Phone number marketplace

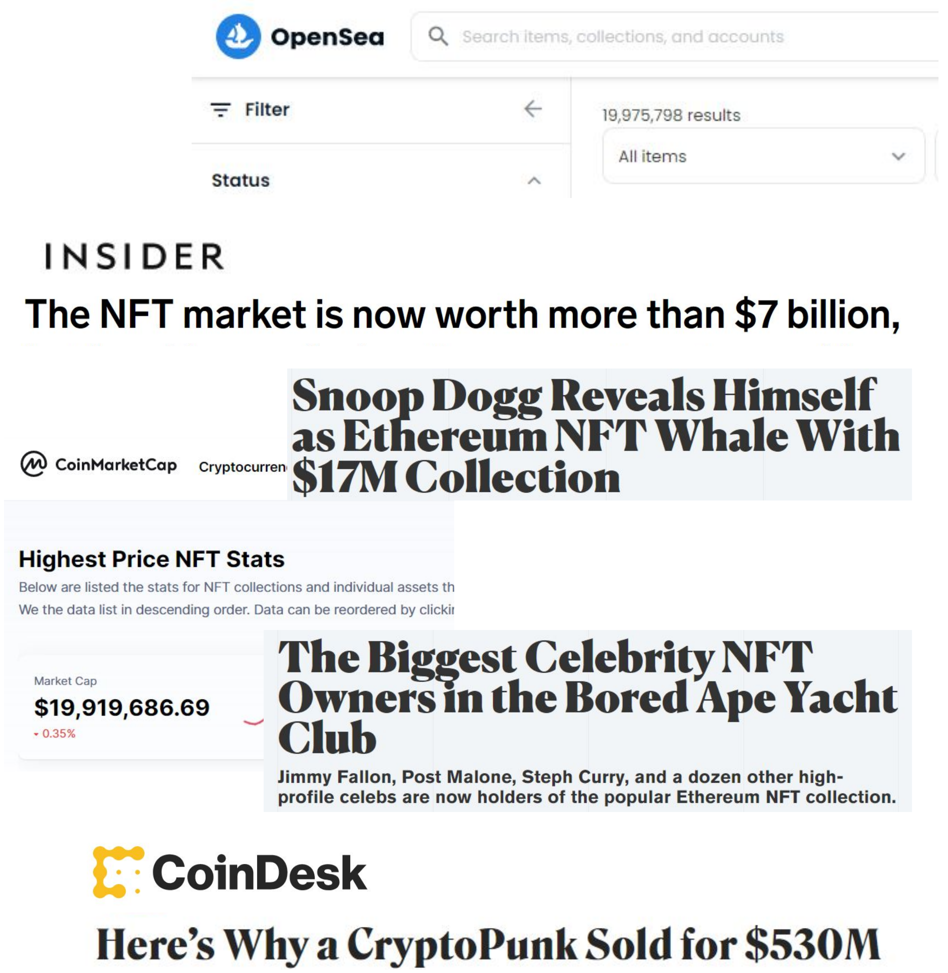
A phone number service run on the Ethereum blockchain

Our team: Alan Barahona Ruiz, Joël Haubold,
Minh-Quang Nguyen, Natalia Markoborodova.

Outline

Our motivation

NFTs is growing fast



OpenSea

Search items, collections, and accounts

Filter

Status

19,975,798 results

All items

INSIDER

The NFT market is now worth more than $7 billion,

CoinMarketCap Cryptocurren

Snoop Dogg Reveals Himself as Ethereum NFT Whale With $17M Collection

Highest Price NFT Stats

Below are listed the stats for NFT collections and individual assets th

We the data list in descending order. Data can be reordered by clicki

Market Cap

$19,919,686.69

0.35%

The Biggest Celebrity NFT Owners in the Bored Ape Yacht Club

Jimmy Fallon, Post Malone, Steph Curry, and a dozen other high-profile celebs are now holders of the popular Ethereum NFT collection.

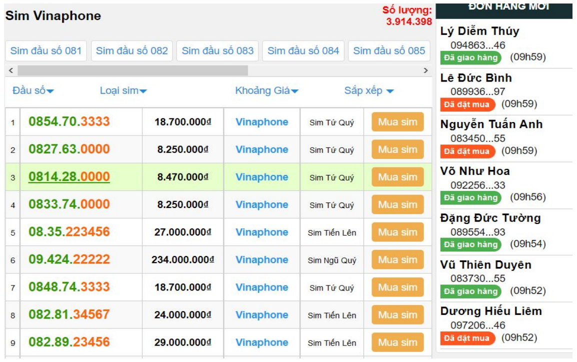
CoinDesk

Here's Why a CryptoPunk Sold for $530M

Phone numbers have a lot in common with NFTs!

Phone number exhibits many similar characteristics to an NFT.

Phone number service is an interesting industry in many countries.



Sim Vinaphone

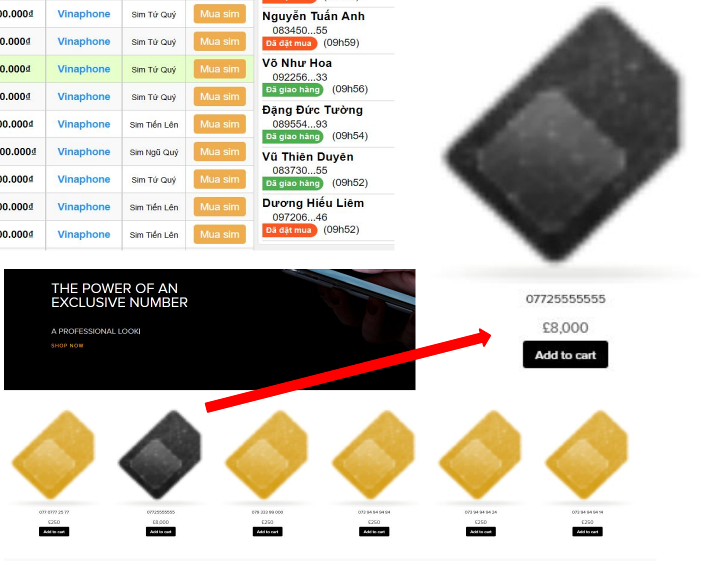
Số lượng: 3.914.398

Sim đầu số 081 | Sim đầu số 082 | Sim đầu số 083 | Sim đầu số 084 | Sim đầu số 085

	Đầu số		Loại sim		Khoảng Giá	Sắp xếp
1	0854.70.3333	18.700.000đ	Vinaphone	Sim Tứ Quý	Mua sim
2	0827.63.0000	8.250.000đ	Vinaphone	Sim Tứ Quý	Mua sim
3	0814.28.0000	8.470.000đ	Vinaphone	Sim Tứ Quý	Mua sim
4	0833.74.0000	8.250.000đ	Vinaphone	Sim Tứ Quý	Mua sim
5	08.35.223456	27.000.000đ	Vinaphone	Sim Tiến Lên	Mua sim
6	09.424.22222	234.000.000đ	Vinaphone	Sim Ngũ Quý	Mua sim
7	0848.74.3333	18.700.000đ	Vinaphone	Sim Tứ Quý	Mua sim
8	082.81.34567	24.000.000đ	Vinaphone	Sim Tiến Lên	Mua sim
9	082.89.23456	29.000.000đ	Vinaphone	Sim Tiến Lên	Mua sim

ĐƠN HÀNG MỚI

Name	Number	Status	Time
Lý Diễm Thúy	094863...46	Đã giao hàng	(09h59)
Lê Đức Bình	089936...97	Đã đặt mua	(09h59)
Nguyễn Tuấn Anh	083450...55	Đã đặt mua	(09h59)
Võ Như Hoa	092256...33	Đã giao hàng	(09h56)
Đặng Đức Tường	089554...93	Đã giao hàng	(09h54)
Vũ Thiên Duyên	083730...55	Đã giao hàng	(09h52)
Dương Hiếu Liêm	097206...46	Đã đặt mua	(09h52)



THE POWER OF AN EXCLUSIVE NUMBER

A PROFESSIONAL LOOK!

