Project School Certificate

Title : Real Estate Application using Ethereum

Faculty Incharge : Dr.Rajasekaran Subramanian

Session Duration : 13/10/2022 – 12/01/2023

Name : Pavan Kumar K N

Roll Number : 245320748036

Class : CSM – A

Signature of faculty **Signature of Student**

Venture World – A Real Estate Application using Ethereum.

> *Product Name: Real Estate Application using Ethereum*
> *Phase: Completed*

TABLE OF CONTENTS

Introduction

Purpose of the Project

This application is a blockchain-based platform for buying and selling real estate properties. It utilizes the power of blockchain technology to provide a secure and transparent way for users to conduct real estate transactions. The application is designed to streamline the buying and selling process, reducing the need for intermediaries and increasing the speed and efficiency of transactions by using various modes of conducting the transaction namely auction, listing and donation

Problem with the existing system

Traditional real estate systems rely on centralized databases and intermediaries, which can make it difficult for buyers and sellers to access accurate and up-to-date information about properties and transactions. The buying and selling process can be time-consuming and costly, due to the need for intermediaries and the need to manually verify and transfer property ownership. It can be difficult to verify the accuracy of property data, such as location and ownership history. Personal and financial information can be shared or sold to third parties without the user's consent in traditional real estate systems. Traditional real estate systems are vulnerable to hacking and fraud, which can lead to the loss of personal and financial information.

Proposed System

Blockchain technology can provide a secure and tamper-proof record of property ownership and transaction history, increasing transparency in the real estate market. Blockchain-based applications can automate the process through the use of smart contracts, reducing the need for intermediaries and increasing the speed and efficiency of transactions. Blockchain technology can provide a tamper-proof record of property data, allowing buyers and sellers to trust that the information they are accessing is accurate. Blockchain technology allows for the creation of private blockchains which can protect users' privacy by encrypting their data and giving them complete control over who has access to it. Blockchain technology provides advanced security measures, such as encryption and secure key storage, which can protect users' personal and financial information.

In addition to this, one can sell their assets by the methods of auctioning them and awarding the property as per the highest bid normally, listing them as per the specified price and donating the asset to any entity as per the requirement.

About Ethereum

Ethereum is a blockchain with a computer embedded in it. It is the foundation for building apps and organizations in a decentralized, permissionless, censorship-resistant way.

In the Ethereum universe, there is a single, canonical computer (called the Ethereum Virtual Machine, or EVM) whose state everyone on the Ethereum network agrees on. Everyone who participates in the Ethereum network (every Ethereum node) keeps a copy of the state of this computer. Additionally, any participant can broadcast a request for this computer to perform arbitrary computation. Whenever such a request is broadcast, other participants on the network verify, validate, and carry out ("execute") the computation. This execution causes a state change in the EVM, which is committed and propagated throughout the entire network.

The EVM's physical instantiation can't be described in the same way that one might point to a cloud or an ocean wave, but it does exist as one single entity maintained by thousands of connected computers running an Ethereum client.

The Ethereum protocol itself exists solely for the purpose of keeping the continuous, uninterrupted, and immutable operation of this special state machine. It's the environment in which all Ethereum accounts and smart contracts live. At any given block in the chain, Ethereum has one and only one 'canonical' state, and the EVM is what defines the rules for computing a new valid state from block to block.

