**CSED490U Blockchain & Cryptocurrency**

**Assignment 7**

**Submitted by- Sajan Maharjan**

**POVIS id- thesajan@postech.ac.kr**

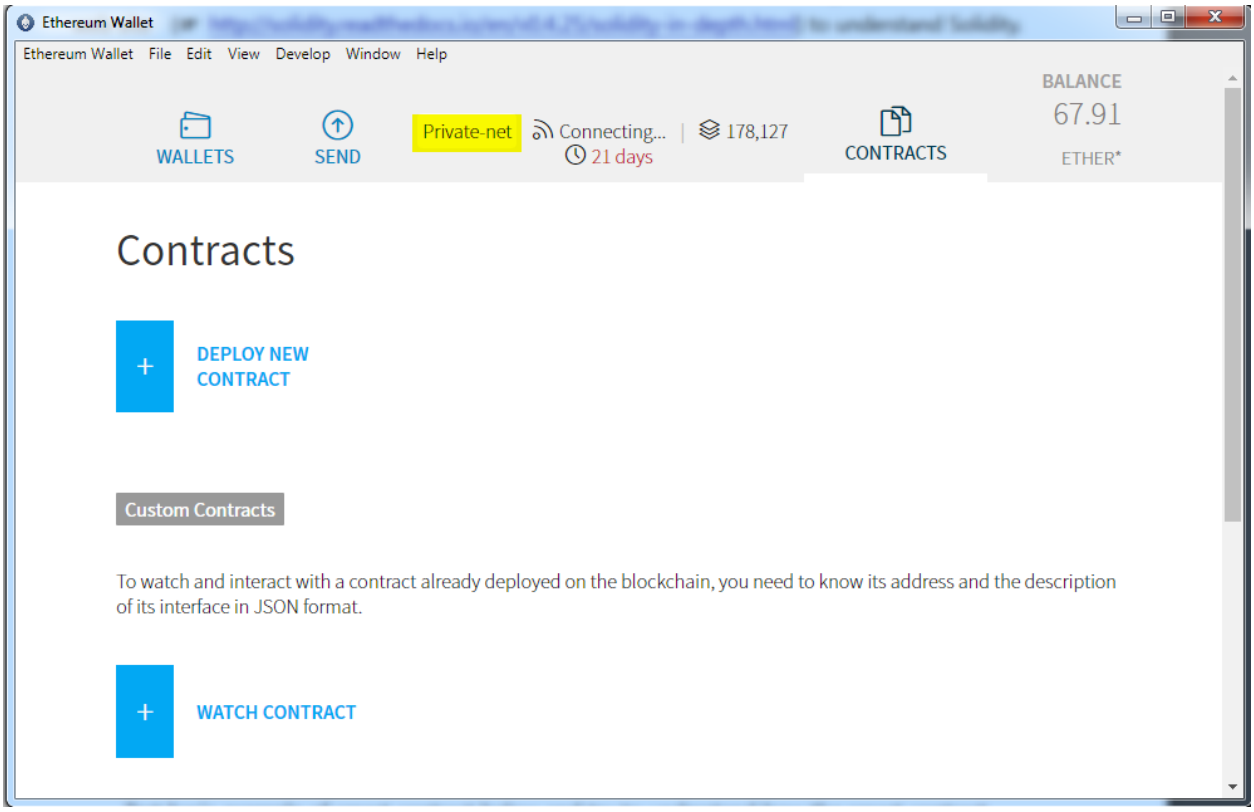
**Registration Number- 20182095**

**<Step. 1> Install Ethereum wallet / Mist**

> Ethereum wallet was installed for the Windows operating system using the link- https://github.com/ethereum/mist/releases

**<Step. 2> Connect to your own private network**

> Geth was initially launched and connected to a private network. Next, ethereum wallet application was launched and ethereum wallet was instantly connected to the operating instance of geth. i.e.




**<Step. 3> Create a simple smart contract**

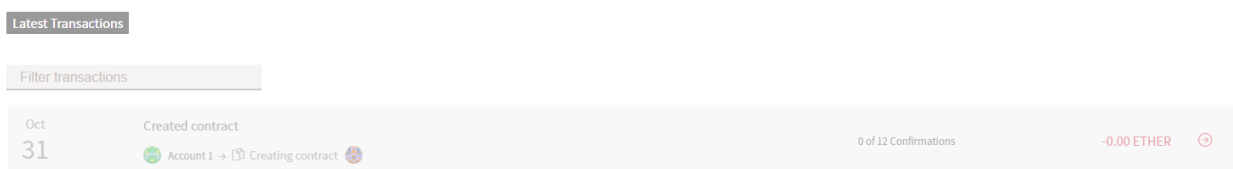
**3-1 Create new smart contract that contains the code below and compile it**

The given source code was saved into a solidity contract SimpleStorage.sol and compiled successfully to generate the contract byte code as shown below-

```
C:\Windows\system32\cmd.exe

C:\Users\sajan>solc --bin SimpleStorage.sol
SimpleStorage.sol:10:2: Warning: Function state mutability can be restricted to
view
        function get() public returns(uint) {
 ^ (Relevant source part starts here and spans across multiple lines).

======= SimpleStorage.sol:SimpleStorage =======
Binary:
6080604052348015610010576000080fd5b5060df8061001f6000396000f3006080604052600436106
0049576000357c0100000000000000000000000000000000000000000000000000000000900463ff
ffffff16806360fe47b114604e5780636d4ce63c146078575b600080fd5b348015605957600080fd
5b506076600480360381019080803590602001909291905050506a0565b005b348015608357600080
fd5b50608a60aa565b604051808281526020019150506040518091039f35b806000819055550505
565b60008054905090560a165627a7a723058208f5a3bcabd03405a670239a3b546759cfbbff476
a77e8ca78cc740ed72cd14ae0029

C:\Users\sajan>
```

**3-2) If you compile successfully, publish the smart contract on the private network**

```
> eth.accounts[0]
"0xf3f1679f34767be51c6dfe79d6750d5c90978338"
> INFO [10-31|18:29:12.662] Setting new local account                address=0xF
3F1679F34767be51c6dFE79d6750D5c90978338
INFO [10-31|18:29:12.682] Submitted contract creation              fullhash=0x92
3da02590a9af159c015b2a15b46ee69838f3358e1d135ee196aa6415db61a1 contract=0xC03FEb
A7F7c396929FBFC7BAf2B2C8B35A2819Fc

> INFO [10-31|18:59:05.194] Regenerated local transaction journal    transaction
s=1 accounts=1
INFO [10-31|19:59:05.194] Regenerated local transaction journal    transactions=
1 accounts=1
```




From the ethereum wallet snapshot shown above, it can be confirmed that the smart contract was sent to the ethereum blockchain but was not yet confirmed, as mining was not yet initiated.

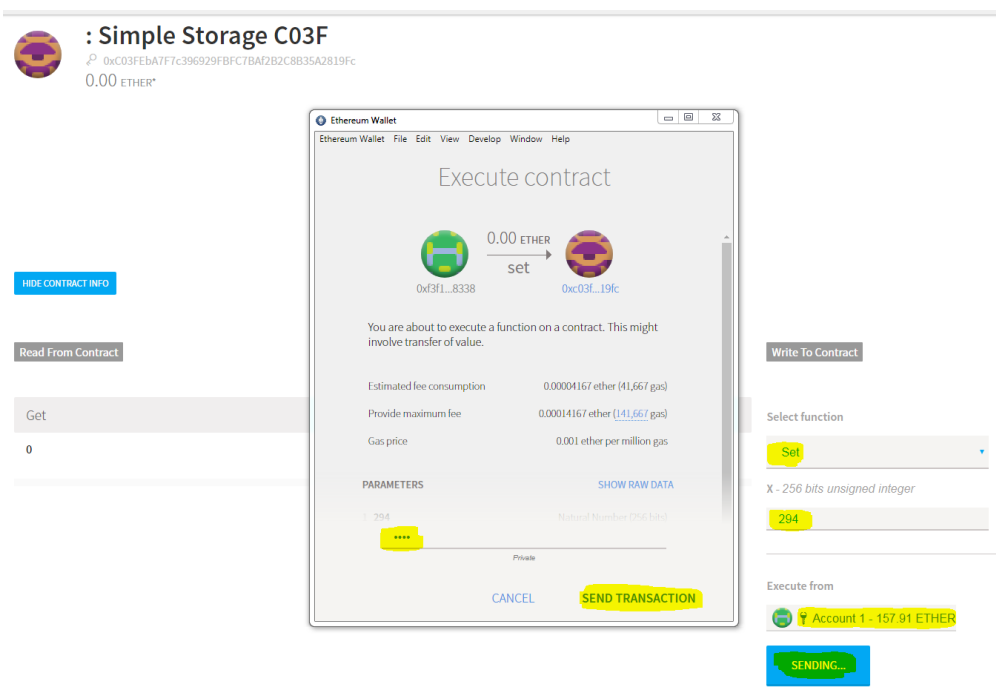
**3-3) After the block contains the transaction generated by the corresponding smart contract is mined, check the status of the smart contract in detail. Capture the status and attach it to the report.**