SHOP NOW

07725555555

£8,000

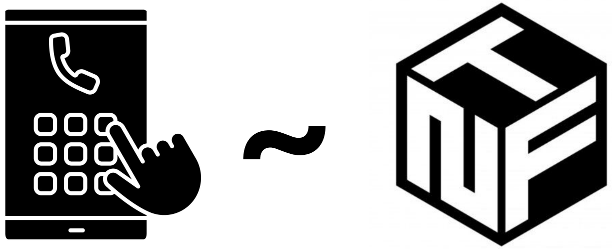
Add to cart

077 0777 25 77	07725555555	079 333 99 000	073 94 94 94 84	073 94 94 94 24	073 94 94 94 14
£250	£8,000	£250	£250	£250	£250
Add to cart	Add to cart	Add to cart	Add to cart	Add to cart	Add to cart

Project Idea

A marketplace for phone number on Ethereum blockchain

- Similarities between phone numbers and NFTs
  - Uniqueness
  - Non-fungible
  - Ownership
  - Indivisibility
  - Interoperability
  - Transparency
- All possible functionalities of tradition services
  - Buy/sell
  - Auction



Why it’s better than traditional solution

- Utilize advantages of blockchain technology and smart contract

+ Full transparency

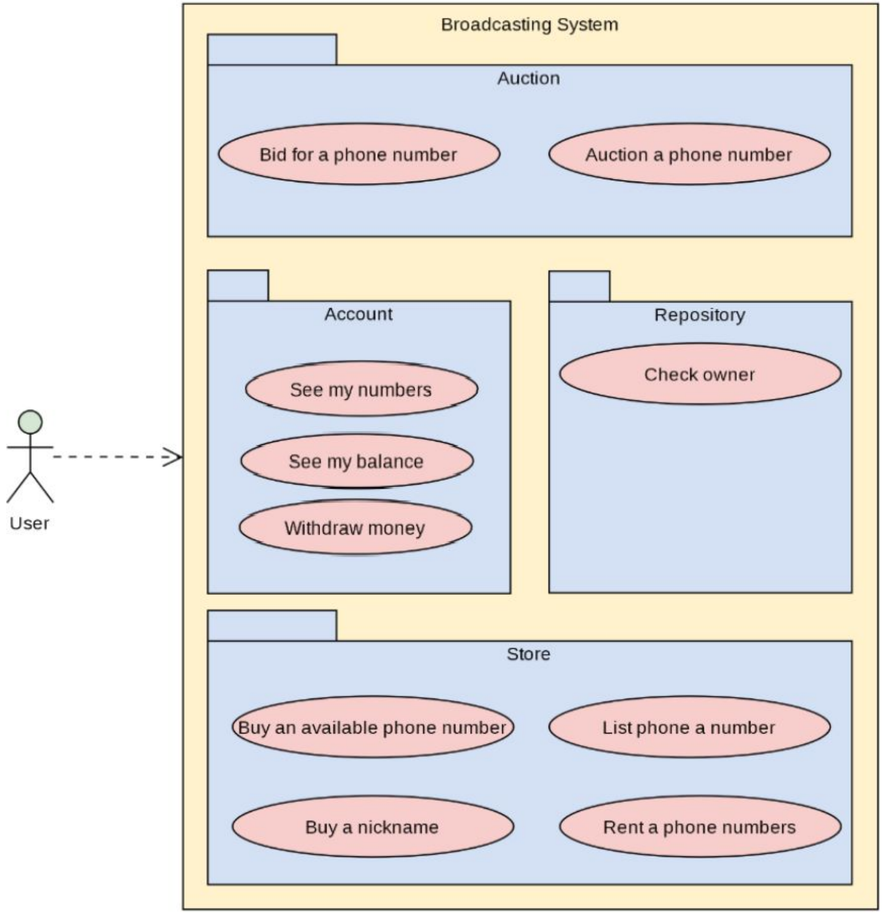
+ Trustless and decentralization

+ Minimize impact from middle-man

Design specification

Main services for user:

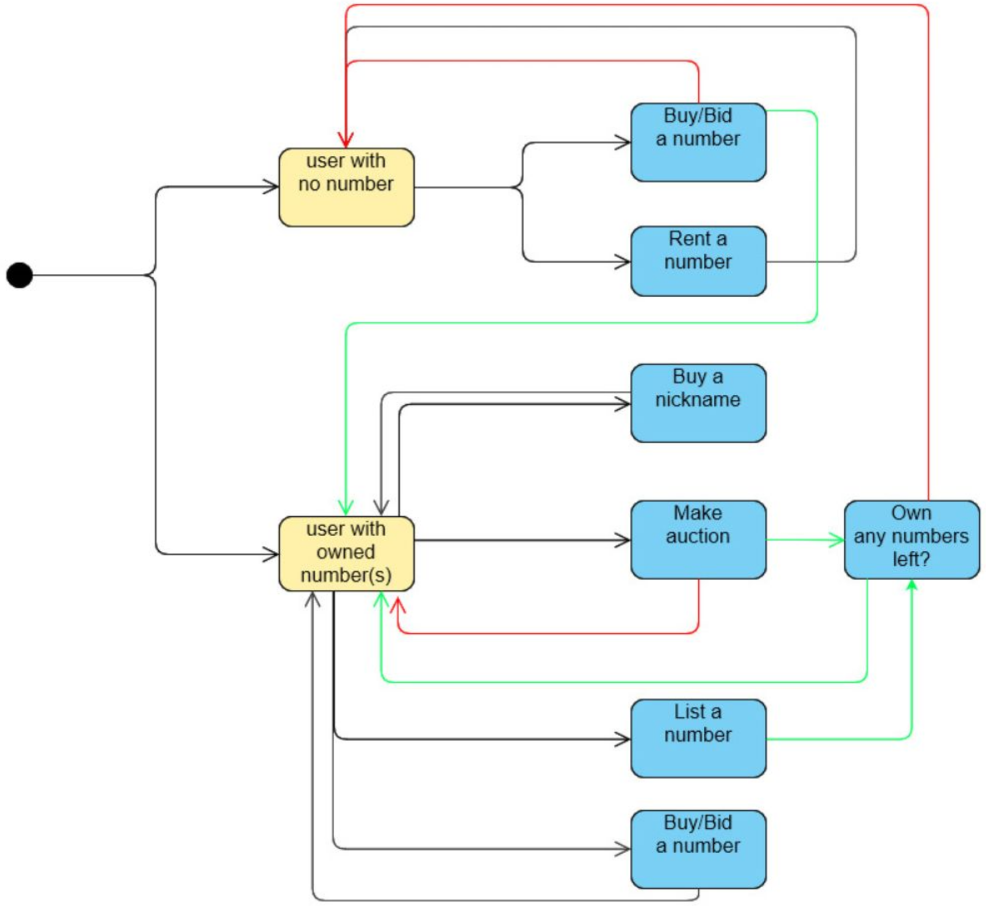
- Buy available phone numbers
- Buy a nickname
- Rent available phone numbers
- List a phone number that you own and want to sell
- Bid on an auction of phone number



