Real Estate Contract in Blockchain

Presented by The Litecoins :

Patricia , Nirvana, Stephan

Objective

The objective is to create a set of smart contracts that enable various functionalities related to real estate management, ownership, rental agreements and crowdfunding for real estate projects.

Can blockachian have an affect on the real estate industry?

- ***Efficiency in Transactions:*** Makes transaction more efficient by automating financial payments such as rent, fund collection and ownership representation.

- ***Blockchain Technology:*** Blockchain ensures a more secure, transparent and decentralized transactions. These smart contracts operate on a blockchain, ensuring secure ownership tracking, transparent rental agreements, and verifiable crowdfunding.

- ***Digital Payments and Transactions:***The Rental Agreement contract, for instance, facilitates digital rent payments, aligning with fintech's focus on digital financial services.

- ***Innovation in Real Estate Financing:*** The Crowdfunding contract mirrors fintech's approach to alternative financing. By enabling multiple contributors to fund a real estate project, it demonstrates innovation in real estate financing, a domain where fintech continually seeks new methods for investment and capital raising.

Smart Contracts

- **The *RealEstateToken* contract** is for creating and managing a digital token related to real estate.

- **The *RentalAgreement* contract** handles the financial transactions and terms of a property rental.

- **The *Crowdfunding contract*** is for raising funds for a project or goal, ensuring that the money can only be withdrawn once the target amount is reached.

RealEstate Token Contract

- **Description**: Creating digital tokens representing real estate ownership or value.
- **Function:** The contract allows minting new tokens and distributing them to represent real estate assets.
- **Key Features**: Enables ownership representation in a digital format.

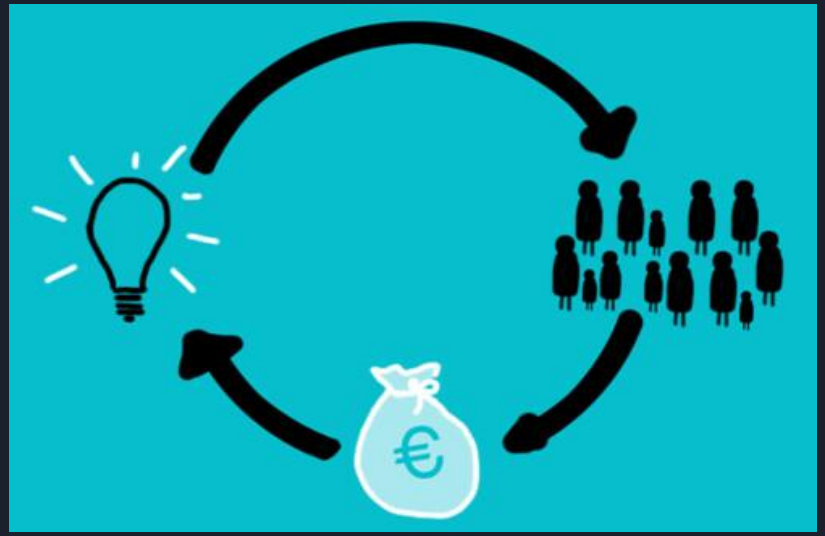
```
pragma solidity ^0.5.0;
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Detailed.s
contract RealEstateToken is ERC20, ERC20Detailed {
    address public admin;
    mapping(uint => address) public propertyOwners;
    mapping(uint => uint256) public propertyValues;
    uint public nextPropertyId;
    modifier onlyAdmin() {
        require(msg.sender == admin, "Only admin can perform this action");
        _;
    }
    constructor(string memory tok) ERC20Detailed("RealEstateToken", tok, 18) public {
        admin = msg.sender;
    }
    function createProperty(uint256 _value) external onlyAdmin {
        propertyOwners[nextPropertyId] = msg.sender;
        propertyValues[nextPropertyId] = _value;
        nextPropertyId++;
    }
    function mint(uint propertyId, address to, uint256 amount) external onlyAdmin {
        require(propertyOwners[propertyId] == msg.sender, "Only property owner can mint tokens");
        _mint(to, amount);
    }
}
```

Real Estate Agreement Contract

- ***Description:*** A digital rental agreement between landlords and tenants.
- ***Function:*** Records lease terms, rent amounts, and security deposits digitally.
- *Key Features:* Facilitates digital rent payments and ensures compliance during the lease period.

```
contract RentalAgreement {
    struct Agreement {
        address payable landlord;
        address payable tenant;
        uint rent;
        uint securityDeposit;
        uint leaseStart;
        uint leaseEnd;
        bool isActive;
    }
    Agreement[] public agreements;
    function createAgreement(
        address payable _tenant,
        uint _rent,
        uint _securityDeposit,
        uint _leaseStart,
        uint _leaseEnd
    ) public {
        Agreement memory newAgreement = Agreement({
            landlord: msg.sender,
            tenant: _tenant,
            rent: _rent,
            securityDeposit: _securityDeposit,
            leaseStart: _leaseStart,
            leaseEnd: _leaseEnd,
            isActive: true
        });
        agreements.push(newAgreement);
    }
    function payRent(uint agreementId) external payable {
        Agreement storage agreement = agreements[agreementId];
        require(msg.sender == agreement.tenant, "Only tenant can pay rent");
        require(block.timestamp > agreement.leaseStart && block.timestamp < agreement.leaseEnd, "Lease not active");
        require(msg.value == agreement.rent, "Incorrect rent amount");
        agreement.landlord.transfer(msg.value);
    }
}
```

