

**`**

**Section:** FT-B

**Project Report**

**Subject:** Blockchain Technology

**Submitted To**: Sir Hammad Majeed

**Title:** Blockchain For Real Estate Transactions

**SUBMITTED BY:**

Mohsin Ali (22I-9839)

Muhammad Anas (22I-9808)

Mohammad Matin Khan (22I-9842)

Amna Arshad (22I-2284)

### 

Table of Contents

[**Introduction** 3](#_Toc184726241)

[**Problem Statement** 3](#_Toc184726242)

[**Proposed Solution: Blockchain Technology** 4](#_Toc184726243)

[**Technological Components** 4](#_Toc184726244)

[**Limitations of the Project** 5](#_Toc184726245)

[**Rationale for Key Decisions** 5](#_Toc184726246)

[**System Architecture** 6](#_Toc184726247)

[**Implementation Plan** 6](#_Toc184726248)

[**Step 3: Blockchain as a Solution** 7](#_Toc184726249)

[**Step 4: Development Plan** 9](#_Toc184726250)

[**Deployment on SepoliaEth (Etherium Testnet)** 14](#_Toc184726251)

[**Testnet Image** 14](#_Toc184726253)

[**Conclusion** 14](#_Toc184726254)

### **Blockchain-Based Real Estate Management System**

# Introduction

The real estate industry, a cornerstone of global economies, has historically relied on centralized systems to facilitate transactions, manage ownership records, and regulate processes. While this approach has enabled the industry to function at scale, it has also introduced significant challenges such as inefficiency, lack of transparency, and susceptibility to fraud. The involvement of multiple intermediaries like real estate agents, banks, and government entities further complicates the transaction process, increasing costs and delays.

Blockchain technology has emerged as a transformative solution to these challenges. By decentralizing processes and ensuring the immutability of data, blockchain offers a platform where stakeholders can transact with enhanced trust, transparency, and efficiency. This report explores a blockchain-based system tailored for real estate management, detailing its components, the rationale for design decisions, and the challenges encountered during the project.

Additionally, this document discusses key decisions, such as why tokenization was not adopted, why MetaMask was chosen as the primary wallet, and why Phantom blockchain was excluded despite its technical advantages. Limitations of the proposed system are also highlighted to provide a balanced understanding of its feasibility.

# Problem Statement

The real estate sector's reliance on centralized systems poses several challenges:

1. Fraud and Ownership Disputes:
   * Centralized property records are vulnerable to tampering, forgery, and unauthorized alterations. This can lead to disputes and mistrust among stakeholders.
   * For instance, disputes over property boundaries and falsified ownership documents are prevalent in many regions due to the lack of a unified and tamper-proof record-keeping system.
2. **Inefficiencies and Delays**:
   * Transactions typically involve multiple intermediaries, each contributing to delays and increasing the likelihood of errors.
   * For example, escrow services, title verifications, and approvals often take weeks to months, causing frustration and financial strain for parties involved.
3. **High Transaction Costs**:
   * Escrow services, legal documentation, and agent fees significantly increase the cost of real estate transactions.
   * Buyers and sellers incur substantial costs for services that could be automated through technology.
4. **Lack of Transparency**:
   * Limited visibility into transaction progress and unclear ownership records lead to mistrust among buyers, sellers, and regulators.
   * A lack of real-time updates exacerbates disputes, creating inefficiencies in closing deals.

# Proposed Solution: Blockchain Technology

Blockchain addresses these challenges by offering a decentralized, transparent, and immutable system. Key advantages include:

1. **Decentralized Ownership Records**:
   * Property ownership details are stored on a blockchain ledger, making them tamper-proof and accessible to all authorized parties in real time.
2. **Smart Contracts for Automation**:
   * Functions like escrow management, title transfer, and ownership verification are automated using smart contracts, reducing reliance on intermediaries.
3. **Immutable Records**:
   * Blockchain’s immutable nature ensures that transaction history cannot be altered, fostering trust among stakeholders.
4. **Improved Transparency**:
   * Every transaction is recorded on the blockchain, providing full visibility to all participants and reducing the risk of disputes.

# Technological Components

The implementation of the blockchain-based system relies on the following technologies:

##### **1. Blockchain Platform: Ethereum**

Ethereum was selected for its advanced ecosystem and smart contract capabilities. Its key features include:

* **Smart Contracts**: Automated agreements that execute predefined actions, eliminating manual intervention.
* **Developer Ecosystem**: Ethereum’s extensive community offers robust tools and support, ensuring project scalability and reliability.

