**Code Description:**

We have used solidity language to generate a smart contract that works on “Proof of Permission”. Proof of Permission allows the members of the network to use or buy an asset of others in the network with their permission. Hence, a user in the network can give permission to the other users in the network to either rent or buy his asset.

To implement this Proof of Permission protocol, we have created functions to rent an asset, sell an asset, find the bidders in the network for an asset. This smart contract is a generalized contract that can be used for any type of asset i.e. car, house, pen, etc.

The owner of the asset creates an entry to the ADAM block only after transacting an asset with others. Any asset in our smart contract is identified with its id, name, cost at which the owner is willing to sell/rent, whether the asset is for rent/sell, the owner’s wallet address, number of days the asset is being rented for, cost per day. The following code gives properties of an asset in the network.

struct Asset {

       uint id;

       string name;

       uint cost;

       string choice;

       address key;

       address rented;

       uint rentDays;

       uint percost;

   }

The owner of the asset who is willing to rent/sell the asset can create an asset in the network and notify the members of the network with the help of the following create asset function.

function newAsset (uint id, string name,uint cost,string choice,uint rentDays,uint percost){

       if (msg.sender == owner){

           a.id = id;

           a.name = name;

           a.cost = cost;

           a.choice = choice;

           a.key = owner;

           a.rentDays = rentDays;

           a.percost = percost;

       }

   }

For a member of the network to use this asset, they must have the permission of the owner. This is done by the bidding process where the members who want to use the asset quote the amount that they are willing to pay for the asset. The properties of a quote are given by the following code.

struct Quote {

       uint assetId;

       uint quote;

       address quoterAddr;

       string choice;

   }

The quote for an asset is raised by the members of the network with the help of the following function.

function Quotes (uint id,uint quote,string choice){

       if (msg.sender == owner  && msg.sender!=a.rented){

           var quoteA = QuoteValues[msg.sender];

           quoteA.assetId = id;

           quoteA.quote = quote;

           quoteA.quoterAddr = msg.sender;

           quoteA.choice = choice;

           Quoters.push(msg.sender);

       }

   }

The owner of the asset will be able to view all the bids/quotes in the network raised for his asset and can choose to rent/sell his asset based on the bids. This is done with the help of get bidders function.

function getQuoters () view public returns (address []) {

       if (msg.sender == owner  && msg.sender!=a.rented){

           return Quoters;

       }

   }

The code in our smart contract allows the transaction to proceed only if it is the owner of the asset who is performing the transaction and the asset is not being rented/sold to anyone in the network. This avoids the double spending problem of the block chain network. Once these conditions are satisfied, the owner can either sell/rent his asset. The renting of the asset is done with the help of following function.

function rent (address Addr,uint percost,uint rentDays,string choice) returns (string){

       if (msg.sender == owner && (compareStrings(choice,"rent"))){

           a.rentDays = rentDays;

           a.percost = percost;

           a.rented = Addr;

           a.cost = percost \* rentDays;

           return "Rented";

       }

   }

Similarly, the selling of an asset is performed by the following code.

function sell (address Addr,uint rate) returns (string){

       if (msg.sender == owner  && msg.sender!=a.rented){

         a.key = Addr;

         a.cost = rate;

         return "Transaction Done";

