dApp 개발 보고서

네트워크최신기술

20191664 정승우

목차

1 환경 설정 3

1.1 Ganache 실행 3

1.2 Metamask 연동 3

1.3 Remix 연동 4

2 파일 설명 5

2.1 SeungwooJeong.html 5

2.2 auctionpage.html 6

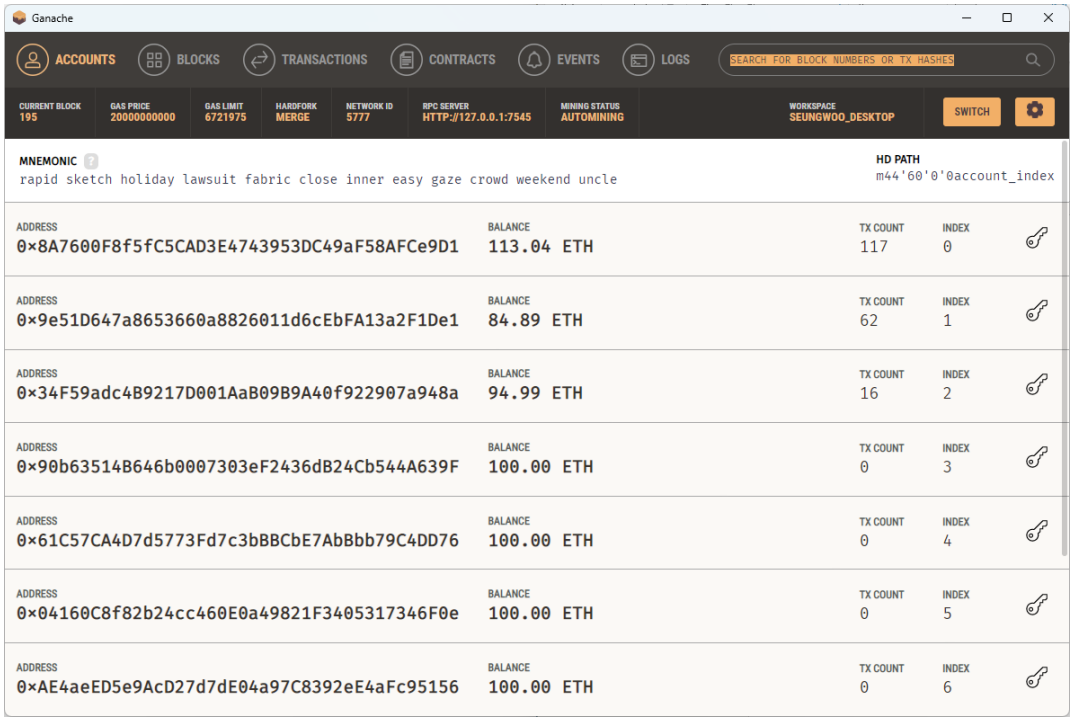
2.3 NameRegistry.sol 9

2.4 SeungwooJeong.sol 10

3 실행 방법 11

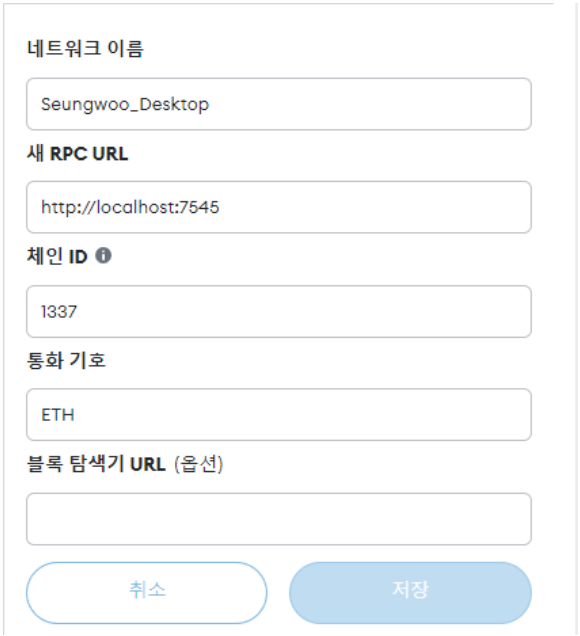
1 환경 설정

1.1 Ganache 실행



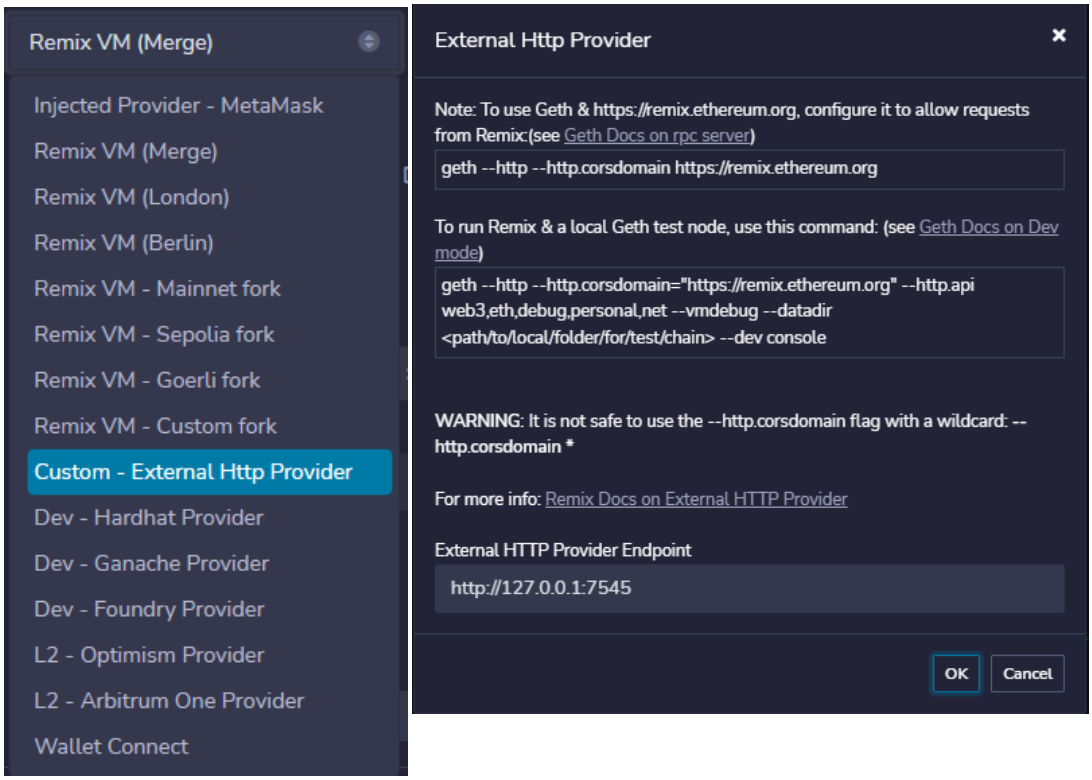
Ganache 를 실행합니다.

1.2 Metamask 연동



메타마스크를 실행후 네트워크에 추가합니다.

1.3 Remix 연동



Remix 와 Ganache 네트워크를 연동합니다.

2 파일 설명

2.1 SeungwooJeong.html

```
var namereg = null;

window.addEventListener('load', entry);

async function entry() {
    if (window.ethereum) {
        window.accounts = await ethereum.request({ method :
"eth_requestAccounts" });
        window.web3 = new Web3(window.ethereum);

        var address = "0xe60AFf4106CCC2fe437Bd5a610FbAc3fE85F9fB3";      //
address of nameregistry
        namereg = await new window.web3.eth.Contract(nameregistryabi, address);
    }
    else if (window.web3) {
        window.web3 = new Web3(web3.currentProvider);
    }
    else {
        console.log('Non-Ethereum browser detected. You should consider trying
