St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2024-2025
Class: BE-ITA/B, Semester: VIII
Subject: BlockChain Lab

Experiment –2

1. **Aim:** To implement smart contract using Remix IDE

2. **Objective:** To …
   - explain what smart contracts are.
   - understand coding smart contracts using solidity language.
   - test a smart contract by using Remix IDE

3. Lab outcome: After performing the experiment, the students will be able to **implement** smart contracts in Ethereum using different development frameworks (PO3, PSO2, BL3)

4. Prerequisite:
- Fundamental knowledge of blockchain
- Knowledge of the Ethereum platform and Remix IDE
- Familiarity with the Solidity programming language

5. Requirements: The following are the requirements –
- Remix IDE

6. Pre-Experiment Theory:

What is Blockchain?
Blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network. An asset can be tangible (a house, car, cash, land) or intangible (intellectual property, patents, copyrights, branding). Virtually anything of value can be tracked and traded on a blockchain network, reducing risk and cutting costs for all involved.

What is Ethereum?
Ethereum is a decentralized blockchain designed to be highly secure, fault-tolerant, and programmable.
Ethereum blockchain is a choice for many developers and businesses. As said programmable, the main task of Ethereum is to securely execute and verify the application code known as **smart contracts**.
Ethereum helps to build native scripting language(solidity) and EVM. Ethereum consensus mechanism
is proof of work to operate to verify the new transaction.

What is Remix IDE?
It is an online IDE for creating solid, **smart contracts**, so you do not need to install or download anything to do any setup. You can develop, deploy, and administer your solidity smart contract using Remix IDE.

What is Solidity?
Solidity is a statically typed, contract-oriented, high-level language for implementing smart contracts that run on the Ethereum Virtual Machine. Smart contracts are programs that are executed inside a peer- to-peer network where nobody has special authority over the execution, and thus they allow anyone to implement tokens of value, ownership, voting, and other kinds of logic.

What is smart contract?
A smart contract is a small program that runs on an Ethereum blockchain. Once the smart contract is deployed on the Ethereum blockchain, it cannot be changed. To deploy the smart contract to Ethereum, you must pay the ether (ETH) cost. Understand it as a digital agreement that builds trust and allows both parties to agree on a particular set of conditions that cannot be tampered with.

7. Laboratory Exercise

A. Steps to be implemented.

To Follow the procedure given below to build smart contract in Remix IDE

1. Open Remix IDE in Google Chrome
2. Write smart contract by creating new file under contract folder, with .sol extension.
3. Write your contract code using solidity language.
4. Click on Solidity compiler icon.
5. Choose compiler version or keep default and click on compile button.
6. Click on Deploy and Run Transaction icon.
7. Choose Environment Remix VM (London). and choose Account Number or keep default.
8. Click on deploy button to deploy smart contract on Ethereum blockchain.
9. Under deployed contract, get the output of your contract.

B. Program Code

1. Write the first smart contract HelloWorld.sol as follows.

```
==========================
// SPDX - License - identifier:
GPL-3.0 pragma solidity >=0.7.0
<0.9.0; contract HelloWorld {
function Greet() public view returns(string memory){
  return "Hello Everyone, I am Joanne";
} }
==========================
```

2. Write a smart contract storage.sol to store a number and to retrieve the number as follows.

```
==========================
// SPDX-License-Identifier:
GPL-3.0 pragma solidity >=0.7.0
<0.9.0; contract storage1 {
uint number;
function storeInt (uint _num) public {
   number = _num; }
function retrieve () public view returns (uint) {
   return number;
} }
==========================
```

3. Write a smart contract employee.sol to create an employee database with fields like empId, empName, empDept, empDesignation. Create functions to addEmployee and getEmployee. Use concept of structure.

8. Post Experimental Exercise-

C. Questions:

1. Write down key properties and advantages of smart contracts
2. Create a Hostel.sol smart contract by following the tutorial from https://blog.loginradius.com/engineering/guest-post/ethereum-smart-contract-tutorial/

D. Results/Observations/Program output:

Present the program input/output results if any and comment on the same.