       }

   }

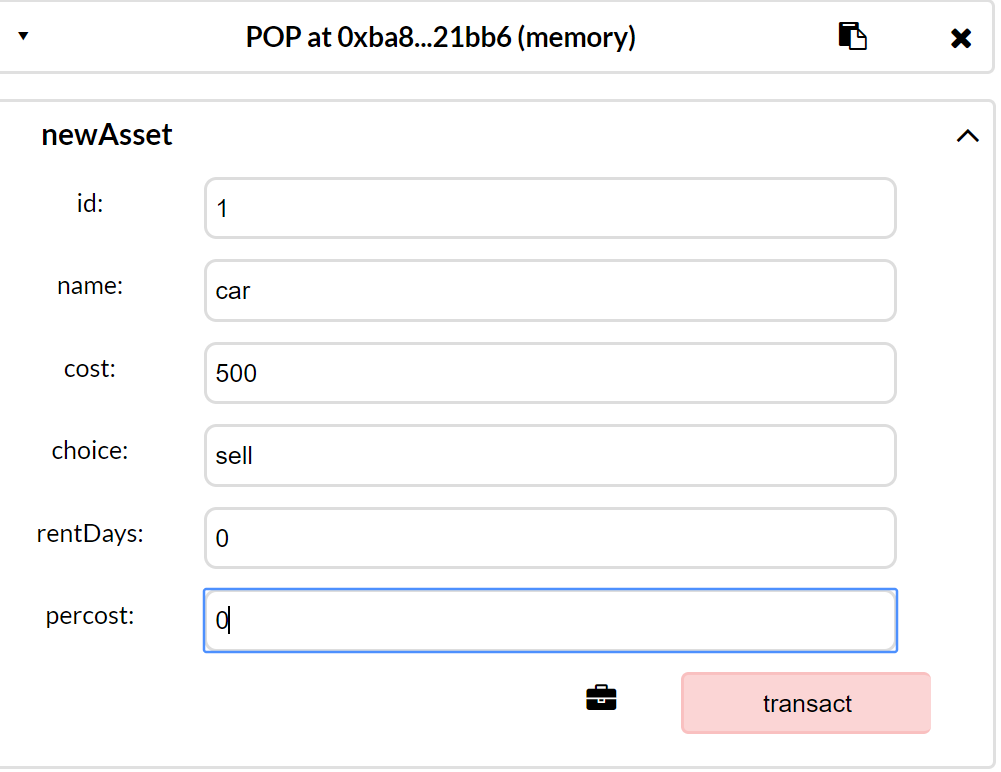
**Installation and Execution:**

We have used the “Remix.ethereum.org” IDE for developing smart contract. This IDE is available online and allows the developers to generate smart contract based on solidity language. Solidity is a contract oriented, high level language that is like C++, Java script and Python that helps in designing smart contracts for Ethereum block chain network.

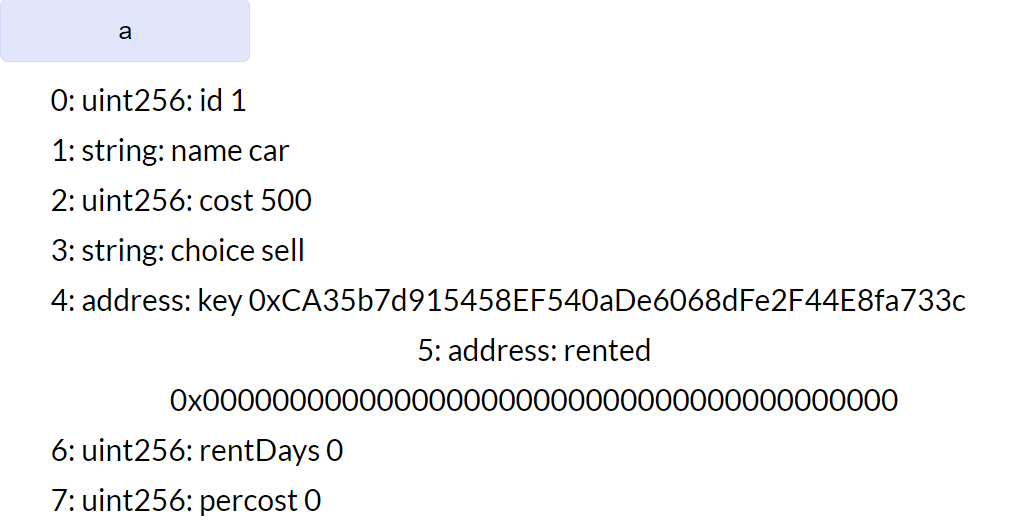
**Output:**

The following screenshots help us understand the working of the smart contract.

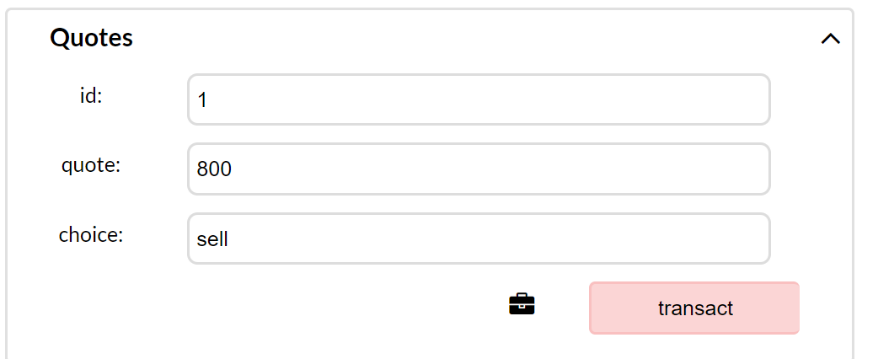
Creation of a new asset:



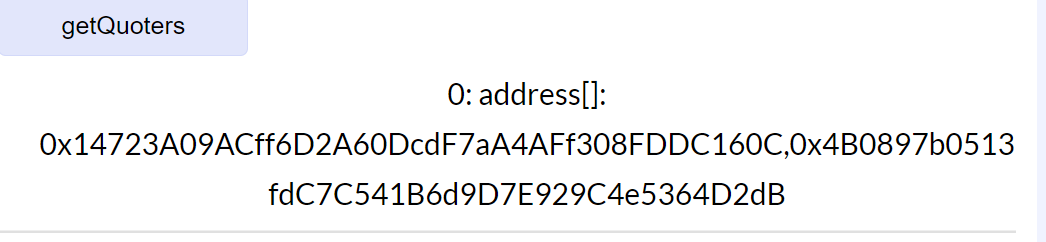
After creating an asset:



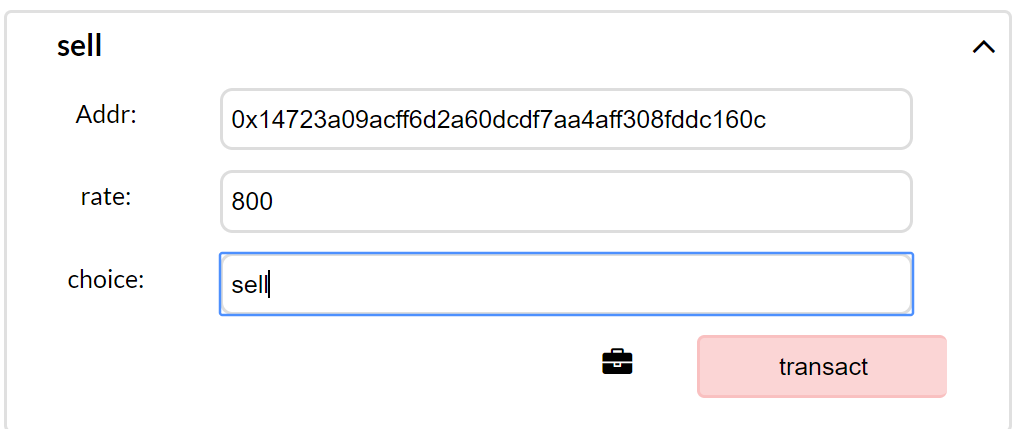
Quotation for the asset:



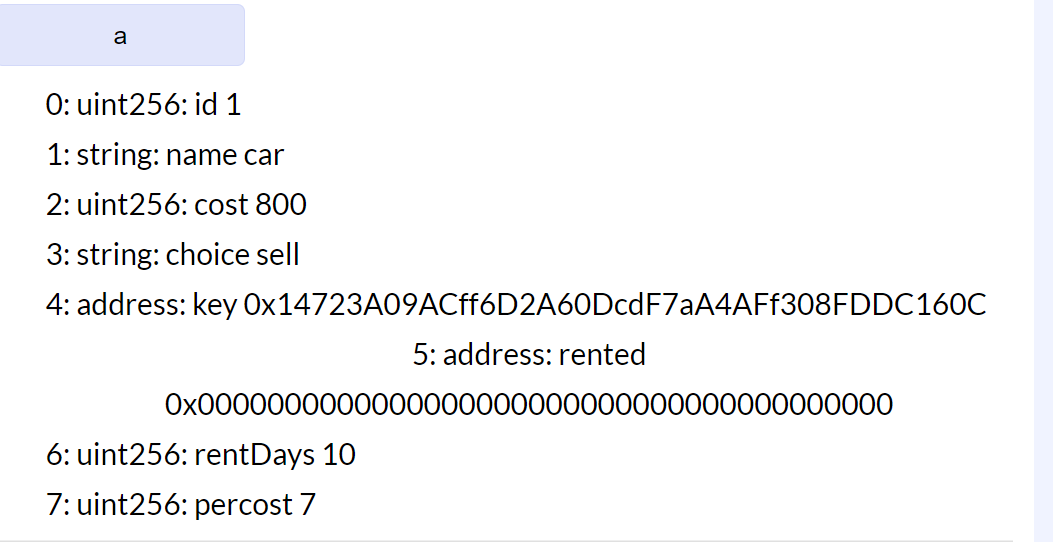
Viewing quotations for the asset:



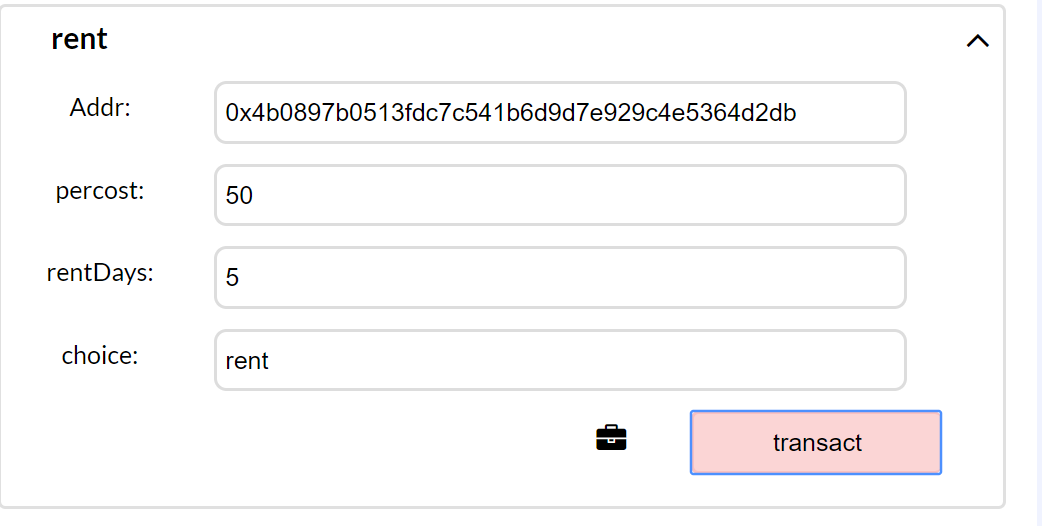
Selling the asset:



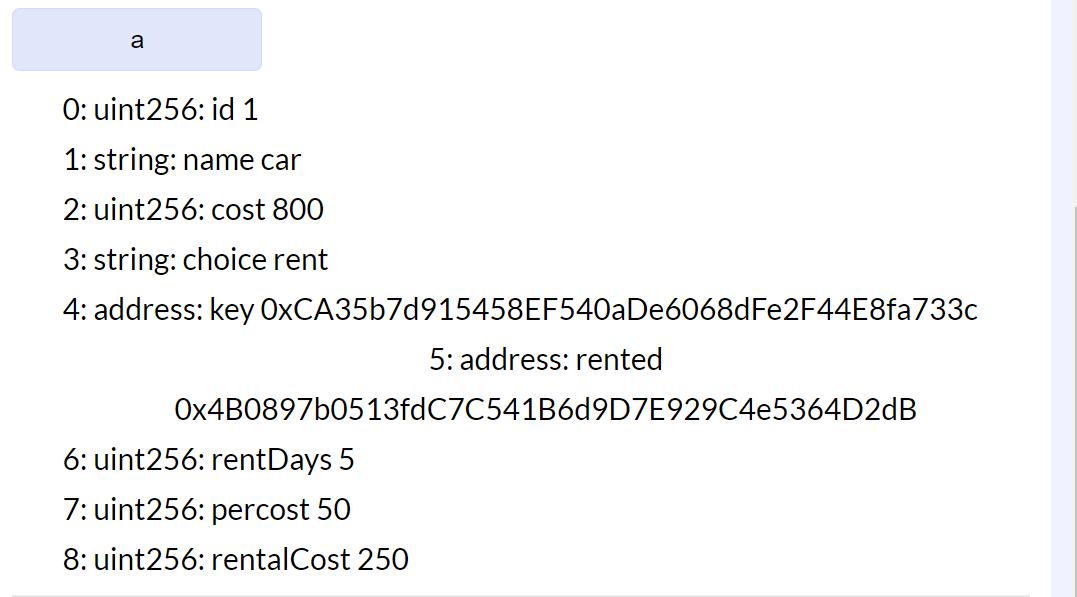
After the asset is sold, the details of the asset are updated:



Renting the asset:



After renting:



**Conclusion:**

The commledger, also called the community ledger, has the current and previous transactions and the timestamp and ADAM keys. The commledger works on POP protocol wherein the members of the network can rent or sell an asset to the other members in the network. The same is implemented in the smart contract. To provide the security for an asset, in commledger TALA is used, similarly, in the provided smart contract, the transactions are being locked with the hash value.

**References:**

<https://solidity.readthedocs.io/en/v0.4.24/>

<https://dzone.com/articles/implementing-a-simple-smart-contract-for-asset-tra>

<https://www.youtube.com/watch?v=v_hU0jPtLto>