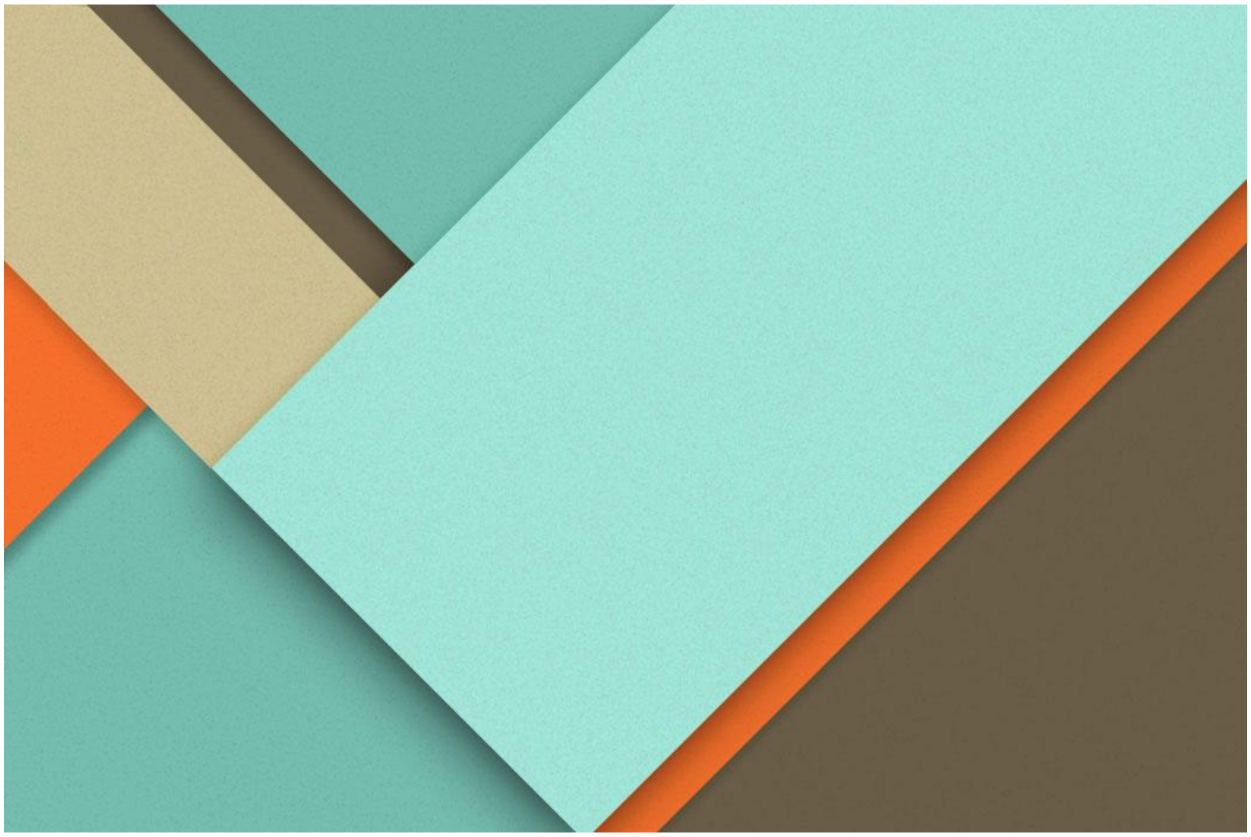


# DLaRS

Decentralized Land Registration System

—

Aman Aggarwal
2018327

Prasham Narayan
2018359

Saad Ahmad
2018409

## Idea

Land registry refers to registering and changing ownership of land which is done through the exchange of documents and performing transactions. Multiple entities might be involved in the process and the registry process goes through multiple intermediaries each of which act on mutual trust. There is generally a central authority involved that overlooks the transactions, ensures legal ownership, manages and stores relevant documents. We propose to bring this process to a distributed ledger to introduce trust and transparency in the process. Specifically, we design smart contracts for the advertisement of land by the owner, request by a prospective buyer to view the details of the land, confirmation of registration by the buyer, and the exchange of ownership.

## Motivation

Blockchain is one of the hottest topics in academic research and industry. The blockchain approach allows trusted trades among untrusted members in the network. Blockchain can bring security, trust, and privacy to the system which are the most crucial things people look for. In the traditional land registry system, records are maintained by a central local ledger, which can lead to tainting and forgery of records. Therefore, a lack of properly maintained records can lead to numerous property frauds. We can reduce these threats by creating a decentralized land registry system based on blockchain technology. Using the distributed and immutability properties of the blockchain we can enhance the security of the land registry system exponentially and reduce hacking, manipulation, and other security risks.

