**Report**

**Prompt:**

I need a detailed smart contract project output should be like a proper project that we can propose to the client , Since I am beginner for this smart contract project guide me from scratch and also give me detailed code.  
  
**Response:**

Smart Contract (solidity):

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract Wallet {

address public owner;

mapping(address => uint256) public balances;

event Deposit(address indexed from, uint256 amount);

event Withdrawal(address indexed to, uint256 amount);

event Transfer(address indexed from, address indexed to, uint256 amount);

constructor() {

owner = msg.sender;

}

modifier onlyOwner() {

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

\_;

}

function deposit() public payable {

require(msg.value > 0, "Deposit amount must be greater than 0");

balances[msg.sender] += msg.value;

emit Deposit(msg.sender, msg.value);

}

function withdraw(uint256 amount) public {

require(amount > 0, "Withdrawal amount must be greater than 0");

require(balances[msg.sender] >= amount, "Insufficient balance");

balances[msg.sender] -= amount;

payable(msg.sender).transfer(amount);

emit Withdrawal(msg.sender, amount);

}

function transfer(address to, uint256 amount) public {

require(to != address(0), "Invalid recipient address");

require(amount > 0, "Transfer amount must be greater than 0");

require(balances[msg.sender] >= amount, "Insufficient balance");

balances[msg.sender] -= amount;

balances[to] += amount;

emit Transfer(msg.sender, to, amount);

}

function getBalance(address account) public view returns (uint256) {

return balances[account];

}

}

Python Script to interact with smart contracts:

from brownie import Contract, network, accounts, config

# Load contract ABI and address

contract\_abi = "path/to/contract\_abi.json"

contract\_address = "0x123...456" # Address of the deployed contract

# Initialize contract instance

def initialize\_contract():

return Contract.from\_abi("Wallet", contract\_address, contract\_abi)

# Deposit ETH into the wallet

def deposit\_eth(amount):

contract = initialize\_contract()

account = accounts.load("your\_account") # Load your Ethereum account

gas\_price = network.gas\_price()

gas\_limit = 500000 # adjust as needed

tx = contract.deposit({"from": account, "value": amount, "gas\_price": gas\_price, "gas\_limit": gas\_limit})

tx.wait(1)

print("Deposited", amount, "ETH into the wallet")

# Withdraw ETH from the wallet

def withdraw\_eth(amount):

contract = initialize\_contract()

gas\_price = network.gas\_price()

gas\_limit = 500000 # adjust as needed

tx = contract.withdraw(amount, {"from": account, "gas\_price": gas\_price, "gas\_limit": gas\_limit})

tx.wait(1)

print("Withdrawn", amount, "ETH from the wallet")

# Transfer ETH from one wallet to another

def transfer\_eth(to\_address, amount):

contract = initialize\_contract()

account = accounts.load("your\_account") # Load your Ethereum account

gas\_price = network.gas\_price()

gas\_limit = 500000 # adjust as needed

tx = contract.transfer(to\_address, amount, {"from": account, "gas\_price": gas\_price, "gas\_limit": gas\_limit})

tx.wait(1)

print("Transferred", amount, "ETH to", to\_address)

# Check balance of a wallet

def check\_balance():

contract = initialize\_contract()

account = accounts.load("your\_account") # Load your Ethereum account

balance = contract.getBalance(account)

print("Current balance:", balance, "ETH")

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

# Example: Deposit 1 ETH

deposit\_eth(1)

# Example: Withdraw 0.5 ETH

withdraw\_eth(0.5)

# Example: Transfer 0.3 ETH to another address

transfer\_eth("0x789...012", 0.3)

# Example: Check balance

check\_balance()  
-----------------------------------------------------------------------------------------------

**Execution:**

To execute this solidity Ethereum code I have used:

>Remix IDE platform (Online compiler)

>MetaMask (Configured in my web-browser)

>Initially I have created a new project which will default will give me a project structure in which I have created new file named “4\_Wallet.sol”.

>In which I have used the code given the LLM(GPT 3.5 & 4) .

>After coding the solidity:

>I have Compiled it and then deployed it by using “Deployment option”.  
  
 >Then I have connected the “MetaMask” account which I have configured with the web-browser and then after configuration .

A screenshot of a computer screen

Description automatically generated>I have started a transaction the “Remix-IDE” will redirect to “MetaMask”.

>MetaMask:

Transaction Request:

A screenshot of a computer

Description automatically generated

Deployment Using Emulation(Used :Remix VM(Cancum)):

1. 1.I have used Javascript VM here the available model is Remix VM(Cancum) .
2. Since the Remix VM(cancum) is the recent updated version I have used this model.
3. 3.These are the transactions detailed report:

|  |  |
| --- | --- |
| **status** | 0x1 Transaction mined and execution succeed |
| **transaction hash** | 0x69dfcb6ec34fa9094184838c3d5f3e650024396b2ccb2a7ccabf076135fc25fd |
| **block hash** | 0x1d3fa2349b7eb163e16c9560f8e3a7b740aeae13d906b64744c23bce49b18e8d |
| **block number** | 3 |
| **contract address** | 0xf8e81D47203A594245E36C48e151709F0C19fBe8 |
| **from** | 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 |
| **to** | Wallet.(constructor) |
| **gas** | 810010 gas |
| **transaction cost** | 704356 gas |
| **execution cost** | 606782 gas |
| **input** | 0x608...90033 |
| **decoded input** | {} |
| **decoded output** | - |
| **logs** | [] |