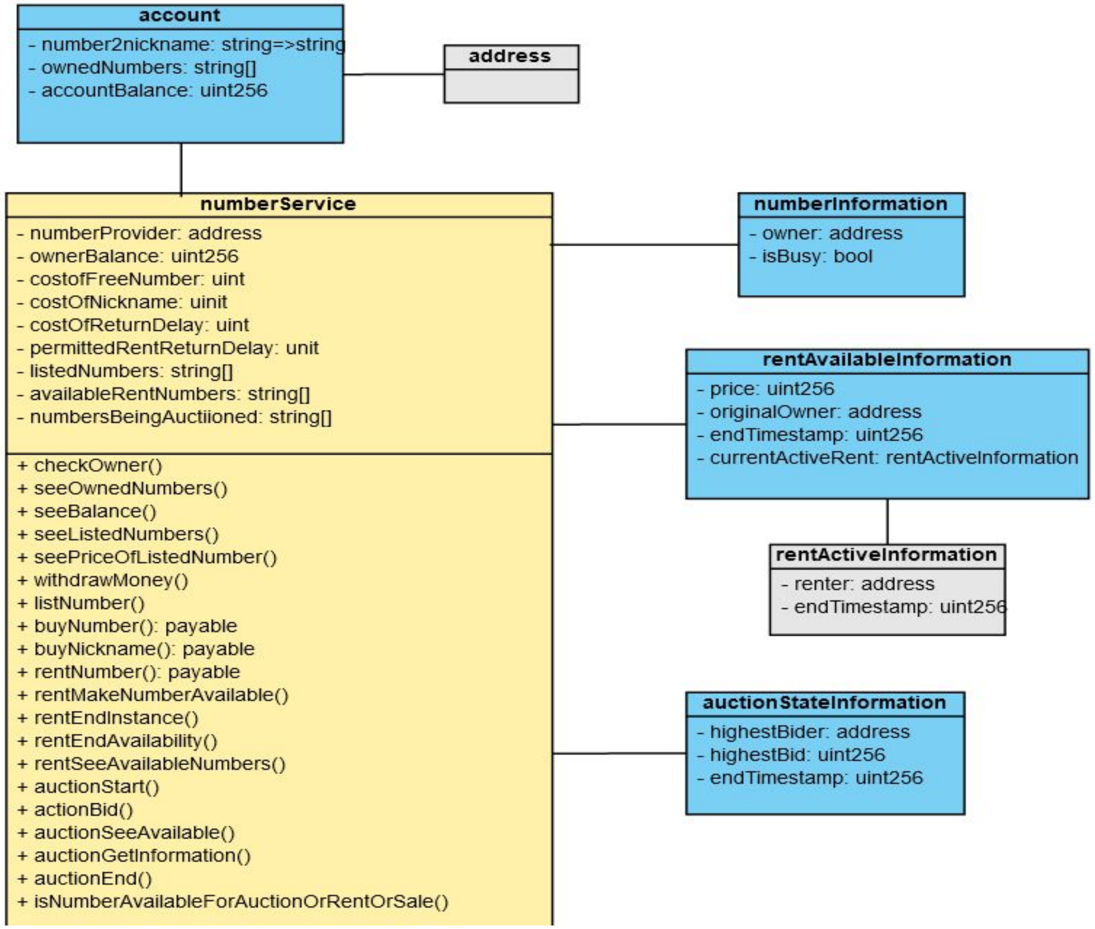
User case diagram

Main services for user:

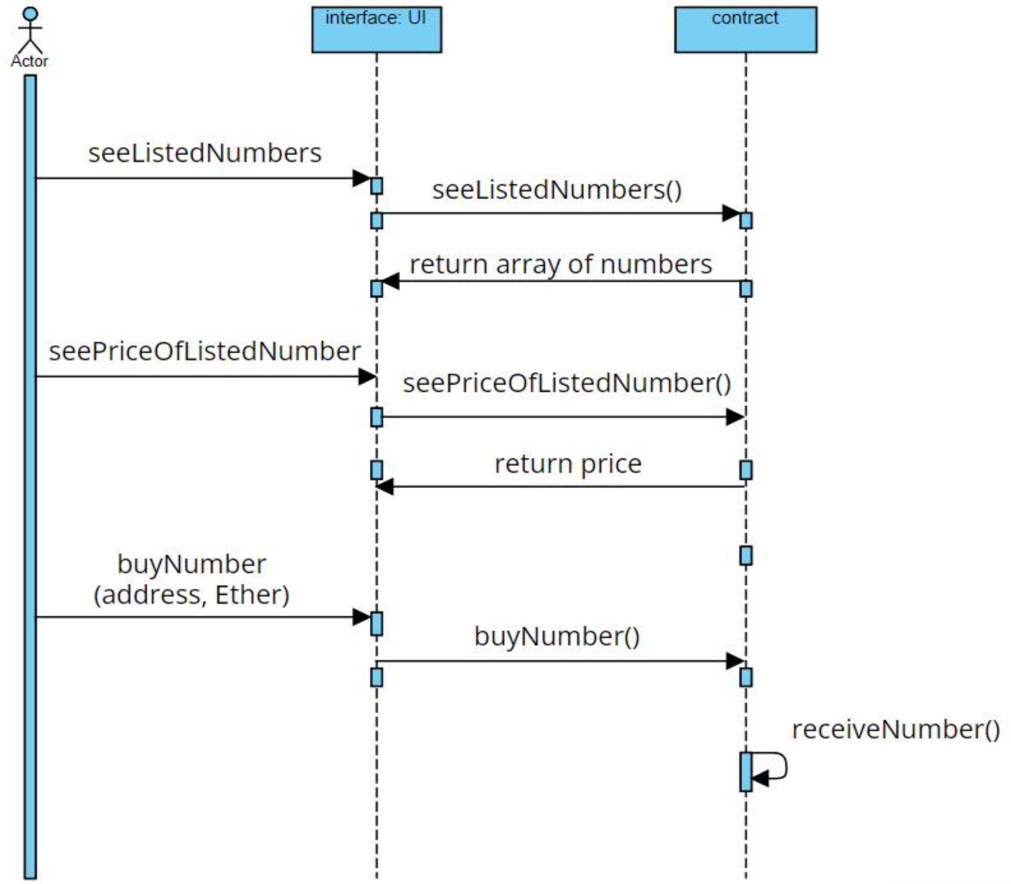
- Buy available phone numbers
- Buy a nickname
- Rent available phone numbers
- List a phone number that you own and want to sell
- Bid on an auction of phone number



