분산컴퓨팅

Assignment #2 Ethereum-based auction system development

컴퓨터 소프트웨어 학부 2016025469 서건식

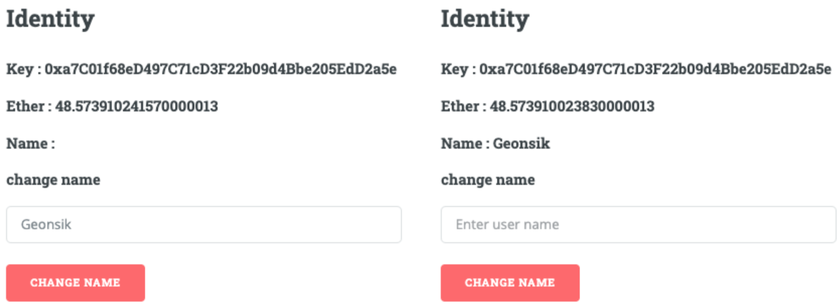
1. getName()

구현한 함수 별로 Client / Solidity 코드, 그리고 실행결과를 설명을 드리겠습니다.

```
348  var getName = async function() {
349      var address = await $('#address').text();
350      var name = await contract.methods.getName().call({from:address});
351      $('#name').text(name);
352  }
353
```

Client

Client 코드에서는 address의 값을 비동기처리 하여 가져오고, name값은 web3.js의 call함수를 사용해서 불러옵니다. (view함수) 그 이후 id가 name인 html 코드에 Jquery로 입력해줬습니다.




```
struct User{
    string Name; // User's Name
    string[] Items; // Items
}

mapping(address => User) users;

function getName() public view returns(string memory){
    return users[msg.sender].Name;
}
```

Solidity

Solidity에서 각 유저마다 고유한 Name과 Item list를 유지시키기 위해 구조체를 선언했고, 이를 mapping함수를 통해서 주소를 key값, User 구조체를 Value값으로 만들었습니다.

getName()은 view함수로써, 위에서 만들었던 storage 변수들의 값 중 한 주소에 매핑되는 구조체의 Name 멤버변수만을 반환합니다.

2. registerName()

```
356 var registerName = async function() {
357     var address = await $('#address').text();
358     var name = await document.getElementById('change_name').value;
359     try{
360         await contract.methods.registerName(name).send({from : address});
361     }
362     catch(err){
363         alert(err);
364     }
365 }
366
```

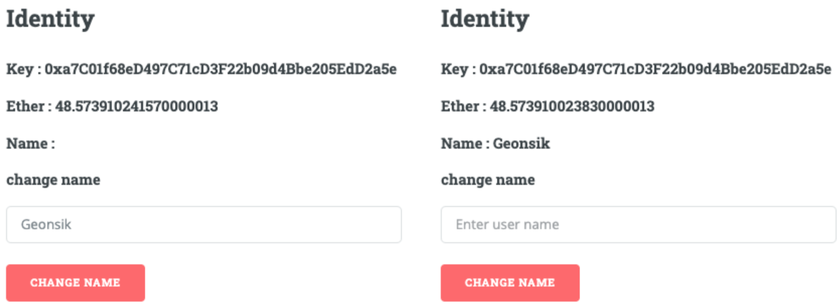
Client

```
function registerName(string memory name) public{
    users[msg.sender].Name = name;

}
```

Solidity

Address와 변한 입력한 이름을 가져오고, send함수를 통해 contract 함수를 실행하며 인자로 name을 넣습니다.

Solidity 함수는 간단합니다. 인자로 받은 name을 msg.sender에 매핑되는 구조체 User의 멤버변수 name에 저장합니다.



3. registerForMyItem() / registerItem()

```
367 var registerForMyItem = async function() {
368     var address = await $('#address').text();
369     var item = await document.getElementById('Item').value;
370
371     try{
372         await contract.methods.registerItem(item).send({from:address});
373
374     }
375     catch(err){
376         alert(err);
377     }
378 }
```

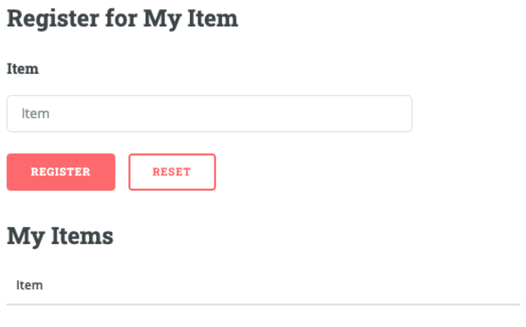
Client

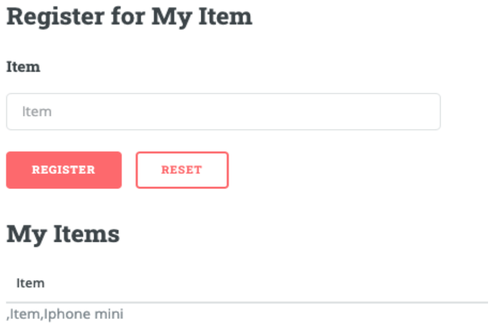
```
function registerItem(string memory name) public{
    users[msg.sender].Items.push(name);
}
```

Solidity

Address와 변한 입력한 이름을 가져오고, 입력된 Item name값을 contract의 registerItem() 함수의 인자값으로 넘습니다.

registerItem()도 Solidity 함수는 간단합니다.
인자로 받은 name을 msg.sender에 매핑되는 구조체 User의 멤버변수 Items 배열에 push 함수로 추가합니다.





Item 목록에 Item / Iphone mini를 등록한 상태입니다.