## What we learned and explored?

- We learned to develop, test, debug, and deploy smart contracts on Ethereum.
- We used "Remix - Ethereum IDE" to compile and test our code to save time.
- We learnt about how to effectively use modifiers, enums, visibilities (pure, view functions).
- We learnt to use Web3 along with Javascript to interact with the blockchain.
- We learnt to compile and deploy the blockchain using Truffle and Ganache.
- We also learned the usage of Metamask wallet to handle the signing of transactions and sending ether tokens from one account to another.

## Compiler Version:

### Solidity v0.8 Series

Solidity v0.8 series is the most recent version of the solidity language.

## Wallet:

### Metamask

We used Metamask to sign various transactions from different accounts connected to a local instance of Ganache.

## Test Network:

### Ganache

We used the Ganache network to deploy the network and get dummy accounts. These accounts are added to the Metamask wallet.

## Frontend and Connector:

### React JS and Web3JS

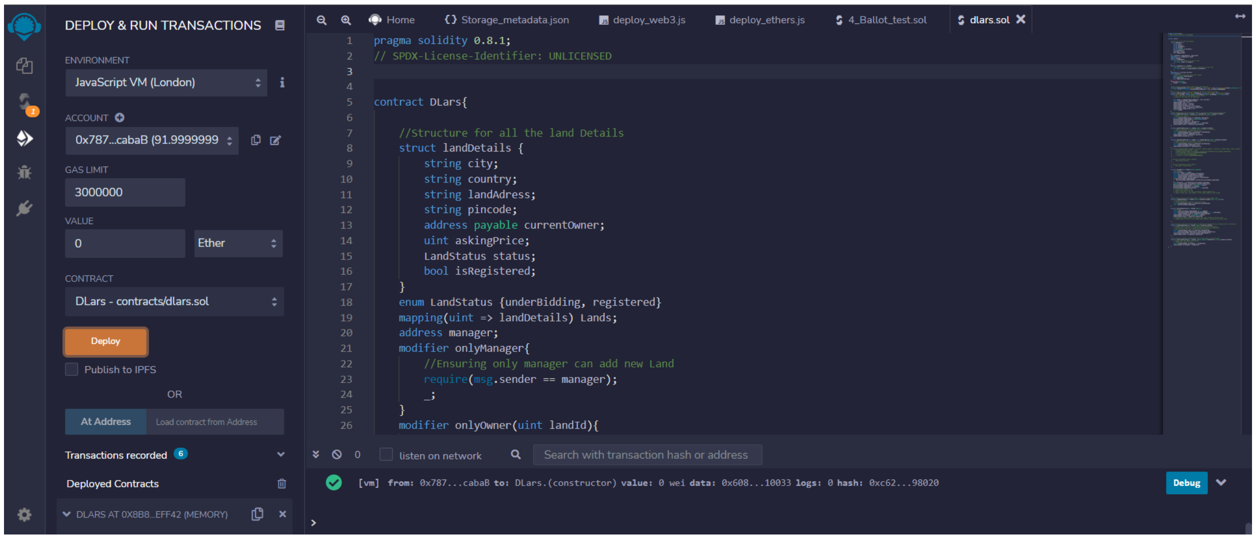
Used this library to interact with the local ethereum network and connect our react application with our solidity code.

## Deploying the blockchain network on Ganache:

We also deployed our smart contract on the Ganache, a test network for the Ethereum chain, to get a feel of working on a real network similar to Ethereum Mainnet.

It provides 10 dummy accounts with 100 ether balance in each account. We tested our DLaRS contract using these accounts and connecting them to metamask.

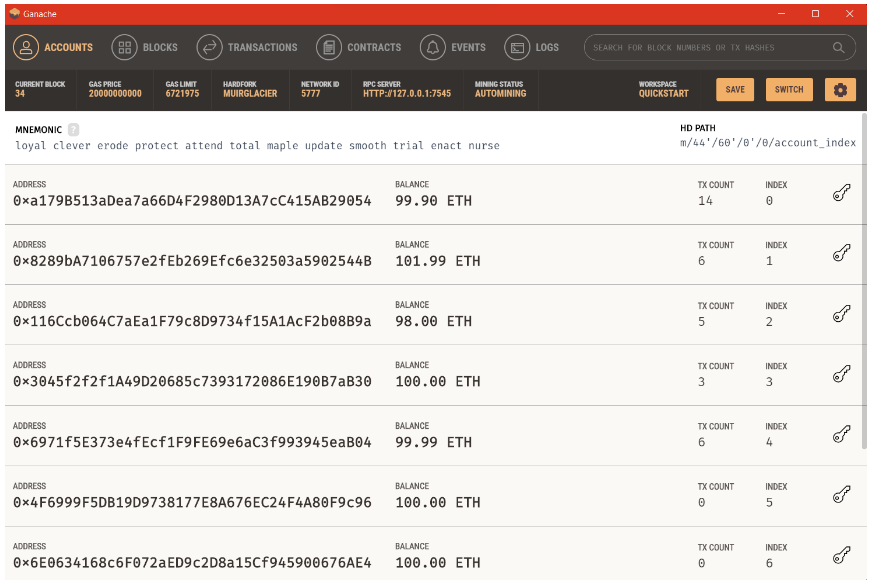
To compile and deploy our smart contract we also used Remix IDE.



