BLOCKCHAIN TECHNOLOGY LAB

(20CP406P)

LAB ASSIGNMENT - 7

B.Tech in Computer Science and Engineering Dept.,

Pandit Deendayal Energy University,

Gandhinagar

Name: Mire Kishorkumar Patel

Roll No.: 19BCP080

Branch: Computer Engineering

Lab Assignment 7

Aim: Create the Banking Application, Deploy it on Testnet through Metamask

Introduction:

- **Testnet:** Testnet is a test network for Bitcoin that is used by developers to test new features or implementations before they are ready for mainnet. Testnet coins are separate and distinct from actual bitcoins, and are never supposed to have any real-world value. This allows developers to experiment with new features or implementations without having to worry about breaking the main network.

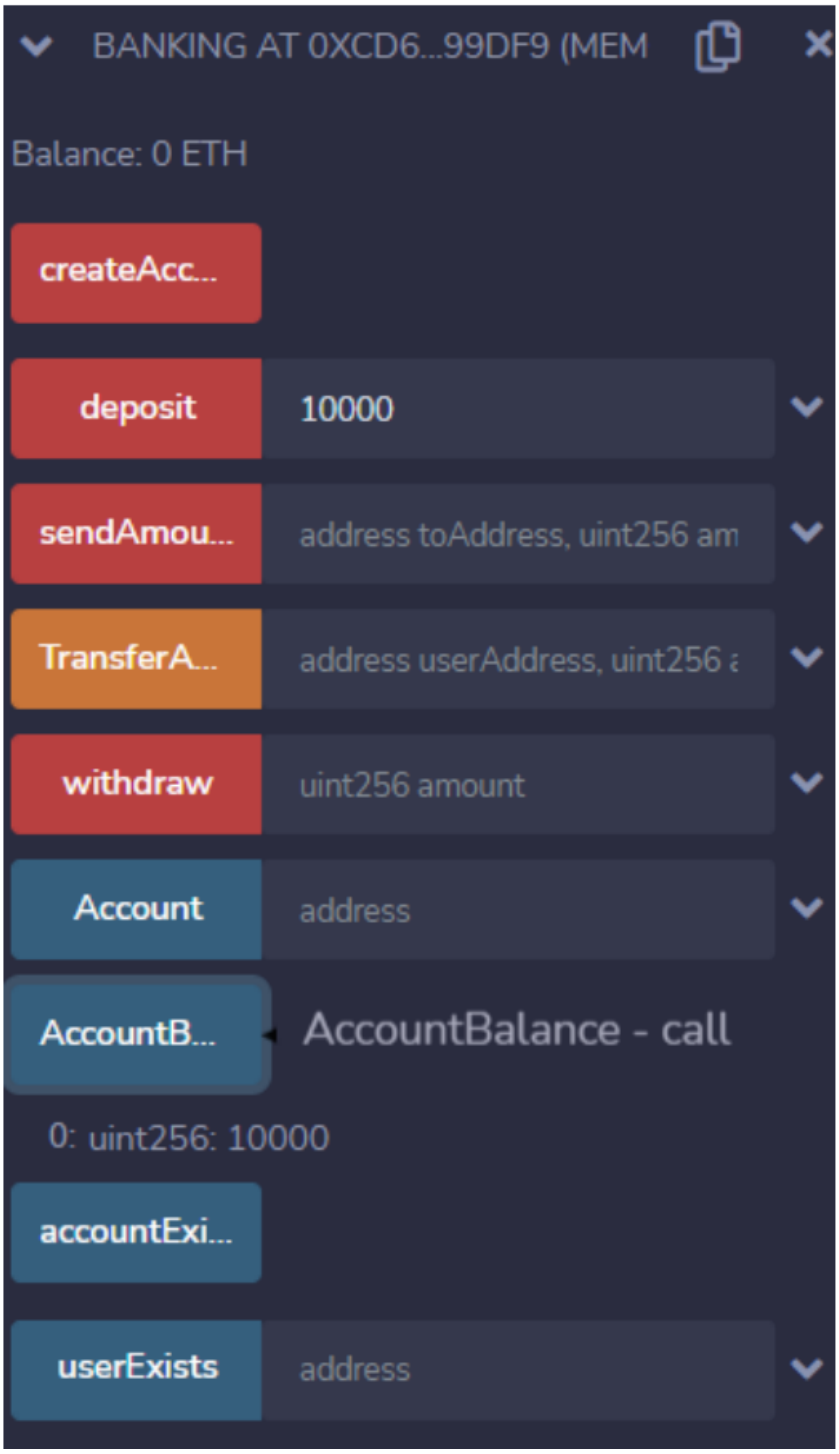
- **Metamask:** Metamask is a digital wallet that allows you to store, send, and receive cryptocurrency. It is one of the most popular wallets in the world and is used by millions of people. Metamask is available on all major platforms, including Windows, Mac, Linux, and Android.

Source Code:

- ***Smart Contract*:**

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

contract banking {
    mapping(address => uint256) public Account;
    mapping(address => bool) public userExists;

    function createAccount() public payable returns (string memory) {
        require(userExists[msg.sender] == false, "Account Already Exists");
        Account[msg.sender] = msg.value;
        userExists[msg.sender] = true;
        return "account created";
    }

    function deposit(uint256 amount) public payable returns (string memory) {
        require(userExists[msg.sender] == true, "Account is not created");
        require(amount > 0, "Value for deposit is Zero");
        Account[msg.sender] = Account[msg.sender] + amount;
        return "Deposited Succesfully";
    }

    function withdraw(uint256 amount) public payable returns (string memory) {
        require(
            Account[msg.sender] > amount,
            "Insufficeint balance in Bank account"
        );
        require(userExists[msg.sender] == true, "Account is not created");
        require(amount > 0, "Enter non-zero value for withdrawal");
        Account[msg.sender] = Account[msg.sender] - amount;
        msg.sender.transfer(amount);
        return "Withdrawal Succesful";
    }

    function TransferAmount(address payable userAddress, uint256 amount)
        public
        returns (string memory)
    {
        require(
            Account[msg.sender] > amount,
            "insufficient balance in Bank account"
        );
        require(userExists[msg.sender] == true, "Account is not created");
        require(
            userExists[userAddress] == true,
            "to Transfer account does not exists in bank accounts "
        );
        require(amount > 0, "Enter non-zero value for sending");
        Account[msg.sender] = Account[msg.sender] - amount;
        Account[userAddress] = Account[userAddress] + amount;
        return "Transfer successful";
    }

    function sendAmount(address payable toAddress, uint256 amount)
        public
        payable
        returns (string memory)
    {
        require(amount > 0, "Enter non-zero value for withdrawal");
        require(userExists[msg.sender] == true, "Account is not created");
        require(
            Account[msg.sender] > amount,
            "insufficient balance in Bank account"
        );
        Account[msg.sender] = Account[msg.sender] - amount;
        toAddress.transfer(amount);
        return "transfer success";
    }

    function AccountBalance() public view returns (uint256) {
        return Account[msg.sender];
    }

    function accountExist() public view returns (bool) {
        return userExists[msg.sender];
    }
}
```

Output:


BANKING AT 0XCD6...99DF9 (MEM
Balance: 0 ETH
createAcc...
deposit
10000
sendAmou...
address toAddress, uint256 am
TransferA...
address userAddress, uint256 a
withdraw
uint256 amount
Account
address
AccountB...
AccountBalance - call
0: uint256: 10000
accountExi...
userExists
address

Deploying the banking smart contract in Ganache UI and Metamask: