Blockchain-Based Smart Real estate Management System

Submitted By

S.BHUVANASHREE-921020104009

A.DEVISHREE-921020104012

J.T.HARSHINI-921020104019

S.INIYASHREE-921020104022

CONTENTS

1. **INTRODUCTION**
   1.1 Project Overview
   1.2 Purpose
2. **LITERATURE SURVEY**
   2.1 Existing problem
   2.2 References
   2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   3.1 Empathy Map Canvas
   3.2 Ideation & Brainstorming

4.**REQUIREMENT ANALYSIS**

4.1 Functional requirement
4.2 Non-Functional requirements

5.**PROJECT DESIGN**

5.1 Data Flow Diagrams & User Stories
5.2 Solution Architecture

6.**PROJECT PLANNING & SCHEDULING**

6.1 Technical Architecture
6.2 Sprint Planning & Estimation
6.3 Sprint Delivery Schedule

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

7.1 Feature 1
7.2 Feature 2
7.3 Database Schema (if Applicable)

8.**PERFORMANCE TESTING**

8.1 Performace Metrics

9. **RESULTS**

9.1 Output Screenshots

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

INTRODUCTION

1.1 Project Overview

Smart real estate management is an emerging field that leverages advanced technologies to optimize the acquisition, operation, and maintenance of real estate properties. This abstract provides an overview of the key components and benefits of smart real estate management.

Smart real estate management integrates Internet of Things (IoT) devices, data analytics, and artificial intelligence to streamline property management processes. It enables real-time monitoring and control of various aspects, such as energy consumption, security, and maintenance, leading to increased operational efficiency and cost savings.

1.2 Purpose

"In recent years, blockchain technology has been making waves in the real estate industry, offering a new paradigm for property management and transactions. With its transparency, smart contract capabilities, and enhanced security, blockchain systems are poised to revolutionize how we buy, sell, and manage real estate. From reducing fraud to automating transactions and enabling fractional ownership, the advantages of a smart real estate management system on the blockchain are reshaping the way we interact with property assets."

2. LITERATURE SURVEY

2.1 Existing problem

Propy: Propy utilizes blockchain technology for international real estate transactions and offers a platform for buying and selling properties using smart contracts.

Harbor: Harbor specializes in tokenizing real estate assets, enabling fractional ownership and simplified property investment through blockchain technology.

BitProperty: BitProperty is a blockchain-based real estate investment platform, allowing users to invest in property tokens representing real-world assets.

RealT: RealT focuses on tokenizing real estate properties, making them accessible to a broader range of investors through fractional ownership

Atlant: Atlant is a decentralized real estate platform that aims to facilitate property ownership and rental transactions through blockchain.

2.2 Reference

Blockchain Technology in Real Estate:

Mougayar, W. (2016) . The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology. This book provides a foundational understanding of blockchain technology and its potential applications in various industries, including real estate.

2.3 Problem Statement Definition

A "Smart Real Estate Management System" refers to a technologically advanced platform that leverages various digital tools, including the internet of things (IoT), blockchain, data analytics, and automation, to streamline and enhance the management of real estate properties. This system is designed to improve efficiency, transparency, and the overall experience for property owners, tenants, and property managers. It may include features like automated rent collection, predictive maintenance, smart security, transparent property records, and accessibility to property data through digital interfaces. The primary goal is to optimize real estate operations and provide a more intelligent, data-driven, and user-friendly approach to managing properties.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstroming

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Functional Requirement	Description
User Management	- User registration and authentication. - Role-based access control.
Property Listing and Search	- Property listing with filters for location, type, size, and price. - Integration with mapping services.
Lease and Rental Management	- Automated lease agreement generation and management. - Rent collection and payment processing.
Functional Requirement	Description

4.2. Non-Functional Requirement

Non-Functional Requirement	Description
Security	- High-level security measures to protect sensitive data. - Data encryption and access control.
Performance	- Fast response times for property searches and data retrieval. - Scalability to handle growing data and user load.
Usability	- Intuitive user interfaces for property owners, tenants, and managers. - Mobile-friendliness and cross-device compatibility.
Non-Functional Requirement	Description