##### **2. Development Tools**

* **Solidity**: Used to create smart contracts that manage escrow, ownership transfers, and payment handling.
* **Web3.js**: Facilitates real-time interaction between the front end and Ethereum blockchain, enabling seamless transaction monitoring.
* **React.js**: Builds a dynamic and user-friendly interface for stakeholders.

##### **3. Off-Chain Storage: IPFS**

IPFS is used for storing large property-related documents, such as titles and inspection reports, off-chain. Only document hashes are recorded on-chain, ensuring:

* **Data Integrity**: Hashes act as unique fingerprints for documents, verifying their authenticity.
* **Cost Efficiency**: Reducing on-chain data storage minimizes blockchain fees.

# Limitations of the Project

While the proposed system offers significant advantages, certain limitations remain:

1. **Scalability Issues**:
   * Ethereum’s network faces congestion during high transaction volumes, leading to increased gas fees and slower transaction times.
2. **Legal and Regulatory Hurdles**:
   * Blockchain-based solutions must comply with varying real estate regulations across jurisdictions. Adapting to these regulations can be time-consuming.
3. **Cost Constraints**:
   * Gas fees on Ethereum can be prohibitively high during peak usage, affecting the cost-effectiveness of the system.
4. **Adoption Challenges**:
   * Encouraging traditional stakeholders to adopt a blockchain-based system requires extensive education and trust-building efforts.

# Rationale for Key Decisions

##### **1. Why Tokenization Was Not Used**

* **Regulatory Complexity**: Tokenizing real estate assets involves navigating intricate regulatory landscapes, which differ widely by region.
* **Increased Complexity**: Tokenization requires frameworks for fractional ownership and liquidity management, adding layers of complexity to the project.
* **Focus on Simplicity**: The project prioritizes streamlining existing processes rather than introducing new layers of innovation.

##### **2. Why MetaMask Was Selected**

* **User-Friendly Interface**: MetaMask provides an accessible wallet for managing Ethereum transactions.
* **Security Features**: It ensures secure private key storage, reducing the risk of unauthorized access.
* **Integration Capabilities**: MetaMask integrates seamlessly with dApps, enabling smooth user interactions.

##### **3. Why Phantom Blockchain Was Excluded**

* **Ecosystem Maturity**: Ethereum's established ecosystem offers more developer resources compared to Phantom.
* **Community Support**: Ethereum’s active community ensures quick resolution of technical challenges.
* **Trade-Off Analysis**: While Phantom provides lower fees and faster transaction speeds, Ethereum's stability and tools outweigh these benefits for this project.

# System Architecture

The system comprises three key layers:

1. **Blockchain Layer**:
   * Stores ownership records and transaction hashes.
   * Executes smart contracts for escrow, payments, and title transfers.
2. **Off-Chain Storage Layer**:
   * Stores large files like property documents using IPFS.
   * Links document hashes to blockchain records for verification.
3. **Application Layer**:
   * Built with React.js and Web3.js for an intuitive user experience.
   * Provides dashboards for buyers, sellers, and regulators to monitor transactions.

# Implementation Plan

The project follows these phases:

1. **Requirement Gathering**:
   * Analyze existing systems and identify stakeholder needs.
   * Define roles and permissions for accessing data.
2. **Design and Development**:
   * Develop smart contracts and integrate IPFS for storage.
   * Build the front-end interface and connect it to Ethereum via Web3.js.
3. **Testing**:
   * Conduct functional, security, and load testing to ensure system reliability.
   * Validate data integrity through comprehensive blockchain and IPFS tests.
4. **Deployment**:
   * Launch the system with pilot users, gathering feedback for improvements.

# Step 3: Blockchain as a Solution

This step focuses on implementing blockchain technology to address the identified problems:

**3.1 Immutable Ownership Records**

* **Approach:**
  + Develop a decentralized ledger on the Ethereum blockchain to store property ownership records.
  + Implement cryptographic algorithms to protect ownership data from tampering and unauthorized access.
* **Key Actions:**

1. Create a blockchain ledger structure that stores each property transaction as a separate block.

2. Use cryptographic hashing to ensure data integrity and immutability.

3. Develop APIs that allow government registries to verify and update ownership data.

**3.2 Smart Contracts for Automation**

* **Approach:**
  + Design and deploy smart contracts using Solidity to automate real estate transactions.
* **Key Actions:**
  + 1. Develop smart contracts to handle escrow management, ownership verification, and title transfers.
    2. Automate escrow payments by holding funds until predefined conditions (e.g., title transfer) are met.
    3. Integrate a verification mechanism to match the buyer's and seller's details with blockchain records.

**Smart Contract (Solidity)**

This contract handles **escrow payments**, **ownership verification**, and **title transfer**.