4. getMyItems() / getMyItems()

```
var getMyItems = async function() {
    var address = await $('#address').text();
    var items = new Array(); //배열선언
    items.push(await contract.methods.getMyItems().call({from:address}));

    $('#myItems').text(items);
}
```

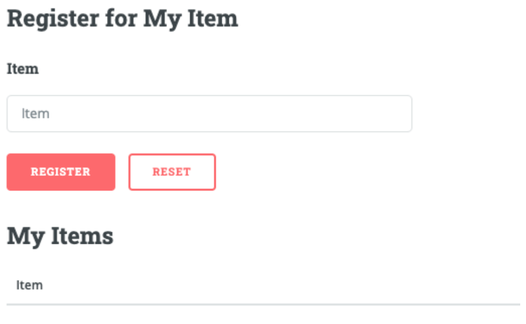
Client

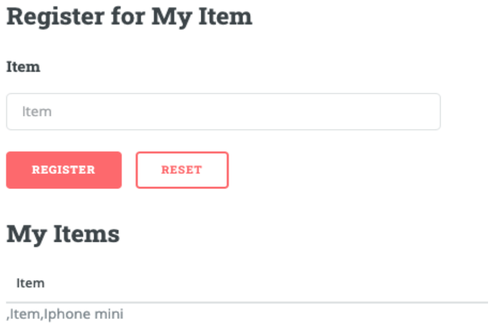
```
function getMyItems() public view returns(string[] memory){
    return users[msg.sender].Items;
}
```

Solidity

Register한 Item의 목록을 가져옵니다. Web3.js에서는 call method로 컨트랙트 내 getMyItems()함수를 호출하여 값을 가져옵니다.

Msg.sender에 매핑되는 User 구조체의 Item 목록을 반환합니다.





Item 목록에 Item / Iphone mini를 등록한 상태입니다.

5. getMyItemsToBeAuctioned() / getMyItems()

```
379  var getMyItemsToBeAuctioned = async function() {
380      var address = await $('#address').text();
381      var items = await contract.methods.getMyItems().call({from:address});
382      // items = items.split(",")
383      for(var i=0 ; i<items.length ; i++)
384      {
385          if(items[i] == "" || items[i] == " ")
386          {
387              continue;
388          }
389          option_value = items[i];
390          option_text = items[i];
391          $('#myitems-category').append('<option value="'+ option_value +'">'+ option_text
392      }
393
```

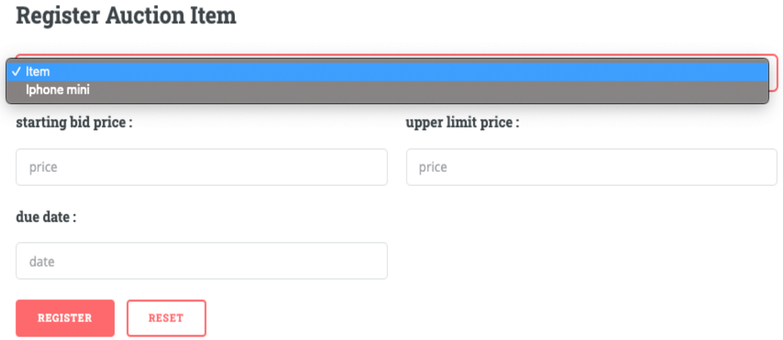
Client

```
function getMyItems() public view returns(string[] memory){
    return users[msg.sender].Items;
}
```

Solidity

Register한 Item의 목록을 가져옵니다. Web3.js에서는 call method로 컨트랙트 내 getMyItems()함수를 호출하여 값을 가져옵니다. myitems-category id를 가진 코드에 option value를 할당하여 코드를 추가하는 식으로 코드를 작성했습니다.

Msg.sender에 매핑되는 User 구조체의 Item 목록을 반환합니다.



Item 목록에 Item / Iphone mini를 등록한 상태입니다.

6. registerAuctionItem() / registerAuctionItem(item,start price,limit price,date)

```
var registerAuctionItem = async function() {
    await document.getElementById('change_name').value;
    var address = await $('#address').text();
    var items = await document.getElementById('myitems-category').value;
    var startingBidPrice = await document.getElementById('startingBidPrice').value;
    var upperLimitPrice = await document.getElementById('upperLimitPrice').value;
    var dueDate = await document.getElementById('dueDate').value;
    try{
        await contract.methods.registerAuctionItem(items,startingBidPrice,upperLimitPrice,dueDate).send({from:address, gas:5000000});
    }
    catch(err){
        alert(err)
    }
}
```

Client

Client코드는 간단합니다. 이전 함수들과 같이 입력된 값을 받아와서 registerAuctionItem() 함수를 실행합니다.

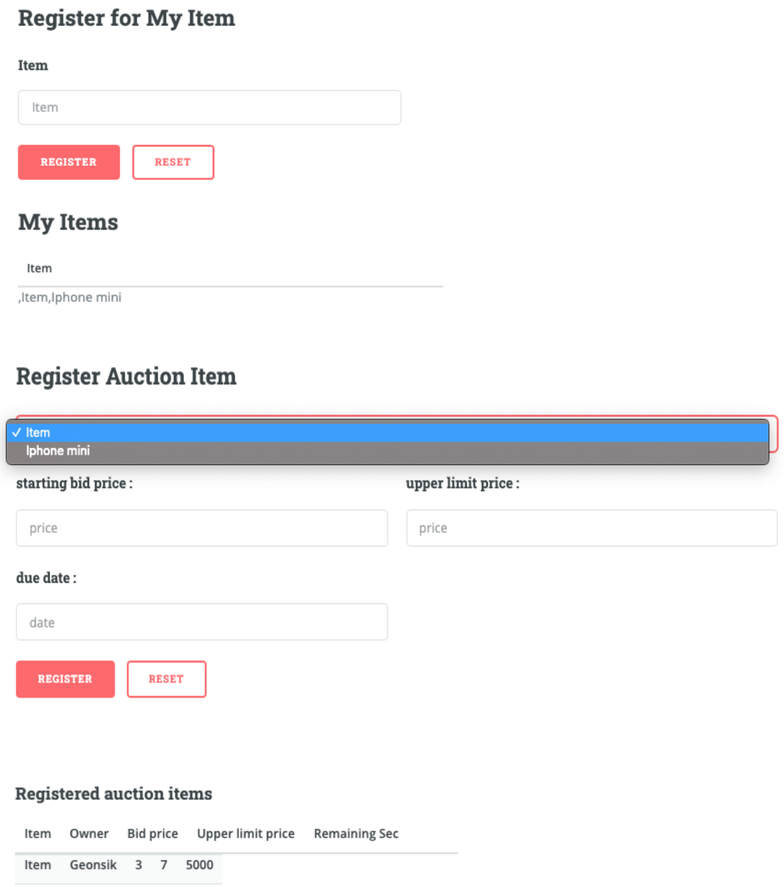
```
struct AuctionItem{

    string Itemname; // Item's name
    string Owner;
    uint start_price;
    uint limit_price;
    uint remaining; // minutes
    uint limitTime;
    uint startTime;
    bool ended;
    address payable add;
}

AuctionItem[] EndedItems;
AuctionItem[] AuctionItems;

function registerAuctionItem(string memory item, uint start_price, uint limit_price, uint date) public{
    require(start_price < limit_price, "start_price is higher than limit_price");
    AuctionItems.push(AuctionItem(item,users[msg.sender].Name, start_price, limit_price,date,date,now,false,msg.sender));
    uint index = 0;
    for(uint i = 0 ; i<users[msg.sender].Items.length ; i++)
    {
        if(stringsEqual(users[msg.sender].Items[i],item))
        {
            index = i;
            break;
        }
    }
    delete users[msg.sender].Items[index];
}
```