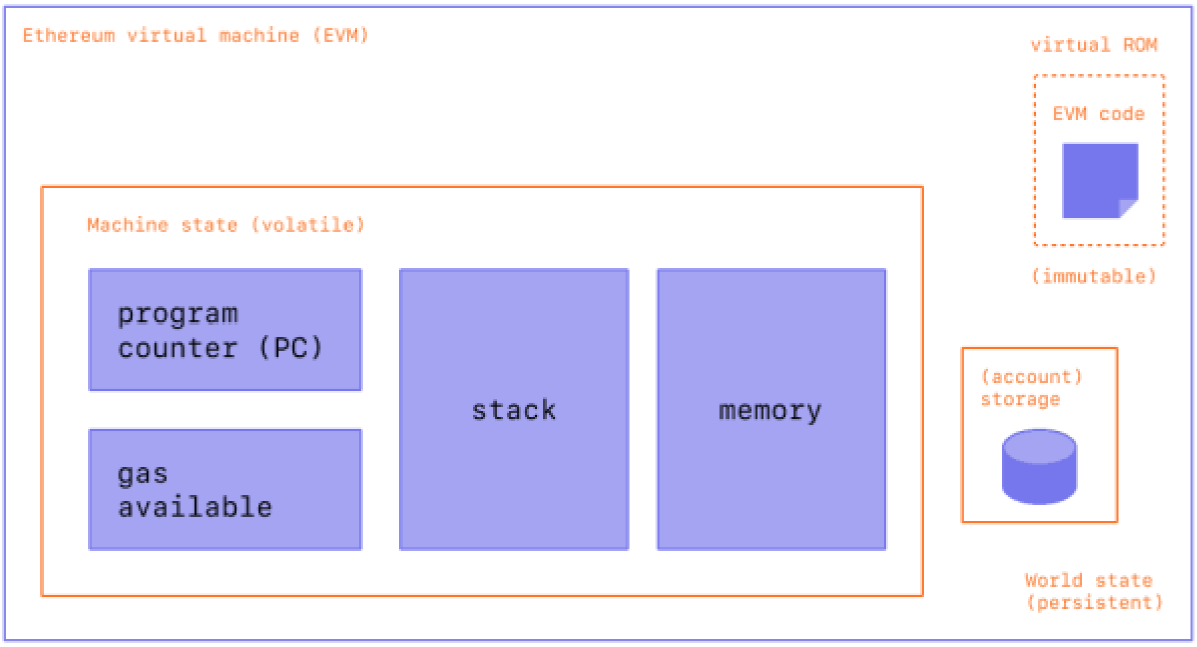
The analogy of a 'distributed ledger' is often used to describe blockchains like Bitcoin, which enable a decentralized currency using fundamental tools of cryptography. The ledger maintains a record of activity which must adhere to a set of rules that govern what someone can and cannot do to modify the ledger. For example, a Bitcoin address cannot spend more Bitcoin than it has previously received. These rules underpin all transactions on Bitcoin and many other blockchains.

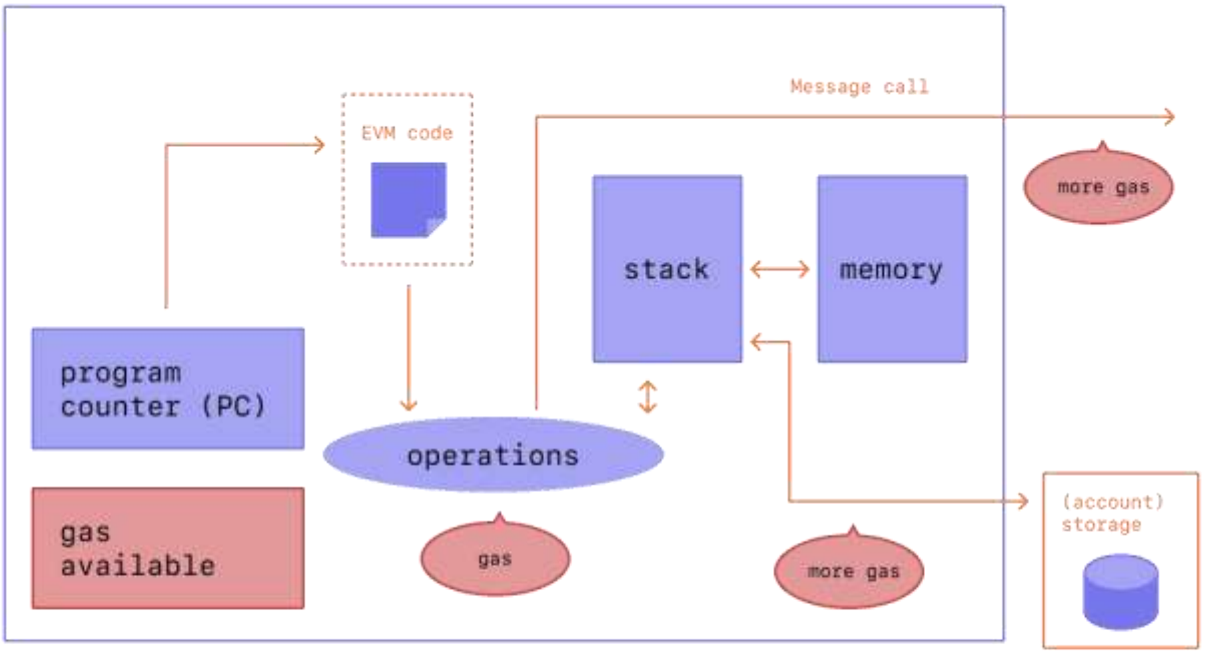
Compiled smart contract bytecode executes as a number of EVM opcodes which perform standard stack operations. The EVM also implements a number of blockchain-specific stack operations, such as ADDRESS, BALANCE etc.

