DEEPAK KUMAR CHAUDHARY

12010440

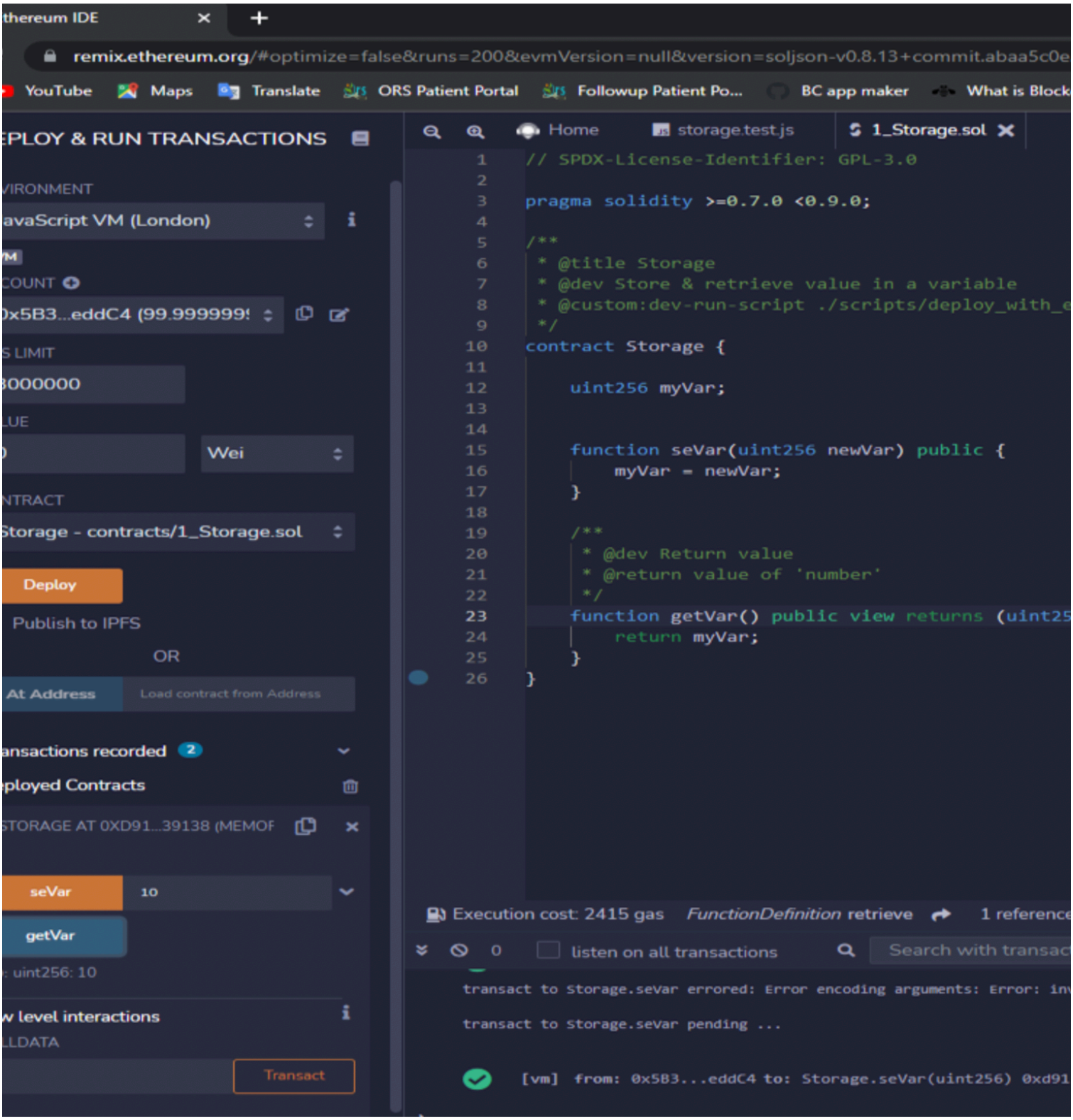
IIT Bhilai

1. Create multiple accounts in the meta mask and perform the balance transfer between the various accounts.

Ans- STEP:-

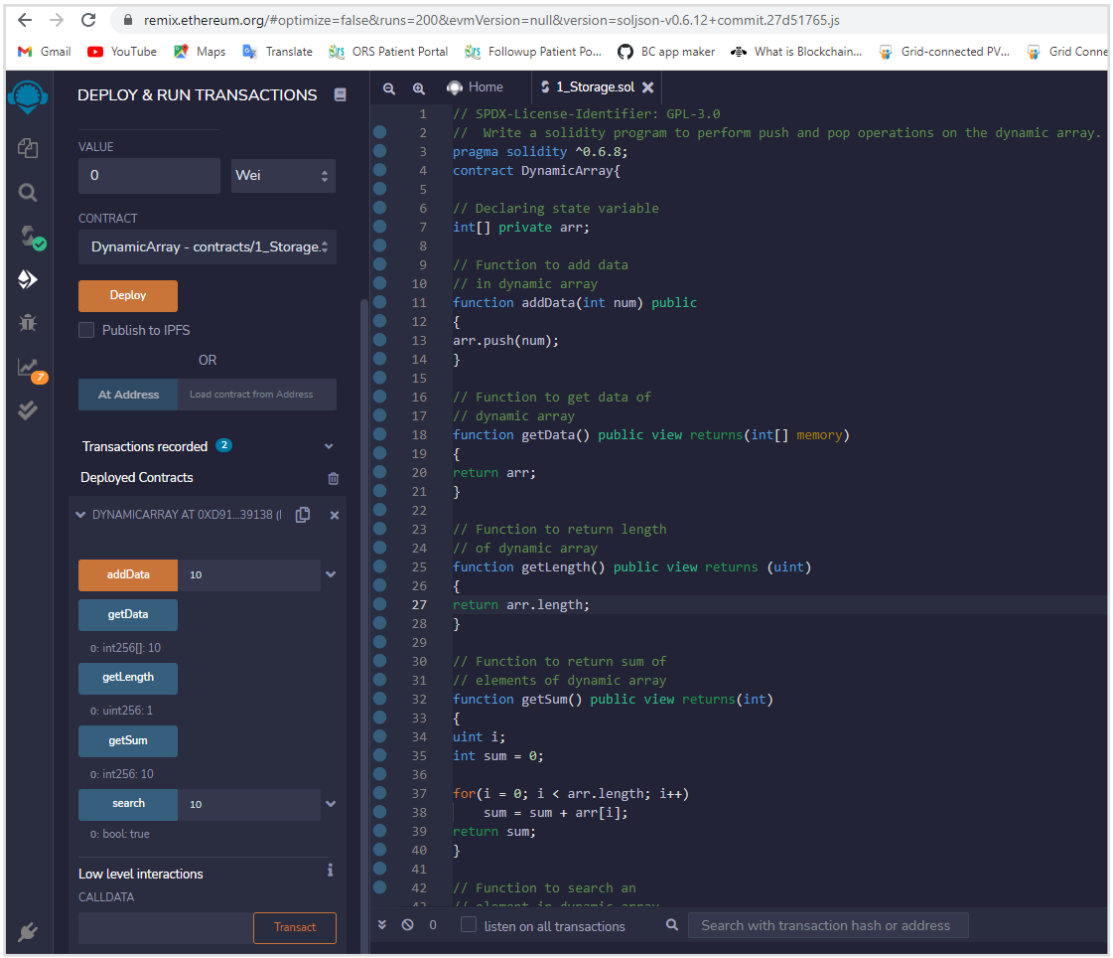
- Firstly open metamask.io & create account 1,2,3…
- Click open(), enter the address, and then take the test ether into account.
- Go to A/C 1 Menu click on Send option. click option “transfer “ between my account .
- After that gas fee are add and click on “Confirm”
- Finally balance transfer from one account to another.

2. Write a solidity program to set variables and get variables.



Caption

3. Write a solidity program to perform push and pop operations on the dynamic array.



4. write a solidity program to set address with a mapping variable

Ans- Code is written bellow.

```
pragma solidity ^0.7.0;
contract LedgerBalance {
mapping(address => uint) public balances;
function updateBalance(uint newBalance) public {
 balances[msg.sender] = newBalance;
 }
}
contract Updater {
function updateBalance() public returns (uint) {
 LedgerBalance ledgerBalance = new LedgerBalance();
    ledgerBalance.updateBalance(10);
  return ledgerBalance.balances(address(this));
}
}
```

5. Write a solidity program to get the factorial of a number.

Ans-

```
pragma solidity ^0.4.19;

contract Factorial {
```

```
  function fact(uint x) returns (uint y) {
    if (x == 0) {
      return 1;
    }
    else {
      return x*fact(x-1);
    }
  }
}
```