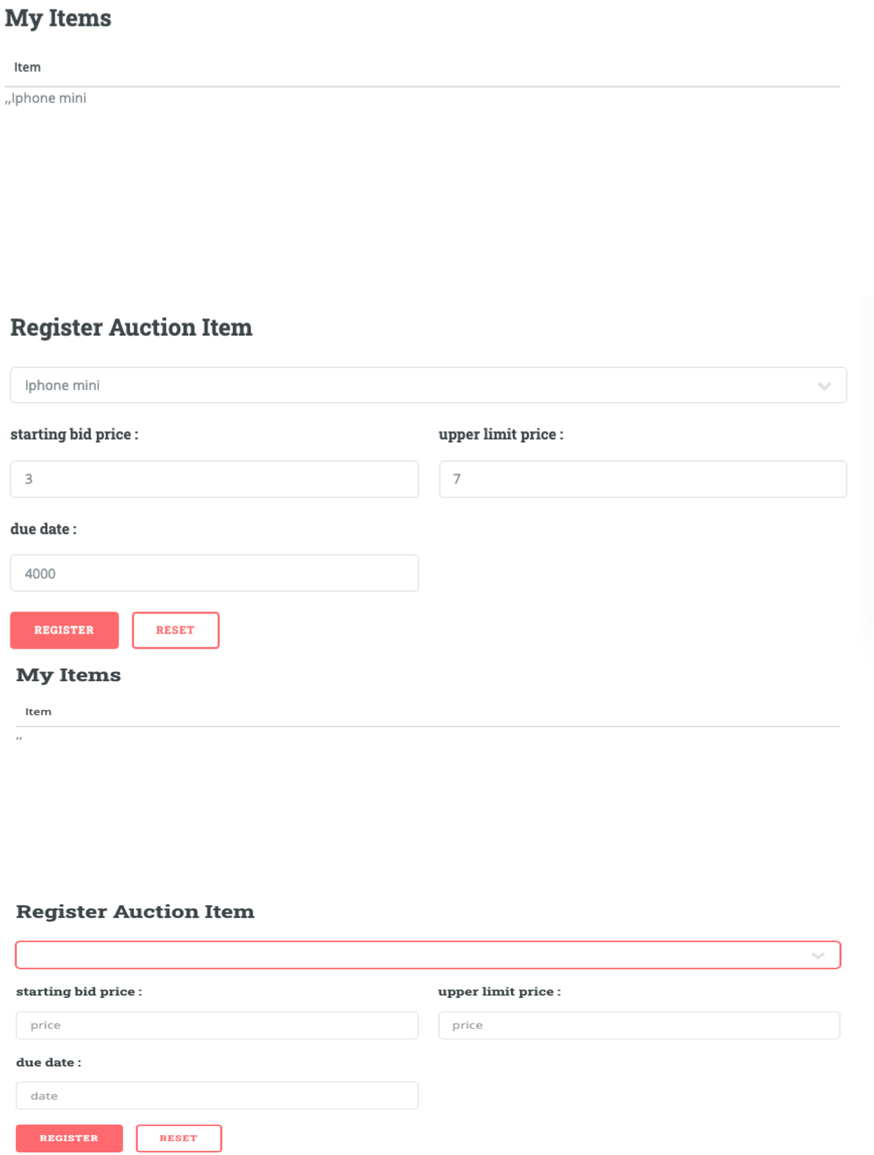
Solidity

Auction Item은 경매에 필요한 정보가 담긴 AuctionItem이라는 구조체 배열에 저장됩니다. Client로 부터 전달받은 item이름 / 시작 금액 / 최대 금액 / 종료시간(초단위)로 입력을 받으면 AuctionItem 변수를 하나 선언하면서 각 변수에 값을 넣어줍니다. 특히 remainin은 남은시간, limitTime은 종료시간, startTime은 now를 넣어주면서 생성된 시간을 넣어줍니다. Ended 변수는 Auction이 끝나면 flag를 바꿔주어 clien에 전달하지 않도록 합니다. 마지막 add는 AuctionItem을 소유한 Owner의 addres입니다.

AuctionItems 배열에 값을 넣었다면 유저가 가지고 있던 Item 목록에서 해당 아이템을 제거합니다. 이중 등록을 막기 위함입니다.

6. registerAuctionItem() / registerAuctionItem(item,start price,limit price,date)





Item, Iphone mini를 경매에 등록하는 과정입니다. 사진을 보시면 Item 목록에 2개의 Item이 있는데 Item을 경매에 등록하자 My Items에 정상적으로 Item이 지워지고 Iphone mini만 남은것을 확인할 수 있습니다. 또한 Registered auction items에 잘 추가가 된 것을 확인할 수 있으며, 이에 대한 코드는 추후 설명하겠습니다.