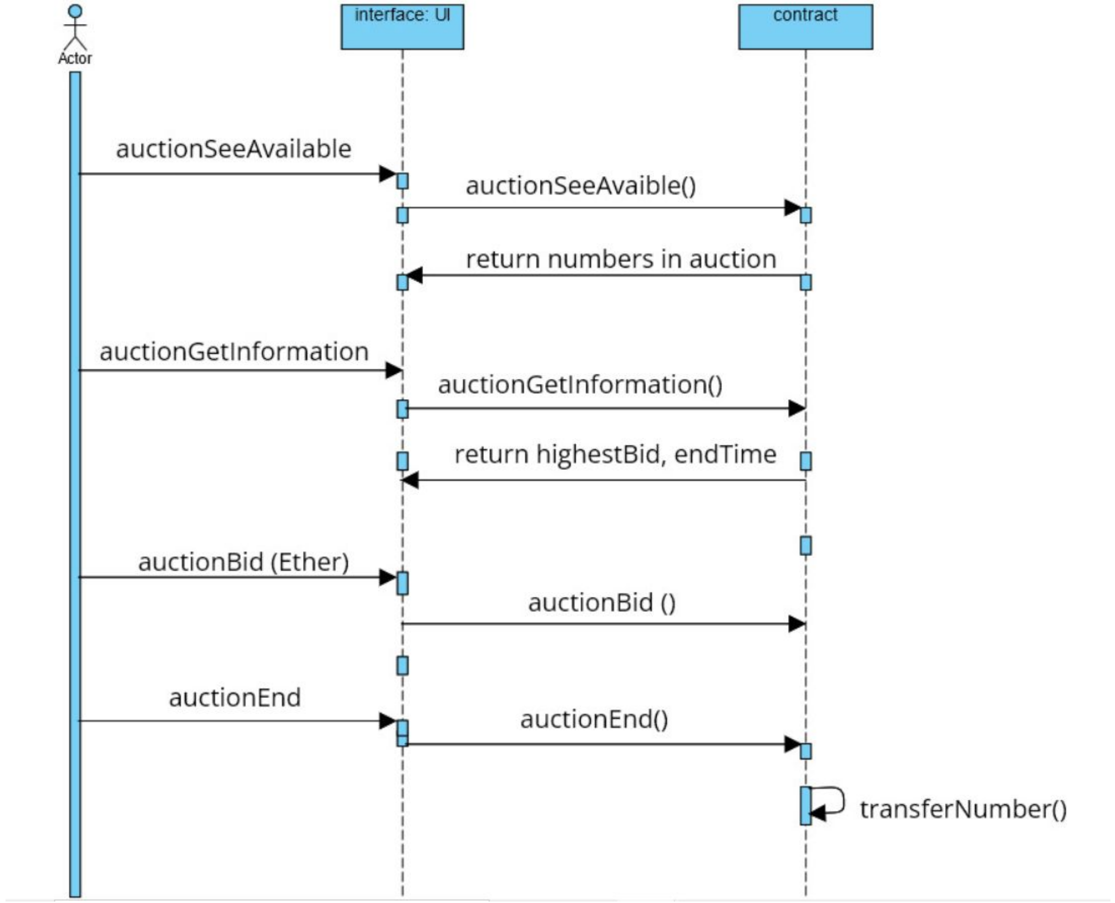
State diagram



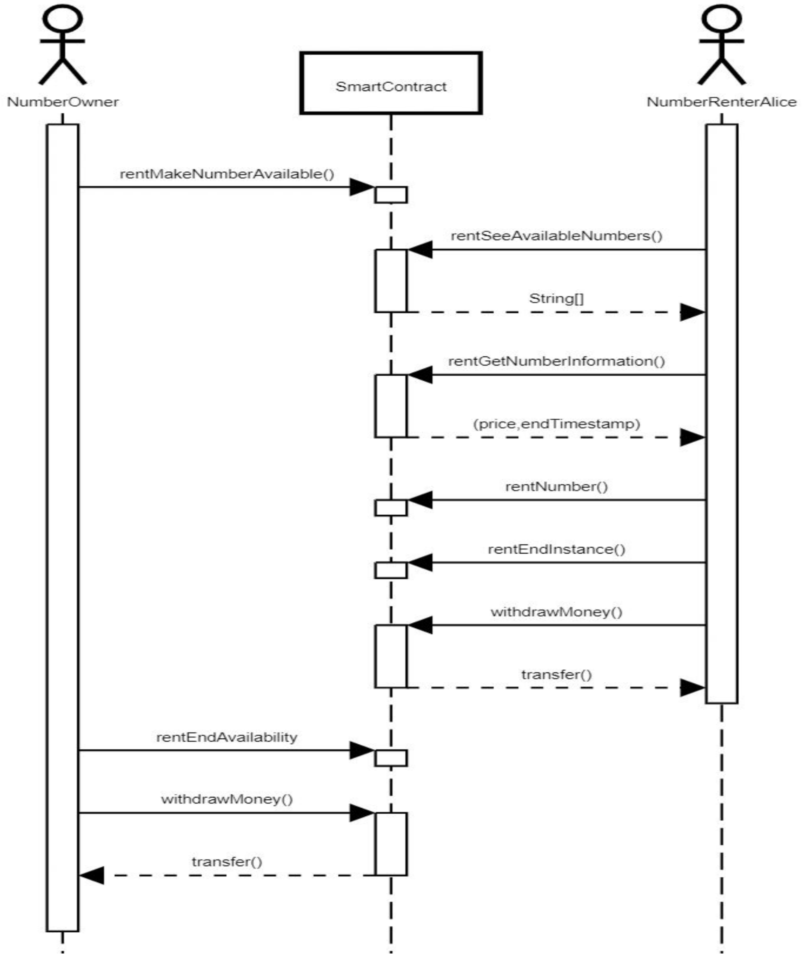
Class diagram



Sequence Diagram 1 - Buy a listed number and receive it

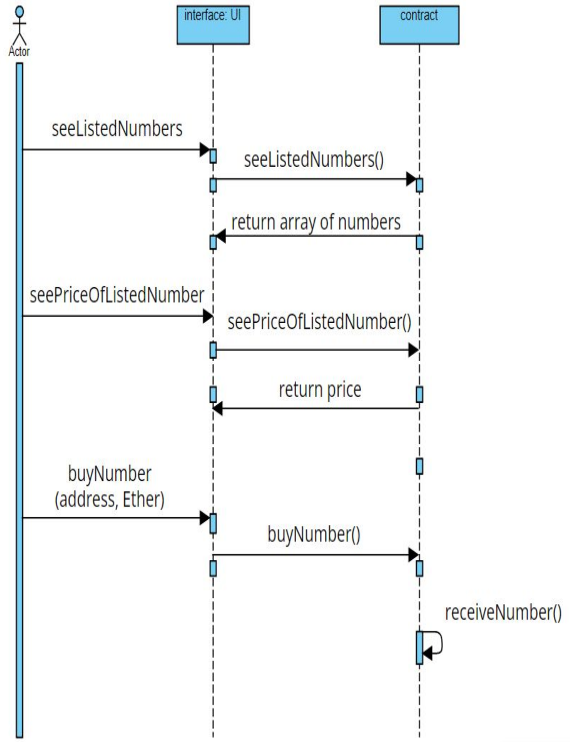


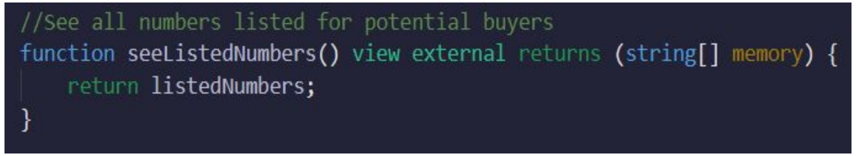
Sequence Diagram 2 - Bid and win in an auction



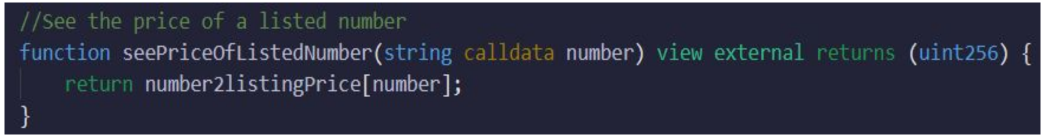
Sequence Diagram 3 - Rent a phone number

Code highlights

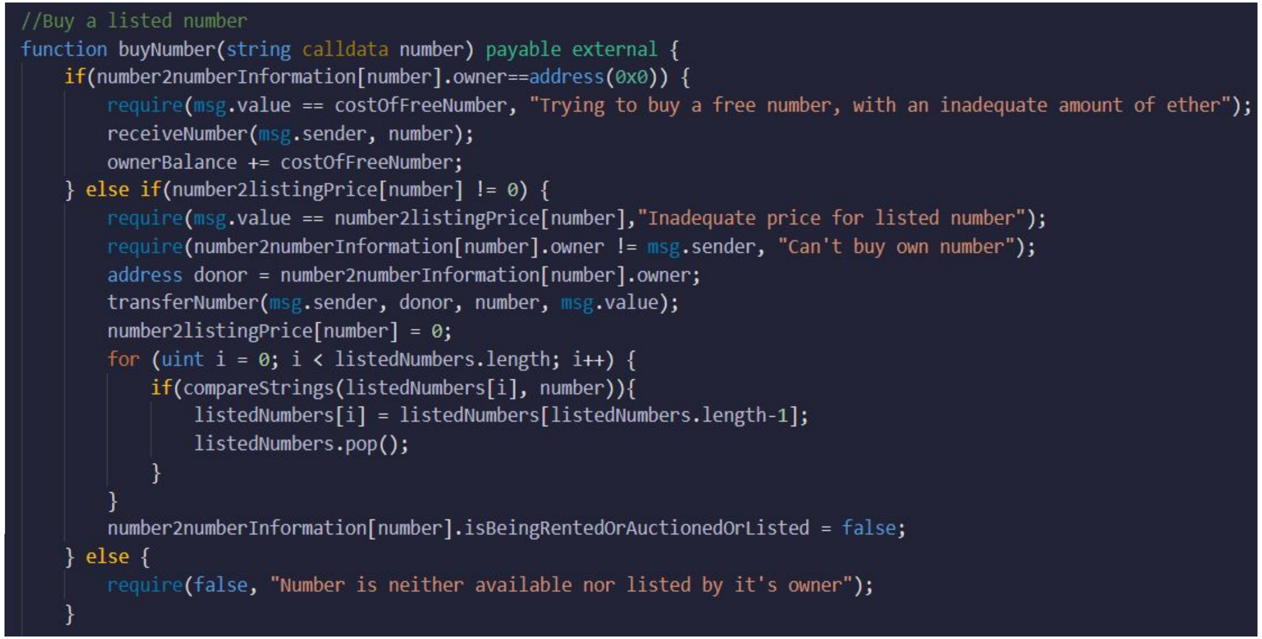




```
//See all numbers listed for potential buyers
function seeListedNumbers() view external returns (string[] memory) {
    return listedNumbers;
}
```