ENVIRONMENT: JavaScript VM (London)
ACCOUNT: 0x787...cabaB (91.9999999
GAS LIMIT: 3000000
VALUE: 0 Ether
CONTRACT: DLars - contracts/dlars.sol
Publish to IPFS

```
pragma solidity 0.8.1;
// SPDX-License-Identifier: UNLICENSED


contract DLars{

    //Structure for all the land Details
    struct landDetails {
        string city;
        string country;
        string landAdress;
        string pincode;
        address payable currentOwner;
        uint askingPrice;
        LandStatus status;
        bool isRegistered;
    }
    enum LandStatus {underBidding, registered}
    mapping(uint => landDetails) Lands;
    address manager;
    modifier onlyManager{
        //Ensuring only manager can add new Land
        require(msg.sender == manager);
        _;
    }
    modifier onlyOwner(uint landId){
```

```
[vm] from: 0x787...cabaB to: DLars.(constructor) value: 0 wei data: 0x608...10033 logs: 0 hash: 0xc62...98020
```

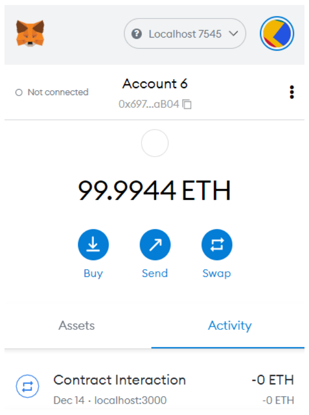


| CURRENT BLOCK | GAS PRICE | GAS LIMIT | HARDFORK | NETWORK ID | RPC SERVER | MINING STATUS | WORKSPACE |
|---|---|---|---|---|---|---|---|
| 34 | 20000000000 | 6721975 | MUIRGLACIER | 5777 | HTTP://127.0.0.1:7545 | AUTOMINING | QUICKSTART |

MNEMONIC
loyal clever erode protect attend total maple update smooth trial enact nurse

HD PATH
m/44'/60'/0'/0/account_index

| ADDRESS | BALANCE | TX COUNT | INDEX |
|---|---|---|---|
| 0×a179B513aDea7a66D4F2980D13A7cC415AB29054 | 99.90 ETH | 14 | 0 |
| 0×8289bA7106757e2fFb269Efc6e32503a5902544B | 101.99 ETH | 6 | 1 |
| 0×116Ccb064C7aEa1F79c8D9734f15A1AcF2b08B9a | 98.00 ETH | 5 | 2 |
| 0×3045f2f2f1A49D20685c7393172086E190B7aB30 | 100.00 ETH | 3 | 3 |
| 0×6971f5E373e4fEcf1F9FE69e6aC3f993945eaB04 | 99.99 ETH | 6 | 4 |
| 0×4F6999F5DB19D9738177E8A676EC24F4A80F9c96 | 100.00 ETH | 0 | 5 |
| 0×6E0634168c6F072aED9c2D8a15Cf945900676AE4 | 100.00 ETH | 0 | 6 |