E. Conclusion:

1. Write what was performed in the experiment
2. Write which tools you used to perform the experiment
3. Write what you inferred from the output obtained

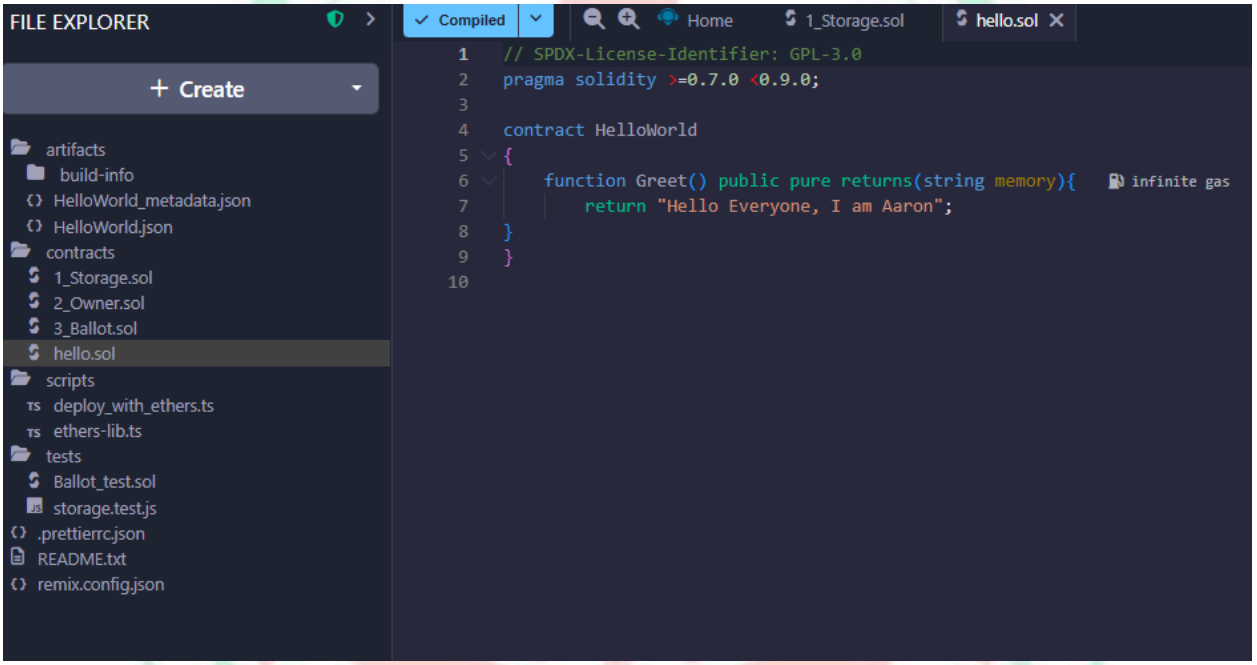
F. References:

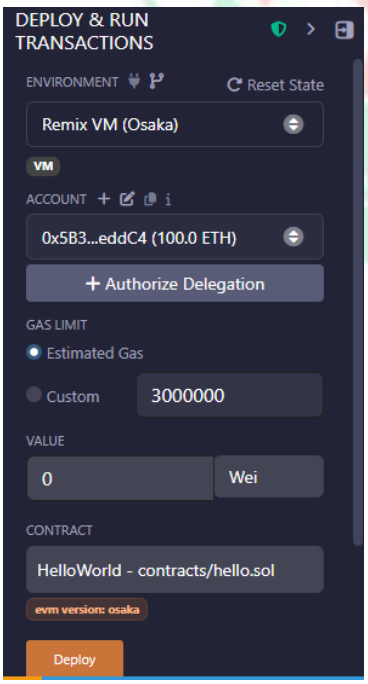
[1] https://www.simplilearn.com/tutorials/blockchain-tutorial/what-is-smart-contract

