Blockchain-Based Land Registration System

FINAL YEAR PROJECT PRESENTATION
APRIL 19, 2024

SUPERVISOR: PROFESSOR YIU SIU MING

GROUP MEMBERS:
1. GOLI, SMARAN
2. AGARWAL, RAHUL
3. BHATIA, DIVTEJ SINGH
4. RAHMAN, MOHAMMAD ABDUR
Introduction

2. Our Solution

3. Methodology

4. Demo

5. Considerations

6. Challenges

7. Future Prospects

8. Conclusion
PROBLEM DEFINITION

Existing Land Registration System Challenges in Hong Kong

01 Legal uncertainties and complexities in property transactions
02 Inconvenience and Higher costs
03 Difficulty in secure storage of paper deeds

Opportunities with Blockchain Technology

01 Reliable, decentralized, transparent and secure land registration system.
02 Adoption of smart contracts on public blockchain platforms
03 Potential for enhancing security and maintenance of land data
PROJECT OBJECTIVES

1. Establish a Blockchain-based land registry
2. Automate property transactions
3. Ensure data security and privacy

INTRO → SOLUTION → METHODOLOGY → LIVE DEMO → CONSIDERATIONS → CHALLENGES → CONCLUSION
PROJECT DELIVERABLES

01 Front-End Application
- Buyer dashboard
- Land Inspector dashboard
- Seller dashboard

02 Smart Contracts
- Land Registration Contract

03 Documentation
- Component
- Smart Contract
- Code

INTRO → SOLUTION → METHODOLOGY → LIVE DEMO → CONSIDERATIONS → CHALLENGES → CONCLUSION
WHY WILL IT WORK?

Expanded Market Participation and Liquidity
Digitizing land registration on blockchain broadens market access and streamlines transactions, simplifying investment entry and boosting liquidity.

Operational Efficiency
Smart contracts automate processes, reduce manual errors and intermediaries, speeding up transactions and cutting costs while real-time updates enhance decision-making.

Enhanced Security and Compliance
Blockchain’s immutable ledger and decentralized validation secure records and reduce fraud, while smart contracts ensure regulatory compliance by embedding legal and procedural rules directly into transactions.
Blockchain Platform and Smart Contracts:
- Ethereum: Offers advanced smart contracts and strong developer support for secure, automated transactions.
- Solidity: Primary language for reliable and efficient smart contracts on Ethereum.

Development Environment:
- Ganache: A personal blockchain for Ethereum development that allows for easy testing and simulation of blockchain conditions locally.

Frontend and User Interface:
- React: JavaScript library selected for responsive and flexible UI development, optimized for blockchain interactions.

Backend and API Integration:
- Node.js: Chosen for its event-driven architecture, ideal for efficient real-time applications on distributed systems.
1st Blockchain Based Land Registration System in Hong Kong

Decentralization and Trust

Role of Government and Stakeholder Collaboration

Enhanced Security and Immutable Data Storage

Efficient Transaction Handling and Validation
Blockchain-Based Land Registration System

System Architectural Design
Defines the structure, behavior, and more views of a system.

Smart Contract Design
Defines the participants, functions, requirements and constraints of smart contract execution.

Security Design
Defines the security rules associated with transactions. Answers some security questions related.
SYSTEM ARCHITECTURAL DESIGN

- **Decentralized Land Registry:** The system utilizes blockchain technology to create a decentralized and tamper-proof land registry. This ensures the reliability of property ownership records.

- **Buyer Dashboard:** This user interface allows potential buyers to view available lands for purchase.

- **Seller Dashboard:** This user interface enables sellers to post lands they are willing to sell.

- **Land Inspector Dashboard:** This user interface allows a land inspector, which is a government entity, to verify lands and authorize sales.
- **Truffle**: Development environment, testing framework, and asset pipeline for Ethereum.
  - Used to develop and deploy smart contracts for the land registration system.
  - Helps in coding, compiling, and testing the smart contracts.

- **Ganache**:
  - Personal blockchain for Ethereum development.
  - Provides a local blockchain environment for testing and simulating smart contract execution.
  - Allows developers to test smart contracts before deploying them to a “test” Ethereum network.
A Smart Contract consists of several key components:

- **Participants**: These are the entities that interact with the contract.
- **State**: This is the current status of the contract.
- **Functions**: These are the operations that the contract can perform.
- **Rules**: These are the conditions that govern how the contract operates.
SMART CONTRACT DESIGN

The 4 Main Use Cases:

**REGISTER**
For Land Registration

**TRANSFER**
Land Transfer

**VERIFY / VIEW**
To verify the authenticity of records

**SELL**
For Property Sales
How Different Functions and Events in our system interact.

Add land flow:
1. `addLand()`: by seller
2. `approveLand()`: by land inspector

Sale flow:
1. `requestLand()`: by buyer
2. `approveRequest()`: by seller
3. `approveSale()`: by land inspector
4. `payment()`: by buyer
1. **addLand()** allows an address to register a new parcel of land with its location and a unique parcel ID. Emits a LandAdded event when land is registered.
2. `approveLand()` allows the land inspector to property to be added to the blockchain network.

