// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract FundManager {

// Address of the contract owner

address public owner;

// Constructor to set the owner as the contract deployer

constructor() {

owner = msg.sender;

}

// Modifier to restrict access to the contract owner

modifier onlyOwner() {

require(msg.sender == owner, "Only the owner can withdraw funds"); \_;

}

// Function to deposit Ether into the contract

function deposit() public payable {

require(msg.value > 0, "You must send some ether to deposit");

}

// Function to withdraw Ether from the contract (only owner)

function withdraw(uint \_amount) public onlyOwner {

require(address(this).balance >= \_amount, "Insufficient balance in contract");

payable(msg.sender).transfer(\_amount);

}

// Function to check contract balance

function getContractBalance() public view returns (uint) {

return address(this).balance;

}

}

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract CompoundInterest {

// State variables

address public owner;

uint256 public principal;

uint256 public rate; // Interest rate in percentage

uint256 public time; // Time in years

// Events

event InterestCalculated(uint256 totalAmount);

// Constructor

constructor() {

owner = msg.sender; // Set the contract creator as the owner

}

// Function to set principal, rate, and time

function setParameters(uint256 \_principal, uint256 \_rate, uint256 \_time) public payable{

require(msg.sender == owner, "Only the owner can set parameters");

principal = \_principal;

rate = \_rate;

time = \_time;

}

// Function to calculate compound interest

function calculateCompoundInterest() public returns (uint256) {

uint256 totalAmount = principal \* (1 + rate / 100) \*\* time;

emit InterestCalculated(totalAmount);

return totalAmount;

}

}

Q.6 Build and test decentralized application (Dapp) for Election Voting System on the local

Ethereum Blockchain Network Ganache using truffle suite. Code:

MyVoting.sol

//SPDX-License-Identifier: MIT

pragma solidity >=0.5.0 <0.8.27;

contract MyVoting {

struct Candidate {

uint256 id;

string name;

uint256 voteCount;

}

mapping(address => bool) public voters;

mapping(uint256 => Candidate) public candidates;

uint256 public candidateCount;

event votedEvent(uint256 indexed \_candidateId);

constructor() public {

addCandidate("Candidate 1");

addCandidate("Candidate 2");

addCandidate("Candidate 3");

}

function addCandidate(string memory \_name) public {

candidateCount++;

candidates[candidateCount] = Candidate(candidateCount, \_name, 0);

}

function vote(uint256 \_candidateId) public {

require(!voters[msg.sender], "You already voted");

require(\_candidateId > 0 && \_candidateId <= candidateCount);

voters[msg.sender] = true;

candidates[\_candidateId].voteCount++;

emit votedEvent(\_candidateId);

}

function getCandidateDetails(uint \_candidateId) public view returns (uint,string memory,uint)

{

return (candidates[\_candidateId].id, candidates[\_candidateId].name, candidates[\_candidateId].voteCount);

}

}

2\_deploy\_contract.js

var MyVoting = artifacts.require("./MyVoting.sol");

module.exports = function(deployer)

{

deployer.deploy(MyVoting);

};

Q.7 Build and test decentralized application, (Dapp) for Banking System on the local Ethereum

Blockchain Network Ganache using truffle suite. Code:

MyBanking.sol

pragma solidity >=0.5.16;

contract MyBanking {

// State variable to store the balance

uint256 private balance;

// Constructor to ini alize balance

constructor () public {

balance = 0;

}

// Func on to add (deposit) amount to the balance

function addAmount(uint256 amount) public {

balance += amount;

}

// Func on to withdraw amount from the balance

function withdrawAmount(uint256 amount) public {

require(amount <= balance, "Insufficient balance");

balance -= amount;

}

// Func on to check the remaining balance

function checkBalance() public view returns (uint256) {

return balance;

}

}

2\_deploy\_contracts.js

const MyBanking = artifacts.require("MyBanking");

module.exports = function (deployer) {

deployer.deploy(MyBanking);

};