While Ethereum has its own native cryptocurrency (Ether) that follows almost exactly the same intuitive rules, it also enables a much more powerful function: smart contracts. For this more complex feature, a more sophisticated analogy is required. Instead of a distributed ledger, Ethereum is a distributed state machine. Ethereum's state is a large data structure which holds not only all accounts and balances, but a machine state, which can change from block to block according to a pre-defined set of rules, and which can execute arbitrary machine code. The specific rules of changing state from block to block are defined by the EVM .

The transactions that we talked about earlier happens through a node that the Hyderledger Fabric features. This node is called an Orderer or an ordering node. This Orderer does the transaction ordering, which along with the other nodes forms the ordering services.

To summarize, to limit the scams that can take place in the real estate business along-side maintaining credibility and privacy with the transparency of necessary details, we have used this framework of the blockchain in our application.





Requirements

Functional Requirements

- Certificate Authorization
- Administrative functions
- Solidity functions to manipulate ownership.
- Smart Contracts
- Communities

Non-Functional Requirements

- Access to Property Details.
- Property Documents as soft copies.
- Scalability
- Reliability
- Identity management
- Solidity functionality
- Maintainability
- Security
- Environmental
- Data Integrity
- Usability

Software Requirements

- Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12, or Windows
- Metamask Extension
- Remix IDE
- Solidity: 0.8.17
- A code editor of your choice, we recommend Visual Studio Code.

Hardware Requirements

- Required CPU Architecture : x86_64
- Preferred Number of CPUs : 2 (minimum 1)
- Minimum Amount of RAM : 4 GB
- Internet Access : Required
- OS required : Any

Product Design

Product Overview

Our project involved us to work with the Ethereum and develop an application to serve as a platform to buy and sell assets within a venture. The assets referred to in this application are basically lands.

We focused on developing a secure transaction system which allows users within the venture to be able to interact with the properties. So, what our application does is fairly simple. A user can perform multiple roles – buyer and seller/owner, and proceed accordingly.

When the user logs in, he/she will be met with a welcome page showing all the available assets in the venture and the description of the venture. He/she can proceed to purchase a listed property or participate in an auction for any property. When an owner initiates a Listing or an Auction, everyone in the network can view it. The owner can cancel the listings and auctions if needed and all the bids are refunded automatically. He/she can also choose to donate their property.

After any transaction, if the payment is verified, the nft will automatically be transferred and ownership changes. The bids canbe cancelled at any time before an Auction ends.

To develop this project, we used Solidity to build the Smart Contract, EthersJS, VueJS, ElementUI on the front-end.

Product Objectives

- Decentralized platform - It provides the user easier and optimal usage of resourcesand services. It also improves data reconciliation thus ensuring data security.
- Transparent platform - Maintains a complete history of previous transactions withinthe network available to all the users.
- Verified Assets - Providing all the verified asset description and thus reducing the chances of fraud by through check of documents. The transaction can be reverted if any malpractice is found by the admin.
- User friendly product - Providing a hassle free UI with customizable filters for both buyers and sellers enabling them to buy/sell the assets in just a few simple steps.
- Scalability - Highly scalable and modular application where we can add a new entity / organization without affecting the existing architecture of the product.

Product Features

Our product handles the ability of a user to do a real estate transaction by using smart contract programs in Ethereum. Ethereum enables the smart contracts and applications built on its blockchain to run smoothly without fraud, downtime, control, or any third-party interference.

The main features of the product are the auction, listing and donation where a user can choose any of the three methods to transact a NFT which represents a property and can cancel before successful transaction in the case of auction and listing if the user wishes to.

