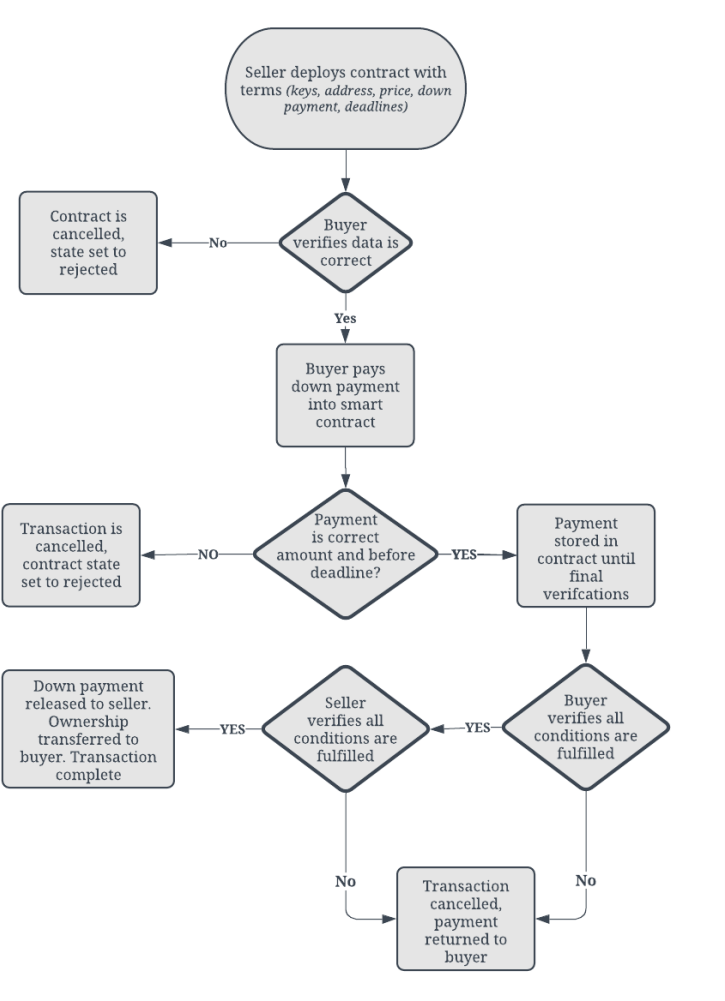
**GROUP 2**

**JOY MAKAU 19/06908 PRAISE NYANDIA 20/00848**

**CAROLINE MWENDE 20/04039 MARY WAMBUI 20/02415**

**REAL ESTATE SYSTEM**

# Work Flow of the System



This Solidity smart contract manages a real estate transaction, handling payments and the transfer of property between a seller and a buyer.

# Constructor

The constructor initializes the contract with:

* Seller and buyer addresses: The Ethereum addresses of the seller and the buyer.
* Property Address: The address of the property being sold.
* Final Price: The total price of the property.
* Down Payment: The initial payment amount required.
* Down Payment Deadline: The deadline by which the down payment must be made.
* Transfer Deadline: The deadline by which the property must be transferred to the buyer.

# Modifiers

* Only Buyer: Ensures that only the buyer can call certain functions.
* Only Seller: Ensures that only the seller can call certain functions.

# Functions

1. **make Down Payment**:
   * Allows the buyer to make the down payment before the specified deadline.
   * Ensures the correct payment amount is provided.
   * Sets down Payment Made to true and logs the event.

# Transfer Property:

* + Allows the seller to transfer the property after the down payment has been made.
  + Checks that the current time is before the transfer deadline and that the down payment is complete.
  + Transfers the remaining balance to the seller and logs the event.

# Refund Down Payment:

* + Allows the seller to refund the down payment if it was not made by the deadline.
  + Transfers the down payment back to the buyer.

# finalize Sale:

* + Allows the buyer to complete the sale after the property has been transferred.
  + Transfers the remaining balance of the final price to the seller.

# Events

* Down Payment Made: Logs when the buyer makes the down payment.
* Property Transferred: Logs when the seller transfers the property to the buyer.

This contract ensures a secure, step-by-step process for making down payments, transferring property, and finalizing real estate transactions.

**SYSTEM CODE**

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract RealEstate {

    address public seller;

    address public buyer;

    string public propertyAddress;

    uint256 public finalPrice;

    uint256 public downPayment;

    uint256 public downPaymentDeadline;

    uint256 public transferDeadline;

    bool public downPaymentMade;

    bool public propertyTransferred;

    event DownPaymentMade(address indexed buyer, uint256 amount);

    event PropertyTransferred(address indexed seller, address indexed buyer, uint256 amount);

    constructor(

        address \_seller,

        address \_buyer,

        string memory \_propertyAddress,

        uint256 \_finalPrice,

        uint256 \_downPayment,

        uint256 \_downPaymentDeadline,

        uint256 \_transferDeadline

    ) {

        seller = \_seller;

        buyer = \_buyer;

        propertyAddress = \_propertyAddress;

        finalPrice = \_finalPrice;

        downPayment = \_downPayment;

        downPaymentDeadline = \_downPaymentDeadline;

        transferDeadline = \_transferDeadline;

        downPaymentMade = false;

        propertyTransferred = false;

    }

    modifier onlyBuyer() {

        require(msg.sender == buyer, "Only the buyer can call this function.");

        \_;

    }

    modifier onlySeller() {

        require(msg.sender == seller, "Only the seller can call this function.");

        \_;

    }

    function makeDownPayment() public payable onlyBuyer {

        require(block.timestamp <= downPaymentDeadline, "Down payment deadline has passed.");

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

        require(!downPaymentMade, "Down payment has already been made.");

        downPaymentMade = true;

        emit DownPaymentMade(buyer, msg.value);

