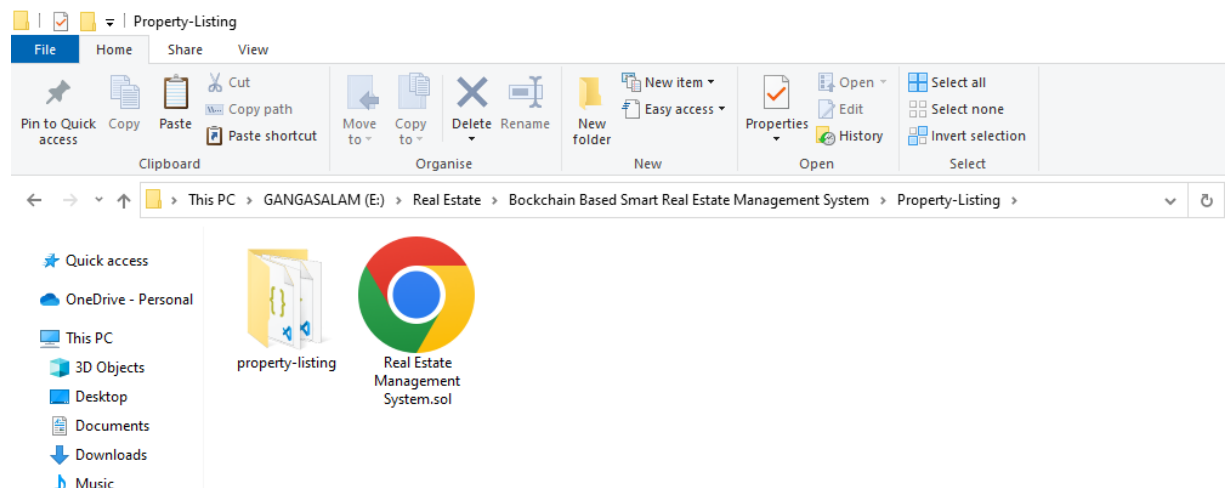
Citations and references for research papers, articles, and resources used during the project development. Links to relevant documentation, libraries, and frameworks.

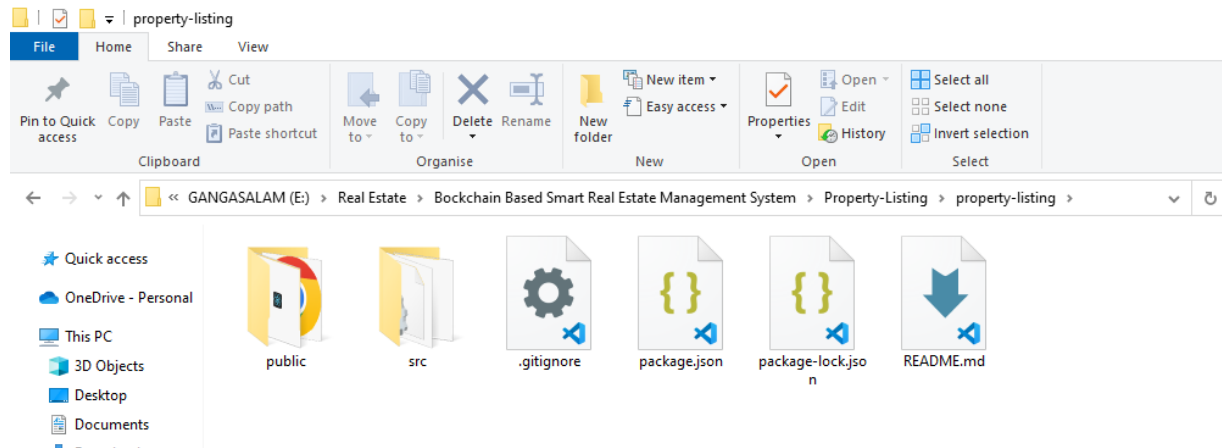
10. Glossary:

Definitions and explanations of technical terms, acronyms, and industry-specific jargon used throughout the project documentation

SOURCE CODE

Folder Structure:





INDEX.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>React App</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.

  To begin the development, run `npm start` or `yarn start`.
  To create a production bundle, use `npm run build` or `yarn build`.
-->
</body>
</html>

```

MANIFEST.JSON

```

{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}

```

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 = "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8"

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>
    </Container>
  </div>
)

```



```

    <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.CSS

.App {

```

    text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }

  to {
    transform: rotate(360deg);
  }
}

```

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;
```

APP.TEST.JS

```

import { render, screen } from '@testing-library/react';
import App from './App';

test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});

```

INDEX.CSS

```

body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
  'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
  sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
  monospace;
}

```

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>
);

```

```

    </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

```

// jest-dom adds custom jest matchers for asserting on DOM nodes.
// allows you to do things like:
// expect(element).toHaveTextContent(/react/i)
// learn more: https://github.com/testing-library/jest-dom
import '@testing-library/jest-dom';

```

PACKAGE.JSON

```

{
  "name": "property-listing",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "bootstrap": "^5.3.1",
    "ethers": "^5.6.6",
    "react": "^18.2.0",
    "react-bootstrap": "^2.8.0",
    "react-dom": "^18.2.0",

```

```

    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}

```

GITHUB& DEMO VIDEO

GitHub Link: <https://github.com/Surya123456789012/Blockchain-Based-Smart-Real-Estate-Management-System>

Demo Video:

https://drive.google.com/file/d/144KvDEUyEv2WF2RYy2voq94DeSRSiQ_T/view?usp=drivesdk