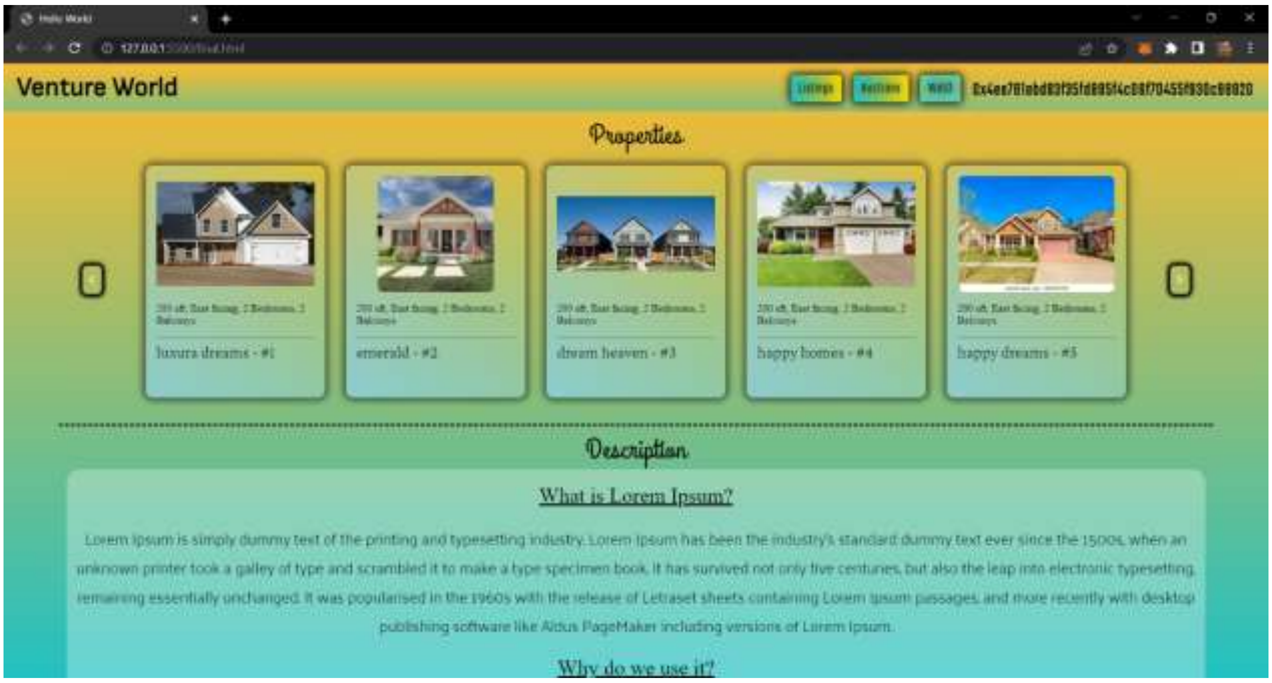
The user can view his/her currently owned NFTs which represent what properties they own and can also view all the live auctions and live listings that he/she has initiated. Also the user can at any time disable approval for the Real Estate smart contract to perform transactions with the NFTs on their behalf.

Writing Solidity

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

pragma solidity >=0.7.0 <0.9.0;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol";
import "@openzeppelin/contracts/utils/Strings.sol";

contract Real_Estate is ERC721Holder{

    address owner;
    address NFTaddress;

    struct Details{
        address currentOwner;
        uint state;
        uint acquire;
    }

    mapping(uint => Details) VentureDetails;
    uint[] allVentureDetails;

    struct Auction{
        uint state;
        uint startingValue;
        uint step;
        address highestBidder;
        uint highestBidderValue;
        mapping(address => uint) allBids;
        address[] allBidsAddresses;
    }

    struct Listing{
        uint state;
        uint listingValue;
        address buyer;
    }

    struct user{
        mapping(uint => Auction) allAuctions;
        uint[] allTokens;
        mapping(uint => Listing) allListings;
        uint[] allTokensListing;
        uint state;
    }

    mapping(address => user) allUsers;
    address[] allUsersAdresses;

    constructor(address nftAddress, uint[] memory tokens){
        owner = msg.sender;
        NFTaddress = nftAddress;
        for(uint i=0 ; i<tokens.length; i++){
            Details storage temp = VentureDetails[tokens[i]];
            temp.currentOwner = msg.sender;
            temp.state = 4;
            temp.acquire = 0;
            allVentureDetails.push(tokens[i]);
        }
    }