MetaMask!');
        window.web3 = null
    }
    registerEventHandlers();
}
```

페이지를 불러들일 때 함수입니다. Address 는 namegistry 의 address 입니다. Web3 를 초기화하고 nameregistry 의 주소와 abi 를 불러옵니다.

```
function registerEventHandlers() {
    let btn = document.getElementById("selectbutton");
    btn.addEventListener('click', goToAuctionPage);
}
```

버튼등에 callback function 을 등록하는 함수입니다.

```
async function goToAuctionPage() {
    var itemname = document.getElementById("itemname").value;
    try {
        var address = await namereg.methods.getAddr(itemname).call();
        if (address != 0) {
            window.location.href = "auctionpage.html?itemname=" +
itemname +
                                   "&address=" + address;
        }
        else {
            let exceptionbox = document.getElementById("exceptionbox");
            exceptionbox.innerHTML = "해당 물품은 경매중이 아닙니다!"
```

```
        }
    }
    catch (error) {
        console.log(error);
        alert(error);
    }
}
```

Select 버튼 클릭 시 auctionpage.html 파일로 넘어가게 해줌과 동시에 nameregistry 에서 주소를 얻어내어 이름과 주소를 auctionpage.html 파일에 넘겨줍니다. 넘겨주는방식은 브라우저 주소창에 인자를 주어 넘겨줍니다.