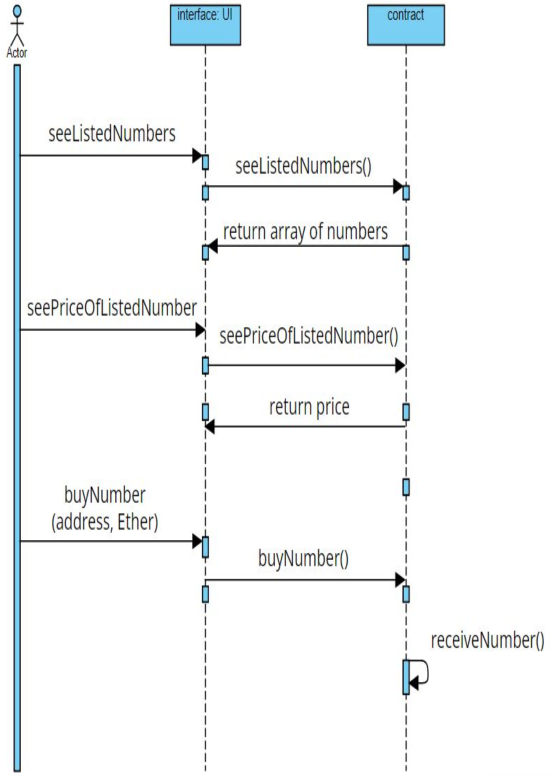


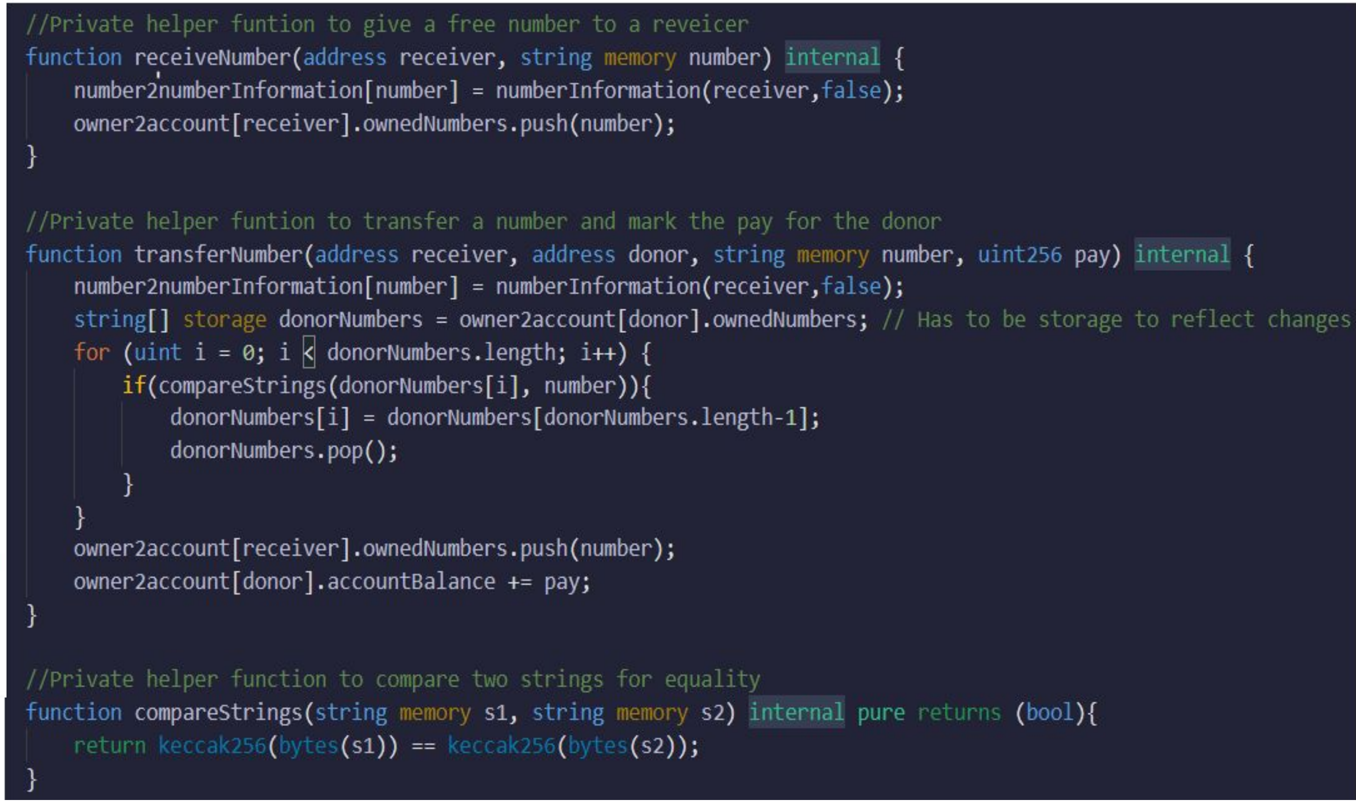
```
//See the price of a listed number
function seePriceOfListedNumber(string calldata number) view external returns (uint256) {
    return number2listingPrice[number];
}
```



```
//Buy a listed number
function buyNumber(string calldata number) payable external {
    if(number2numberInformation[number].owner==address(0x0)) {
        require(msg.value == costOfFreeNumber, "Trying to buy a free number, with an inadequate amount of ether");
        receiveNumber(msg.sender, number);
        ownerBalance += costOfFreeNumber;
    } else if(number2listingPrice[number] != 0) {
        require(msg.value == number2listingPrice[number],"Inadequate price for listed number");
        require(number2numberInformation[number].owner != msg.sender, "Can't buy own number");
        address donor = number2numberInformation[number].owner;
        transferNumber(msg.sender, donor, number, msg.value);
        number2listingPrice[number] = 0;
        for (uint i = 0; i < listedNumbers.length; i++) {
            if(compareStrings(listedNumbers[i], number)){
                listedNumbers[i] = listedNumbers[listedNumbers.length-1];
                listedNumbers.pop();
            }
        }
        number2numberInformation[number].isBeingRentedOrAuctionedOrListed = false;
    } else {
        require(false, "Number is neither available nor listed by it's owner");
    }
```