> miner.start() command was used to start mining and confirmation on the contract were increased slowly

```
> miner.start()
INFO [10-31|20:52:59.106] Updated mining threads                    threads=4
null
> INFO [10-31|20:54:10.611] Successfully sealed new block             number=1781
28 sealhash=27c9fe.851a55 hash=cd8ab6.6ef9bb elapsed=3m39.364s
INFO [10-31|20:54:10.620] ?? mined potential block                   number=17812
8 hash=cd8ab6.6ef9bb
INFO [10-31|20:54:10.655] Commit new mining work                    number=178129
 sealhash=954385.c7e581 uncles=0 txs=0 gas=0      fees=0             elapsed=35.00
2ms
INFO [10-31|20:54:30.877] Successfully sealed new block             number=178129
 sealhash=954385.c7e581 hash=37005b.dbc5bc elapsed=20.256s
INFO [10-31|20:54:30.886] ?? mined potential block                   number=17812
9 hash=37005b.dbc5bc
INFO [10-31|20:54:30.972] Commit new mining work                    number=178130
 sealhash=80252d.85ce1c uncles=0 txs=0 gas=0      fees=0             elapsed=86.00
4ms
```

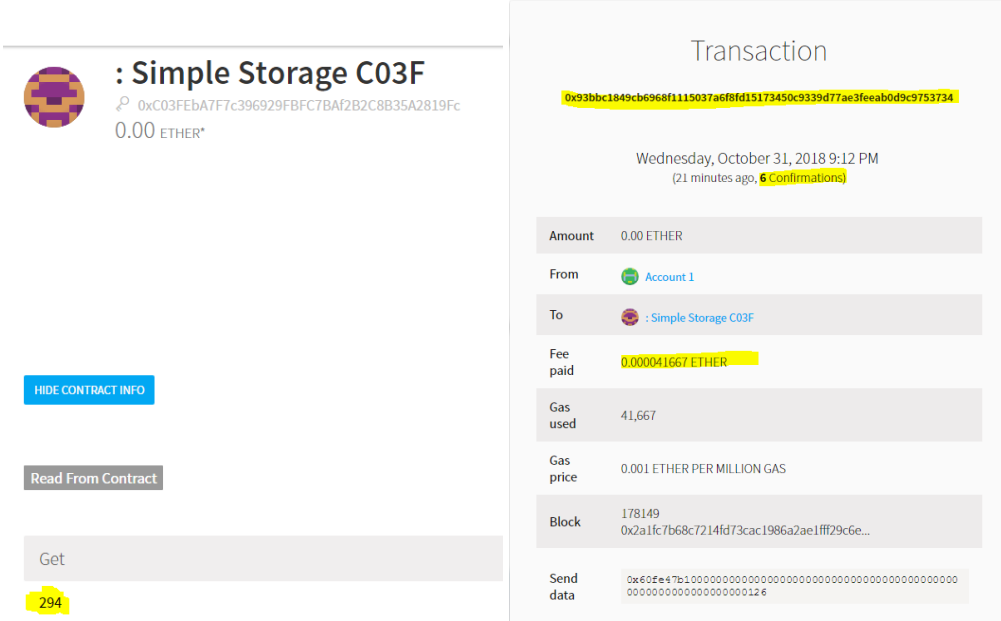

### 3-4) Call functions included in the smart contract (Send a transaction)

> Once the smart contract was written to the blockchain, it was possible to call the functions/ send transactions via the ethereum wallet interface-




```
1ms
INFO [10-31|21:12:41.877] Submitted transaction                     fullhash=0x93
bbc1849cb6968f1115037a6f8fd15173450c9339d77ae3feeab0d9c9753734 recipient=0xC03FE
bA7F7c396929FBFC7BAf2B2C8B35A2819Fc
INFO [10-31|21:12:42.510] Commit new mining work                    number=178149
 sealhash=2b3857.2b4bc9 uncles=0 txs=1 gas=41667  fees=4.1667e-05   elapsed=0s
INFO [10-31|21:13:11.130] Successfully sealed new block             number=178149
 sealhash=2b3857.2b4bc9 hash=2a1fc7.59bb21 elapsed=28.620s
```

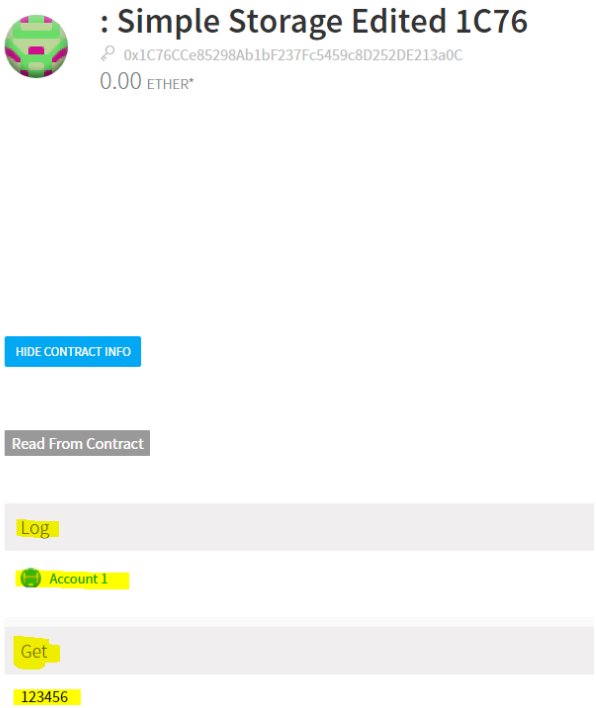
Once the transaction received sufficient confirmations, the value of the uint inside our contract- storedData is updated in the contract as follows-




**3-5) Add logging function to keep track of who recently changed data (Then, test it)**

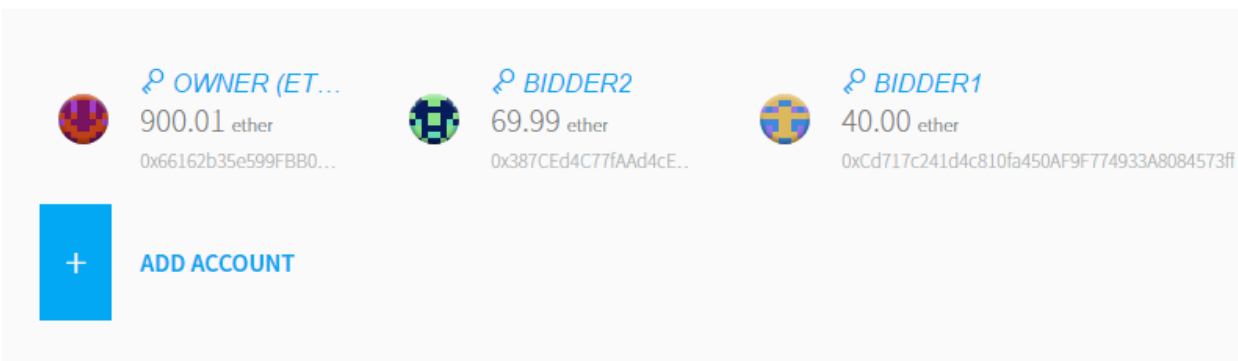
> A new solidity class file was created with the addition of the log function and deployed as a new contract and the functions were executed.

```
pragma solidity ^0.4.18;

contract SimpleStorageEdited {
    uint storedData;
    address changeAddress;

    function set(uint x) public {
        storedData = x;
        changeAddress = msg.sender;
    }

    function get() public view returns(uint) {
        return storedData;
    }

    function log() public view returns(address) {
        return changeAddress;
    }
}
```




**<Step. 4> Create a smart contract for auction**

**4-0) Prepare three accounts for testing your contract. (Account 1: Owner, Account 2: Bidder 1, Account 3: Bidder 2)**




**4-1) Complete Auction smart contract and save it as Auction.sol**

```
pragma solidity ^0.4.18;

contract Auction {
        address public owner; // address of the owner who can withdraw from the auction sale
        address public highestBidder; //the highest bidder's address
        uint public highestBid; //the amount of the highest bid
        mapping(address => uint) public userBalances; //mapping for the amount to return
        bool sold;

        function Auction() public {
```

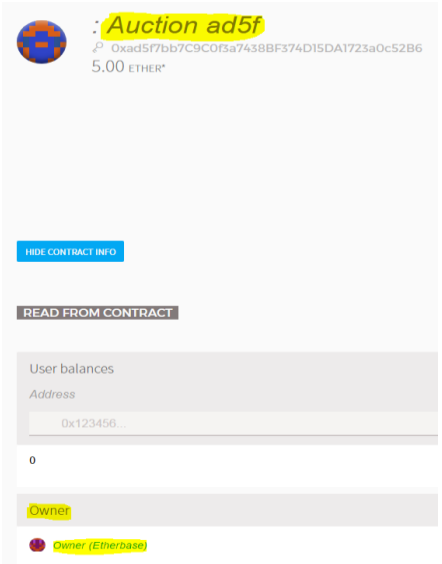
```
		sold = false;
		owner = 0x66162b35e599FBB0B0D66837C55be33304385D7e;
		highestBid = 0; //initialize highest bid with value 0
		highestBidder = msg.sender;
	}

	function bid() public payable {
		if(msg.value > highestBid) {
			if(highestBidder != 0) {
				userBalances[highestBidder] = highestBid;
			}
			highestBid = msg.value;
			highestBidder = msg.sender;
		}

	}

	function withdraw() public {
		// function to withdraw the amount of bid to return
		// 1. check if the amount to return is greater than zero
		// 2. update status variable and return bid
		address withdrawalAccount = msg.sender;
		uint withdrawalAmt = userBalances[withdrawalAccount];

		require(withdrawalAmt != 0);

		//send the amount back
		if(msg.sender.send(withdrawalAmt)) {
			userBalances[withdrawalAccount] -= withdrawalAmt;
		}
	}
}
```