A LandApproved event is emitted when land is approved.

```solidity
// Function for the land inspector to approve land
function approveLand(uint _landId) public onlyLandInspector {
    require(!landApprovalStatus[_landId], "Land has already been approved");
    landApprovalStatus[_landId] = true;
    // RequestStatus[landsCount] = false;
    // RequestedLands[landsCount] = false;

    // Emit an event that land has been approved
    emit LandApproved(_landId);
}
```
3. **requestLand()**: This function is used when a potential buyer requests a property from a seller.

This emits a Landrequested event to be triggered.

```solidity
function requestLand(address _sellerId, uint _landId) public {
    require(isBuyer(msg.sender));

    requestsCount++;
    RequestsMapping[requestsCount] = LandRequest(
        requestsCount,
        _sellerId,
        msg.sender,
        _landId
    );
    RequestStatus[requestsCount] = false;
    RequestedLands[requestsCount] = true;

    emit Landrequested(_sellerId);
}
```
4. **approveRequest()**: Seller approves the request to sell his property to buyer who has requested land.
   - State of RequestStatus changes from ‘false‘ to ‘true‘

```solidity
function approveRequest(uint _reqId) public {
    require(isSeller(msg.sender));
    RequestStatus[_reqId] = true;
}
```
5. **approveSale()**: This function is used when a LandInspector approves the sale of land before property is transferred from seller to buyer.
- Emits a SaleApproved event.

```solidity
// Function for the land inspector to approve the sale
function approveSale(uint _reqId) public onlyLandInspector {
    require(
        RequestedLands[RequestsMapping[_reqId].landId],
        "Land has not been requested"
    );
    require(!saleApprovalStatus[_reqId], "Sale has already been approved");

    RequestStatus[_reqId] = true;
    saleApprovalStatus[_reqId] = true; // Mark sale as approved by the land inspector

    // Emit an event that sale has been approved
    emit SaleApproved(
        RequestsMapping[_reqId].landId,
        RequestsMapping[_reqId].sellerId,
        RequestsMapping[_reqId].buyerId
    );
}
```
6. payment(): Ensures payment is completed by buyer to seller before the asset is transferred

```solidity
function payment(address payable _seller, uint _landId) public payable {
    // Transfer the land price to the seller
    _seller.transfer(msg.value);

    // Update the land ownership
    LandOwner[_landId] = msg.sender;

    // Mark the payment as received
    PaymentReceived[_landId] = true;

    // Additional logic can be added here if needed, like updating the land registry
    // to indicate the land is no longer available for sale, etc.

    // Emit an event that payment has been made
    emit PaymentMade(_landId, _seller, msg.sender);

    // Emit an event, if you have one for successful payment/transfer
    emit LandOwnershipTransferred(_landId, _seller, msg.sender);
}
```
Step 1: Set up the Smart Contracts

Solidity Smart Contract:
- The project utilizes a Solidity smart contract called "Land.sol"
- This contract is responsible for handling the transfer of property and maintaining a decentralized and tamper-proof land registry.
Step 2: Set up the Truffle Suite - Ganache

Setup some test accounts with initial ETH balance. Gas fee is a commonly used term for the cost that certain blockchain protocol users pay to network validators each time they wish to perform a state change in the contract.
Blockchain-Based Land Registration System

SUPERVISOR: PROFESSOR YIU SIU MING

GROUP MEMBERS:
1. GOLI, SMARAN
2. AGARWAL, RAHUL
3. BHATIA, DIVTEJ SINGH
4. RAHMAN, MOHAMMAD ABDUR

ASK FOR LIVE DEMONSTRATION
SECURITY DESIGN

Key Questions:

1. How to ensure each property is added only once?
Key Questions:

2. How to make sure of user identity?

- HKID Verification with OTP
- Facial ID Verification with HKID similar to the one in bank account opening / using HKSAR Govt. Services
SECURITY DESIGN

Key Questions:

3. How to make sure payment is transferred?

1. Verify that the contract has sufficient balance to cover the payment amount.
2. We use a robust transfer method like send or call.value to handle potential transfer failures.
3. Handle any errors or exceptions that may occur during the transfer process.
4. Thoroughly test the payment functionality in different scenarios and edge cases.
5. Validated the contract's behavior on a testnet or in a local development environment.
SECURITY DESIGN

Key Questions:

4. Can you sell to yourself.

Simple Answer = No

Checks in-place to make sure the transaction happens from a unique user to a different unique user.
**TESTING AND FEEDBACK**

**TESTING STRATEGIES EMPLOYED**

- **Stress Testing**
  Tested system performance under varied load conditions to confirm stability and scalability.

