CSE-586

Project 3 on Blockchain

Phase 2 Report, Use case Explanation, Architecture and Pseudo Code

Name: Rajiv Ranjan UbId: 50249099

BlockChain in Real Estate

Fraud is high in land- and property-related transactions. This is often because even if someone claims to be the rightful owner of a piece of land, that person might have been duped by another seller who sold the property with fake documents. Tracking the ownership of a property is crucial in land deals. Let's consider this scenario where a buyer approaches a seller to purchase a piece of land. The buyer asks the seller to show the details of legal ownership of the property. Also, the buyer wants to know the historical data that lists the ownership of the land for the past 100 years. Traceability is something justly registered in the Ethereum blockchain storage. The beauty here is data once stored can't be modified or deleted, which gives the blockchain the edge over any traditional database. Let's find out how to trace the previously recorded ownership from the Ethereum blockchain. You will need two different addresses, one for the buyer and one for the seller. Then you will need a struct for the property details and will need a list of such properties.

Land registry is a special type of use case suitable to being implemented on Ethereum. Land registration data should be publicly available, traceable, secure, and nonmodifiable. Privacy, performance, and scalability are not core features needed in land registry. Hence, Ethereum is the right kind of public blockchain implementation that will address most land registry requirements.

Implementation Details

I have created a contract which is a basic land registry program where Only the contract initiator can assign properties to people may be after some document verification. LandRegistryAdmin is the user who will initiate the contract. Property ownership is a historical data that has a seller, an owner/buyer and a date of transaction. PropertySold is an event that informs on success or failure of a transaction. Land registration authority may alter ownership to any of the existing properties. If the propertyId does not exist in record, hence create a new transaction and add to registered properties.

Architecture

Fig 1. Block Diagram

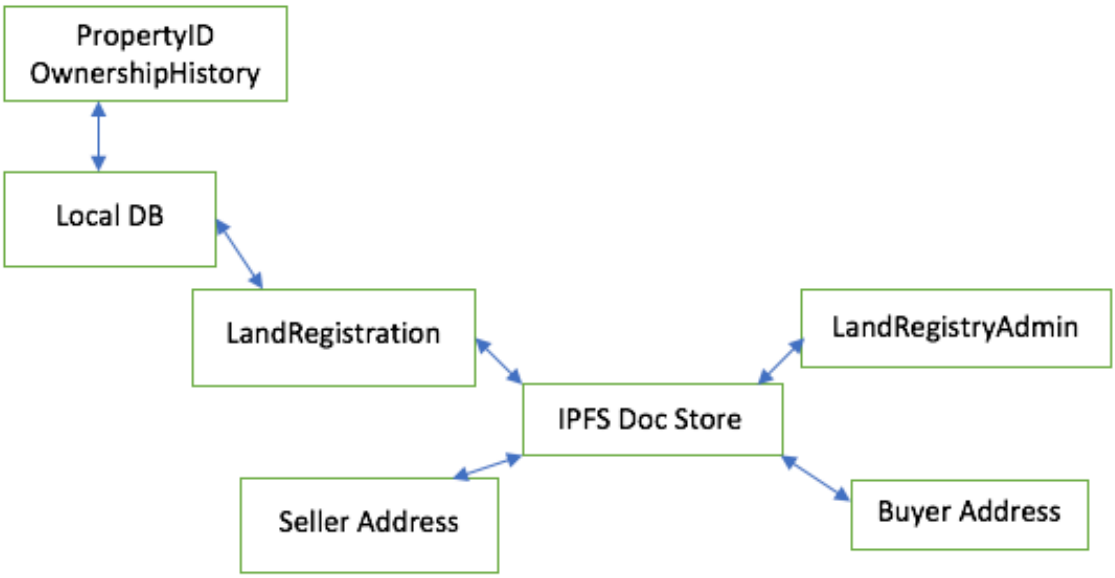
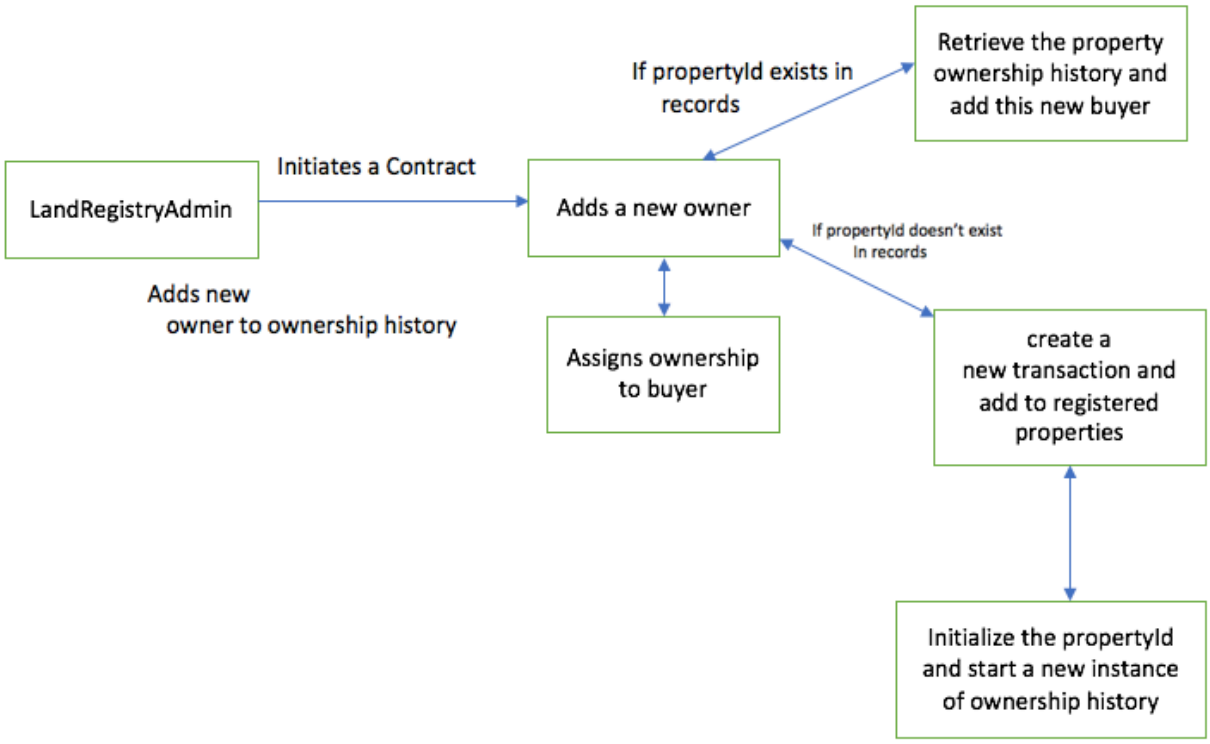