    function addProperties(uint[] memory tokens) public{
        require(msg.sender == owner, "Not Venture Owner");
        for(uint i=0 ; i<tokens.length; i++){
            Details storage temp = VentureDetails[tokens[i]];
            require(temp.state == 0, "Property Already Exists");
            temp.currentOwner = msg.sender;
            temp.state = 4;
            temp.acquire = 0;
            allVentureDetails.push(tokens[i]);
        }
    }

    function createAuction(uint tokenId, uint startValue, uint stepValue)public {
        require(VentureDetails[tokenId].state == 4, "The TokenID is
invalid");
        require(stepValue > startValue*5/100, "StepValue Must be greater
than 5 %");
        ERC721 token = ERC721(NFTaddress);
        require(token.ownerOf(tokenId) == msg.sender, "You must own the
NFT");
        require(token.isApprovedForAll(msg.sender, address(this)), "Approval
Not Valid");
        token.safeTransferFrom(msg.sender, address(this), tokenId);
        user storage temp = allUsers[msg.sender];
        Auction storage temp1 = temp.allAuctions[tokenId];
        require(temp1.state == 0 || temp1.state == 2 || temp1.state == 3,
"Auction is Live");
        if(temp.state == 0){
            temp.state = 1;
            allUsersAdresses.push(msg.sender);
        }
        if(temp1.state == 0){
            temp.allTokens.push(tokenId);
        }
        temp1.state = 1;
        temp1.startingValue = startValue;
        temp1.step = stepValue;
    }

    function updateAuction(uint tokenId, uint startValue, uint stepValue)
public{
        user storage temp = allUsers[msg.sender];
        Auction storage temp1 = temp.allAuctions[tokenId];
        require(temp1.state == 1, "Auction Invalid");
        require(stepValue > startValue*5/100, "StepValue Must be greater
than 5 %");
        temp1.startingValue = startValue;
        temp1.step = stepValue;
    }

    function cancelAuction(uint tokenId) public{
        user storage temp = allUsers[msg.sender];
        Auction storage temp1 = temp.allAuctions[tokenId];
        require(temp1.state == 1, "Auction Invalid");
        temp1.state = 3;
        ERC721(NFTaddress).safeTransferFrom(address(this), msg.sender,
tokenId);
        for(uint j=0; j<temp1.allBidsAddresses.length; j++){
            uint temp2 = temp1.allBids[temp1.allBidsAddresses[j]];
            temp1.allBids[temp1.allBidsAddresses[j]] = 0;
            payable(temp1.allBidsAddresses[j]).transfer(temp2);
        }
    }

    function assertAuction(uint tokenId, address buyer) public{
        require(allUsers[msg.sender].allAuctions[tokenId].state == 1,
"Auction Invalid");
        user storage temp = allUsers[msg.sender];
        Auction storage temp1 = temp.allAuctions[tokenId];
        uint tempval = temp1.allBids[buyer];
        temp1.allBids[buyer] = 0;
        ERC721(NFTaddress).safeTransferFrom(address(this), buyer, tokenId);
        payable(msg.sender).transfer(tempval);
        endAuction(tokenId);
        Details storage temp2 = VentureDetails[tokenId];
        temp2.currentOwner = buyer;
        temp2.acquire = 1;
    }

    function endAuction(uint tokenId) internal{
        user storage temp = allUsers[msg.sender];
        Auction storage temp1 = temp.allAuctions[tokenId];
        temp1.state = 2;
        for(uint i=0; i<temp1.allBidsAddresses.length; i++){
            if(temp1.allBids[temp1.allBidsAddresses[i]] == 0) continue;
            uint tempval = temp1.allBids[temp1.allBidsAddresses[i]];
            temp1.allBids[temp1.allBidsAddresses[i]] = 0;
            payable(temp1.allBidsAddresses[i]).transfer(tempval);
        }
    }

    function placeBid(address auc, uint tokenId) public payable{
        require(allUsers[auc].allAuctions[tokenId].state == 1 ,"Auction
Invalid");