6. Write a solidity program to store information about a student (name, roll no., institute, Age) using a structure.

Ans-

```
pragma solidity ^0.7.0;

contract Student {
   struct detail {
      string institute;
      string name;
      uint roll;
   }
    detail student;

   function setStudent() public {
      student = detail( 'IIT', 'Deepak', 12010440);
   }
   function getroll() public view returns (uint) {
      return student.roll;
   }
```

```
}
```

7. Write a smart contract using a solidity program to perform the balance transfer from the contract to other accounts.

Ans-

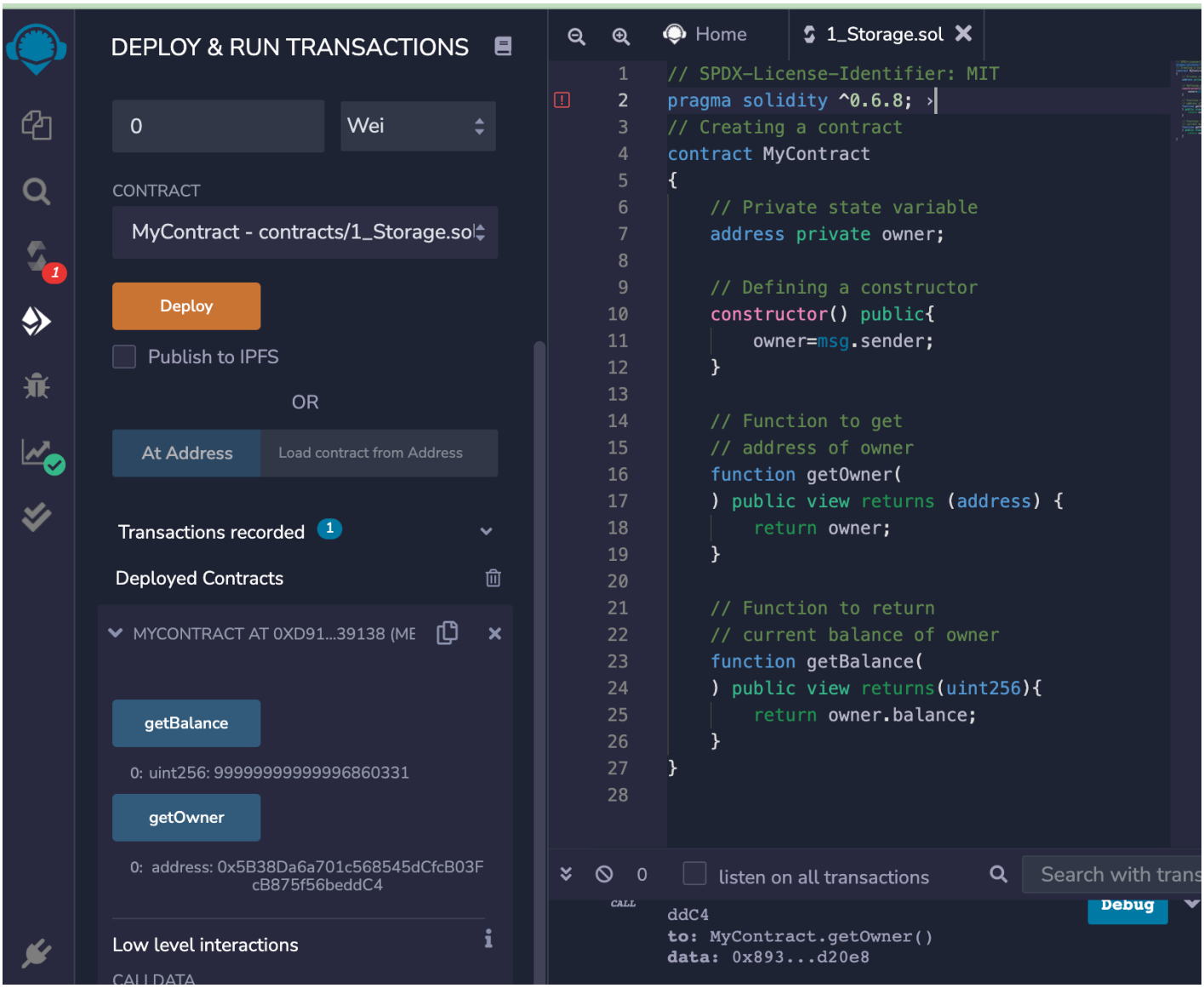
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.7.0;

contract CommunityChest {
    function deposit(uint256 amount) payable public {
        require(msg.value == amount);
        // nothing else to do!
    }