Fig 2. Functional Diagram



Code(pseudocode)

```
contract PropertyTransaction {
Property[] properties;
address landRegistryAdmin;

struct Property{
uint propertyId;
bytes32[] ownershipHistory;
}

event PropertySold(uint propertyId, string ownership, bool flag, string message);

constructor() public {
landRegistryAdmin = msg.sender;
Property memory property0 = Property(0, new bytes32[](0));
properties.push(property0);
properties[properties.length-1].ownershipHistory.
push("Buyer:b0, Seller:s0, DOT:dt0");
Property memory property1 = Property(1, new bytes32[](0));
properties.push(property1);
properties[properties.length-1].ownershipHistory.
push("Buyer:b1, Seller:s1, DOT:dt1");
```

```
Property memory property2 = Property(2, new bytes32[](0));
properties.push(property2);
properties[properties.length-1].ownershipHistory.
push("Buyer:b2, Seller:s2, DOT:dt2");
}

function addNewOwner(uint propertyId, bytes32 ownership)
public {
if (msg.sender != landRegistryAdmin) {
emit PropertySold(propertyId,
bytes32ToString(ownership),
false,
'Only land registry department
can assign ownership to buyer');
}

for (uint i = 0; i < properties.length; i++) {
if (properties[i].propertyId == propertyId) {
properties[i].ownershipHistory.push(ownership);
emit PropertySold(propertyId,
bytes32ToString(ownership),
true,
'New ownership added to existing
property');
break;
}
}


Property memory property = Property(properties.length,
new bytes32[](0));
properties.push(property);
properties[properties.length-1].ownershipHistory.
push(ownership);
emit PropertySold(propertyId,
bytes32ToString(ownership),
true,
'New Property added');
}

function retrievePropertyHistory(uint propertyId) public
view returns(bytes32[]){
for (uint i = 0; i < properties.length; i++) {
if (properties[i].propertyId == propertyId) {
return properties[i].ownershipHistory;
```

```
}
}
}
function bytes32ToString(bytes32 x) private pure returns
(string) {
bytes memory bytesString = new bytes(32);
uint charCount = 0;
for (uint j = 0; j < 32; j++) {
byte char = byte(bytes32(uint(x) * 2 ** (8 * j)));
if (char != 0) {
bytesString[charCount] = char;
charCount++;
}
}
bytes memory bytesStringTrimmed = new bytes(charCount);
for (j = 0; j < charCount; j++) {
bytesStringTrimmed[j] = bytesString[j];
}
return string(bytesStringTrimmed);
}}
```