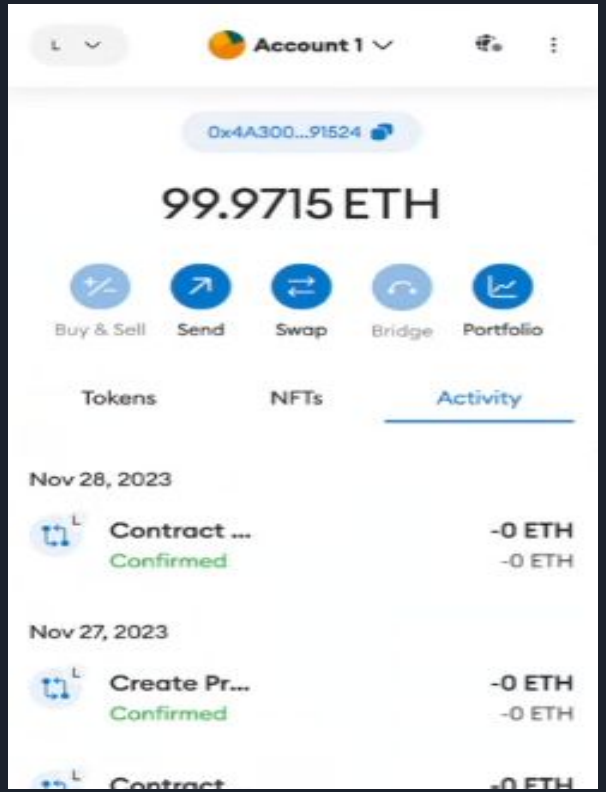
Crowdfunding Contract

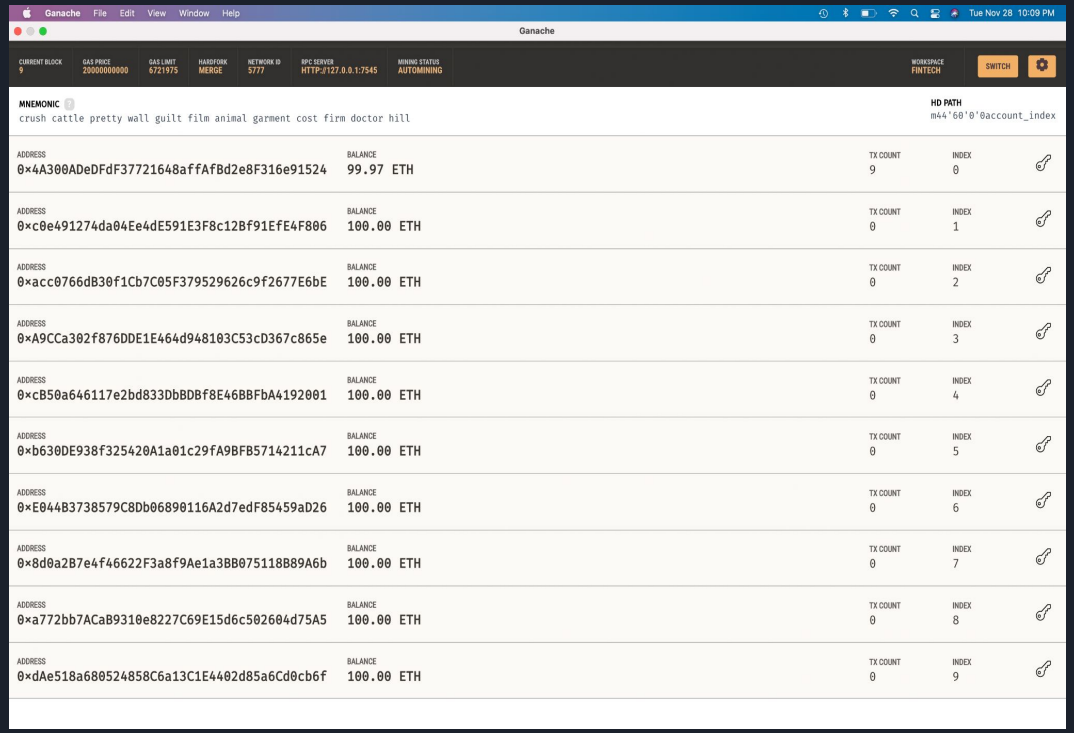


- ***Description:*** Online fundraising platform through smart contracts.
- ***Function:*** Collects contributions for specific goals or projects.
- ***Key Features:*** Stops accepting contributions upon reaching the funding goal, enabling withdrawal thereafter.

```
contract RealEstateCrowdfunding {
    struct Project {
        uint id;
        uint goal;
        uint raisedAmount;
        uint end;
        bool isFunded;
    }
    RealEstateToken public token;
    address public admin;
    Project[] public projects;
    mapping(uint => mapping(address => uint)) public contributions;
    constructor(address _token) public {
        token = RealEstateToken(_token);
        admin = msg.sender;
    }
    function createProject(uint _goal, uint _duration) public {
        uint projectId = projects.length;
        projects.push(Project({
            id: projectId,
            goal: _goal,
            raisedAmount: 0,
            end: block.timestamp + _duration,
            isFunded: false
        }));
    }
    function contribute(uint projectId) external payable {
        Project storage project = projects[projectId];