    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

8. Write a smart contract using a solidity program to perform balance transfer with mapping and make sure only the owner can transfer the balance from the contract to other accounts

Ans-

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.6.8;

// Creating a contract


contract MyContract
{
   // Private state variable
   address private owner;

   // Defining a constructor
   constructor() public{
      owner=msg.sender;

   // Function to get
   // address of owner
   function getOwner(
   ) public view returns (address) {
```

```
    return owner;
```

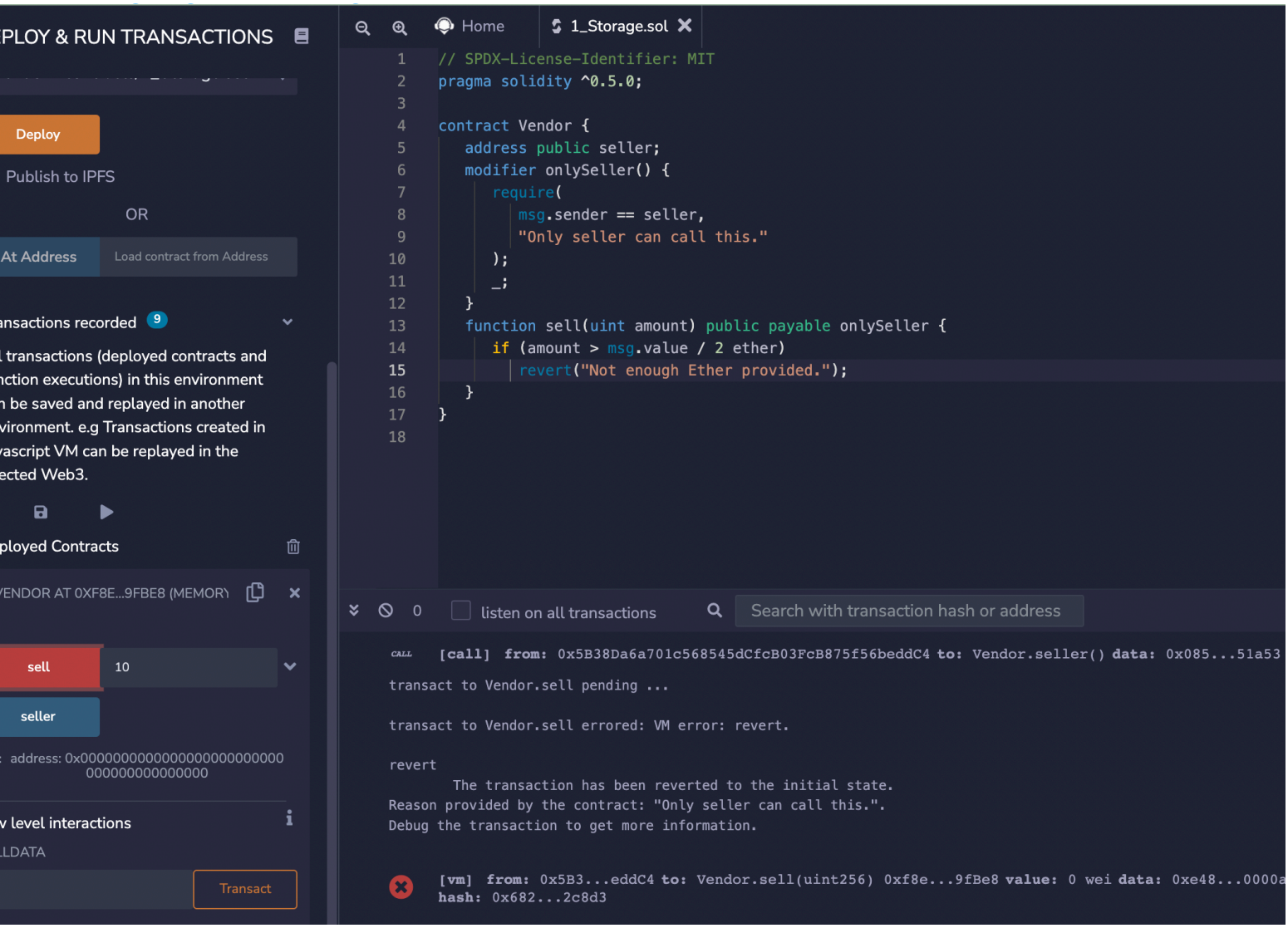



```
    }

    // Function to return
    // current balance of owner
    function getBalance(
    ) public view returns(uint256){
        return owner.balance;
    }
}
```

9. Write a solidity program to perform the exception handling and describe the details with screenshots.

Ans -