5. PROJECT DESIGN

5.1 Data Flow Diagram & User Stories

User Stories

User Type	Functional Requirement	User Story Number	User Task	Acceptance Criteria	Priority	Team Member
Real Estate Agent	Property Listing	US001	Create and manage property listings	- Agent can input property details. - Images are uploadable. - Listings can be edited or deleted.	High	Agent Module
Property Owner	Property Listing	US002	List properties with details and set availability dates	- Property details are added. - Availability dates are set. - Listings are live on the platform.	High	Owner Module
Prospective Buyer	Property Search	US003	Search for properties based on criteria	- Search criteria are applied successfully. - Relevant properties are displayed. - User can view property details.	High	Search Module
Renter	Property Filter	US004	Filter rental properties based on features	- Filters are applied successfully. - Matching rental properties are displayed. - User can view property details.	High	Filter Module
Real Estate Agent	Property Viewings	US005	Schedule and manage property viewings	- Agent can schedule viewings. - Clients can book appointments online. - Notifications are sent to both parties.	High	Agent Module
Property Owner	Notifications	US006	Receive notifications of interest	- Owner is notified when someone expresses interest in a property. - Notifications are timely.	Medium	Owner Module
User	Property Details	US007	View detailed property information	- Property details, photos, price, and contact info are displayed. - User can contact the agent or owner.	High	General Module

5.2 Solution Architecture

6. PROJECT DESIGN & SCHEDULING

6.1 Technical Architecture

Layer	Component	Description
Front-End	User Interfaces	Property listings, search, user profiles
Mobile and Web Applications	Accessibility for users	
API Integration	Communication with blockchain and back-end components	
Blockchain	Smart Contracts	Handling property transactions and agreements
Immutable Property Records	Secure and tamper-proof property data on the blockchain	
Back-End	Server or Cloud Infrastructure	Hosting, business logic, data processing
Business Logic	Handling core functionality	
User Authentication and Authorization	User access control	
Database	Supporting data for blockchain interactions	
External Services	Third-Party Integrations	Payment gateways, mapping services, and more
Cloud Storage	Media files storage (images, videos)	
Security	Authentication and Authorization	User access control, data security
Encryption	Data protection	
Threat Protection	Protection against cyber threats	
Infrastructure	Cloud Hosting	Scalability, availability
Load Balancing	Even distribution of incoming blockchain requests	
Caching	Caching Layer	Optimization of blockchain interactions
Messaging Services	Email, SMS, Real-Time Chat	Notifications and communication through the system
Search Engine	Search Engine (e.g., Elasticsearch)	Property search functionality

6.2 Sprint Planning & Estimation

- Sprint planning crucial for defining objectives and setting goals.
- Task breakdown and estimation help allocate work efficiently.
- Agile approach allows for adaptability and incremental development.
- Continuous collaboration with the product owner ensures alignment.
- Regular retrospectives for process improvement.

6.3 Sprint Delivery Schedule

Sprint	Goal	Duration
Sprint 1	Project Kick-off and Planning	2 weeks
Sprint 2	User Management	3 weeks
Sprint 3	Property Listings	3 weeks
Sprint 4	Property Search and Filters	3 weeks
Sprint 5	Property Details and Viewing	2 weeks
Sprint 6	Real-Time Notifications	2 weeks
Sprint 7	Reporting and Analytics	3 weeks
Sprint 8	Admin Panel	3 weeks
Sprint 9	Testing, Optimization, and Documentation	2 weeks
Sprint 10	Final Testing and Deployment	2 weeks

7. CODING & SOLUTIONING

7.1 Feature 1

User Registration and Verification

1. Smart Contract (Solidity):

Develop a Solidity smart contract to manage user registration and verification. Include functions for user registration, email verification, and account activation.

2. Front-end (React):

Create a user registration form in your React app using HTML and CSS. Implement a form submission handler in JavaScript. Connect Metamask to the app to facilitate user registration. Send a transaction to the Ethereum network to register the user on the smart contract.