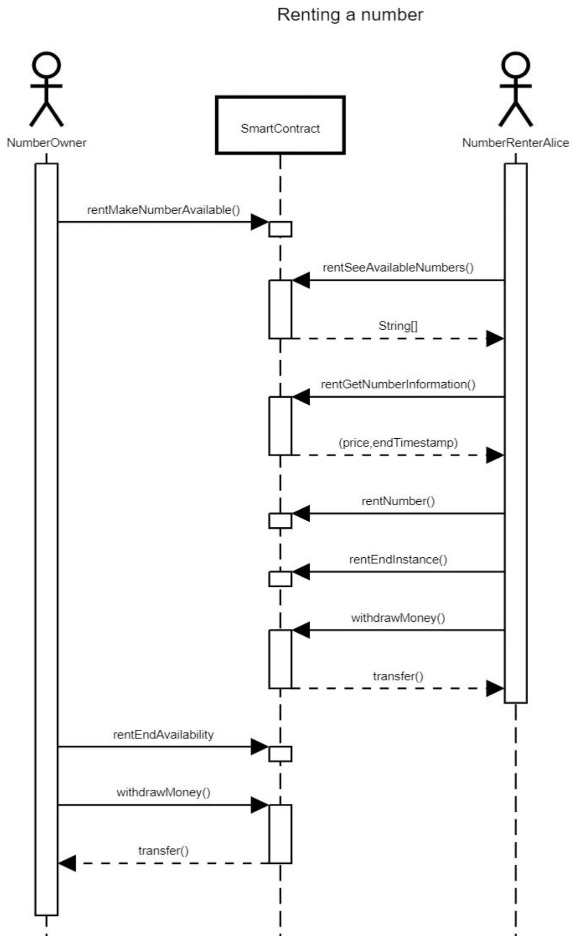
Buy a listed number and receive it (1/2)

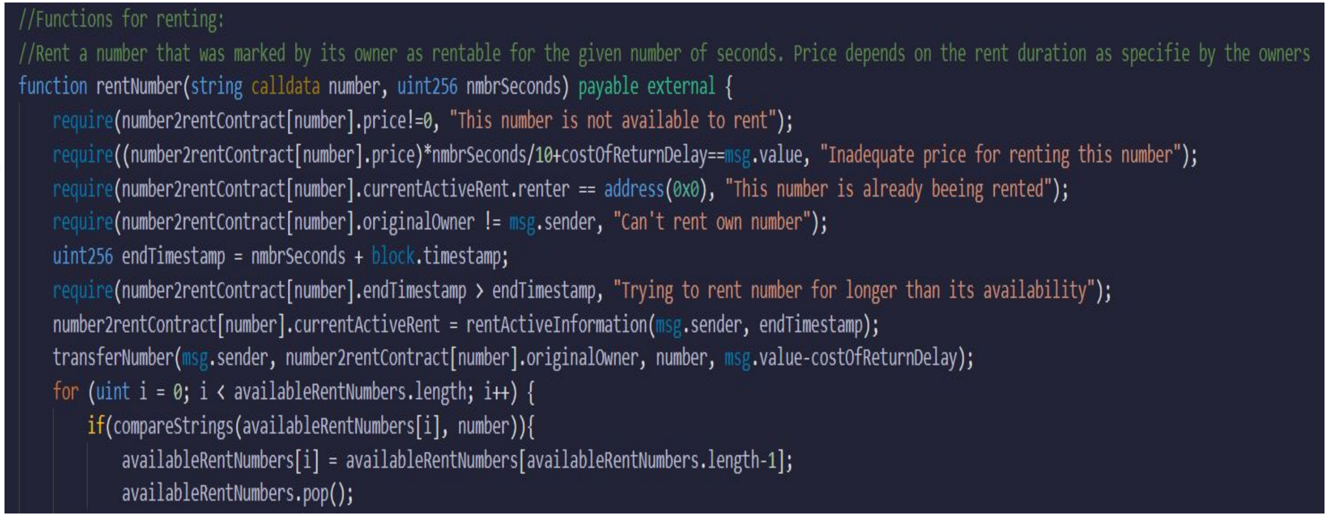




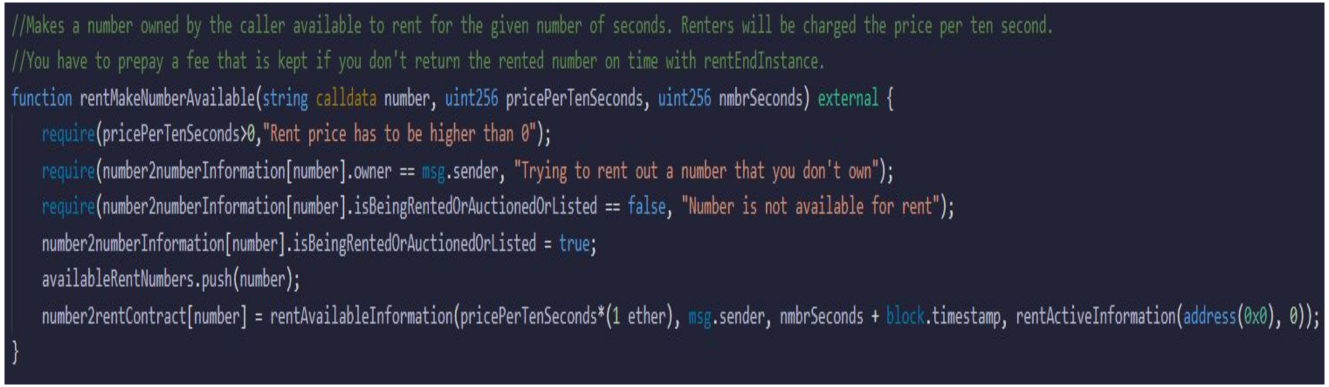
```
//Private helper funtion to give a free number to a reveicer
function receiveNumber(address receiver, string memory number) internal {
    number2numberInformation[number] = numberInformation(receiver,false);
    owner2account[receiver].ownedNumbers.push(number);
}

//Private helper funtion to transfer a number and mark the pay for the donor
function transferNumber(address receiver, address donor, string memory number, uint256 pay) internal {
    number2numberInformation[number] = numberInformation(receiver,false);
    string[] storage donorNumbers = owner2account[donor].ownedNumbers; // Has to be storage to reflect changes
    for (uint i = 0; i < donorNumbers.length; i++) {
        if(compareStrings(donorNumbers[i], number)){
            donorNumbers[i] = donorNumbers[donorNumbers.length-1];
            donorNumbers.pop();
        }
    }
    owner2account[receiver].ownedNumbers.push(number);
    owner2account[donor].accountBalance += pay;
}

//Private helper function to compare two strings for equality
function compareStrings(string memory s1, string memory s2) internal pure returns (bool){
    return keccak256(bytes(s1)) == keccak256(bytes(s2));
}
```

Buy a listed number and receive it (2/2)