Iphone mini까지 등록을 완료하며, 3/7/4000(초)로 등록을 하겠습니다.

7. getRegisteredAuctionItems() / getAllAuctionedItems()

```
431  var getRegisteredAuctionItems = async function() {
432      var address = await $('#address').text();
433      // await contract.methods.refreshTime().send({from:address});
434      var AuctionItemInfo = await contract.methods.getAllAuctionedItems().call({from:address});
435      for(var i =0; i<AuctionItemInfo.length ; i++)
436      {
437          if(AuctionItemInfo[i][7] == true){
438              continue;
439          }
440          var rowItem = "<tr>"
441          for(var j=0 ; j<AuctionItemInfo[i].length-4; j++){
442              var element = AuctionItemInfo[i][j];
443              rowItem += "<th>"+element+"</th>";
444          }
445          rowItem += "</tr>"
446          $('#registeredCars').append(rowItem)
447      }
448
449  }
```

Client

```
function getAllAuctionedItems() public view returns(AuctionItem[] memory ){
    return AuctionItems;
}
```

Solidity

Registered auction items

Item	Owner	Bid price	Upper limit price	Remaining Sec
Item	Geonsik	3	7	5000
Iphone mini	Geonsik	3	7	4000

auctionBidding을 설명하기 전 경매에 등록된 아이템 / 끝난 아이템들을 불러오는 함수부터 설명드리겠습니다.

Client에서는 contract내 getAllAuctionedItems()함수를 호출하며 솔리디티 코드를 보시면 AuctionItem 구조체 배열을 반환합니다.

클라이언트에서는 이를 받아와서, 반복문을 수행합니다. 다만 AuctionItemInfo 중 7번째 멤버는 ended로 Auction이 끝났는지 여부를 알려주기 때문에, 이것이 true라면 client 프로그램에 띄워주지 않고 넘어갑니다.

그 외 불필요한 정보는 넘어가고, 필요한 5가지의 정보만을 가져와서 행을 추가합니다.

8. getClosedAuctionItems() / getAllEndedAuctionedItems()

```
451  var getClosedAuctionItems = async function() {
452      var address = await $('#address').text();
453      var ClosedItemInfo = await contract.methods.getAllEndedAuctionedItems().call({from:address});
454      for(var i =0; i<ClosedItemInfo.length ; i++)
455      {
456          var rowItem = "<tr>"
457          for(var j=0 ; j<ClosedItemInfo[i].length-6 ; j++){
458              var element = ClosedItemInfo[i][j];
459              rowItem += "<th>"+element+"</th>";
460          }
461          rowItem += "</tr>"
462          $('#carsOnSale').append(rowItem)
463      }
464  }
```

Client

```
function getAllEndedAuctionedItems() public view returns(AuctionItem[] memory){
    return EndedItems;
}
```

Solidity

Closed auction items

Item	Owner	Winning bid
b		4
b	Hi	8
Iphone mini	Mike	8

Client에서는 contract내 getAllEndedAuctionedItems()함수를 호출하며 솔리디티 코드를 보시면 AuctionItem 구조체 배열을 반환하는데, EndedItems를 반환합니다. 이는 auctionEnd() 함수에서 에서 추가됩니다.

클라이언트에서는 이를 받아와서, 반복문을 수행합니다.여기서 필요한 아이템이름과 바뀌게 된 주인, 낙찰가만 받아온 후 그 외 불필요한 정보는 넘어가고 행을 추가합니다.

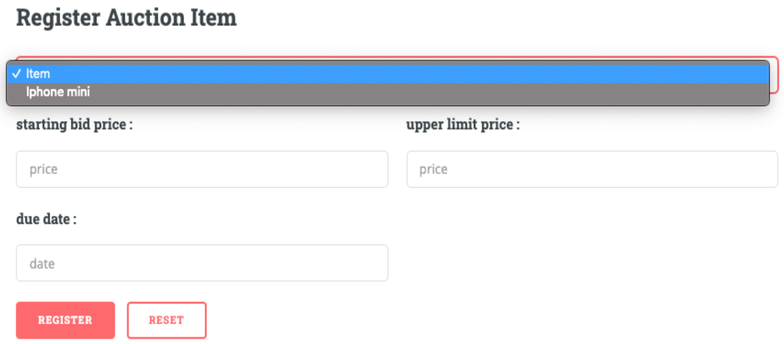
9. getItemsRegisteredAtAuction() /getAllAuctionedItems()

```
var getItemsRegisteredAtAuction = async function() {
    var address = await $('#address').text();
    var AuctionItemInfo = await contract.methods.getAllAuctionedItems().call({from:address});
    for(var i=0 ; i<AuctionItemInfo.length ; i++)
    {
        if(AuctionItemInfo[i][7] == true){
            continue;
        }
        var option_value = AuctionItemInfo[i][0];
        option_value += "/";
        option_value += AuctionItemInfo[i][1];

        var option_text = AuctionItemInfo[i][0];
        option_text += "/" ;
        option_text += AuctionItemInfo[i][1];

        $('#auction-category').append('<option value="'+ option_value +'">'+ option_text +'</option>');
    }
}
```

Client

```
function getAllAuctionedItems() public view returns(AuctionItem[] memory ){
    return AuctionItems;
}
```

Solidity



Client에서는 contract내 getAllAuctionedItems()함수를 호출하며 솔리디티 코드를 보시면 AuctionItem 구조체 배열을 반환합니다.

클라이언트에서는 이를 받아와서, 반복문을 수행합니다. 다만 AuctionItemInfo 중 7번째 멤버는 ended로 Auction이 끝났는지 여부를 알려주기 때문에, 이것이 true라면 client 프로그램에 띄워주지 않고 넘어갑니다.

이들을 auction-bidding이 가능하도록 category에 추가해줍니다.