require((allUsers[auc].allAuctions[tokenId].highestBidderValue)+(allUsers[a
uc].allAuctions[tokenId].step)  <=
allUsers[auc].allAuctions[tokenId].allBids[msg.sender]+msg.value, "The
Amount is less than the Highest Bid");
        user storage temp = allUsers[auc];
        Auction storage temp1 = temp.allAuctions[tokenId];
        bool chk = false;
        for(uint i=0; i<temp1.allBidsAddresses.length; i++){
            if(msg.sender == temp1.allBidsAddresses[i]){
                chk = true;
                break;
            }
        }
        if(chk == false){
            temp1.allBidsAddresses.push(msg.sender);
        }
        if(temp1.allBids[msg.sender] != 0)
payable(msg.sender).transfer(temp1.allBids[msg.sender]);
        temp1.allBids[msg.sender] = msg.value;
        temp1.highestBidder = msg.sender;
        temp1.highestBidderValue = temp1.allBids[msg.sender];
    }

    function cancelBid(address auc, uint tokenId) public{
        require(allUsers[auc].allAuctions[tokenId].state == 1, "Auction 
Invalid");
        user storage temp = allUsers[auc];
        Auction storage temp1 = temp.allAuctions[tokenId];
        uint tempval = temp1.allBids[msg.sender];
        temp1.allBids[msg.sender] = 0;
        payable(msg.sender).transfer(tempval);
        if(msg.sender == temp1.highestBidder){
            temp1.highestBidder = 
0x0000000000000000000000000000000000000000;
            temp1.highestBidderValue = 0;
            address x = temp1.allBidsAddresses[0];
            uint y = temp1.allBids[temp1.allBidsAddresses[0]];
            for(uint i=0; i<temp1.allBidsAddresses.length; i++){
                if(temp1.allBids[temp1.allBidsAddresses[i]] > y){
                    y = temp1.allBids[temp1.allBidsAddresses[i]];
                    x = temp1.allBidsAddresses[i];
                }
            }
            if(x != temp1.highestBidder){
                temp1.highestBidder = x;
                temp1.highestBidderValue = y;
            }
        }
    }

    function donateProperty(address reciever, uint tokenId) public{
        require(VentureDetails[tokenId].state == 4, "Invalid Token ID");
        ERC721 token = ERC721(NFTaddress);
        require(token.ownerOf(tokenId) == msg.sender, "You must own the 
NFT");
        require(token.isApprovedForAll(msg.sender, address(this)), "Approval 
Not Valid");
        token.safeTransferFrom(msg.sender, reciever, tokenId);
        Details storage temp = VentureDetails[tokenId];
        temp.currentOwner = reciever;
        temp.acquire = 3;
    }

    function placeListing(uint tokenId, uint listVal) public{
        require(VentureDetails[tokenId].state == 4, "Invalid Token ID");
        user storage temp = allUsers[msg.sender];
        Listing storage temp1 = temp.allListings[tokenId];
        require(temp1.state == 0 || temp1.state == 2 || temp1.state == 3, 
"Listing is Active");
        ERC721 token = ERC721(NFTaddress);
        require(token.ownerOf(tokenId) == msg.sender, "You must own the 
NFT");
        require(token.isApprovedForAll(owner, address(this)), "Approval Not 
Valid");
        token.safeTransferFrom(msg.sender, address(this), tokenId);
        if(temp.state == 0){
            allUsersAdresses.push(msg.sender);
            temp.state = 1;
        }
        if(temp1.state == 0){
            temp.allTokensListing.push(tokenId);
        }
        temp1.state = 1;
        temp1.listingValue = listVal;
    }
    function buyListing(address auc, uint tokenId) public payable{
        require(allUsers[auc].allListings[tokenId].state == 1, "Listing
Invalid");
        require(allUsers[auc].allListings[tokenId].listingValue ==
msg.value, "Place Match the Listing Price");
        user storage temp = allUsers[auc];
        Listing storage temp1 = temp.allListings[tokenId];
        temp1.state = 2;
        temp1.buyer = msg.sender;
        ERC721(NFTaddress).safeTransferFrom(address(this), msg.sender,
tokenId);
        payable(auc).transfer(msg.value);
        Details storage temp2 = VentureDetails[tokenId];
        temp2.currentOwner = msg.sender;
        temp2.acquire = 2;
    }

    function updateListing(uint tokenId, uint listVal) public{
        require(allUsers[msg.sender].allListings[tokenId].state == 1,
"Lisiting Invalid");
        user storage temp = allUsers[msg.sender];
        Listing storage temp1 = temp.allListings[tokenId];
        temp1.listingValue = listVal;
    }