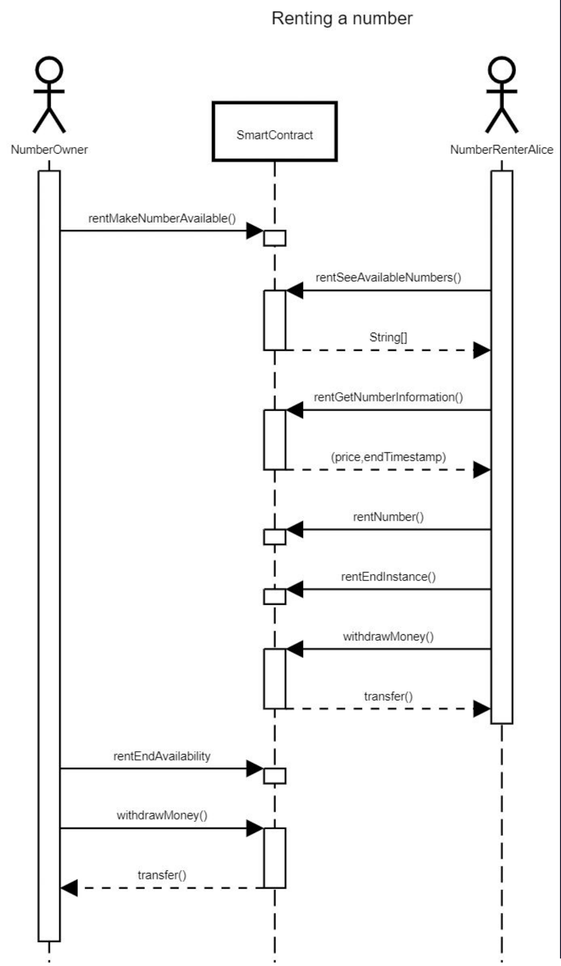


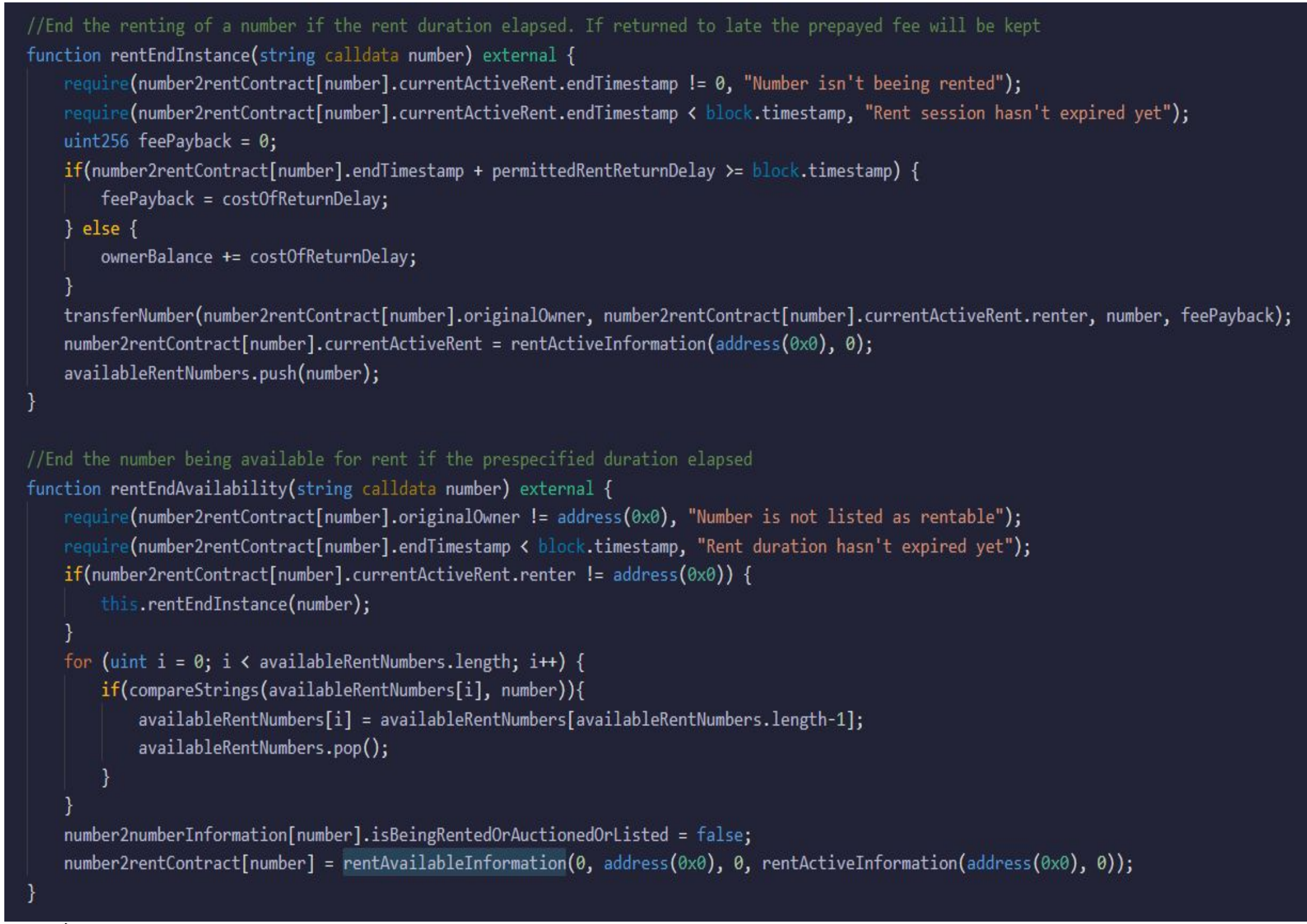
```
//Functions for renting:
//Rent a number that was marked by its owner as rentable for the given number of seconds. Price depends on the rent duration as specifie by the owners
function rentNumber(string calldata number, uint256 nmbrSeconds) payable external {
    require(number2rentContract[number].price!=0, "This number is not available to rent");
    require((number2rentContract[number].price)*nmbrSeconds/10+costOfReturnDelay==msg.value, "Inadequate price for renting this number");
    require(number2rentContract[number].currentActiveRent.renter == address(0x0), "This number is already beeing rented");
    require(number2rentContract[number].originalOwner != msg.sender, "Can't rent own number");
    uint256 endTimestamp = nmbrSeconds + block.timestamp;
    require(number2rentContract[number].endTimestamp > endTimestamp, "Trying to rent number for longer than its availability");
    number2rentContract[number].currentActiveRent = rentActiveInformation(msg.sender, endTimestamp);
    transferNumber(msg.sender, number2rentContract[number].originalOwner, number, msg.value-costOfReturnDelay);
    for (uint i = 0; i < availableRentNumbers.length; i++) {
        if(compareStrings(availableRentNumbers[i], number)){
            availableRentNumbers[i] = availableRentNumbers[availableRentNumbers.length-1];
            availableRentNumbers.pop();
```



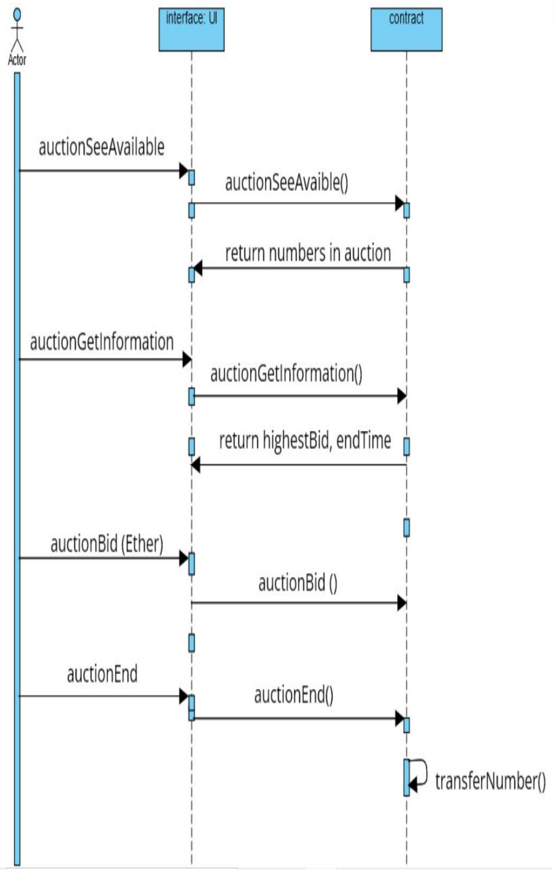
```
//Makes a number owned by the caller available to rent for the given number of seconds. Renters will be charged the price per ten second.
//You have to prepay a fee that is kept if you don't return the rented number on time with rentEndInstance.
function rentMakeNumberAvailable(string calldata number, uint256 pricePerTenSeconds, uint256 nmbrSeconds) external {
    require(pricePerTenSeconds>0,"Rent price has to be higher than 0");
    require(number2numberInformation[number].owner == msg.sender, "Trying to rent out a number that you don't own");
    require(number2numberInformation[number].isBeingRentedOrAuctionedOrListed == false, "Number is not available for rent");
    number2numberInformation[number].isBeingRentedOrAuctionedOrListed = true;
    availableRentNumbers.push(number);
    number2rentContract[number] = rentAvailableInformation(pricePerTenSeconds*(1 ether), msg.sender, nmbrSeconds + block.timestamp, rentActiveInformation(address(0x0), 0));
}
```