10. auctionBidding() - Client

```
410  var auctionBidding = async function() {
411      var address = await $('#address').text();
412      var item = await document.getElementById('auction-category').value;
413      item = String(item).split('/'); //item[0]에 아이템 이름 들어감
414      var bidprice = await document.getElementById('bidPrice').value;
415      try{
416          await contract.methods.auctionBidding(item[0],item[1]).send({from:address,gas:5000000, value : bidprice*Math.pow(10,18)});
417      }
418      catch(err){
419          alert(err);
420      }
421  }
```

Client

Client와 Solidity 코드를 나눠 설명하겠습니다.

Client는 간단합니다. Item을 선택하고 자신이 경매에 제시할 금액을 ether 단위로 넣고, 실제로는 send 함수에서 wei로 값이 들어가기 때문에 10^18을 곱하여 전송합니다. 여기서 컨트랙트 내의 actionBidding 함수를 실행합니다.

10. auctionBidding() - Solidity

```
function auctionBidding(string memory item, string memory name) payable public{
    refreshTime();
    uint index = 0;
    for(uint i = 0 ; i<AuctionItems.length ;i++){
        if(stringsEqual(AuctionItems[i].Itemname,item)&& stringsEqual(AuctionItems[i].Owner,name) && AuctionItems[i].ended==false){
                index = i;
        }
    }

    require(msg.value/(1 ether) > AuctionItems[index].start_price);
    require(!stringsEqual(AuctionItems[index].Owner,users[msg.sender].Name));


    if(msg.value/(1 ether)<AuctionItems[index].limit_price){
        AuctionItems[index].start_price = msg.value/(1 ether);
        balanceTransfer(msg.sender);
    }
    else if (AuctionItems[index].remaining == 0){
        balanceTransfer(msg.sender);
        users[AuctionItems[index].add].Items.push(AuctionItems[index].Itemname);
        auctionEnd(index);
    }
    else if(msg.value/(1 ether) >= AuctionItems[index].limit_price){
        address payable seller = AuctionItems[index].add;

        AuctionItems[index].limit_price = msg.value/(1 ether);
        AuctionItems[index].start_price = msg.value/(1 ether);
        balanceTransfer(seller);
        changeItemOwner(index);
        auctionEnd(index);
    }


}
```

Solidity

스마트 컨트랙트 내의 auctionBidding 함수입니다. 함수가 여러 개 있기 때문에 하나씩 설명하겠습니다.

제일 먼저 refreshTime() 함수로 남은 시간을 갱신합니다. 이는 auctionBidding 전에 남은시간이 0인 아이템은 bidding할 수 없도록 하게 하기 위함입니다.

일단 인자로 item 이름과 주인의 이름을 받습니다. 그리고 경매에 등록된 아이템들을 탐색하며 인자로 받은 2개의 parameter 와 일치하는 AuctionItem을 찾으면 index를 저장합니다. 또한 ended 멤버변수가 false여야만 index를 저장합니다.

이렇게 index를 찾았다면, bidding할 item을 찾은 것 입니다. 여기서 require 문이 3개 있습니다. Transaction이 revoke될 2가지 경우입니다.

1. 제시한 금액이 start_price보다 낮은 경우

2. 자신의 아이템에 bidding을 한 경우

이 경우를 제외한다면 제어문으로 들어갑니다. Bidding 금액이 start_price와 limit_price 사이라면 해당 item의 start_price를 제시한 금액으로 바꿔주고 들어온 돈은 메시지를 보낸 사람에게 다시 환불해줍니다. 구매를 하지 않았기 때문입니다.

또한 남은시간이 0초라면 메시지를 보낸 사람 에겐 환불을 진행하고 기존 주인에게 아이템을 돌려줍니다.

만약 bidding 금액이 limit_price보다 높다면 물건을 구매하는 것이 됩니다. 먼저 해당 아이템의 add 멤버변수를 가져와 파는 사람의 주소를 가져옵니다. 이 후 limit_price / start_price를 제시한 금액으로 설정 해준 후, balanceTransfer(seller)로 판 사람에게 송금합니다. 그리고 changeItemOwner(index)로 주인을 바꿔준 후, auctionEnd(index)로 경매를 종료합니다.

10. auctionBidding() - Solidity

```
mapping (address => uint) balances;

function Deposit() public payable{
    balances[msg.sender] += msg.value;
}


function balanceTransfer(address payable _receiver) public{
    Deposit();
    _receiver.transfer(balances[msg.sender]);
    balances[msg.sender] = 0;
}



function auctionEnd(uint index) public{
    EndedItems.push(AuctionItems[index]);
    AuctionItems[index].ended = true;
}


function changeItemOwner(uint index) public{
    AuctionItems[index].Owner = users[msg.sender].Name;
    AuctionItems[index].add = msg.sender;
    users[msg.sender].Items.push(AuctionItems[index].Itemname);

}
```

Solidity

1. balanceTransfer(_receiver)

Deposit()함수를 통해서 전달받은 value를 msg.sender에 매핑되는 balance에 추가해줍니다. 그 이후 _receiver에게 transfer함수를 시행합니다. _receive는 위 페이지에 있는 seller 이며 판매자 입니다. 보낸 후에 계좌 잔액을 0으로 만듭니다.

2. auctionEnd(index)

경매가 종료되면 EndedItems에 해당 Item을 넣고, flag(ended)를 true로 바꿉니다.

3. changeItemOwner(index)

구매자가 확정되면 해당 아이템의 주인과 주소를 바꾸고, 구매자의 Item 목록에 구매한 Item을 추가합니다. 나중에 다시 이 물건을 경매에 등록하는 것도 가능합니다.