**4-2) Compile the code using solc. (After installing solc, use solc –abi –bin)**

```
C:\Users\sajan>solc --abi --bin Auction.sol
Auction.sol:10:2: Warning: Defining constructors as functions with the same name
 as the contract is deprecated. Use "constructor(...) { ... }" instead.
        function Auction() public {
 ^ (Relevant source part starts here and spans across multiple lines).

======= Auction.sol:Auction =======
Binary:
6080604052348015610010576000080fd5b5060006004600061010000a81548160ff0219169083151502179055507366162b35e599fbb0b0d66837c55be33304385d7e600080610100 0a81548173ffffff
ffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffff
ffffff16021790555060006002819055503360016000610100 0a81548173ffffffffffffffffffff
ffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffff16021790
555061044780610 0d86000396000f3006080604052600436106100785760003 57c0100000000000
000000000000000000000000000000000000000000000000900463ffffffff1680631998aeef1461007d
57806326224c6414610087578063 3ccfd60b146100de5780638da5cb5b146100f557806391f90157
1461014c578063d57bde79146101a3575b600080fd5b610085610 1ce565b005b348015610093576 0
0080fd5b506100c860048036038101908080357 3ffffffffffffffffffffffffffffffffffffffff
1690602001909291905050506102ce565b604051808281526020019150506040518091039 0f35b34
80156100ea57600080fd5b506100f36102e6565b005b348015610101576000080fd5b5061010a6103
ca565b604051808273ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffff
ffffffffffffffffffff1681526020019150506040518091039 0f35b348015610158576000080fd
5b506101616103ef565b604051808273ffffffffffffffffffffffffffffffffffffffff1673ffff
ffffffffffffffffffffffffffffffffff1681526020019150506040518091039 0f35b34801561
01af57600080fd5b506101b8610415565b604051808281526020019150506040518091039 0f35b60
02543411156102cc576000600160009054906101000a900473ffffffffffffffffffffffffffff
ffffffff1673ffffffffffffffffffffffffffffffffffffffff161415156102835760025460 03
6000600160009054906101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffff
ffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff
16815260200190815260200160002819055505b346002819055550336001600061010 0a81548173
ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffff
ffffffffff1602179055505b565b600360205280600052604060002060009150905054815 65b60
0080339150600360008373ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffff
ffffffffffffffffffff16815260200190815260200160002054905060008114151515610 3
3e57600080fd5b3373ffffffffffffffffffffffffffffffffffffffff166108fc8290815029060
40516000604051808303818588 8f193505050501561 03c65780600360008473ffffffffffffffff
ffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200 1
908152602001600020600082825403925050819055505b5050565b600080905490610100 0a900473
ffffffffffffffffffffffffffffffffffffffff1681565b600160009054906101000a900473ffff
ffffffffffffffffffffffffffffffffff1681565b600254815600a165627a7a7230582037236 7
97e87037e4afc429e86fd8e11907f466911854c8699e1dccce23341cdd0029
Contract JSON ABI
[{"constant":false,"inputs":[],"name":"bid","outputs":[],"payable":true,"stateMu
tability":"payable","type":"function"},{"constant":true,"inputs":[{"name":"","ty
pe":"address"}],"name":"userBalances","outputs":[{"name":"","type":"uint256"}],"
payable":false,"stateMutability":"view","type":"function"},{"constant":false,"in
puts":[],"name":"withdraw","outputs":[],"payable":false,"stateMutability":"nonpa
yable","type":"function"},{"constant":true,"inputs":[],"name":"owner","outputs":
[{"name":"","type":"address"}],"payable":false,"stateMutability":"view","type":"
function"},{"constant":true,"inputs":[],"name":"highestBidder","outputs":[{"name
":"","type":"address"}],"payable":false,"stateMutability":"view","type":"functio
n"},{"constant":true,"inputs":[],"name":"highestBid","outputs":[{"name":"","type
":"uint256"}],"payable":false,"stateMutability":"view","type":"function"},{"inpu
ts":[],"payable":false,"stateMutability":"nonpayable","type":"constructor"}]
```

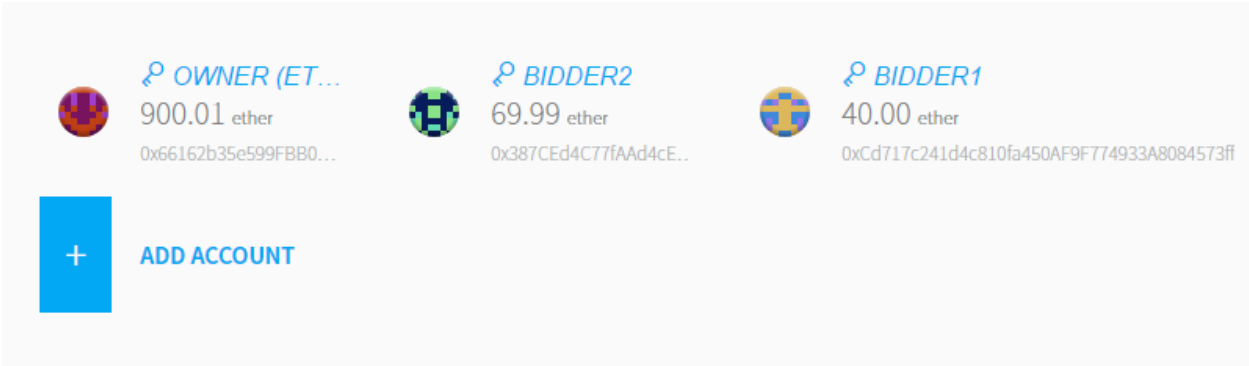
**4-3) Send transaction to create the contract on account 1 (owner)**

> As seen from step 4-0, 3 different accounts had been created prior. The contract was created using ethereum wallet by owner account



**4-4) Check the balance of contract account and account 1 and account 2, the amount of highest bid and the address of the higest bidder**

> The pre-existing balance of owner, account 1 and account 2 were as follows-

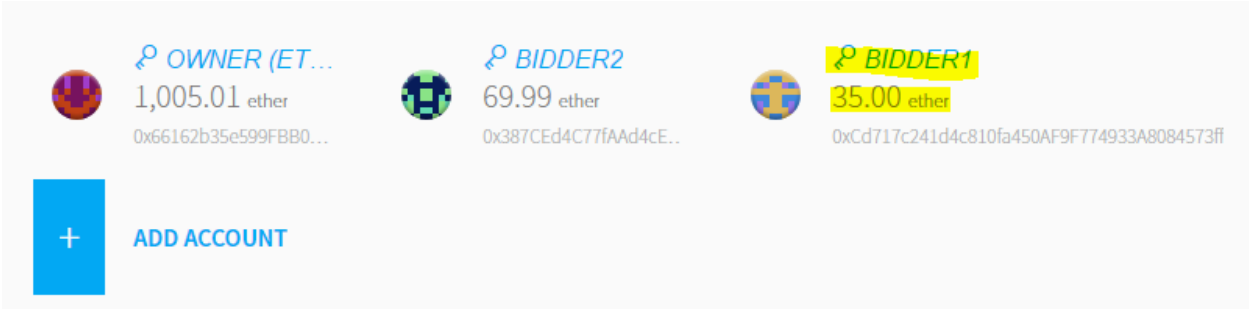


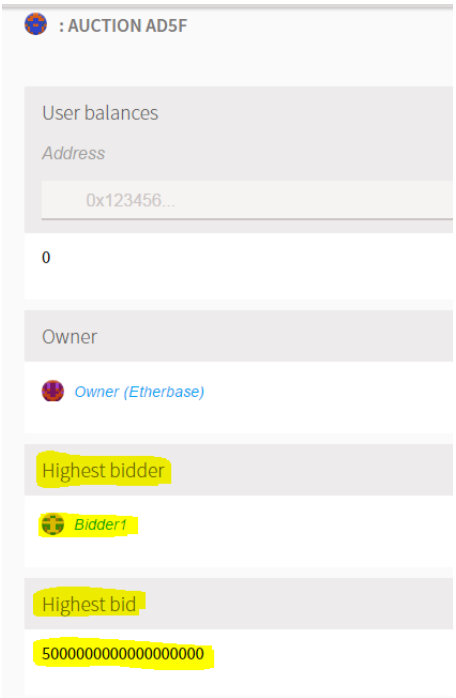

Highest bid was initialized to 0 and highest bidder was initialized to *msg.sender* in the constructor