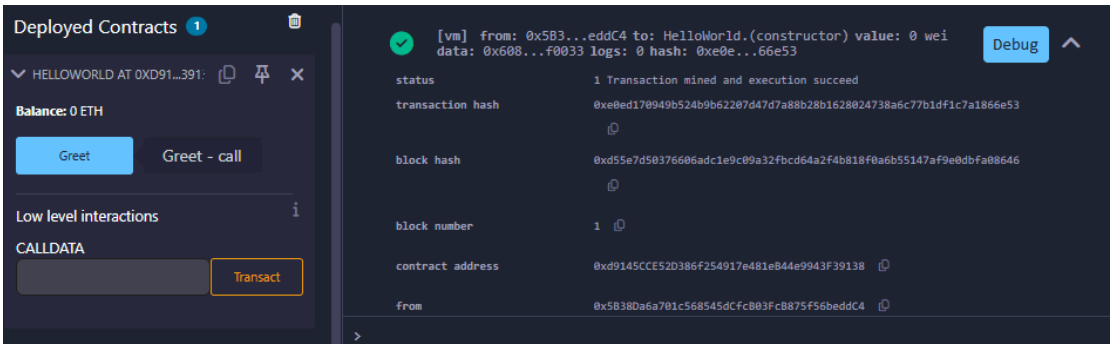
[2] https://ethereum.org/en

[3] Mastering Ethereum, Building Smart Contract and Dapps, Andreas M. Antonopoulos Dr. Gavin Wood, O'reilly