2.2 auctionpage.html

```
var highestbidBox;
var aucontractInstance = null;
var namereg = null;
var itemname = new
URLSearchParams(window.location.search).get("itemname");
var itemAddr = new
URLSearchParams(window.location.search).get("address");

window.addEventListener('load', auction);
```

앞으로 필요한 변수들을 선언해줍니다. Itemname 과 itemAddr 은 브라우저 주소창에서 얻어옵니다. 또한 해당 html 접속시 auction() 함수를 callback function 으로 등록해줍니다.

```
async function auction() {
    if (itemname == null || itemAddr == null)
    {
        alert("Item or ItemAddress is null");
        window.location.href = "SeungwooJeong.html";
    }
    if (window.ethereum) {
        window.accounts = await ethereum.request({ method:
'eth_requestAccounts' });

        window.web3 = new Web3(window.ethereum);

        var address = "0xe60AFf4106CCC2fe437Bd5a610FbAc3fE85F9fB3";
// address of nameregistry
        namereg = await new window.web3.eth.Contract(nameregistryabi,
address);

        let checkItemAddr = await
namereg.methods.getAddr(itemname).call();

        if (itemAddr != checkItemAddr)
```

```
        {
            alert("Wrong Item name or Item Address.");
            window.location.href = "SeungwooJeong.html";
        }
```

초기화를 하는 부분입니다. 일단 사용자가 임의로 auctionpage.html 을 접속하지는 못하게 하기 위해 itemname 과 itemAddr 이 모두 있는지 확인하여 null 이라면 다시 초기화면으로 넘어가게 설정합니다. 이후 web3 를 초기화합니다. address 는 방금 전 파일과 동일하게 nameregistry 의 contract address 입니다. 또한 itemname 과 itemAddr 이 모두 브라우저 주소창에 존재하더라도 해당 이름과 주소가 유효한지 nameregistry 에 검색하여 확인합니다.

```
aucontractInstance = await new window.web3.eth.Contract(auctionAbi,
itemAddr);
        if (aucontractInstance == null)
        {
            alert("aucontractInstance is null");
        }

        let itemnameBox = document.getElementById("itemnameBox")
        let itemDescriptionBox =
document.getElementById("itemDescriptionBox");
        let deadlineBox = document.getElementById("deadlineBox");
        highestbidBox = document.getElementById("highestbidBox");

        itemnameBox.innerHTML = itemname;
        itemDescriptionBox.innerHTML = await
namereg.methods.getDescription(itemname).call();
        highestbidBox.innerHTML = await
aucontractInstance.methods.highestBid().call();

        let deadlinevalue = await
aucontractInstance.methods.deadline().call();
        let date = new Date(parseInt(deadlinevalue) * 1000);
        deadlineBox.innerHTML = date;
        // Legacy dapp browsers...
    } else if (window.web3) {
        window.web3 = new Web3(web3.currentProvider);
        // Acccounts always exposed
        //    web3.eth.sendTransaction({/* ... */});
    }
    // Non-dapp browsers...
    else {
        console.log('Non-Ethereum browser detected. You should consider
trying MetaMask!');
        window.web3 = null
    }
    registerEventHandlers();
}
```

```
function registerEventHandlers() {
    let btn = document.getElementById("bidButton");
    btn.addEventListener('click', clickBidButton);

    btn = document.getElementById("checkButton");
    btn.addEventListener('click', clickCheckButton);
}
```

이후 auctionpage.html 의 화면구성을 초기화하고 버튼에 callback function 을 등록합니다.

```
async function clickBidButton() {
    if (aucontractInstance != null) {
        try {
            let val = document.getElementById("bidAmount").value;
            if (val <= highestbidBox.innerHTML)
            {
                alert("lower than highest bid");
            }
            else if (val == "" || val < 10000000000000)
            {
                alert("bid amount is least 0.1ether");
            }
            else {
                let value = await
aucontractInstance.methods.Bid().send({ from: accounts[0], value:
val });
                highestbidBox.innerHTML = val;
            }
        } catch (error) {
            alert("error occured at clickBidButton");
            alert(error);
            console.log(error);
        }
    }
}
```