**4-5) Send transaction to execute bid() function in order to bid 5ETH, on account 2(Bidder 1)**

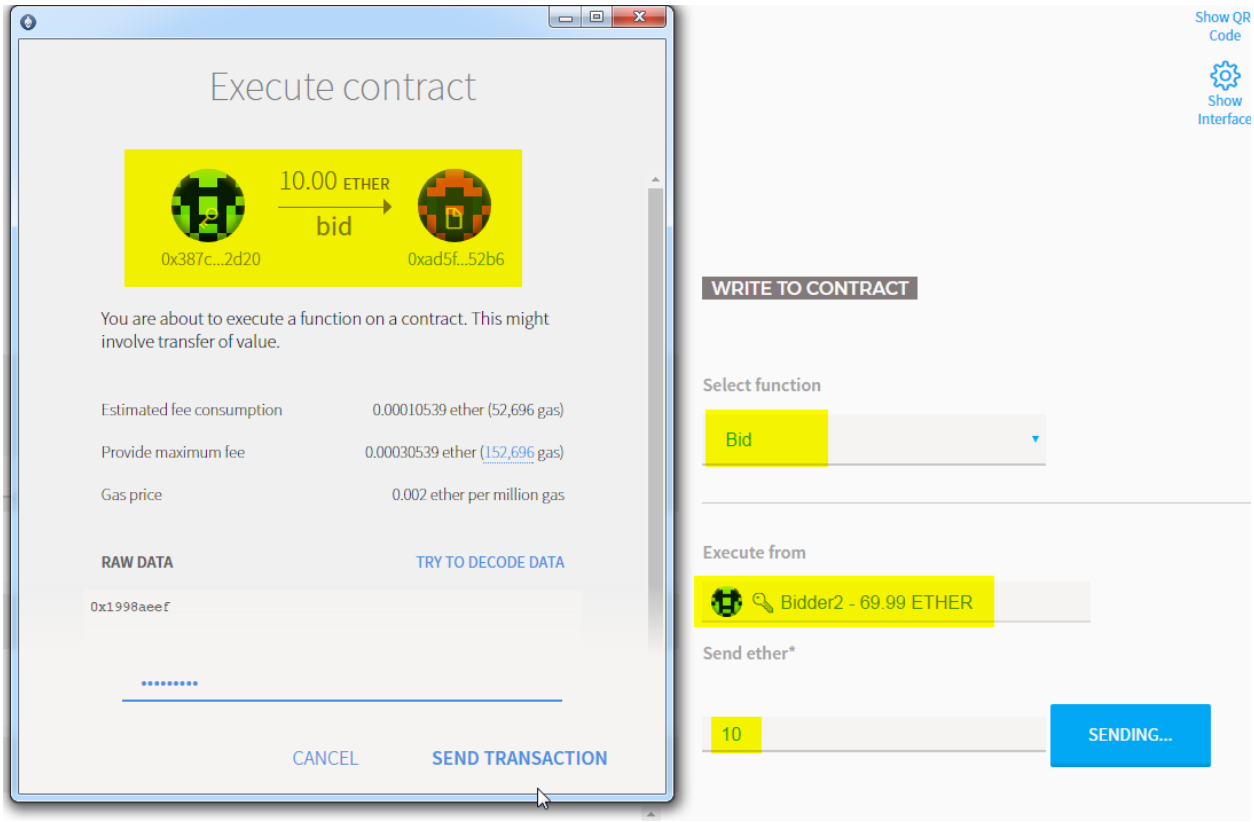
> Bidder 1 put a bid() with 5ETH using the ethereum wallet

4-6) Check the balance of the contract account, account 2 and account 3, the amount of highest bid and the address of the highest bidder

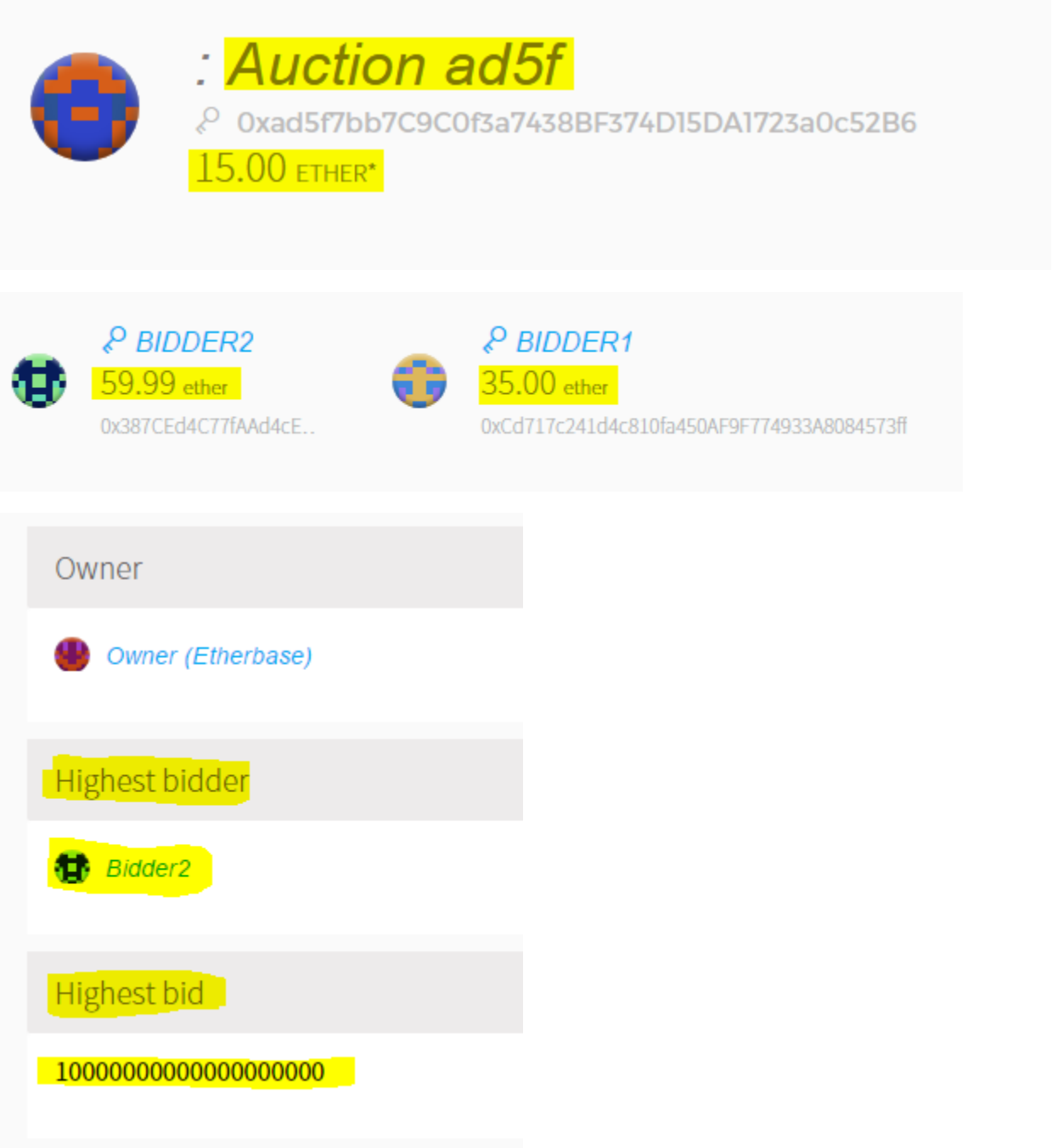





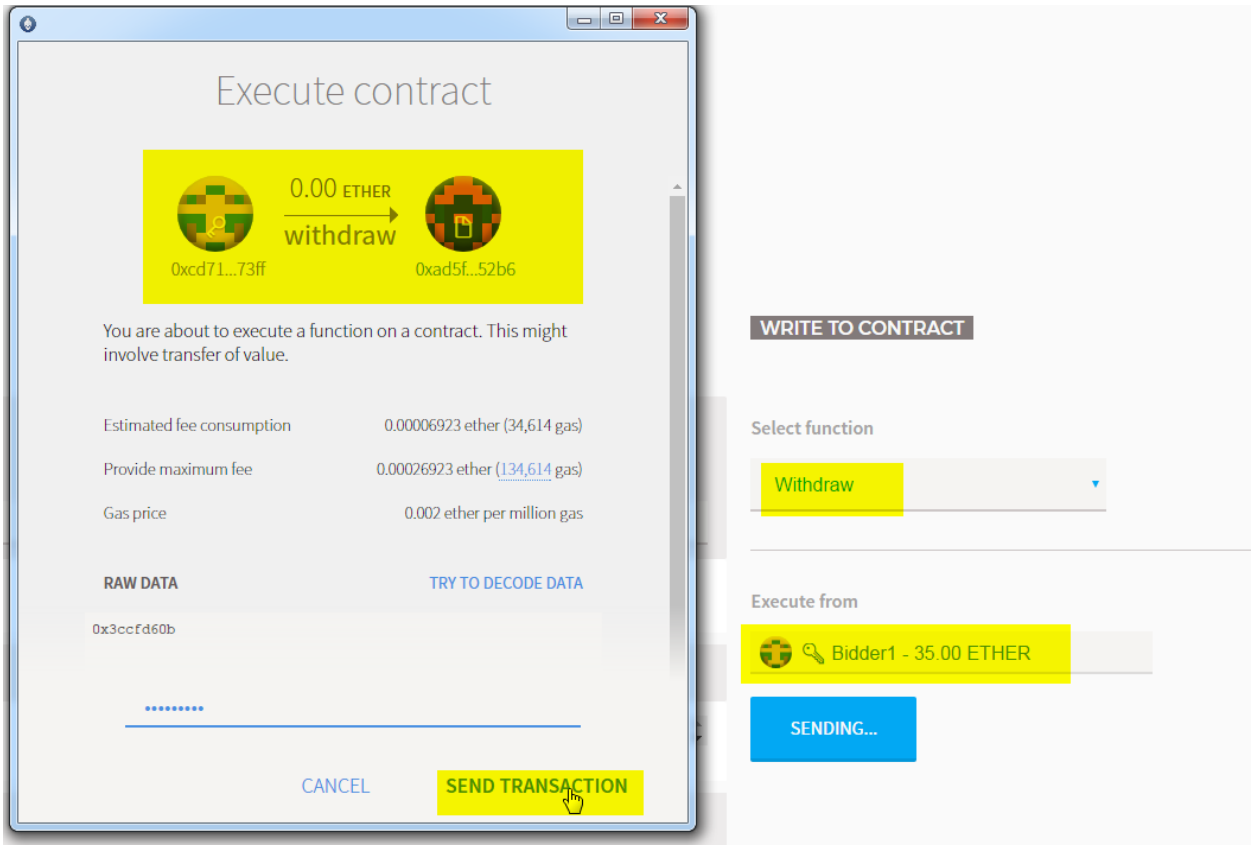

**4-7) Send transaction to execute bid() function in order to bid 10ETH, on account 3(bidder 2).**



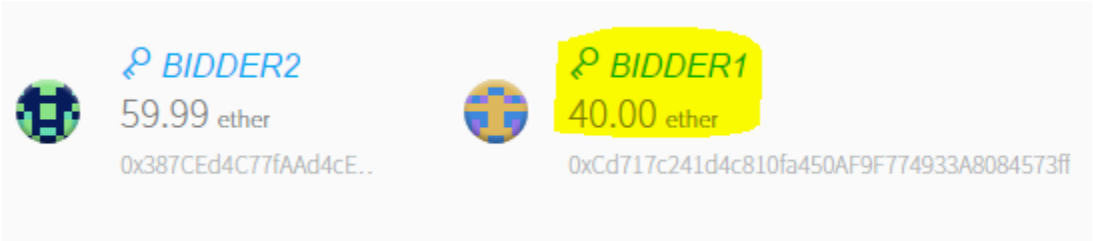

**4-8) Check the balance of contract account and account 3, the amount of the highest bid and the address of the highest bidder**