1. Write the first smart contract HelloWorld.sol as follows

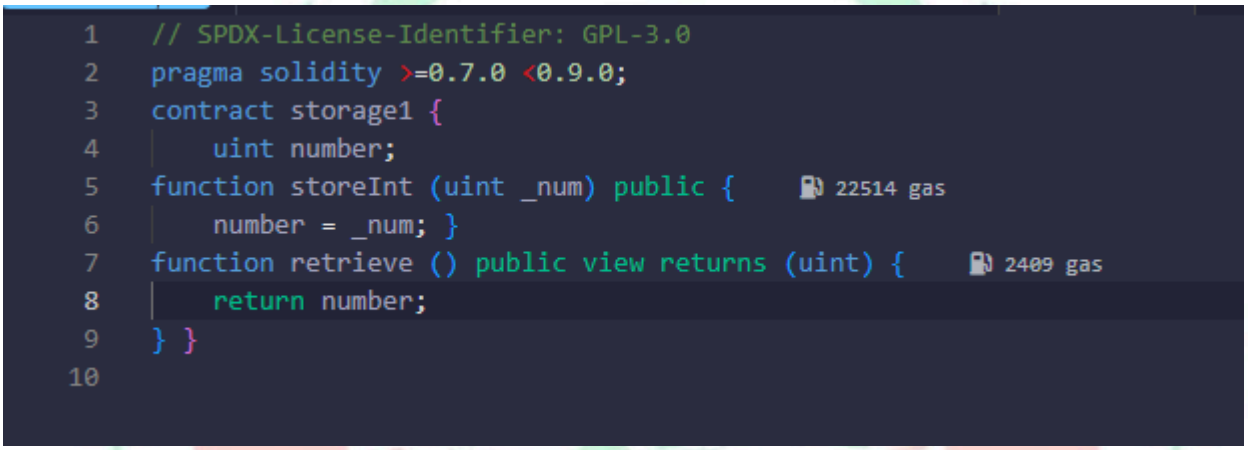






\

2. Write a smart contract storage.sol to store a number and to retrieve the number as follows



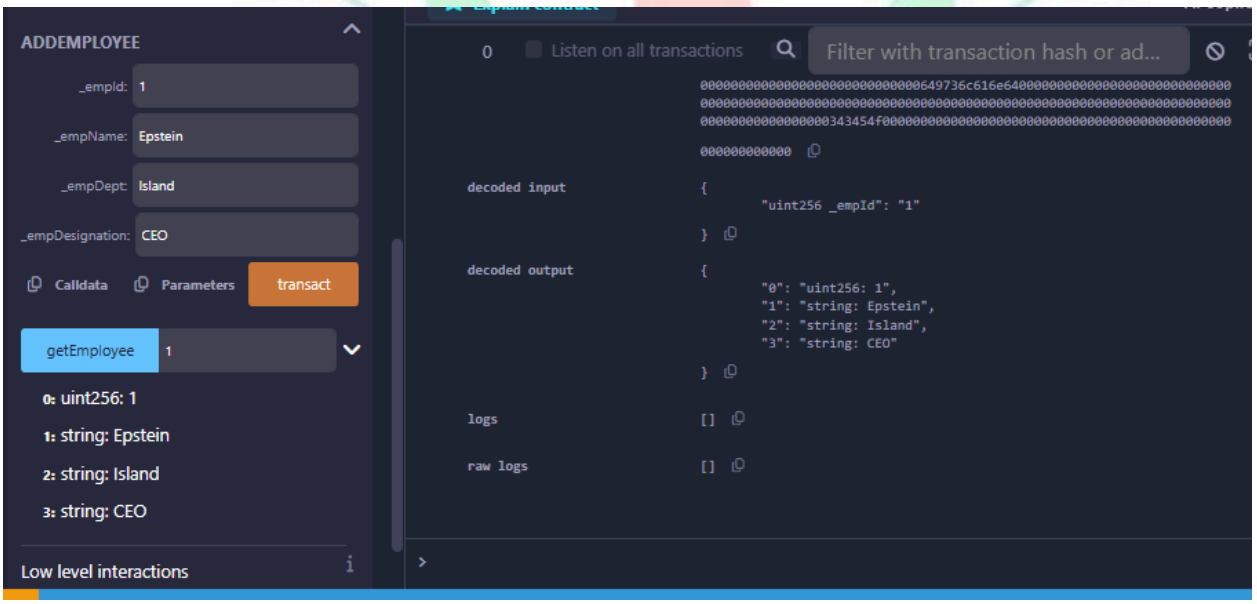
Write a smart contract employee.sol to create an employee database with fields like empId,

empName, empDept, empDesignation. Create functions to addEmployee and getEmployee. Use concept of structure.

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract EmployeeDatabase {
    struct Employee {
        uint256 empId;
        string empName;
        string empDept;
        string empDesignation;
    }
    mapping(uint256 => Employee) private employees;

    function addEmployee(    infinite gas
        uint256 _empId,
        string memory _empName,
        string memory _empDept,
        string memory _empDesignation
    ) public {
        employees[_empId] = Employee(
            _empId,
            _empName,
            _empDept,
            _empDesignation
        );
    }
    function getEmployee(uint256 _empId)    infinite gas
        public
        view
        returns (
            uint256,
            string memory,
            string memory,
            string memory
        )
    {
        Employee memory emp = employees[_empId];
        return (
            emp.empId,
            emp.empName,
            emp.empDept,
            emp.empDesignation
        );
    }
```



Post Experiment

3. Create a Hostel.sol smart contract by following the tutorial from

Hostel:

Code:

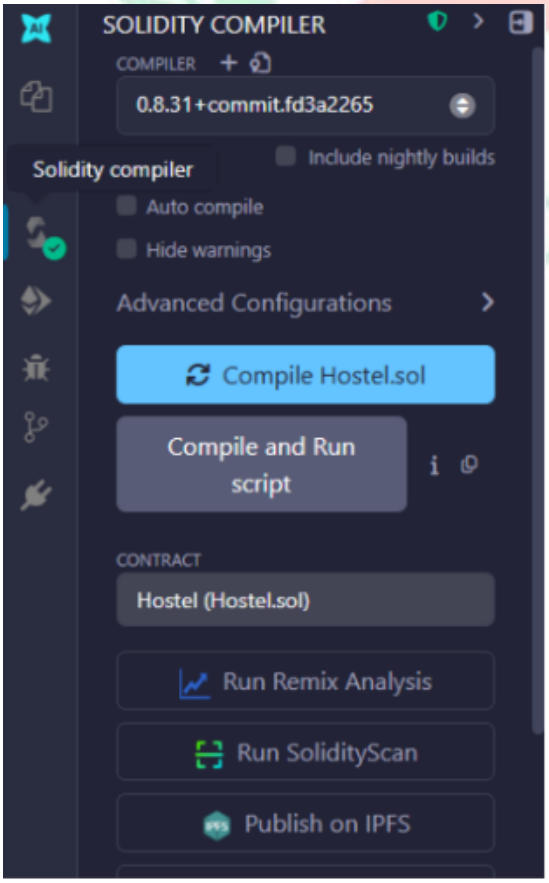
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Hostel {
address payable public landlord;
uint public no_of_rooms = 0;
uint public no_of_agreement = 0;
uint public no_of_rent = 0;
struct Room {
uint roomid;
uint agreementid;
string roomname;
string roomaddress;
uint rent_per_month;
uint securityDeposit;
uint timestamp;
bool vacant;
address payable landlord;
address payable currentTenant;
}
mapping(uint => Room) public Room_by_No;
struct RoomAgreement {
uint agreementid;
uint roomid;
string roomname;
string roomaddress;
uint rent_per_month;
uint securityDeposit;
uint lockInPeriod;
uint timestamp;
address payable landlord;
address payable tenant;
}
mapping(uint => RoomAgreement) public RoomAgreement_by_No;
struct Rent {
uint rentno;
uint roomid;
uint agreementid;
string roomname;
string roomaddress;
uint rent_per_month;
uint timestamp;
```