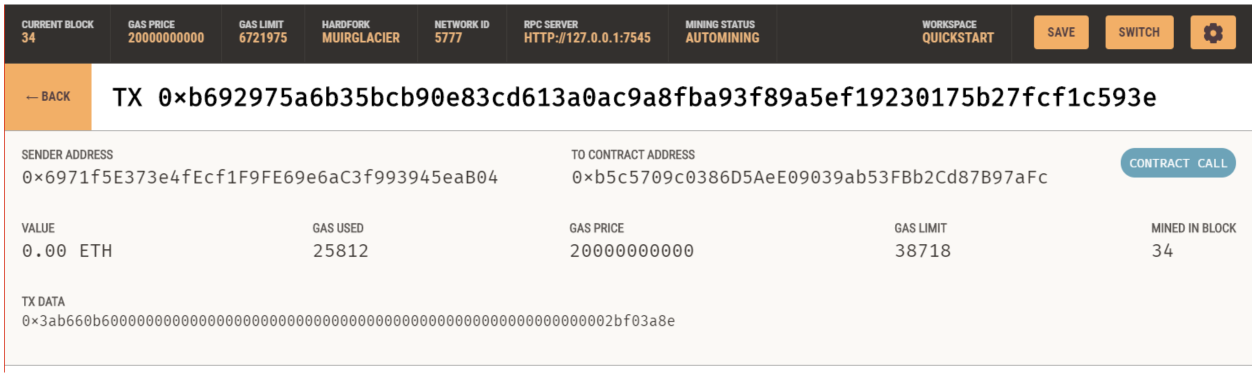


Account 6
0x697...aB04

99.9944 ETH

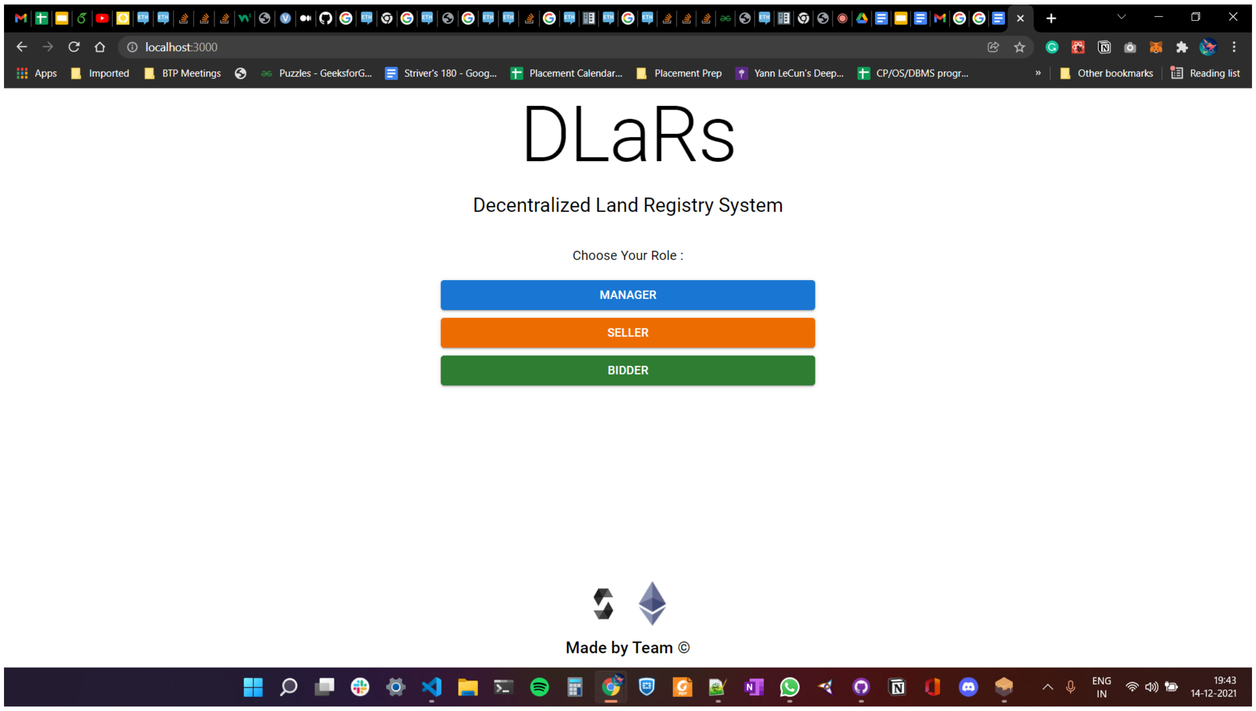
Contract Interaction -0 ETH
Dec 14 · localhost:3000 -0 ETH

The details of the transaction of our contract deployed on the VM.



```
TX 0×b692975a6b35bcb90e83cd613a0ac9a8fba93f89a5ef19230175b27fcf1c593e

SENDER ADDRESS                                   TO CONTRACT ADDRESS
0×6971f5E373e4fEcf1F9FE69e6aC3f993945eaB04       0×b5c5709c0386D5AeE09039ab53FBb2Cd87B97aFc

VALUE          GAS USED          GAS PRICE          GAS LIMIT          MINED IN BLOCK
0.00 ETH       25812             20000000000        38718              34

TX DATA
0×3ab660b600000000000000000000000000000000000000000000000000000000002bf03a8e
```

This particular transaction is made by the “manager” who registers land.