**4-9) Send transaction to execute withdraw() function in order to withdraw the amount of bid, on account 2 (bidder 1)**




**4-10) Check the balance of contract account and account 2(bidder 1)**

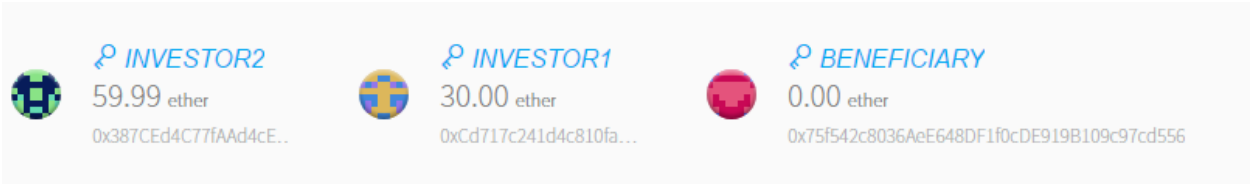




**<Step. 5> Create a smart contract for crowd funding**

**5-0) Prepare three accounts for testing your contract. (Account 1: Beneficiary, Account 2: Investor1, Account 3: Investor2)**

> The three accounts were prepared as-




**5-1) Complete crowdfunding smart contract and save it as "CrowdFunding.sol"**

```
pragma solidity ^0.4.18;
	contract CrowdFunding {
	// Investor struct
	struct Investor {
		address addr; // investor's address
		uint amount;	// investment amount
	}
	address public beneficiary;	// contract beneficiary
	uint public numInvestors;	// the number of investors
	uint public deadline;	// deadline for this contract to be closed
	string public status; // "Funding", "Campagin Success", "Campagin Failed"
	bool public end;	// the end of funding
	uint public goalAmount; // target amount
	uint public totalAmount;	//total ammout
	Investor[] public investors; //declare an array of structs of Investors to store records

	// 1. Create modifier to limit to beneficiary
	modifier onlyBeneficiary() {
		require(msg.sender == beneficiary);
		_;
	}

	modifier beforeDeadline() {
		require(now < deadline);
		_;
	}

	modifier afterDeadline() {
		require(now >= deadline);
		_;
	}

	modifier notEnded() {
		require(end == false);
		_;
	}

	// Constructor
	function CrowdFunding(uint _duration, uint _goalAmount) public {
```

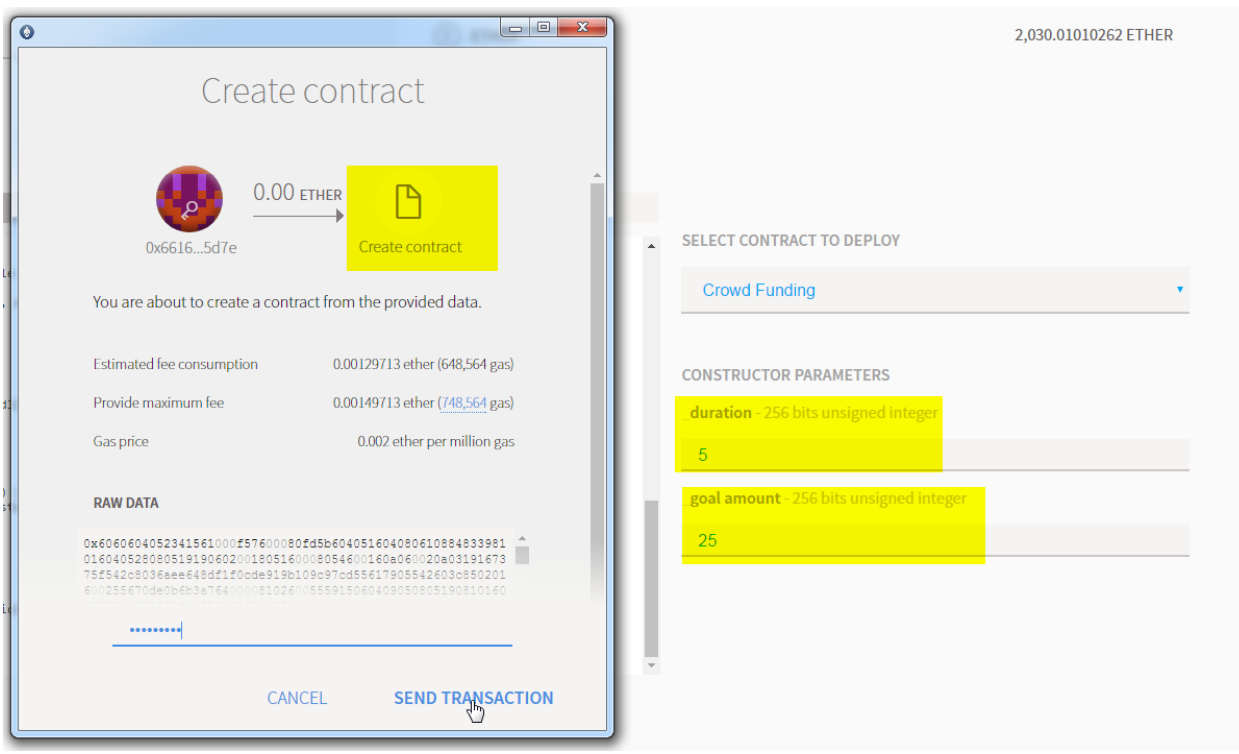
```
        beneficiary = 0x75f542c8036AeE648DF1f0cDE919B109c97cd556;
        deadline = now + _duration * 1 minutes;
        goalAmount = _goalAmount * 1e18;
        status = "Funding";
        end = false;
        numInvestors = 0;
        totalAmount = 0;
    }

    // Function to be called when investing
    function fund() public beforeDeadline payable notEnded {
        uint amt = msg.value;
        Investor memory i = Investor(msg.sender, amt);
        investors.push(i);
        totalAmount += amt;
        numInvestors++;

    }

    function checkGoalReached() public afterDeadline {
        if(totalAmount >= goalAmount) {
            beneficiary.transfer(totalAmount);
            status = "Campaign Success";
        }
        else {
            for(uint k=0; k < numInvestors; k++) {
                investors[k].addr.transfer(investors[k].amount);
            }
            status = "Campagin Failed";
        }
        end = true;
    }

    function destroyContract() public onlyBeneficiary {
        selfdestruct(beneficiary);
    }

}
```

**5-2) Compile the code using solc**

(You can use ***solc –abi –bin CrowdFunding.sol*** command to check)

```
C:\Users\sajan>solc --bin --abi CrowdFunding.sol
CrowdFunding.sol:39:2: Warning: Defining constructors as functions with the same
 name as the contract is deprecated. Use "constructor(...) { ... }" instead.
        function CrowdFunding(uint _duration, uint _goalAmount) public {
 ^ (Relevant source part starts here and spans across multiple lines).

======= CrowdFunding.sol:CrowdFunding =======
Binary:
6080604052348015610010576000080fd5b506040516040806 10aec833981018060405281019080 80
5190602001909291908051906020019092919050505073 75f542c8036aee648df1f0cde919b109c9
7cd556600080610100 0a81548173ffffffffffffffffffffffffffffffffffffffff021916908373
ffffffffffffffffffffffffffffffffffffffff16021790555060 3c820242016002819055508060
058190555060408051908101604052806007815260 20017f46756e64696e67000000000000000000
000000000000000000000000000000000081525060039080519060 2001906100ef929190610122565b
5060006004600061010 00a81548160ff021916908315150217905555060006001819055506000600 6
819055505050610 1c7565b828054600181600116156101000203166002900490600052602060002 0
90601f016020900481019282601f106101635780516 0ff191683800117855561019156 5b82800160
```

**5-3) Send transaction to create contract (When creating contract, set duration to 5mins and goal amount to 25ETH)**




**5-4) Check deadline and goal amount and end state**

> We here consider a latency of 1 minute (while taking snapshot and writing the contract in the blockchain) causing remaining time to be 4 mins as shown below. The Goal amount is shown in Wei units which is 1 ETH = 10^18 Wei.