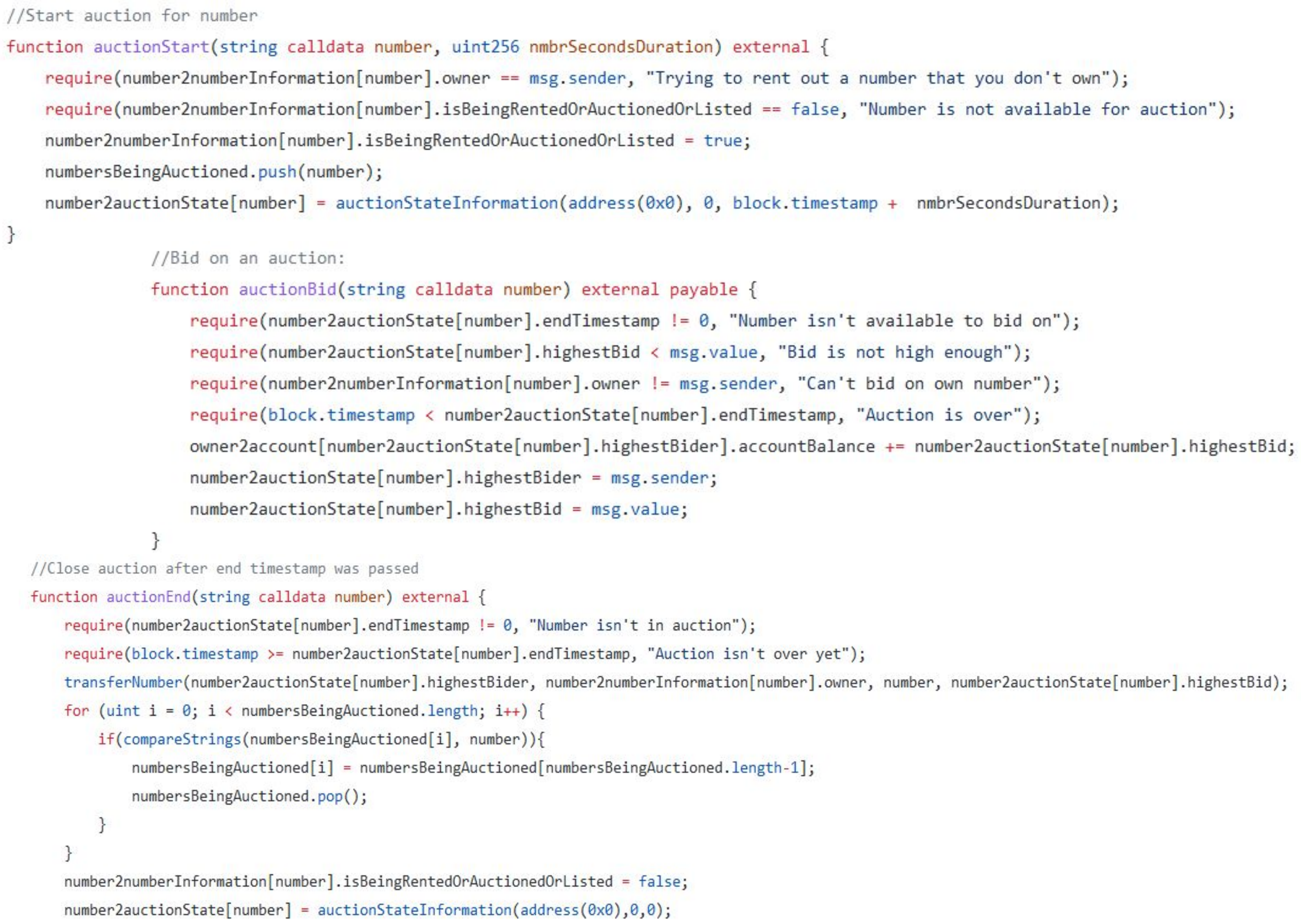
Rent a listed number (1/2)




```
//End the renting of a number if the rent duration elapsed. If returned to late the prepayed fee will be kept
function rentEndInstance(string calldata number) external {
    require(number2rentContract[number].currentActiveRent.endTimestamp != 0, "Number isn't beeing rented");
    require(number2rentContract[number].currentActiveRent.endTimestamp < block.timestamp, "Rent session hasn't expired yet");
    uint256 feePayback = 0;
    if(number2rentContract[number].endTimestamp + permittedRentReturnDelay >= block.timestamp) {
        feePayback = costOfReturnDelay;
    } else {
        ownerBalance += costOfReturnDelay;
    }
    transferNumber(number2rentContract[number].originalOwner, number2rentContract[number].currentActiveRent.renter, number, feePayback);
    number2rentContract[number].currentActiveRent = rentActiveInformation(address(0x0), 0);
    availableRentNumbers.push(number);
}

//End the number being available for rent if the prespecified duration elapsed
function rentEndAvailability(string calldata number) external {
    require(number2rentContract[number].originalOwner != address(0x0), "Number is not listed as rentable");
    require(number2rentContract[number].endTimestamp < block.timestamp, "Rent duration hasn't expired yet");
    if(number2rentContract[number].currentActiveRent.renter != address(0x0)) {
        this.rentEndInstance(number);
    }
    for (uint i = 0; i < availableRentNumbers.length; i++) {
        if(compareStrings(availableRentNumbers[i], number)){
            availableRentNumbers[i] = availableRentNumbers[availableRentNumbers.length-1];
            availableRentNumbers.pop();
        }
    }
    number2numberInformation[number].isBeingRentedOrAuctionedOrListed = false;
    number2rentContract[number] = rentAvailableInformation(0, address(0x0), 0, rentActiveInformation(address(0x0), 0));
}
```

Rent a listed number (2/2)





```
//Start auction for number
function auctionStart(string calldata number, uint256 nmbrSecondsDuration) external {
    require(number2numberInformation[number].owner == msg.sender, "Trying to rent out a number that you don't own");
    require(number2numberInformation[number].isBeingRentedOrAuctionedOrListed == false, "Number is not available for auction");
    number2numberInformation[number].isBeingRentedOrAuctionedOrListed = true;
    numbersBeingAuctioned.push(number);
    number2auctionState[number] = auctionStateInformation(address(0x0), 0, block.timestamp +  nmbrSecondsDuration);
}
```

```
//Bid on an auction:
function auctionBid(string calldata number) external payable {
    require(number2auctionState[number].endTimestamp != 0, "Number isn't available to bid on");
    require(number2auctionState[number].highestBid < msg.value, "Bid is not high enough");
    require(number2numberInformation[number].owner != msg.sender, "Can't bid on own number");
    require(block.timestamp < number2auctionState[number].endTimestamp, "Auction is over");
    owner2account[number2auctionState[number].highestBider].accountBalance += number2auctionState[number].highestBid;
    number2auctionState[number].highestBider = msg.sender;
    number2auctionState[number].highestBid = msg.value;
}
```

```
//Close auction after end timestamp was passed
function auctionEnd(string calldata number) external {
    require(number2auctionState[number].endTimestamp != 0, "Number isn't in auction");
    require(block.timestamp >= number2auctionState[number].endTimestamp, "Auction isn't over yet");
    transferNumber(number2auctionState[number].highestBider, number2numberInformation[number].owner, number, number2auctionState[number].highestBid);
    for (uint i = 0; i < numbersBeingAuctioned.length; i++) {
        if(compareStrings(numbersBeingAuctioned[i], number)){
            numbersBeingAuctioned[i] = numbersBeingAuctioned[numbersBeingAuctioned.length-1];
            numbersBeingAuctioned.pop();
        }
    }
    number2numberInformation[number].isBeingRentedOrAuctionedOrListed = false;
    number2auctionState[number] = auctionStateInformation(address(0x0),0,0);
```

Auction on a number (1/1)

Demo

Thank you!