입찰 버튼 callback function 입니다. 입찰 가격을 받아온 뒤 send 합니다. 단 현재 최고입찰가보다 낮은 입찰가를 입력하였거나 입력을 안 한 상태에서 클릭시 알려주고 send 하지않고 종료합니다.

```
async function clickCheckButton() {
    if (aucontractInstance != null) {
        try {
            await aucontractInstance.methods.endAuction().send({ from:
accounts[0] });
            let isEnd = await aucontractInstance.methods.end().call();
            if (isEnd == false) {
                let winningbidderBox =
document.getElementById("winningbidderBox");
                winningbidderBox.innerHTML = "The auction is still
ongoing.";
```

```
                let highestbidBox =
document.getElementById("winningBidBox");
                highestbidBox.innerHTML = "";
            }
            else {
                let winningBidder = await
aucontractInstance.methods.highestBidder().call();
                let highestBidval = await
aucontractInstance.methods.highestBid().call();

                let winningbidderBox =
document.getElementById("winningbidderBox");
                winningbidderBox.innerHTML = "Winner : " +
winningBidder;

                let highestbidBox =
document.getElementById("winningBidBox");
                highestbidBox.innerHTML = "Winning Price : " +
highestBidval;
            }
        }
        catch (error) {
            alert("error occured at clickCheckButton");
            alert(error);
            console.log(error);
        }
    }
}
```

경매가 끝났는지 확인하는 check 버튼의 callback function 입니다. 만약 경매가 끝나지 않았다면 아직 경매가 진행중이라는 문구를 표시하게하고, 경매가 끝났다면 최종 입찰자와 입찰가를 표시해줍니다.

2.3 NameRegistry.sol

이부분은 예제에서 살짝 변경한 내용만 서술하겠습니다.

```
function unregister(string memory _name) public returns (bool) {
    if (contracts[_name].owner == msg.sender) {
        contracts[_name].owner = address(0x0);
        contracts[_name].addr = address(0x0);
        contracts[_name].description = "";
         numContracts--;
         return true;
    } else {
        return false;
    }
}
```

Unregister 시 입력된 이름에 해당하는 주소, 설명, owner 모두 한번에 0 으로 초기화 되도록 하였습니다.

2.4 SeungwooJeong.sol

```
//SPDX-License-Identifier: UNLICENSED

pragma solidity >=0.7.0;
contract auction {
    address owner;
    address public highestBidder;
    uint public highestBid;
    uint public deadline;
    bool public end;

constructor(uint duration) payable {
    end = false;
    owner = msg.sender;
    highestBidder = msg.sender;
    highestBid = 0;
    deadline = block.timestamp + duration;
}
```

Contract 설정 및 초기화 부분입니다. Deploy 시 duration 을 입력하며 해당 시간동안 경매를 진행하게 됩니다.

```
function Bid() payable public {
    require(msg.value > highestBid);
    require(end == false);

    uint refundAmmount = highestBid;
    address refundAddress = highestBidder;

    highestBid = msg.value;
    highestBidder = msg.sender;

    if (!payable(refundAddress).send(refundAmmount)) {
        revert();
    }
}
```

입찰하는 함수입니다. 만약 보낸 가격이 가장 높은 입찰가이고 아직 경매가 진행중이라면 등록하고 아니라면 revert()합니다. 등록시에 이전에 가장 높은 가격으로 입찰했던 입찰자에게 ether 를 돌려줍니다.

```
function endAuction() public returns(bool) {
    if (block.timestamp <= deadline) {
        return false;
    }
    end = true;

    if (!payable(owner).send(address(this).balance)) {
        revert();
    }
    return true;
}
```

경매가 끝났는지 확인하고. 끝났다면 최종 입찰가를 이 경매를 deploy 한 owner 에게 balance 를 전송합니다.