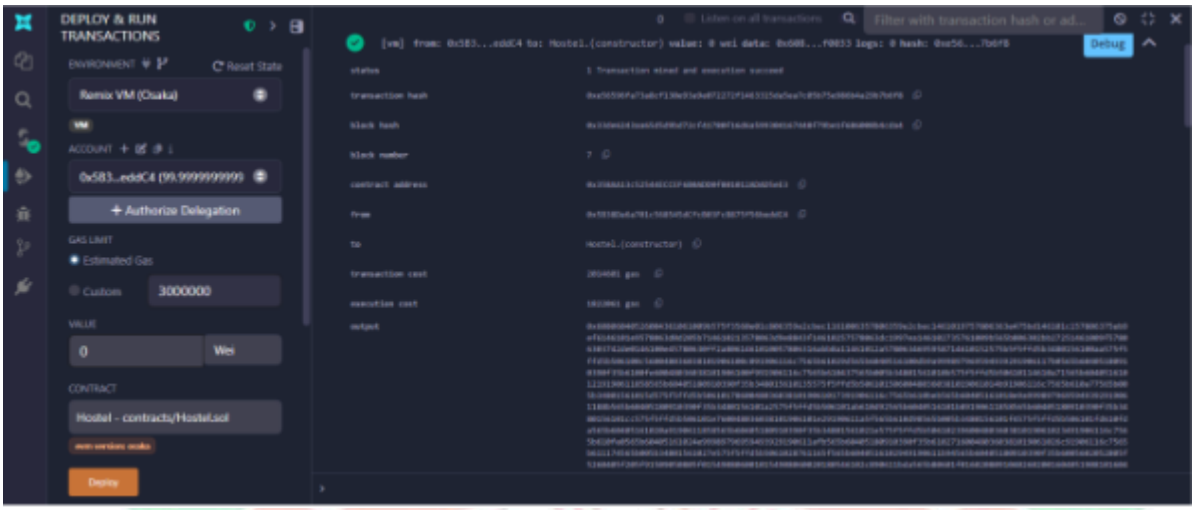
```
address payable landlord;
address payable tenant;
}
mapping(uint => Rent) public Rent_by_No;
constructor() {
landlord = payable(msg.sender);
}
modifier onlyLandlord() {
require(msg.sender == landlord, "Only landlord can access this");
_;
}
modifier notLandlord(uint _index) {
require(msg.sender != Room_by_No[_index].landlord, "Landlord cannot be a tenant");
_;
}
function addRoom(
string memory _roomname,
string memory _roomaddress,
uint _rentcost,
uint _securitydeposit
) public onlyLandlord {
no_of_rooms++;
Room_by_No[no_of_rooms] = Room(
no_of_rooms, 0, _roomname, _roomaddress, _rentcost, _securitydeposit, 0, true,
payable(msg.sender), payable(address(0))
);
}
function signAgreement(uint _index) public payable notLandlord(_index) {
require(Room_by_No[_index].vacant == true, "Room is currently occupied");
require(msg.value == (Room_by_No[_index].rent_per_month +
Room_by_No[_index].securityDeposit), "Please send total amount (Rent + Security)");
no_of_agreement++;
Room_by_No[_index].currentTenant = payable(msg.sender);
Room_by_No[_index].vacant = false;
Room_by_No[_index].agreementid = no_of_agreement;
Room_by_No[_index].timestamp = block.timestamp;
RoomAgreement_by_No[no_of_agreement] = RoomAgreement(
no_of_agreement, _index, Room_by_No[_index].roomname,
Room_by_No[_index].roomaddress,
Room_by_No[_index].rent_per_month, Room_by_No[_index].securityDeposit, 365 days,
block.timestamp, Room_by_No[_index].landlord, payable(msg.sender)
);
// Replaced transfer() with call()
(bool success, ) = Room_by_No[_index].landlord.call{value: msg.value}("");
require(success, "ETH transfer failed");
```