Goal amount

25000000000000000000

Deadline

1541145133 *(in 4 minutes)*

Beneficiary

*Beneficiary*

End

NO

: CROWD FUNDING EE5F

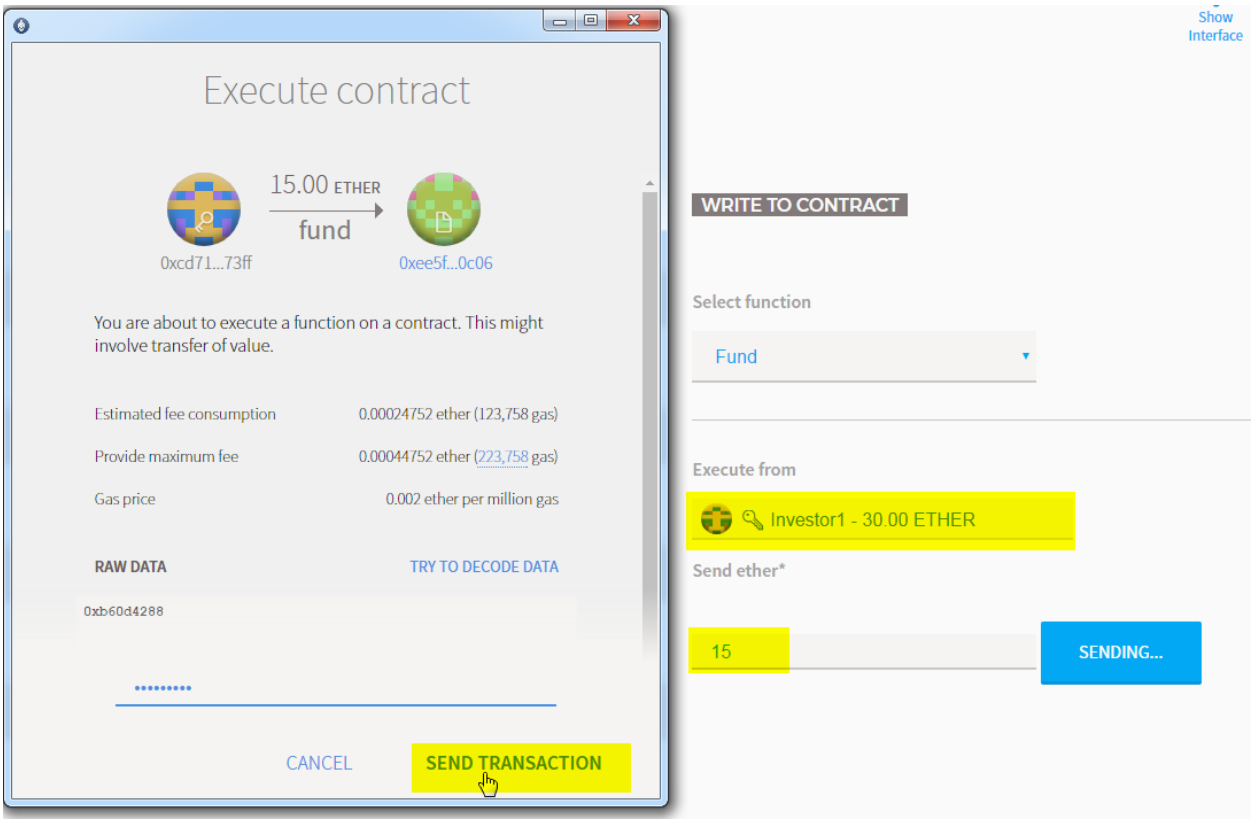
25000000000000000000

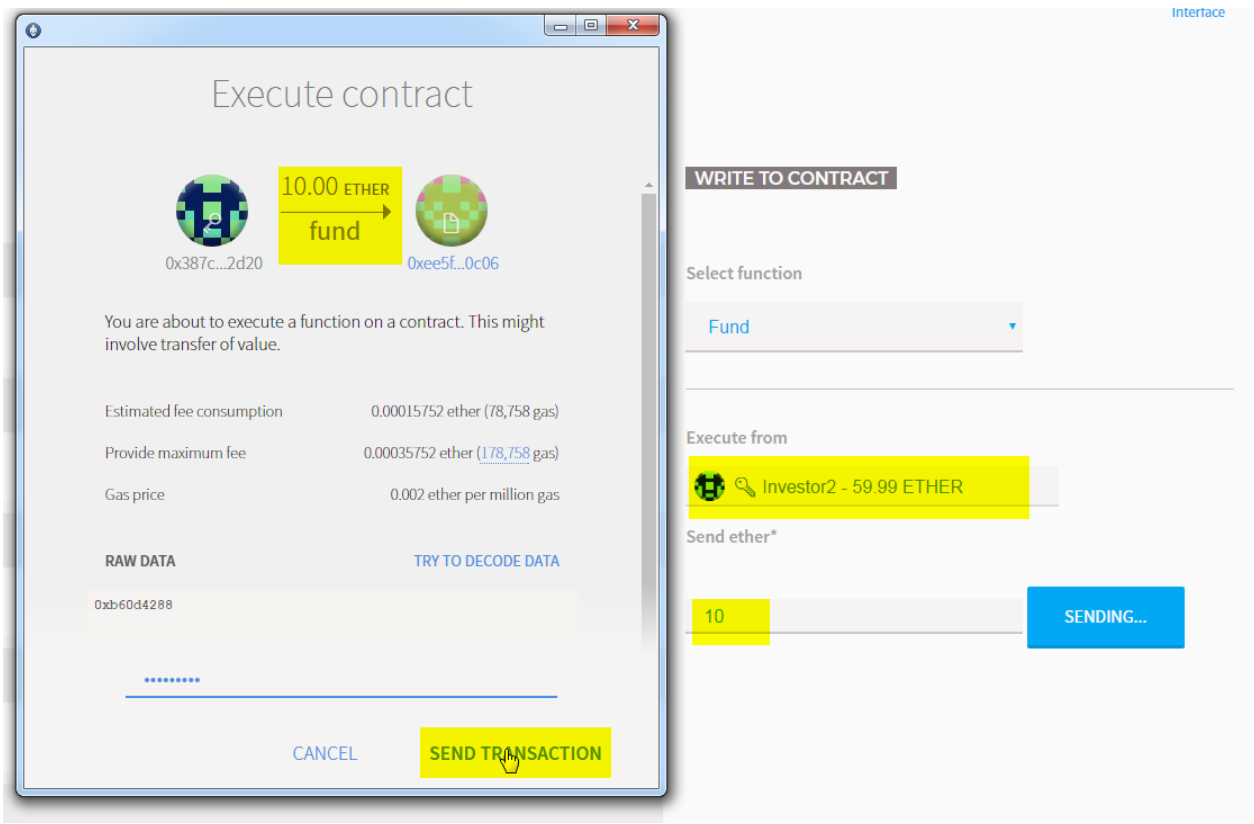
Status

Funding

**<<Case of Successful Funding>>**

**5-5) Send transaction to execute fund() function in order to fund 15ETH and 10ETH from Investor 1 and Investor 2 respectively**


Execute contract
15.00 ETHER
fund
0xcd71...73ff
0xee5f...0c06
You are about to execute a function on a contract. This might involve transfer of value.
Estimated fee consumption 0.00024752 ether (123,758 gas)
Provide maximum fee 0.00044752 ether (223,758 gas)
Gas price 0.002 ether per million gas
RAW DATA
TRY TO DECODE DATA
0xb60d4288
CANCEL
SEND TRANSACTION
Show Interface
WRITE TO CONTRACT
Select function
Fund
Execute from
Investor1 - 30.00 ETHER
Send ether*
15
SENDING...


Execute contract
10.00 ETHER
fund
0x387c...2d20
0xee5f...0c06
You are about to execute a function on a contract. This might involve transfer of value.
Estimated fee consumption 0.00015752 ether (78,758 gas)
Provide maximum fee 0.00035752 ether (178,758 gas)
Gas price 0.002 ether per million gas
RAW DATA
TRY TO DECODE DATA
0xb60d4288
CANCEL
SEND TRANSACTION
Interface
WRITE TO CONTRACT
Select function
Fund
Execute from
Investor2 - 59.99 ETHER
Send ether*
10
SENDING...

```
> INFO [11-02|17:15:50.444] Submitted transaction                    fullhash=0x
dd617ec9aa5c4dd5a4f98c4109d7fa0cd0c064c1fb553ea2f90b78e8b3958af4 recipient=0xEE5
F02F97cB0B9B013c59b7F88517703E3A70c06
INFO [11-02|17:17:02.333] Submitted transaction                    fullhash=0xbb
8d37d682ef1f2e7af154f26bb5a158c03cb01716e7f6b1f24ebadd75d6a453 recipient=0xEE5F0
2F97cB0B9B013c59b7F88517703E3A70c06
```