10. auctionBidding() - Solidity

```
function refreshTime() public {
    for(uint i =0 ; i < AuctionItems.length ; i++){
        if(AuctionItems[i].remaining < (now - AuctionItems[i].startTime) && AuctionItems[i].ended == false){
            AuctionItems[i].remaining = 0;
            auctionEnd(i);
            users[AuctionItems[i].add].Items.push(AuctionItems[i].Itemname);
        }

        else{
            AuctionItems[i].remaining = AuctionItems[i].limitTime - ((now - AuctionItems[i].startTime));
        }
    }
}
```

Solidity

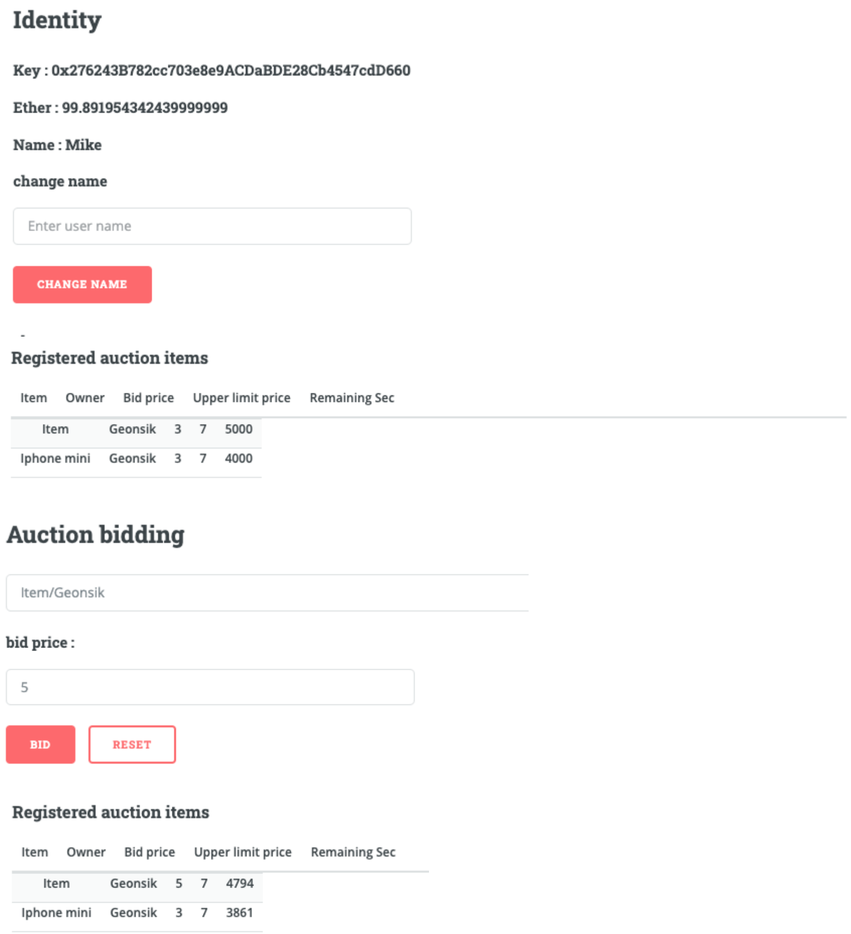
refreshTime 함수는 경매에 등록된 각 Item들의 시간을 갱신합니다.

만약 남은 시간이, 현재시간 – 경매를 시작한 시간 보다 작고 / 아직 경매가 끝나지 않은 물품이라면 남은시간을 0으로 바꿔주고 auctionEnd함수를 시행합니다. 그리고 원래 주인에게 다시 물건을 돌려줍니다.

그것이 아니라면, remaining 멤버변수를 제한 시간에서 현재시간 – 경매를 시작한 시간 을 빼줍니다. 이렇게 되면 bidding 시 마다 refresh된 time으로 경매가 가능합니다.

기존에는 홈페이지를 refresh할 때 마다 client 쪽에서 refreshTime 함수를 호출하도록 했는데 gas limit 문제가 있어 코드에 문제는 없되 gas limit을 넘지 않도록 auctionBidding 함수에 넣게 되었습니다.