- **Usability Testing**
  Engaged peers in testing the interface, collecting insights to improve user interaction and design.

**FEEDBACK MECHANISMS USED**

- **Feedback Sessions**
  Organized casual feedback sessions with friends and fellow students to gain deeper insights and discuss potential enhancements.

- **Regular Update Meetings**
  Met regularly with our project group to review feedback and decide on necessary adjustments.
"Collaborative efforts from all stakeholders are crucial for evolving our land registration system into a modern, efficient, and transparent platform."
IN LAND REGISTRY

Centralized Systems

Pros:
- Central control and oversight.
- Simplified governance structure.

Cons:
- Higher risk of fraud and data tampering.
- Single point of failure can lead to system-wide disruptions.

Decentralized Systems (Blockchain)

Pros:
- Improved security
- Greater transparency and reduced fraud risk.
- Quicker transactions by removing intermediaries.

Cons:
- Requires more technical infrastructure and understanding.
- Potential scalability issues as the network grows.
REAL WORLD IMPLICATIONS

IS IT FEASIBLE IN THE REAL WORLD?

- Scalability and Network Infrastructure
  - Efficiently handles high volume
  - Robust and reliable infrastructure
  - Readily accessible platform

- Technical Compatibility
  - Smooth transition with existing systems
  - Compatibility with diverse applications

- Regulatory Compliance
  - Full compliance with data protection and privacy regulations:
  - Robust security measures
CHALLENGES AND LESSONS LEARNED

CHALLENGE FACED

Technical Complexity:
Mastering blockchain technology was initially daunting due to its novelty and the depth of technical knowledge required.

LESSON LEARNED

Embrace Continuous Learning:
The fast-evolving nature of blockchain technology taught us the importance of staying updated and adaptable.
Regulatory Uncertainty:
Adapting to evolving legal frameworks that govern digital assets and blockchain technology proved challenging.

Proactive Regulatory Alignment:
We aligned closely with existing regulatory frameworks early on, enabling us to anticipate compliance challenges and adapt our development process.
CHALLENGES AND LESSONS LEARNED

CHALLENGE FACED

Technical Interoperability:
Ensuring that our blockchain solution could interact seamlessly with various other technology systems and platforms.

LESSON LEARNED

Emphasizing Technical Flexibility:
We learned the importance of designing systems with adaptability in mind, allowing for easier integration with other technologies and future upgrades.
Introduction

- Representing real-world assets, such as properties, as digital tokens on the blockchain.

Benefits of Tokenization

- Enhanced Liquidity:
- Accessibility:
- Partial Ownership of Real Estate.
CONCLUSION

Next Steps
Further Testing and Real World Implementation

Blockchain Revolution
Property Transaction Transformation

Achievements
Ready to launch MVP

Future Outlook
Total Asset Digitalization
QUESTIONS?
Blockchain-Based Land Registration System

FINAL YEAR PROJECT PRESENTATION
APRIL 19, 2024

SUPERVISOR: PROFESSOR YIU SIU MING

GROUP MEMBERS:
1. GOLI, SMARAN
2. AGARWAL, RAHUL
3. BHATIA, DIVTEJ SINGH
4. RAHMAN, MOHAMMAD ABDUR

THE END
**PROJECT SCHEDULE - SUPPLEMENTARY**

**PHASE 1**
- **SEPT-OCT**
  - **Phase 1 Deliverables**
  - Project Plan (October 1, 2023)

**Phase 2 Deliverables**
- First Presentation (January 8-12th 2024)
- Interim Project Report (January 21st)

**PHASE 2**
- **NOV-JAN**
  - **Key Milestones**
  - Back testing and optimisations
  - Integrating front-end & back-end
  - Final documentation

**PHASE 3**
- **FEB-MAR**
  - **Key Milestones**
  - Back testing and optimisations
  - Integrating front-end & back-end
  - Final documentation

**Phase 4 Deliverables**
- Final Project Report
- Final Project Presentation (Due date: April 15-19th 2023)

**PHASE 4**
- **APR**

**LEAVE THIS SPACE FOR CONTENT INDICATORS, TBA LATER....**
WORK DISTRIBUTION - SUPPLEMENTARY

RAHMAN, Mohammad Abdur
- Smart Contract
- Frontend

GOLI, Smaran
- Web3 Integration
- Documentation

BHATIA, Divtej Singh
- Smart Contract
- Logic Implementation

AGARWAL, Rahul
- Frontend
- Web3 Integration