**5-6) Check investment of Investor 1 and 2**

Investors

*256 bits unsigned integer*

0

Addr

*Investor1*

Amount

15000000000000000000

End

NO

Investors

*256 bits unsigned integer*

1

Addr

*Investor2*

Amount

10000000000000000000

End

NO

**5-7) Check total investment of the contract and the balance of the contract account**

Num investors

2

Total amount

25000000000000000000

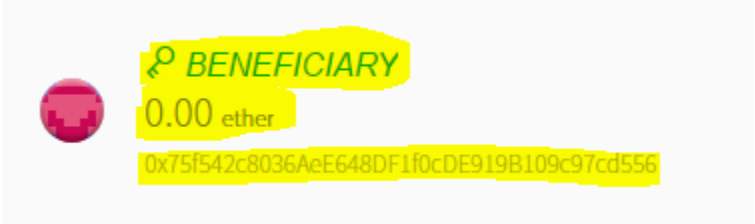
: Crowd Funding ee5f

0xEE5F02F97cB0B9B013c59b7F88517703E3A70c06

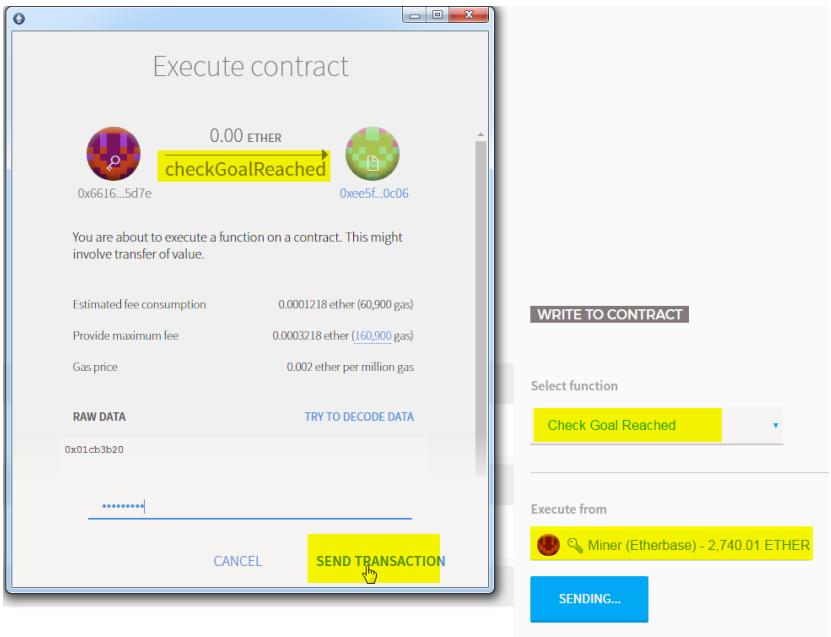
25.00 ETHER*

### 5-8) Check Beneficiary's account

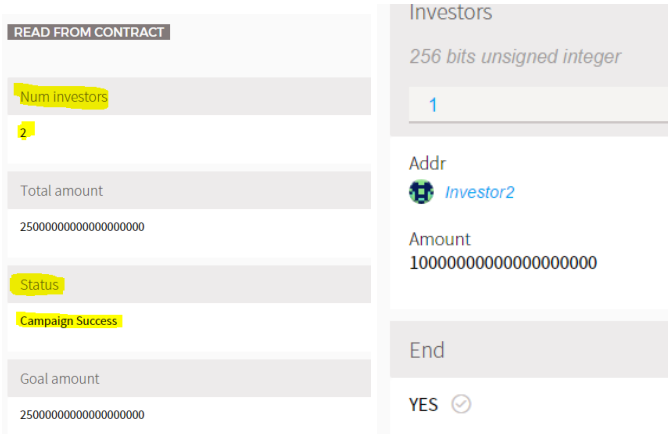
> We initialized the Beneficiary's account with balance 0 in the first place and we can check the beneficiary's account balance from ethereum wallet GUI as-




### 5-9) Send transaction to execute *checkGoalReached()* function to check the fundraising results (After deadline)

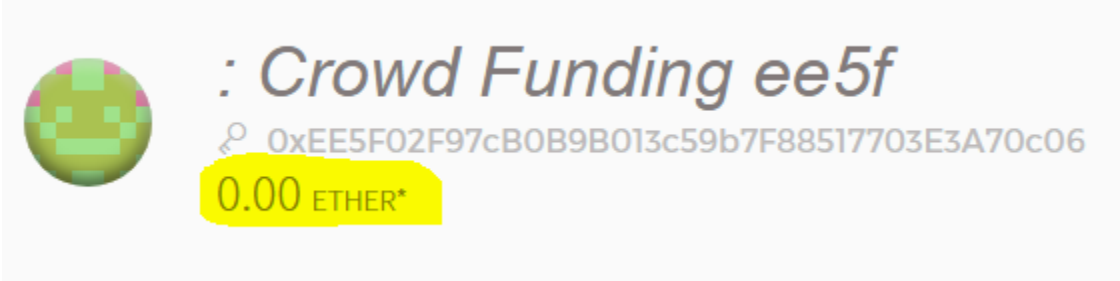


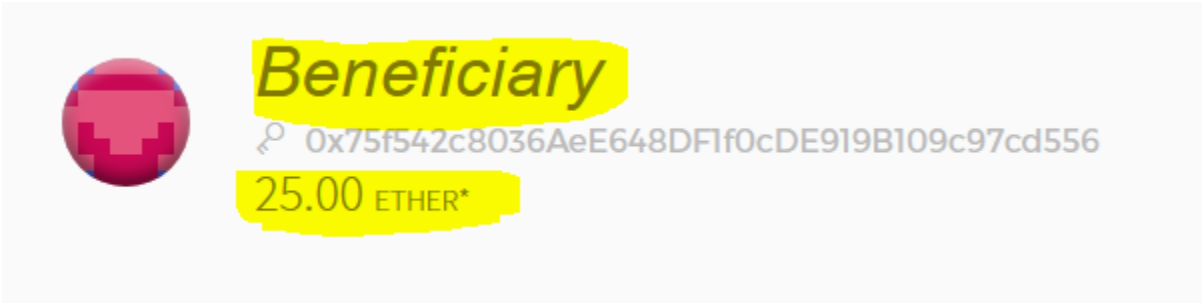

### 5-10) Check deadline and end state



**5-11) Check balance of contract account and owner**

> On success of crowdfunding, the collected fund is moved from contract account to the beneficiary account as shown below-

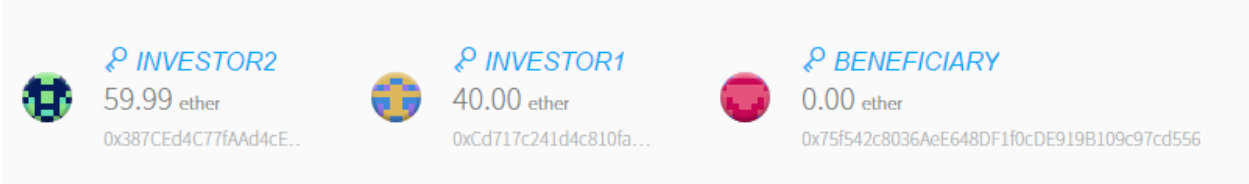


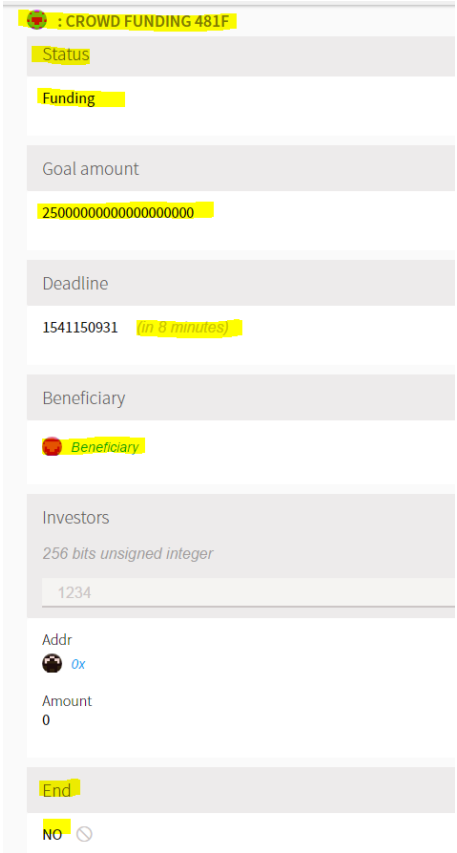




**<<Case of failed funding>>**

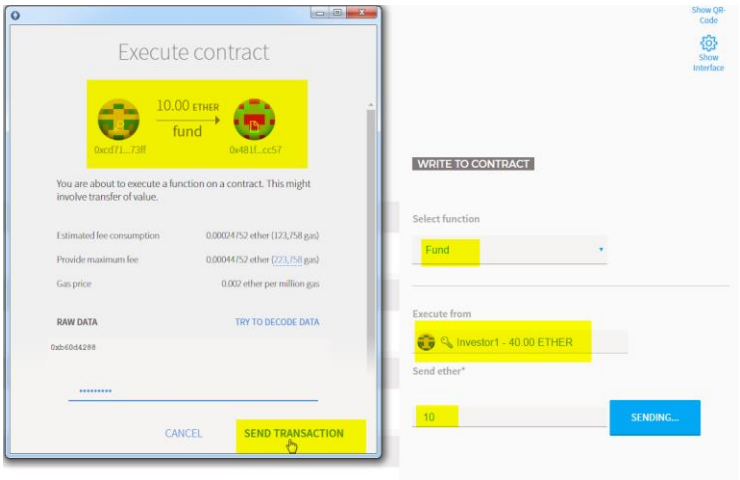
**5-12) Follow steps from 5-3) and 5-4)**