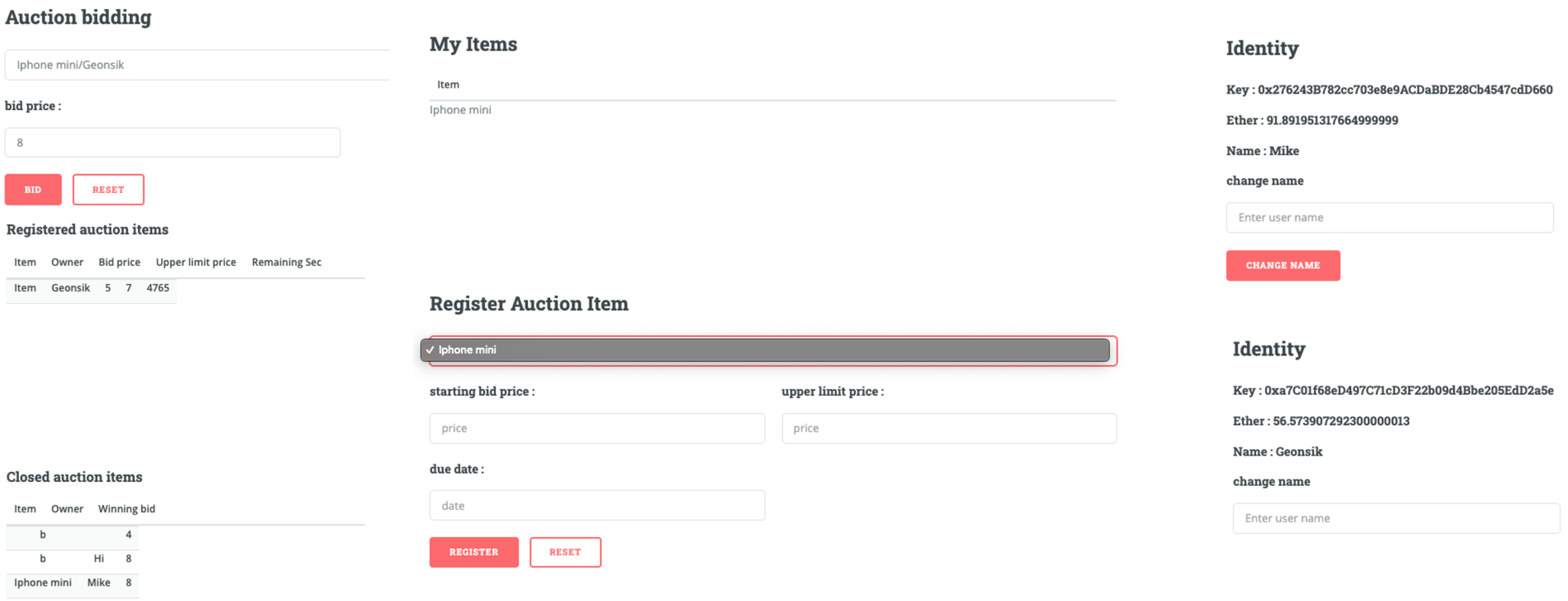
10. auctionBidding() - Solidity



Start_price 와 limit_price 사이의 금액을 bidding했을 때의 경우입니다.

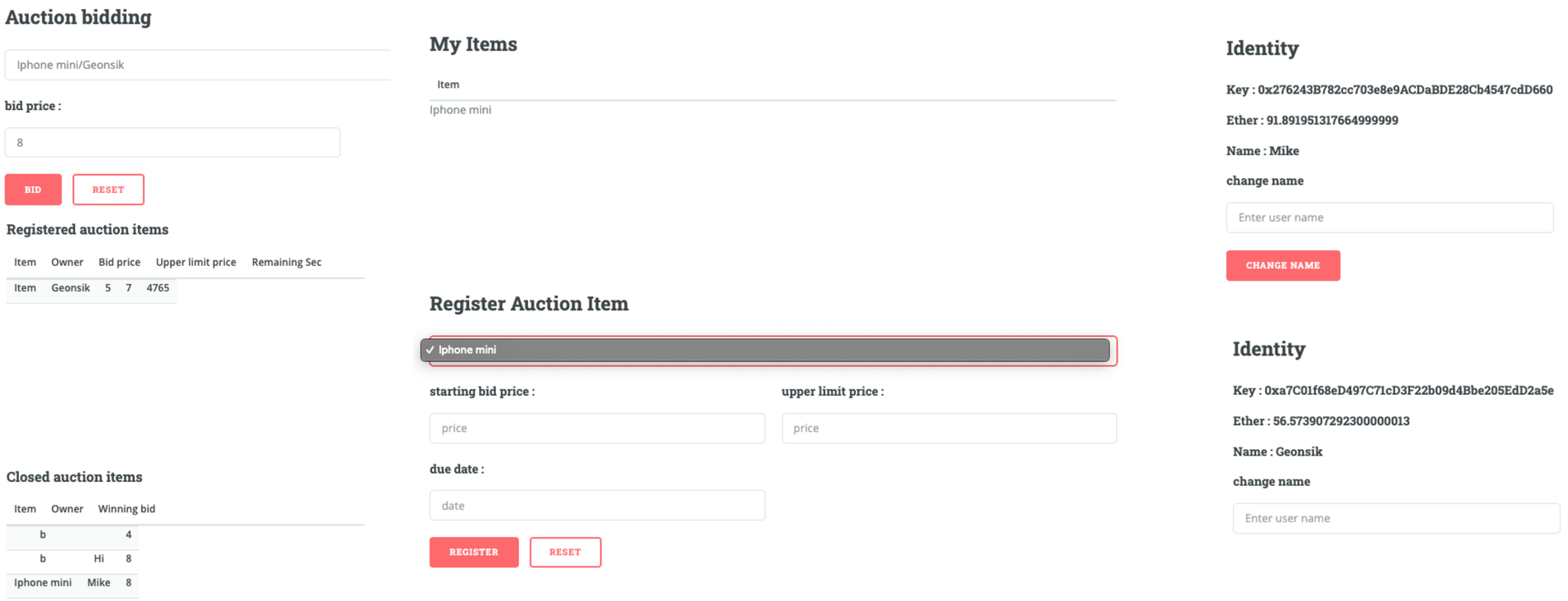
먼저 다른 계정으로 로그인하고, 이름을 Mike로 등록합니다. 이후 5 ether로 bidding 했을 때, 정상적으로 transaction이 전송되어 반영되는 것을 확인할 수 있습니다.

10. auctionBidding() - Solidity



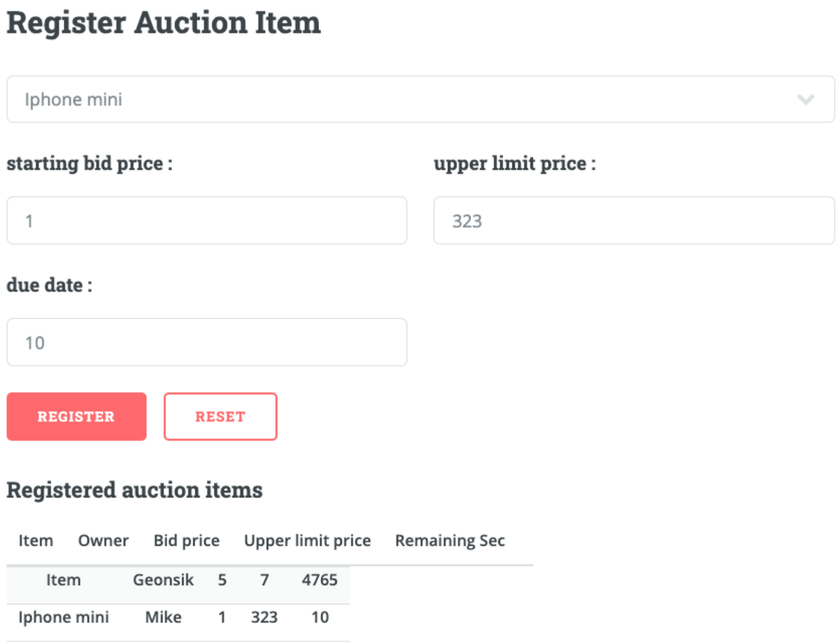
Limit Price보다 높은 금액을 Bidding하게 되면 Mike가 아이템을 소유하게 되면서 Owner가 바뀝니다. My Items에 Iphone mini가 생긴 것을 확인할 수 있습니다. 또한 금액이 Transfer 됩니다. 위 페이지에서의 Mike의 ether 값에서 8을 뺀 만큼 반영되고, Geonsik에게는 8 ether가 송금되었음을 확인이 가능합니다.