```
pragma solidity ^0.5.0;


contract Vendor {
  address public seller;
  modifier onlySeller() {
    require(
      msg.sender == seller,
      "Only seller can call this."
    );
    _;
  }



  function sell(uint amount) public payable onlySeller {
    if (amount > msg.value / 2 ether)
      revert("Not enough Ether provided.");
    // Perform the sell operation.
  }
```


PLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
nsactions recorded 9
l transactions (deployed contracts and
nction executions) in this environment
n be saved and replayed in another
vironment. e.g Transactions created in
vascript VM can be replayed in the
ected Web3.
ployed Contracts
VENDOR AT 0XF8E...9FBE8 (MEMOR\
sell
10
seller
address: 0x0000000000000000000000000
000000000000000
v level interactions
LLDATA
Transact
Home
1_Storage.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.5.0;
3
4 contract Vendor {
5 address public seller;
6 modifier onlySeller() {
7 require(
8 msg.sender == seller,
9 "Only seller can call this."
10);
11 _;
12 }
13 function sell(uint amount) public payable onlySeller {
14 if (amount > msg.value / 2 ether)
15 revert("Not enough Ether provided.");
16 }
17 }
18
0
listen on all transactions
Search with transaction hash or address
CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Vendor.seller() data: 0x085...51a53
transact to Vendor.sell pending ...
transact to Vendor.sell errored: VM error: revert.
revert
The transaction has been reverted to the initial state.
Reason provided by the contract: "Only seller can call this.".
Debug the transaction to get more information.
[vm] from: 0x5B3...eddC4 to: Vendor.sell(uint256) 0xf8e...9fBe8 value: 0 wei data: 0xe48...0000a
hash: 0x682...2c8d3

Caption

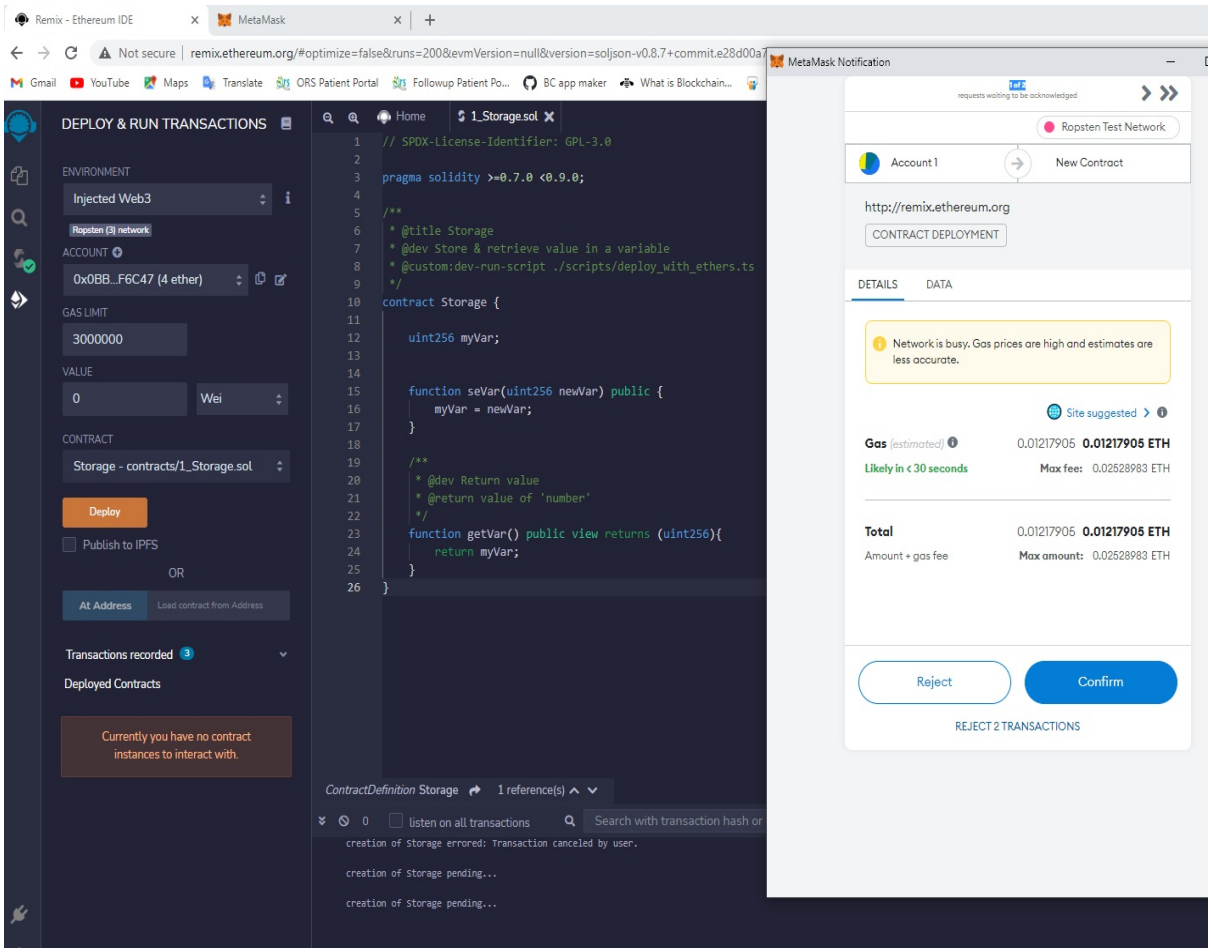
}

10. Connect the following tool with their mixed environment, perform balance transfer between the accounts with the smart contract and share the screenshots.

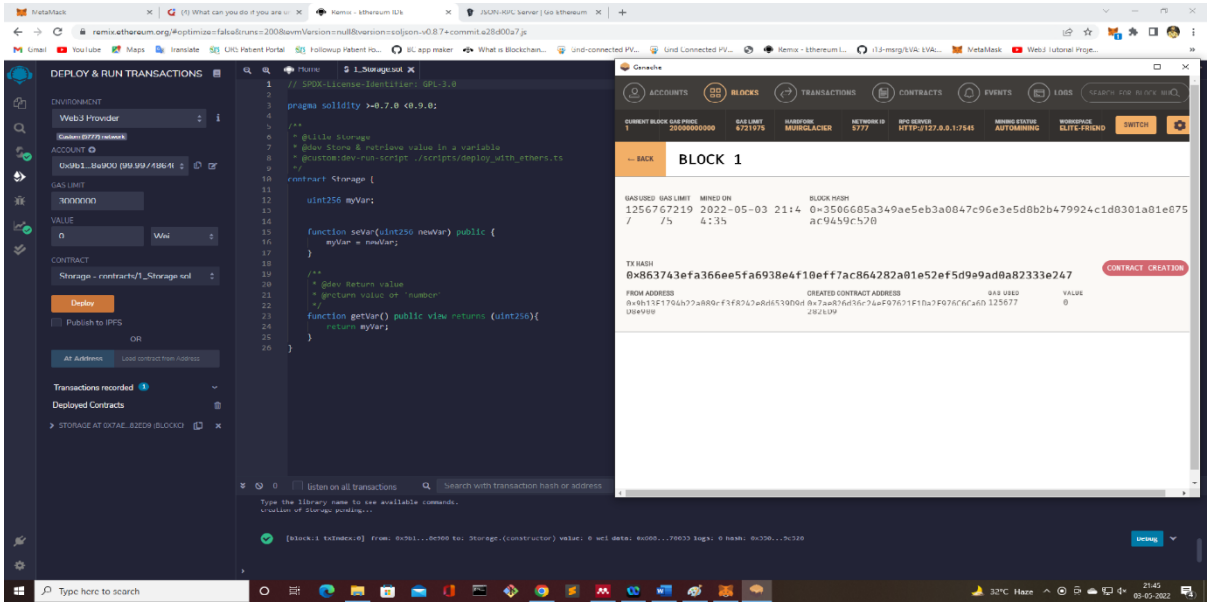
    a. Ganache

    b. Meta mask

Ans- firstly change the environment to “injected web3” and deploy the sol code. And after we get confirmation from meta mask shown in figure below.



-for connecting ganache, we change the environment to “web3 provider”, finally ganache is connected and we can see the transaction on Block wise shown in screenshot below.


DEPLOY & RUN TRANSACTIONS
Web3 Provider
Deploy
Transactions recorded
Deployed Contracts
BLOCK 1
CONTRACT CREATION

**********END**********