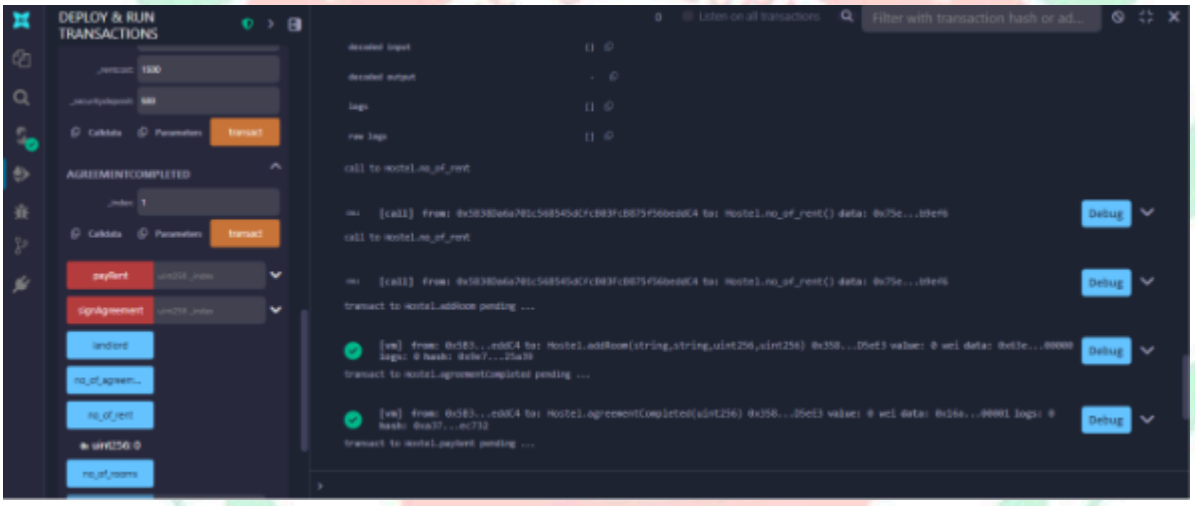
```
}
function payRent(uint _index) public payable {
require(msg.sender == Room_by_No[_index].currentTenant, "Only tenant can pay rent");
require(msg.value == Room_by_No[_index].rent_per_month, "Incorrect rent amount");

no_of_rent++;
Rent_by_No[no_of_rent] = Rent(
no_of_rent, _index, Room_by_No[_index].agreementid, Room_by_No[_index].roomname,
Room_by_No[_index].roomaddress, Room_by_No[_index].rent_per_month,
block.timestamp, Room_by_No[_index].landlord, payable(msg.sender)
);
// Replaced transfer() with call()
(bool success, ) = Room_by_No[_index].landlord.call{value: msg.value}("");
require(success, "ETH transfer failed");
}
function agreementCompleted(uint _index) public onlyLandlord {
require(block.timestamp > Room_by_No[_index].timestamp + 365 days, "Agreement period not
over yet");
Room_by_No[_index].vacant = true;
Room_by_No[_index].currentTenant = payable(address(0));
}
}
```

Output:




DEPLOY & RUN TRANSACTIONS
Remix VM (Osaka)
+ Authorize Delegation
3000000
Hostel - contracts/Hostel.sol
Deploy
Debug


DEPLOY & RUN TRANSACTIONS
AGREEMENTCOMPLETED
payRent
signAgreement
landlord
no_of_agreem...
no_of_rent
no_of_rooms
Debug


DEPLOY & RUN TRANSACTIONS
payRent
signAgreement
landlord
no_of_agreem...
no_of_rent
no_of_rooms
Rent_by_No
Room_by_No
RoomAgreeme...
Debug