Project Title: Fair and Transparent Blockchain based Tendering Framework - A Step Towards Open Governance

Logo:

Aims:

Main aim of this project is to create Auction between Government and citizens (Bidders) using blockchain.

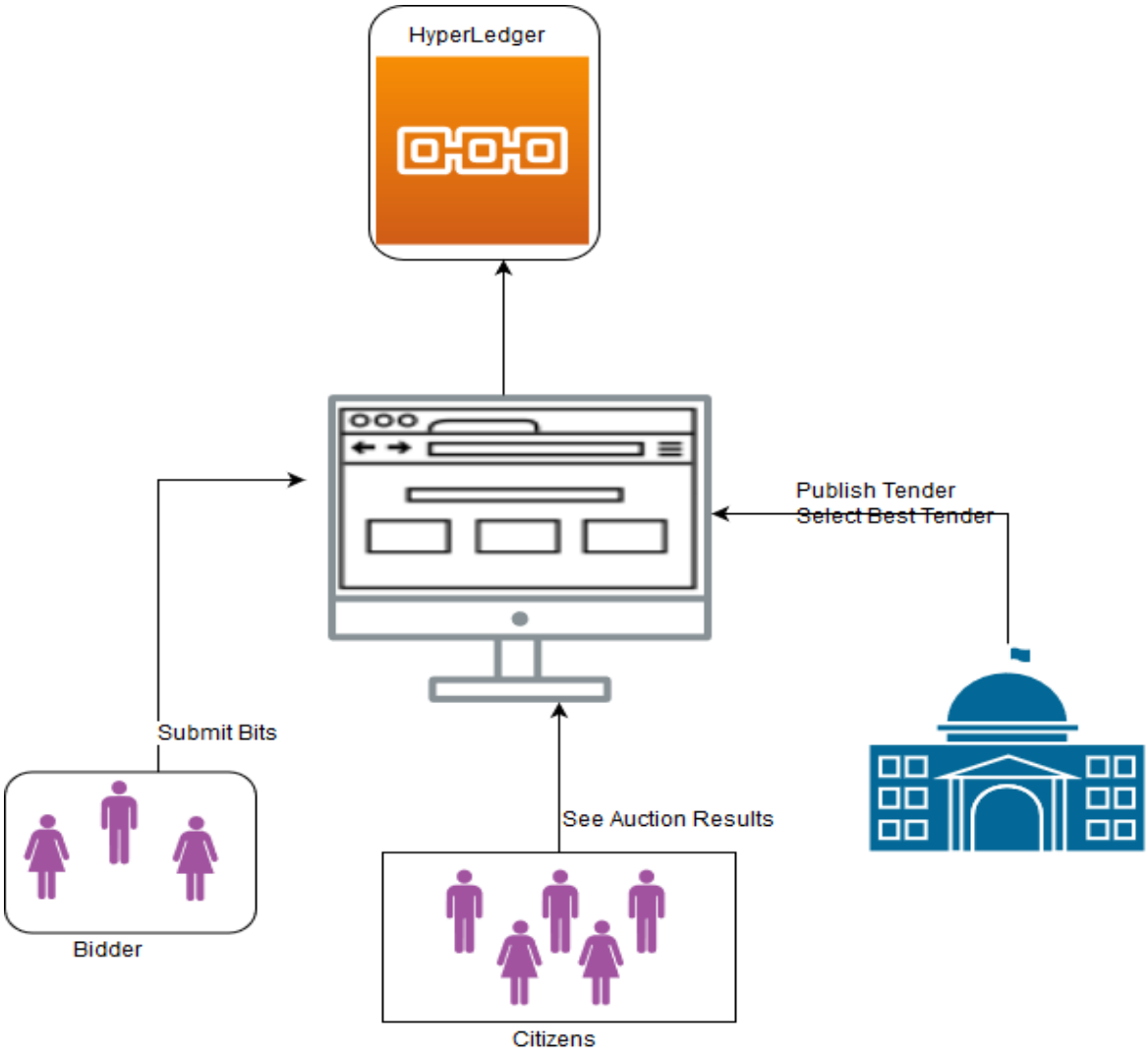
Stake Holders:

1. Government (Upload Auction)
2. Bidder (Do Bidding)
3. Public (See Winner)

Steps:

1. Government create auction for tender.
2. Set time limit (During that time limit no one can see status of bidding process). Once auction period started, government cannot modify its tender.
3. Once bidding period activate, everyone can bid having participantID.
4. After bidding period end, government review best proposal.
5. Government publish result
6. Everyone can review bids.

System Architecture:



Transaction:

A. Government Open Auction:
    a. auctionId
    b. comment
    c. date
    d. signature
B. Bidder Transaction
    a. auctionId
    b. participantId
    c. date
    d. comment
    e. biddingValue
    f. signature

We simply push data of winner in Bidder Transaction and caught using comments.

Accompanied:

a. Complete Flow
b. Hyperledger connection (deploy online)
c. Stunning UI
d. Report

Smart Contract:

```
pragma solidity >=0.4.22 <0.6.0;

contract auction{

  //event for printing bids
  event reviewEachBidsLogger(address _bidderAddress, uint _bidderValue);

  //event for printing loadBiddingDetailsForBidders
  event showBiddingDetailsForBidders(string autionStatement, uint biddingTime);

  //Government Initial Operations
  string public autionStatement;
  address public governmentAddress;
  uint biddingNumber;

  //Highest Bid
  address public highestBidderAddress;
  uint public highestBidValue;


  //Time Operations
  bool public biddingPeriodActivated;
  uint public biddingTimeStart;
  uint public biddingDuration;

  //Bids
  address[] public bidsAddresses;
  uint[] public bidsValues;


  constructor(
```

```
    uint _biddingNumber,
    string memory _autionStatement,
    uint _biddingDuration) public
  {
    biddingNumber = _biddingNumber;
    autionStatement = _autionStatement;
    biddingDuration = _biddingDuration;
    checkIsTimeExpires();
  }

  //Check is biddingPeriodActivated
  function checkIsTimeExpires() public
  {

    if(now > biddingTimeStart + biddingDuration)
    {
      //period is over
      biddingPeriodActivated = true;
    }
    else{
      //bidding in progress
      biddingPeriodActivated = false;
    }
  }

  //Bidder loads bids
  function loadBiddingDetailsForBidders() public
  {
    checkIsTimeExpires();
    if(biddingPeriodActivated)
    {
      emit showBiddingDetailsForBidders(autionStatement, biddingTimeStart + biddingDuration);
    }
  }

 //Set each bids if bidding period is active
  function setBid(address _bidderAddress, uint _bidderValue) public
  {
    checkIsTimeExpires();
    if(biddingPeriodActivated)
    {
      bidsAddresses[bidsAddresses.length] = _bidderAddress;
```

```
        bidsValues[bidsValues.length] = _bidderValue;
    }
}

//Government review Bids if biddingPeriodActivated is false
function reviewBids() public
{
    checkIsTimeExpires();
    if(!biddingPeriodActivated)
    {
        for(uint i=0; i<bidsAddresses.length; i++)
        {
            emit reviewEachBidsLogger(bidsAddresses[i], bidsValues[i]);
        }
    }
}

//Goverment publish results
function pushResults(address _highestBidderAddress, uint _highestBidValue) public{
    highestBidderAddress = _highestBidderAddress;
    highestBidValue = _highestBidValue;
}

//People Review Winner Address
function getHighestBidderAddress() public returns (address)
{
    checkIsTimeExpires();
    if(!biddingPeriodActivated)
    {
        return highestBidderAddress;
    }
}

//People Review Winner Uint
function getHighestBidderValue() public returns (uint)
{
    checkIsTimeExpires();
    if(!biddingPeriodActivated)
    {
        return highestBidValue;
    }
}
```