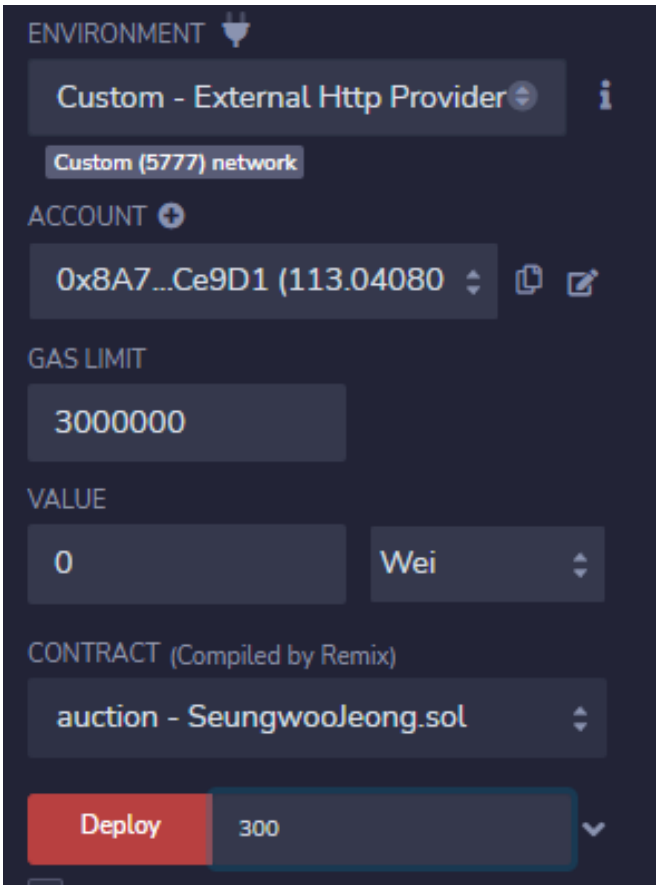
3 실행 방법

실행 방법은 다음과 같습니다. (환경설정이 완료된 상태에서)

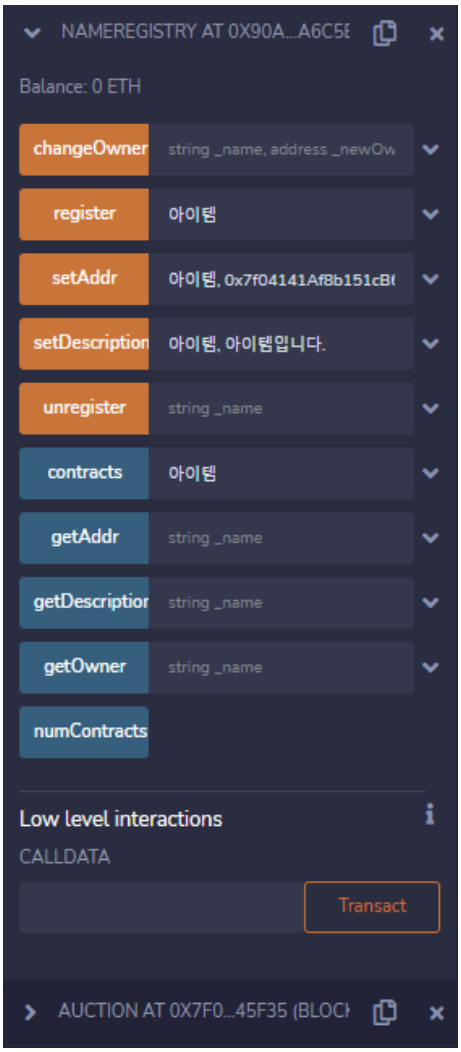
1. NameRegistry.sol 을 컴파일 후 Deploy
2. NameRegistry Contract Address 를 SeungwooJeong.html, auctionpage.html 의 address 변수에 입력

```
var address = "0x90ACd8ad37bF5abB81c83446185A2cb2912a6c5E"; // address of nameregistry
```

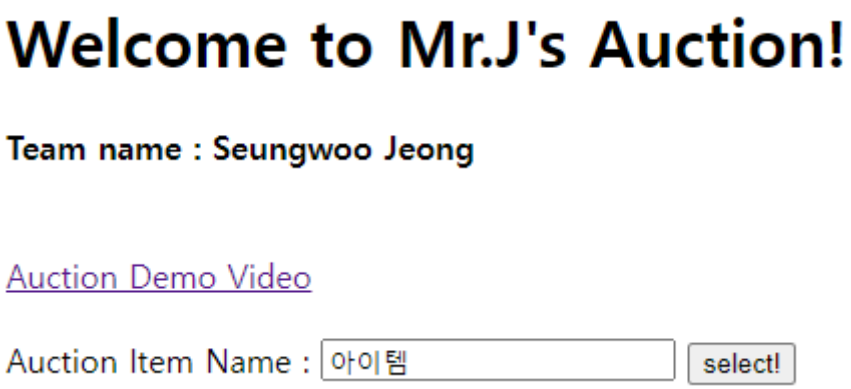
3. 경매를 시작할(아이템을 등록할) 계정을 선택 후 SeungwooJeong.sol 을 Deploy