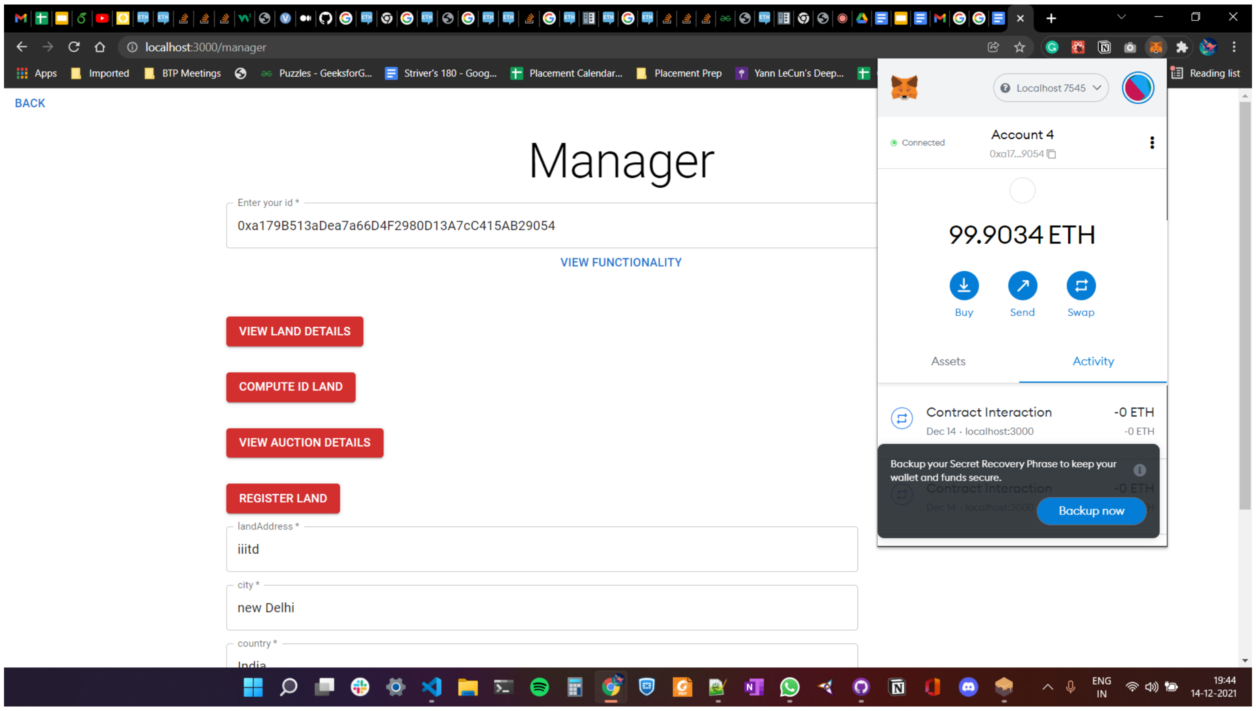


DLaRs

Decentralized Land Registry System

Choose Your Role :

Made by Team ©



localhost:3000/manager

BACK

Manager

Enter your id *
0xa179B513aDea7a66D4F2980D13A7cC415AB29054

VIEW FUNCTIONALITY

VIEW LAND DETAILS

COMPUTE ID LAND

VIEW AUCTION DETAILS

REGISTER LAND

landAddress *
iiitd

city *
new Delhi

country *
India

Localhost 7545

Connected

Account 4
0xa17...9054

99.9034 ETH

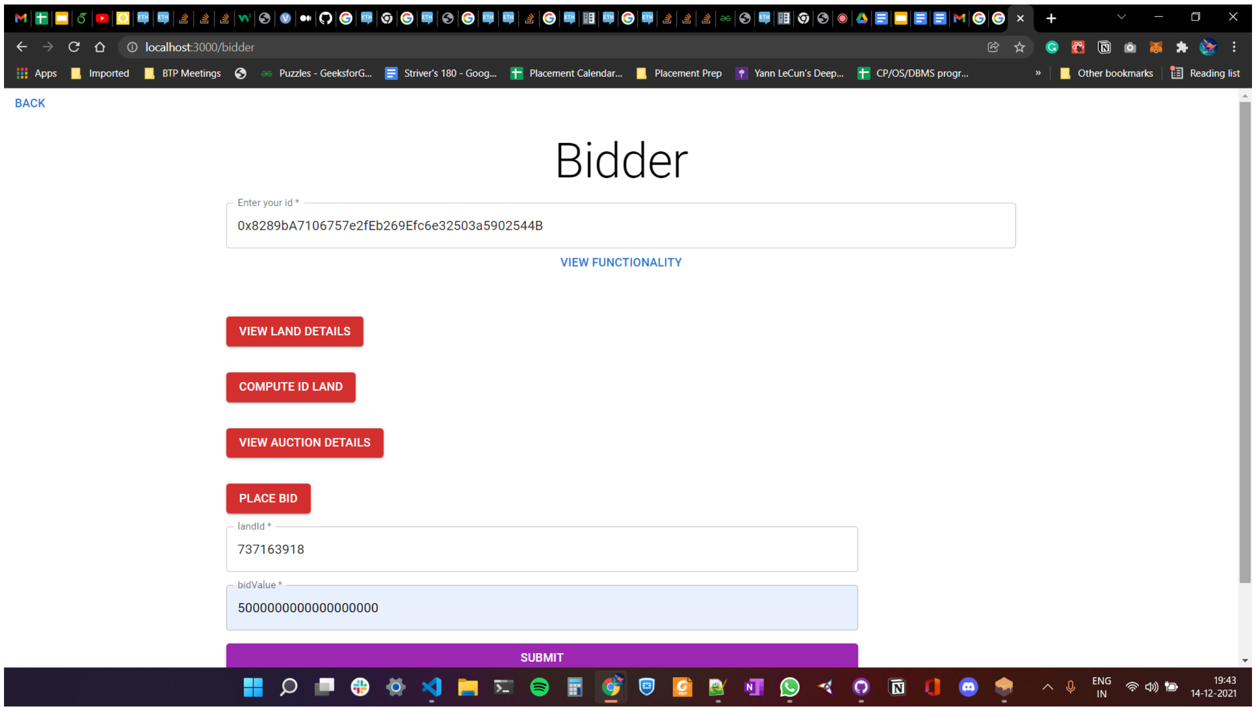
Buy Send Swap

Assets Activity

Contract Interaction -0 ETH
Dec 14 · localhost:3000 -0 ETH

Backup your Secret Recovery Phrase to keep your wallet and funds secure.