3. Verification Email (JavaScript and JSON):

Generate a unique verification token and store it in a JSON file or the blockchain. Send a verification email with the token using JavaScript.

4. Account Activation:

Create an activation endpoint in your smart contract. Implement account activation in your React app by interacting with the smart contract. Verify the user's token from the verification email.

7.2 Feature 2

Document Submission and Storage

1. Smart Contract (Solidity):

- Extend your smart contract to allow users to submit documents.
- Include functions to upload documents and record document details

2. Front-end (React):

- Add a document submission form to your React app using HTML Create a file input field and a form to gather document details.
- Implement a file upload handler using JavaScript.
- Use Metamask to send transactions for document submission.

3. Document Storage (Solidity and Ethereum):

- Store document data on the Ethereum blockchain or IPFS.
- Record document references in your smart contract.

4. User Access:

- Implement user authentication to ensure that only authorized users can access their submitted documents.
- Use Metamask for user authentication when accessing documents.

7.3 Database Schema

The database schema for a blockchain-based smart real estate management system comprises several interconnected tables. The "Properties" table stores property details like ID, address, owner, price, and specific descriptors. An "Owners" table contains information on property owners, including their ID, name, contact details, and associated wallet addresses for transactions. The "Transactions" table records transaction specifics such as property ID, buyer, seller, amount, date, and transaction status. A "Contracts" table manages smart contract data for property transactions, outlining terms, parties involved, and unique blockchain addresses. Additionally, a "Blockchain Ledger" table maintains immutable ledger entries, including block IDs, transaction details, hashes, and other blockchain-specific data. These tables are interconnected, ensuring seamless coordination between property ownership, transactions, smart contracts, and blockchain records, establishing a comprehensive system for managing real estate within a blockchain framework.

8. PERFORMANCE TESTING

8.1 Performace Metrics

Performance testing involves the measurement and analysis of various performance metrics to assess a system's behavior and identify areas for improvement. Here are some common performance metrics used in performance testing:

Response Time:

The time taken for a system to respond to a user request. It's a critical metric for assessing user experience and system responsiveness.

Throughput:

The number of transactions or requests processed by the system in a unit of time. It indicates the system's capacity to handle a load and is often measured in requests per second (RPS)

Concurrency:

The number of users or processes interacting with the system simultaneously. Measuring concurrency helps evaluate how well the system manages multiple simultaneous requests.

Error rate:

The percentage of failed transactions or requests during testing. A high error rate may indicate performance issues or system instability.

Resource Utilization:

Monitoring the utilization of system resources like CPU, memory, disk, and network bandwidth. High resource utilization can lead to performance bottlenecks

Server and Network Error:

The numberand thype of server and network errors are encountered during testing

CPU Utilization:

The percentage of CPU capacity used during testing. High CPU utilization can lead to system slowdowns.

Memory Usage:

Monitoring memory consumption helps identify memory leaks and inefficient memory management.

Network Latency:

The delay in data transmission between the client and the server. High network latency can affect response times.

Page Load Time:

Relevant for web applications, this metric measures the time it takes to load a web page, including all its resources (HTML, CSS, images, scripts).

Transaction Response Time:

Specific to database systems, it measures the time it takes for a database transaction to complete.

Hit Rate:

In caching systems, this metric indicates the percentage of requests served from cache, reducing the load on the backend.

Database Query Time:

The time taken to execute database queries, important for applications heavily reliant on database operations.

Transaction Success Rate:

The percentage of successfully completed transactions or operations during testing.

9. RESULTS

9.1.Output and Screenshots:

localhost:3000
Property Listing on Blockchain
0x1d62B...20291
ABC886
Axis land
xxxx city
Land is 500sqfeet
Add Property To Blockchain
AAA886
Get property Detail
Enter Property Id
Enter new Owner Metamask Addr
Transfer property

localhost:3000
Property Listing on Blockchain
0x1d62B...20291
ABC886
Axis land
xxxx city
Land is 500sqfeet
Add Property To Blockchain
AAA886
Get property Detail
ABC886
Axis land
xxx city
Land is 500sqfeet
Enter Property Id
Enter new Owner Metamask Addr
Transfer property

10. ADVANTAGES & DISADVANTAGES

Advantages:

- Efficiency and time savings in real estate operations.
- Enhanced user experience with smart search and user-friendly interfaces.
- Real-time and accurate property information.
- Transparency and trust through blockchain technology.
- Paperless transactions for convenience.