    }

    function transferProperty() public onlySeller {

        require(downPaymentMade, "Down payment has not been made.");

        require(block.timestamp <= transferDeadline, "Transfer deadline has passed.");

        require(!propertyTransferred, "Property has already been transferred.");

        propertyTransferred = true;

        payable(seller).transfer(finalPrice - downPayment);

        emit PropertyTransferred(seller, buyer, finalPrice);

    }

    function refundDownPayment() public onlySeller {

        require(block.timestamp > downPaymentDeadline, "Down payment deadline has not passed.");

        require(!downPaymentMade, "Down payment has already been made.");

        payable(buyer).transfer(downPayment);

    }

    function finalizeSale() public onlyBuyer {

        require(downPaymentMade, "Down payment has not been made.");

        require(propertyTransferred, "Property has not been transferred.");

        payable(seller).transfer(finalPrice - downPayment);

    }

}

**Contract Overview**

**The Real Estate contract manages the process of selling a property, including handling down payments, property transfers, and deadlines.**

### **State Variables**

address public seller;

    address public buyer;

    string public propertyAddress;

    uint256 public finalPrice;

    uint256 public downPayment;

    uint256 public downPaymentDeadline;

    uint256 public transferDeadline;

    bool public downPaymentMade;

    bool public propertyTransferred;

* **seller**: Address of the property seller.
* **buyer**: Address of the property buyer.
* **propertyAddress**: Address of the property being sold.
* **finalPrice**: Total price of the property.
* **downPayment**: Amount required as an upfront payment.
* **downPaymentDeadline**: Deadline for the buyer to make the down payment.
* **transferDeadline**: Deadline by which the property must be transferred.
* **downPaymentMade**: Indicates whether the down payment has been made.
* **propertyTransferred**: Indicates whether the property has been transferred to the buyer.

### Events

event DownPaymentMade(address indexed buyer, uint256 amount);

    event PropertyTransferred(address indexed seller, address indexed buyer, uint256 amount);

* **DownPaymentMade**: Emitted when the buyer makes the down payment.
* **PropertyTransferred**: Emitted when the property is transferred to the buyer.

### Constructor

constructor(

        address \_seller,

        address \_buyer,

        string memory \_propertyAddress,

        uint256 \_finalPrice,

        uint256 \_downPayment,

        uint256 \_downPaymentDeadline,

        uint256 \_transferDeadline

    ) {

        seller = \_seller;

        buyer = \_buyer;

        propertyAddress = \_propertyAddress;

        finalPrice = \_finalPrice;

        downPayment = \_downPayment;

        downPaymentDeadline = \_downPaymentDeadline;

        transferDeadline = \_transferDeadline;

        downPaymentMade = false;

        propertyTransferred = false;

    }

* Initializes the contract with the seller’s and buyer’s addresses, the property address, final price, down payment amount, deadlines, and initial status of down payment and property transfer.

### Modifiers

modifier onlyBuyer() {

        require(msg.sender == buyer, "Only the buyer can call this function.");

        \_;

    }

    modifier onlySeller() {

        require(msg.sender == seller, "Only the seller can call this function.");

        \_;

    }

* **onlyBuyer**: Ensures that only the buyer can call functions with this modifier.
* **onlySeller**: Ensures that only the seller can call functions with this modifier.

### Functions

1. **makeDownPayment**

function makeDownPayment() public payable onlyBuyer {

        require(block.timestamp <= downPaymentDeadline, "Down payment deadline has passed.");

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

        require(!downPaymentMade, "Down payment has already been made.");

        downPaymentMade = true;

        emit DownPaymentMade(buyer, msg.value);

    }

**** Purpose: Allows the buyer to make the down payment.

 Checks:

* The current time is before the down payment deadline.
* The sent value matches the required down payment amount.
* The down payment has not already been made.

 Actions:

* Sets downPaymentMade to true.
* Emits the DownPaymentMade event**.**

2.’Transferproperty’

  function transferProperty() public onlySeller {

        require(downPaymentMade, "Down payment has not been made.");

        require(block.timestamp <= transferDeadline, "Transfer deadline has passed.");

        require(!propertyTransferred, "Property has already been transferred.");

        propertyTransferred = true;

        payable(seller).transfer(finalPrice - downPayment);

        emit PropertyTransferred(seller, buyer, finalPrice);

    }

 **Purpose**: Allows the seller to transfer the property to the buyer.

 **Checks**:

* The down payment has been made.
* The current time is before the transfer deadline.
* The property has not already been transferred.

 **Actions**:

* Sets propertyTransferred to true.
* Transfers the remaining balance of the final price to the seller.
* Emits the PropertyTransferred event.

3.refundDownPayment

function refundDownPayment() public onlySeller {

        require(block.timestamp > downPaymentDeadline, "Down payment deadline has not passed.");

        require(!downPaymentMade, "Down payment has already been made.");

        payable(buyer).transfer(downPayment);

    }

 **Purpose**: Allows the seller to refund the down payment to the buyer if it has not been made by the deadline.

 **Checks**:

* The current time is after the down payment deadline.
* The down payment has not been made.

 **Actions**:

* Transfers the down payment amount back to the buyer.

1. finalizeSale

  function finalizeSale() public onlyBuyer {

        require(downPaymentMade, "Down payment has not been made.");

        require(propertyTransferred, "Property has not been transferred.");

        payable(seller).transfer(finalPrice - downPayment);

    }

 **Purpose**: Allows the buyer to complete the sale by paying the remaining balance.

 **Checks**:

* The down payment has been made.
* The property has been transferred.

 **Actions**:

* Transfers the remaining amount of the final price to the seller.

Deployment