Backup now



localhost:3000/bidder

BACK

Bidder

Enter your id *
0x8289bA7106757e2fEb269Efc6e32503a5902544B

VIEW FUNCTIONALITY

VIEW LAND DETAILS

COMPUTE ID LAND

VIEW AUCTION DETAILS

PLACE BID

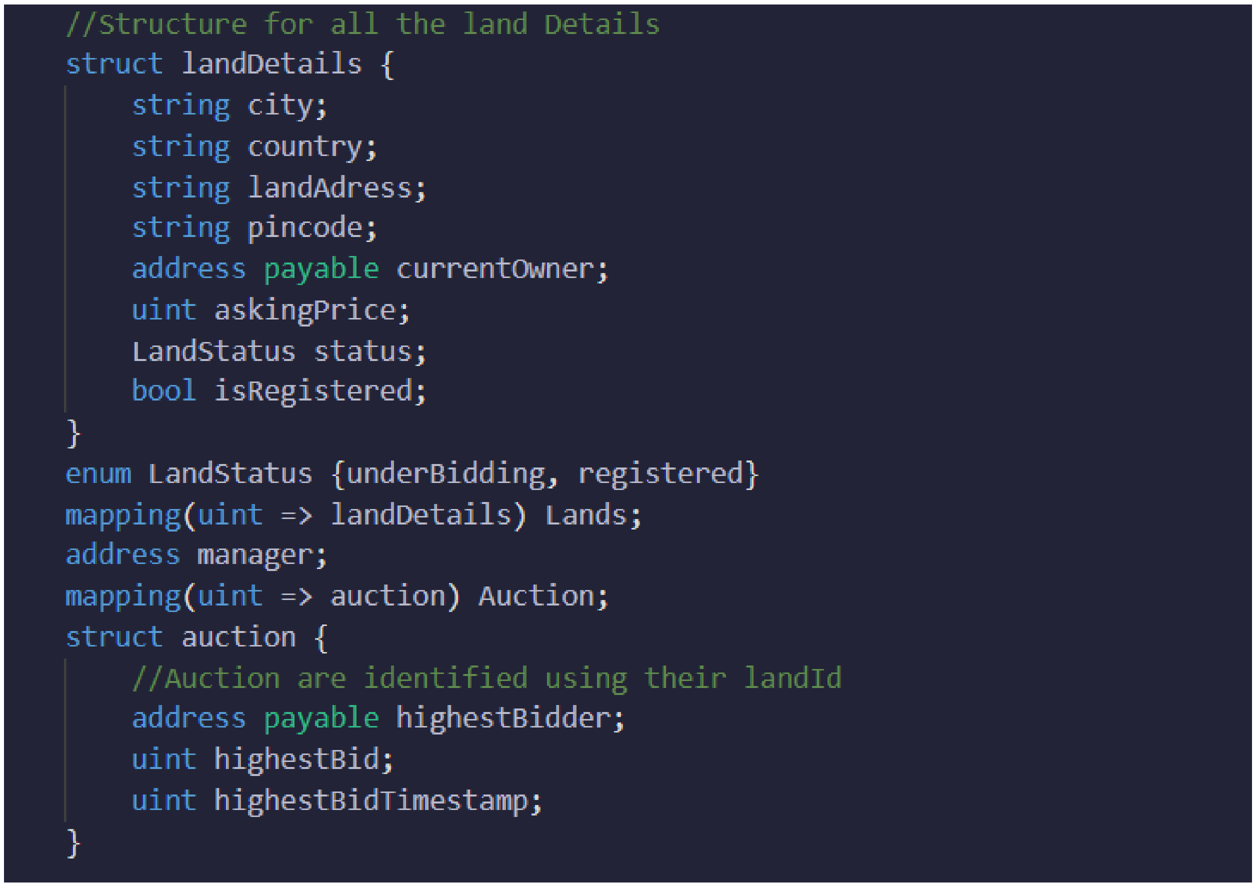
landId *
737163918

bidValue *
5000000000000000000

SUBMIT

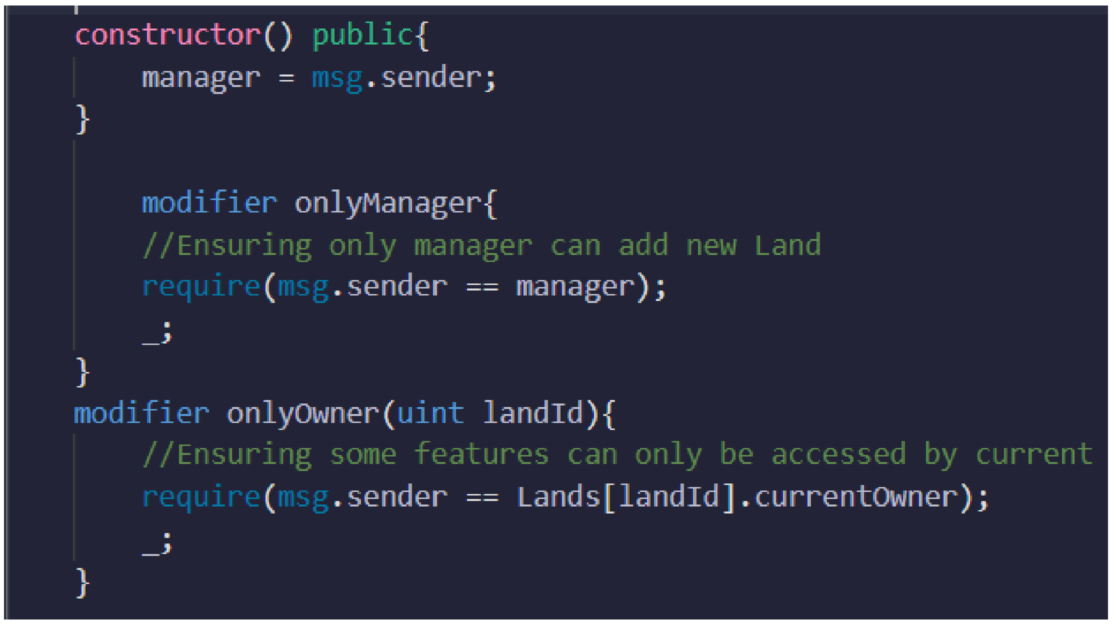
## Code Snippets:

DLaRS Contract variables



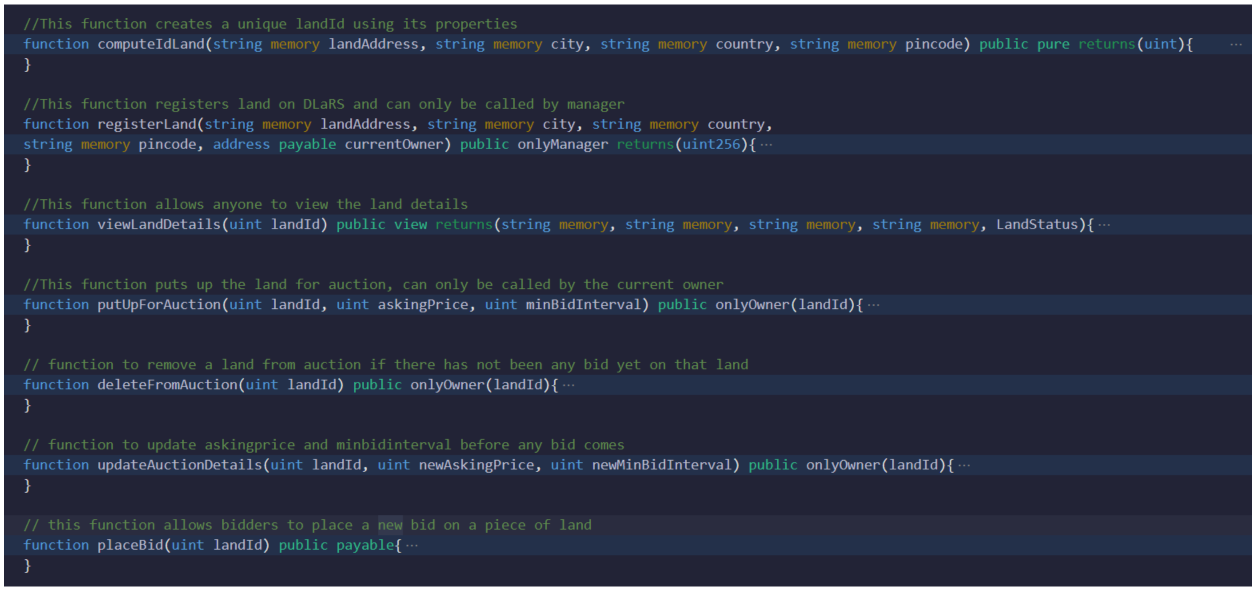
```
//Structure for all the land Details
struct landDetails {
    string city;
    string country;
    string landAdress;
    string pincode;
    address payable currentOwner;
    uint askingPrice;
    LandStatus status;
    bool isRegistered;
}
enum LandStatus {underBidding, registered}
mapping(uint => landDetails) Lands;
address manager;
mapping(uint => auction) Auction;
struct auction {
    //Auction are identified using their landId
    address payable highestBidder;
    uint highestBid;
    uint highestBidTimestamp;
}
```