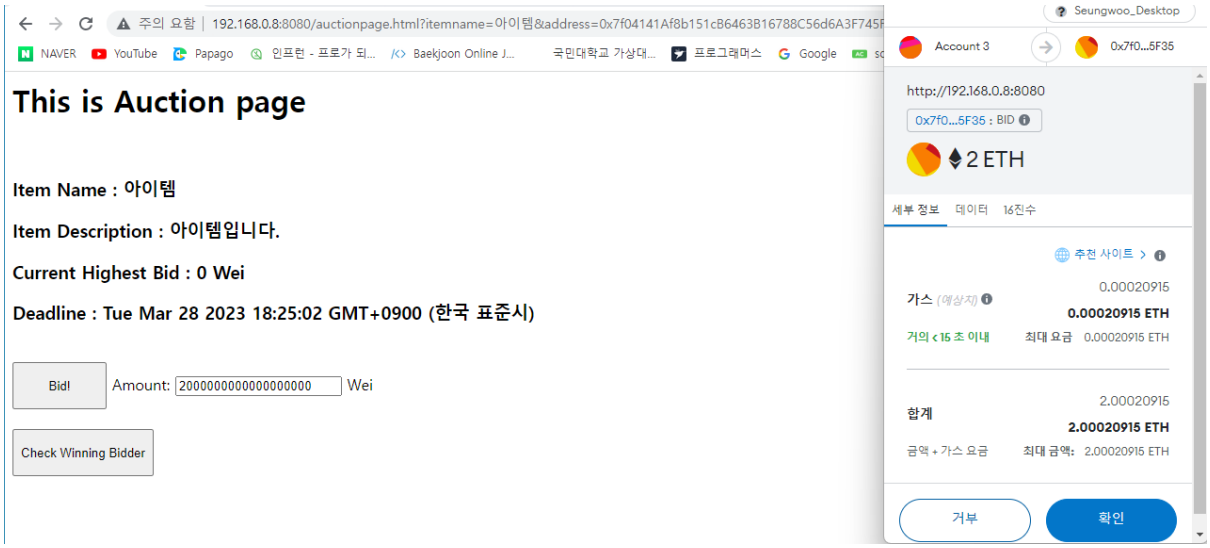
4. Deploy 한 계약(아이템)을 register()



5. SeungwooJeong.html 에 해당 아이템 이름을 입력 후 접속

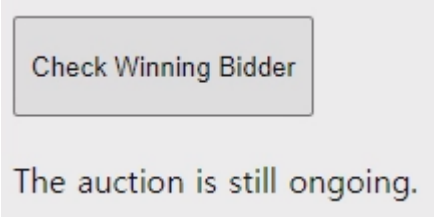


6. auctionpage.html 에서 Bid 버튼을 눌러 경매 진행



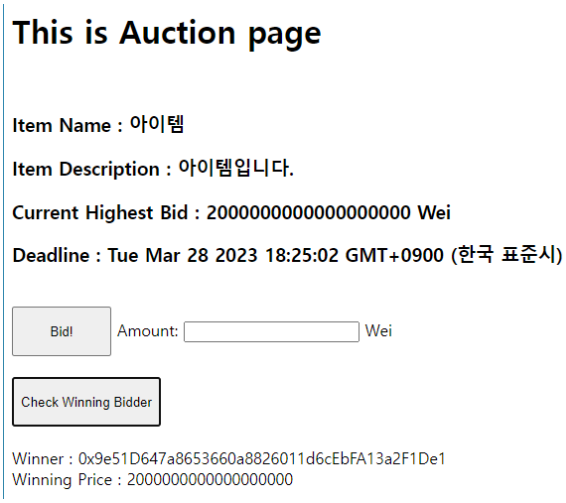
7. 경매 종료 후 check 버튼을 눌러 최종 입찰자 확인

만약 끝나지 않았다면



끝나지 않았다는 문구가.

끝났다면



최종 입찰자와 입찰가가 표시됩니다.

이상입니다. 감사합니다.