**RealEstateEscrow.sol**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract RealEstateEscrow {

// Define buyer, seller, and escrow agent

address public buyer;

address public seller;

address public escrowAgent;

// Property details

uint256 public propertyId;

uint256 public price;

bool public ownershipTransferred;

// Events

event PaymentDeposited(address indexed buyer, uint256 amount);

event OwnershipTransferred(address indexed seller, address indexed buyer);

constructor(

address \_buyer,

address \_seller,

uint256 \_propertyId,

uint256 \_price

) {

buyer = \_buyer;

seller = \_seller;

escrowAgent = msg.sender; // Deployer is escrow agent

propertyId = \_propertyId;

price = \_price;

}

// Function to deposit funds into escrow

function depositPayment() public payable {

require(msg.sender == buyer, "Only buyer can deposit");

require(msg.value == price, "Incorrect payment amount");

emit PaymentDeposited(buyer, msg.value);

}

// Function to transfer ownership once conditions are met

function transferOwnership() public {

require(msg.sender == escrowAgent, "Only escrow agent can release funds");

require(address(this).balance == price, "Payment not yet deposited");

require(!ownershipTransferred, "Ownership already transferred");

// Transfer funds to the seller

payable(seller).transfer(price);

ownershipTransferred = true;

// Emit event

emit OwnershipTransferred(seller, buyer);

}

}

**3.3 Reduced Costs**

* **Approach:**
  + Eliminate intermediaries like escrow agents and title companies through automation.
* **Key Actions:**

1. Implement smart contract logic that eliminates manual verification and escrow management.
2. Use blockchain-based automated processes to reduce administrative and operational costs.

**3.4 Enhanced Transparency**

* **Approach:**
  + Provide a transparent and auditable transaction record for all stakeholders.
* **Key Actions:**
  + 1. Develop a user interface using React.js to display real-time transaction data to buyers and sellers.
    2. Ensure that transaction details and status updates are visible to all stakeholders through a blockchain explorer.

# Step 4: Development Plan

This step outlines the technical roadmap for developing the blockchain-based real estate system.

**4.1 System Design**

* **Approach:**
  + Identify key stakeholders and define roles and permissions.
* **Key Actions:**
  + 1. **Buyers:** View property listings, make offers, and initiate escrow payments.
    2. **Sellers:** List properties, negotiate offers, and transfer ownership titles upon payment.
    3. **Escrow Service (Smart Contract):** Hold and release funds based on predefined conditions.
    4. **Government Registries:** Verify and update public property records with limited access.

**4.2 Smart Contracts**

* **Approach:**
  + Develop core smart contracts to manage interactions between buyers, sellers, and escrow agents.
* **Key Actions:**
  + 1. **Escrow Payments:** Automate holding and release of funds based on contract conditions.
    2. **Ownership Verification:** Match ownership data with blockchain records.
    3. **Title Transfer:** Automate transfer of ownership upon successful payment verification.

**4.3 Data Storage Strategy**

* **Approach:**
  + Implement a hybrid storage model combining on-chain and off-chain storage.
* **Key Actions:**
  + Store critical transaction data (ownership details, escrow status) on-chain for immutability.
  + Use IPFS (InterPlanetary File System) for storing large documents like property titles and inspection reports.
  + Store document hashes on the blockchain to ensure data integrity and authenticity.

**4.4 Development Tools**

* **Approach:**
  + Use a combination of blockchain and web development tools for the platform.
* **Key Actions:**
  + - 1. **Blockchain Platform:** Ethereum for smart contract deployment.
      2. **Development Tools:**
    - Solidity for smart contract development.
    - Web3.js for blockchain interactions.
    - React.js for frontend development.
    - IPFS for decentralized storage of property documents.

**IPFS Integration (JavaScript)**

To upload property documents to IPFS and store the hash on the Ethereum blockchain. We have used the following code:

**ipfsUpload.js**

*import* { create } *from* "ipfs-http-client"**;**

*import* Web3 *from* "web3"**;**

const web3 **=** **new** Web3(Web3**.***givenProvider*)**;**

const ipfs **=** create({

  host**:** "ipfs.infura.io"**,**

  port**:** "5001"**,**

  protocol**:** "https"**,**

})**;**

*export* const uploadToIPFS **=** *async* (**file**) **=>** {

**try** {

    const addedFile **=** **await** ipfs**.**add(file)**;**

    console**.**log("File uploaded to IPFS:"**,** addedFile**.***path*)**;**

**return** addedFile**.***path***;** *// Return the IPFS hash*

  } **catch** (error) {

    console**.**error("Error uploading to IPFS:"**,** error)**;**

  }

}**;**

*export* const storeHashOnBlockchain **=** *async* (**hash,** **contractAddress,** **abi**) **=>** {