Disadvantages:

- Complexity and technical expertise required.
- High initial implementation costs.
- ·Legal and regulatory complexities.
- Scalability and speed challenges on some blockchains.
- Data privacy and security concerns in a transparent system.

11. CONCLUSION

In conclusion, the Smart Real Estate Management System project using blockchain technology has the potential to transform the real estate industry by enhancing transparency and security. While challenges exist, such as cost and regulatory issues, the system's future looks promising. Successful adoption and adaptation will lead to more efficient and trustworthy real estate transactions.

12. FUTURE SCOPE

The future scope for a Smart Real Estate Management System project in blockchain is promising, with opportunities for wider adoption, global transactions, tokenization of assets, and improved user experiences. We anticipate advancements in data analytics, smart contracts, and enhanced security and privacy. This project holds the potential to revolutionize the real estate industry, making transactions more efficient, secure, and accessible.

13.APPENDIX

Source code
Frontend - VScode

File Edit Selection View Go Run ··· 5_Problem_Statement5_Property_Listing

EXPLORER

5_PROB...
- Property-Listing
  - property-listing
    - public
    - src
      - Page
        - connector.js
        - Home.js
      - App.css
      - App.js
      - App.test.js
      - index.css
      - index.js
      - logo.svg
      - reportWebVitals.js
      - setupTests.js
    - .gitignore
    - package-lock.json
    - package.json
    - README.md
  - Property Listing.sol

{} package.json | # index.css

Property-Listing > property-listing > src > # index.css > body

```
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;
}
```

Edit Selection View Go Run ··· 5_Problem_Statement5_Property_Listing

EXPLORER

5_PROBLEM_STATEMENT5...
- Property-Listing
  - property-listing
    - public
    - src
      - Page
        - connector.js
        - Home.js
      - App.css
      - App.js
      - App.test.js
      - index.css
      - index.js
      - logo.svg
      - reportWebVitals.js
      - setupTests.js
    - .gitignore
    - package-lock.json
    - package.json
    - README.md
  - Property Listing.sol

{} package.json | JS App.js

Property-Listing > property-listing > src > JS 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;
```

Edit Selection View Go Run ··· 5_Problem_Statement5_Property_Listing

EXPLORER ··· {} package.json JS connector.js ×

5_PROBLEM_STATEMENT5...
- Property-Listing
  - property-listing
    - public
    - src
      - Page
        - connector.js
        - Home.js
      - App.css
      - App.js
      - App.test.js
      - index.css
      - index.js
      - logo.svg
      - reportWebVitals.js
      - setupTests.js
    - .gitignore
    - package-lock.json
    - package.json
    - README.md
  - Property Listing.sol

Property-Listing > property-listing > src > Page > connector.js > abi > "inputs" > "type"

```
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",
```

File Edit Selection View Go Run ··· 5_Problem_Statement5_Property_Listing

EXPLORER ··· {} package.json JS reportWebVitals.js ×

5_PROB...
- Property-Listing
  - property-listing
    - public
    - src
      - Page
        - connector.js
        - Home.js
      - App.css
      - App.js
      - App.test.js
      - index.css
      - index.js
      - logo.svg
      - reportWebVitals.js
      - setupTests.js
    - .gitignore
    - package-lock.json
    - package.json
    - README.md
  - Property Listing.sol

Property-Listing > property-listing > src > 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;
```

Remix IDE

Metamask Connection

GitHub & Project Demo Link

Github Repository Link:

https://github.com/Iniyashree-S/nm-smart-real-estate-management

Demo link:

https://drive.google.com/file/d/1-9miKVK-l1LTQ3tqJ0DkayrirfJfRNbG/view?usp=sharing