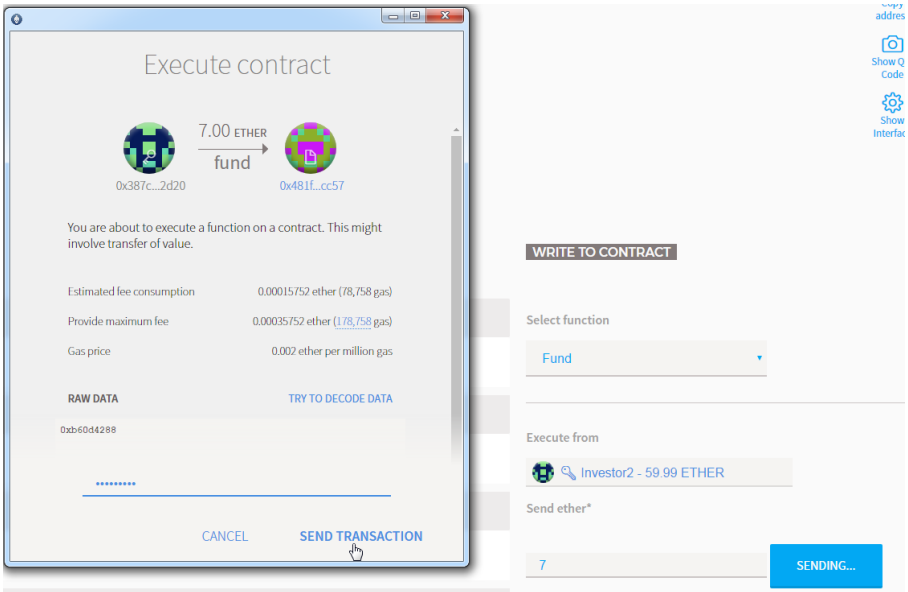
> A new contract was created with the goal amount of 25ETH and a deadline in the next 10 minutes and we ***re-started*** with the following balances for investor 1, investor 2 and beneficiary



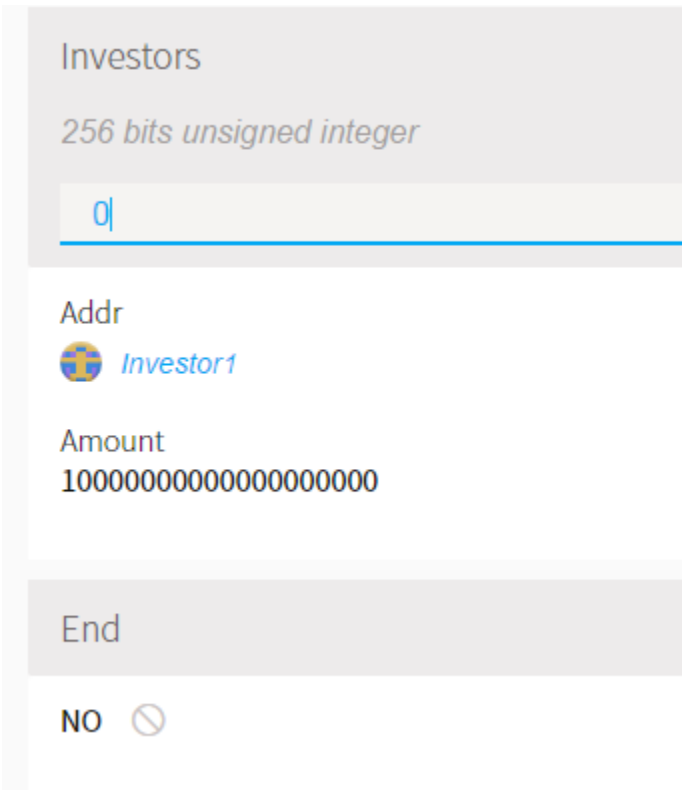


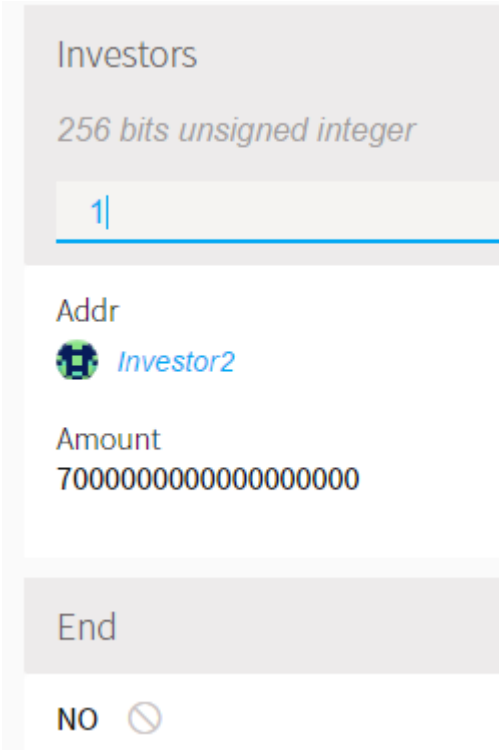

**5-13) Send transaction to execute fund() function in order to fund 10ETH from 7ETH from Investor 1 and Investor 2 respectively**






**5-14) Check investment of investor 1 and 2**




**5-15) Check total investment and balance of the contract account**

Num investors

2

Total amount

17000000000000000000

Status

Funding

Goal amount

25000000000000000000

: Crowd Funding 481f

0x481F6dc601Bd87A3DaE828F624B2c4bE8D4fCC57

17.00 ETHER*

**5-16) Check balance of investor 1, investor 2 and beneficiary account**

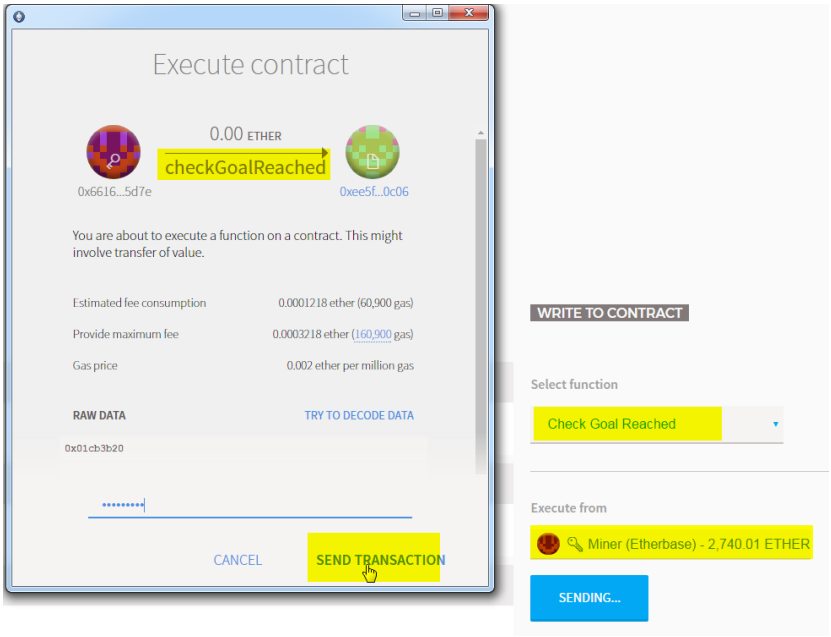
INVESTOR2
52.99 ether
0x387CEd4C77fAAd4cE..

INVESTOR1
30.00 ether
0xCd717c241d4c810fa...

BENEFICIARY
0.00 ether
0x75f542c8036AeE648DF1f0cDE919B109c97cd556

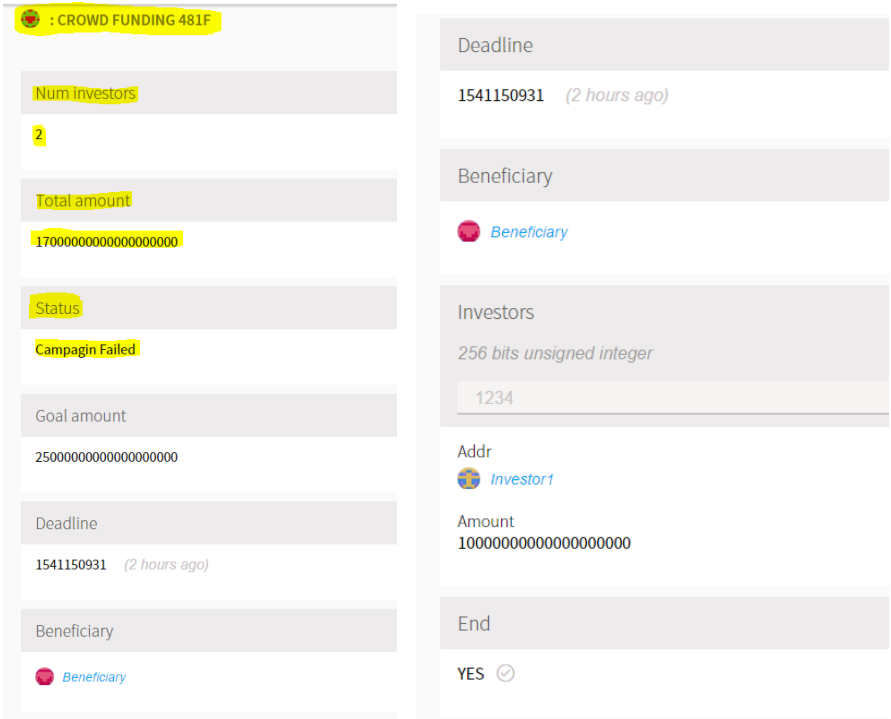
**5-17) Send transaction to execute checkGoalReached() function to check the fundraising results (After deadline)**

> checkGoalReached function was executed via the GUI interface of etheruem wallet. The function being submitted to the blockchain can be seen from the snapshot below-





## 5-18) Check deadline and end state



**5-19) Check balance of contract account**


**5-20) Check balance of investor 1, investor 2 and beneficiary account**

> After the failure of funding, the collected funds were re-distributed to the original investors automatically, which can be confirmed by the snapshot below-