Constructor and Modifiers

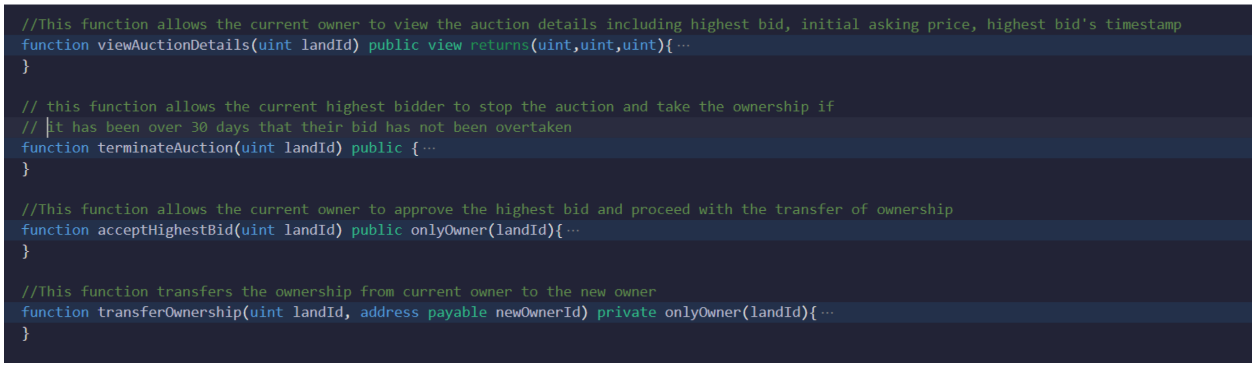


```
constructor() public{
    manager = msg.sender;
}

    modifier onlyManager{
    //Ensuring only manager can add new Land
    require(msg.sender == manager);
    _;
}
modifier onlyOwner(uint landId){
    //Ensuring some features can only be accessed by current
    require(msg.sender == Lands[landId].currentOwner);
    _;
}
```

Functions



```
//This function creates a unique landId using its properties
function computeIdLand(string memory landAddress, string memory city, string memory country, string memory pincode) public pure returns(uint){ ···
}

//This function registers land on DLaRS and can only be called by manager
function registerLand(string memory landAddress, string memory city, string memory country,
string memory pincode, address payable currentOwner) public onlyManager returns(uint256){ ···
}

//This function allows anyone to view the land details
function viewLandDetails(uint landId) public view returns(string memory, string memory, string memory, string memory, LandStatus){ ···
}

//This function puts up the land for auction, can only be called by the current owner
function putUpForAuction(uint landId, uint askingPrice, uint minBidInterval) public onlyOwner(landId){ ···
}

// function to remove a land from auction if there has not been any bid yet on that land
function deleteFromAuction(uint landId) public onlyOwner(landId){ ···
}

// function to update askingprice and minbidinterval before any bid comes
function updateAuctionDetails(uint landId, uint newAskingPrice, uint newMinBidInterval) public onlyOwner(landId){ ···
}

// this function allows bidders to place a new bid on a piece of land
function placeBid(uint landId) public payable{ ···
}
```



```
//This function allows the current owner to view the auction details including highest bid, initial asking price, highest bid's timestamp
function viewAuctionDetails(uint landId) public view returns(uint,uint,uint){ ···
}

// this function allows the current highest bidder to stop the auction and take the ownership if
// it has been over 30 days that their bid has not been overtaken
function terminateAuction(uint landId) public { ···
}

//This function allows the current owner to approve the highest bid and proceed with the transfer of ownership
function acceptHighestBid(uint landId) public onlyOwner(landId){ ···
}

//This function transfers the ownership from current owner to the new owner
function transferOwnership(uint landId, address payable newOwnerId) private onlyOwner(landId){ ···
}
```

## Links:

- Github: https://github.com/DaasDaham/DLaRS
- Demo Video: https://youtu.be/sWQ7uIY0wQw
- Complete Presentation Video (including Demo) - https://drive.google.com/file/d/1irsHVYZA1ulGXd61Usi2bhukTXq9xvj_/view?usp=sharing

## Future Prospects:

There are a few functionalities we would definitely want to see in our project in future, that includes:

1. Verifying details of the users and allowing sharing of documents using IPFS.
2. Incorporating a 2nd contract that acts as a government officer and adds additional security to our system.
3. Use solidity design patterns such as Guard check to implement functionality and reduce security risks.
4. Try and deploy the contract on mainnet to check real-world feasibility.

## Contribution:

All team members gave equal contribution to the project throughout the semester.