        require(block.timestamp < project.end, "Crowdfunding ended");
        contributions[projectId][msg.sender] += msg.value;
        project.raisedAmount += msg.value;
    }
    function withdraw(uint projectId) external {
        Project storage project = projects[projectId];
        require(msg.sender == admin, "Only admin can withdraw");
        require(block.timestamp >= project.end, "Crowdfunding not ended");
        require(project.raisedAmount >= project.goal, "Goal not reached");
        address(uint160(admin)).transfer(address(this).balance);
    }
}
```

Metamask & Ganache





Problems Encountered

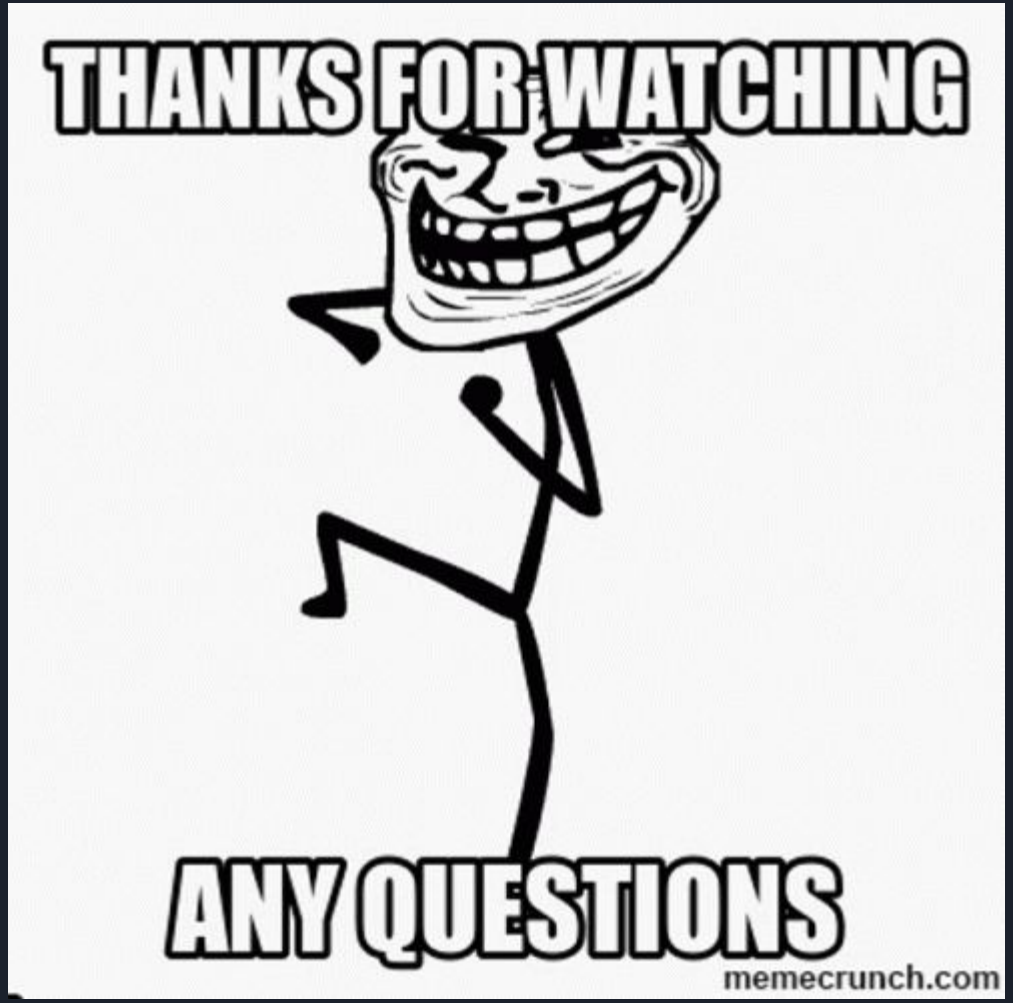
- Importing the libraries
- Creating function
- Deployment of the token
- Issues running the correct pragma version.

Potential Future Ideas

- Environmental Impact Tracking
- Smart Lease Agreements
- AI-Powered Property Valuation

Conclusions

- That smart contracts will enhance the process of buying and selling real estate properties by making transaction more transparent.
- It will also gives more authority and power to the individuals who are making the transactions
- Finally, incorporating smart contracts and blockchain technology will the development of the real estate industry.


THANKS FOR WATCHING
ANY QUESTIONS
memecrunch.com