  const contract **=** **new** web3**.***eth***.**Contract(abi**,** contractAddress)**;**

  const accounts **=** **await** web3**.***eth***.**getAccounts()**;**

**try** {

**await** contract**.***methods*

**.**storeDocumentHash(1**,** hash)

**.**send({ from**:** accounts[0] })**;**

    console**.**log("IPFS hash stored on blockchain")**;**

  } **catch** (error) {

    console**.**error("Error storing hash on blockchain:"**,** error)**;**

  }

}**;**

**Frontend Interaction (React.js + Web3.js)**

This React component allows users to list properties and initiate escrow payments.

*import* React**,** { useState } *from* "react"**;**

*import* Web3 *from* "web3"**;**

*import* { uploadToIPFS**,** storeHashOnBlockchain } *from* "./ipfsUpload"**;**

const PropertyListing **=** () **=>** {

  const [propertyId**,** setPropertyId] **=** useState("")**;**

  const [price**,** setPrice] **=** useState("")**;**

  const [buyerAddress**,** setBuyerAddress] **=** useState("")**;**

  const [transactionStatus**,** setTransactionStatus] **=** useState("")**;**

  const web3 **=** **new** Web3(Web3**.***givenProvider*)**;**

  const contractAddress **=** "0xYourContractAddress"**;** *// Replace with your contract's address*

  const contractABI **=** [

*/\* Your contract ABI \*/*

  ]**;**

  const depositPayment **=** *async* () **=>** {

    const contract **=** **new** web3**.***eth***.**Contract(contractABI**,** contractAddress)**;**

    const accounts **=** **await** web3**.***eth***.**getAccounts()**;**

**try** {

**await** contract**.***methods***.**depositPayment(1)**.**send({

        from**:** accounts[0]**,**

        value**:** web3**.***utils***.**toWei(price**,** "ether")**,**

      })**;**

      setTransactionStatus("Payment deposited successfully!")**;**

    } **catch** (error) {

      console**.**error("Error depositing payment:"**,** error)**;**

      setTransactionStatus("Error occurred while depositing payment.")**;**

    }

  }**;**

  const handleIPFSUpload **=** *async* (**event**) **=>** {

    event**.**preventDefault()**;**

    const file **=** event**.***target***.***files*[0]**;**

**if** (file) {

      const ipfsHash **=** **await** uploadToIPFS(file)**;**

**if** (ipfsHash) {

        setTransactionStatus(`File uploaded successfully: ${ipfsHash}`)**;**

**await** storeHashOnBlockchain(ipfsHash**,** contractAddress**,** contractABI)**;**

      }

    }

  }**;**

**return** (

    <div>

      <h2>Property Listing</h2>

      <label>Property ID:</label>

      <input

*type***=**"text"

*value***=**{propertyId}

*onChange***=**{(**e**) **=>** setPropertyId(e**.***target***.***value*)}

      />

      <label>Price (ETH):</label>

      <input

*type***=**"text"

*value***=**{price}

*onChange***=**{(**e**) **=>** setPrice(e**.***target***.***value*)}

      />

      <label>Buyer Address:</label>

      <input

*type***=**"text"

*value***=**{buyerAddress}

*onChange***=**{(**e**) **=>** setBuyerAddress(e**.***target***.***value*)}

      />

      <label>Upload Property Document:</label>

      <input *type***=**"file" *onChange***=**{handleIPFSUpload} />

      <button *onClick***=**{depositPayment}>Deposit Payment</button>

      {transactionStatus **&&** (

        <div>

          <p>{transactionStatus}</p>

        </div>

      )}

    </div>

  )**;**

}**;**

*export* **default** PropertyListing**;**

# ****Deployment on SepoliaEth (Etherium Testnet)****

# **The Following image shows the testing of Real Estate smart contract implemenation on Etherium Testnet named SepoliaEth. The project is working as one of our Group members attached his own personal Etherium address to validate as shown in the following image:**

# 

# ****Conclusion****

This blockchain-based real estate management system represents a significant step toward modernizing the industry. By automating processes, reducing costs, and improving transparency, the platform addresses the most pressing challenges in real estate transactions. Although limitations exist, the strategic use of Ethereum, MetaMask, IPFS and SepoliaEth Testnet ensures the project’s feasibility and long-term impact. We have attached the whole video of the project.

As blockchain adoption grows, this project has the potential to set a benchmark for innovation, streamlining real estate processes and fostering trust among all stakeholders.