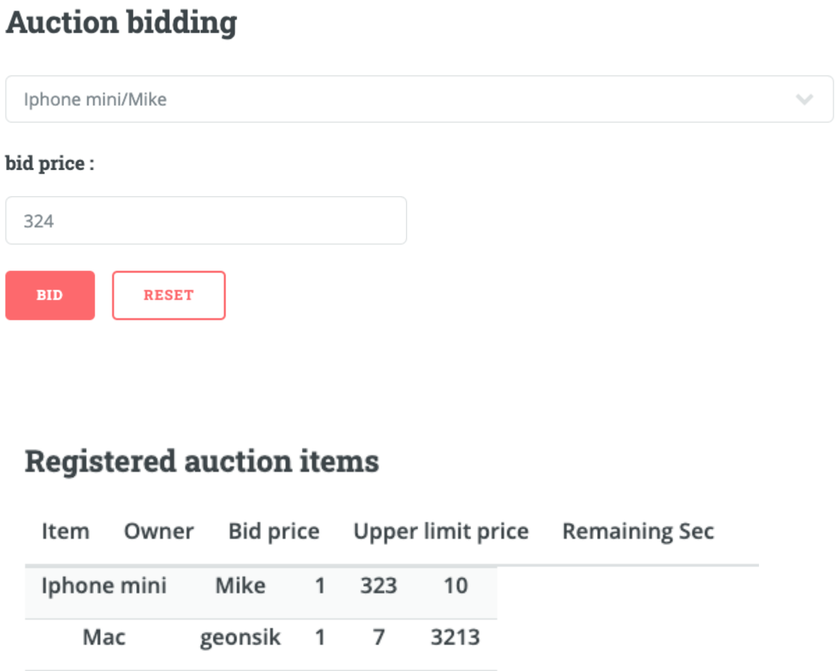
10. auctionBidding() - Solidity



Limit Price보다 높은 금액을 Bidding하게 되면 Mike가 아이템을 소유하게 되면서 Owner가 바뀝니다. My Items에 Iphone mini가 생긴 것을 확인할 수 있습니다. 또한 금액이 Transfer 됩니다. 위 페이지에서의 Mike의 ether 값에서 8을 뺀 만큼 반영되고, Geonsik에게는 8 ether가 송금되었음을 확인이 가능합니다.

10. auctionBidding() - Solidity

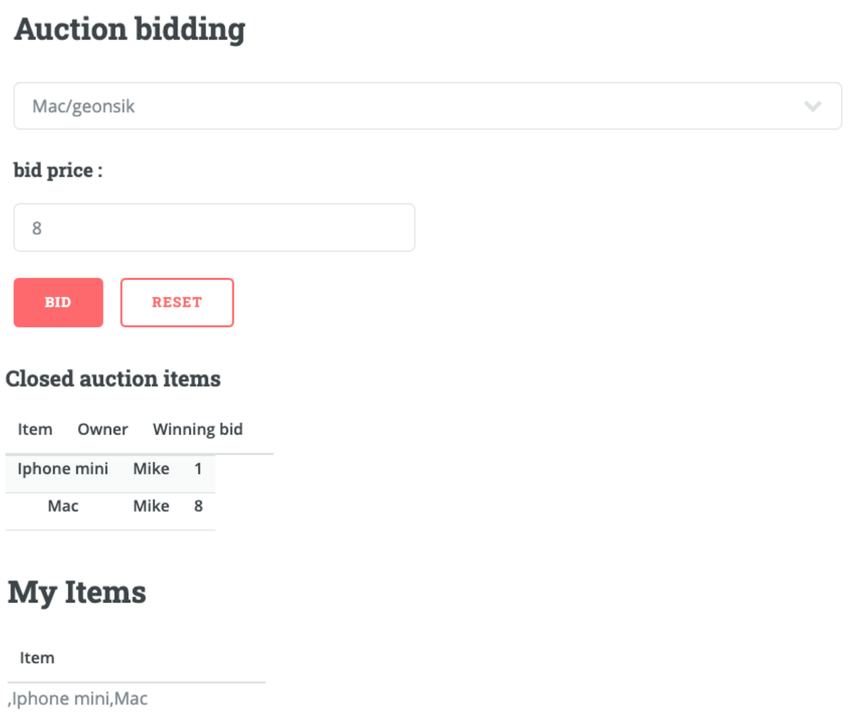




제한시간이 종료되어 자신에게 돌아가는 경우를 확인하겠습니다.

먼저 Mike가 Iphone mini를 1 / 323 / 10초로 설정하여 경매를 등록하고, geonsik으로 접속하여 10초후 limit price보다 높은 금액을 제시하더라도 경매에서 사라지지 않습니다. 여기서 Mike에게 바로 돌아가지 않는 이유는 경매 종료처리가 되었는데 require문 때문에 refresh가 안되는 것입니다.(gas의 한계) 따라서 geonsik이 Mac을 등록한 후 Mike가 Mac을 사는 경우를 생각해보겠습니다.

10. auctionBidding() - Solidity



Mike가 geonsik의 Mac을 8 이더에 구입하자
Closed Auction Items에 제한시간이 끝난
Iphone mini와 방금 경매가 끝난 Mac이
반영되었음을 확인할 수 있고, Mike의 Item
목록에 2개가 추가됨을 확인 가능합니다.

감사합니다.

컴퓨터 소프트웨어 학부 2016025469 서건식