    function cancelListing(uint tokenId) public{
        require(allUsers[msg.sender].allListings[tokenId].state == 1,
"Lisiting Invalid");
        user storage temp = allUsers[msg.sender];
        Listing storage temp1 = temp.allListings[tokenId];
        temp1.state = 3;
        ERC721(NFTaddress).safeTransferFrom(address(this), msg.sender,
tokenId);
    }

    function allAuctions() public view returns(string memory res){
        user storage temp = allUsers[msg.sender];
        if(temp.allTokens.length == 0) return "";
        res = string.concat("{", '"',"User",'"', ":",
'"',Strings.toHexString(uint160(msg.sender), 20),'"', ",",
'"',"Auctions",'"', ":{");
        for(uint i=0; i<temp.allTokens.length; i++){
            Auction storage temp1 = temp.allAuctions[temp.allTokens[i]];
            res = string.concat(res,
'"',Strings.toString(temp.allTokens[i]),'"', ":{",'"',"State",'"',":",'"',
Strings.toString(temp1.state),'"',
",",'"',"StartingValue",'"',":",'"',Strings.toString(temp1.startingValue),'
"',",",'"',"HighestBidder",'"',":",'"',Strings.toHexString(uint160(temp1.hi
ghestBidder), 20),'"', ",", '"',"HighestBidderValue",'"', ":",
'"',Strings.toString(temp1.highestBidderValue),'"', "}");
            if(i+1 != temp.allTokens.length){
                res = string.concat(res, ",");
            }
        }
        res = string.concat(res, " } }");
        return res;
    }

    function allListings() public view returns(string memory res){
        user storage temp = allUsers[msg.sender];
        if(temp.allTokensListing.length == 0) return "";
        res = string.concat("{", '"',"User",'"', ":",
'"',Strings.toHexString(uint160(msg.sender), 20),'"', ",",
'"',"Listings",'"', ":{");
        for(uint i=0; i<temp.allTokensListing.length; i++){
            Listing storage temp1 =
temp.allListings[temp.allTokensListing[i]];
            res = string.concat(res,
'"',Strings.toString(temp.allTokensListing[i]),'"', ":{",
'"',"State",'"',":",'"',Strings.toString(temp1.state),'"',",",'"',"ListingV
alue",'"',":",'"',Strings.toString(temp1.listingValue),'"', "}");
            if(i+1 != temp.allTokensListing.length){
                res = string.concat(res, ",");
            }
        }
        res = string.concat(res, " } }");
        return res;
    }

    function viewProperties() public view returns(string memory res){
        res = string.concat(res, "{");//string.concat("{ "));
        if(allVentureDetails.length == 0) return "";
        for(uint i=0; i<allVentureDetails.length; i++){
            Details memory temp = VentureDetails[allVentureDetails[i]];
            res = string.concat(res,
'"',Strings.toString(allVentureDetails[i]), '"', ":{", '"', "CurrentOwner",
'"', ":", '"', Strings.toHexString(uint160(temp.currentOwner), 20),'", "',
"AcquiredState", '"',":", '"',Strings.toString(temp.acquire),'"', "}");
            if(i+1 != allVentureDetails.length){
                res = string.concat(res, ",");
            }
        }
        res = string.concat(res, "}");
        return res;
    }

    function viewWhole() public view returns(string memory res){
        if(allUsersAdresses.length == 0) return "";
        res = string.concat("{",'"',"Auctions",'"', ":{");
        for(uint i=0; i<allUsersAdresses.length; i++){
            user storage temp = allUsers[allUsersAdresses[i]];
            if(temp.allTokens.length == 0) return "";
            res = string.concat(res, '"',
Strings.toHexString(uint160(allUsersAdresses[i]), 20),'"', ": {");
            for(uint j=0; j<temp.allTokens.length; j++){
                Auction storage tempA =
temp.allAuctions[temp.allTokens[j]];
                res = string.concat(res,
'"',Strings.toString(temp.allTokens[j]),'"', ":{",'"',"State",'"', ":",
'"',Strings.toString(tempA.state),'"', ",",'"',"StartingValue",'"', ":",
'"',Strings.toString(tempA.startingValue),'"', ",",'"',"Step",'"', ":",
'"',Strings.toString(tempA.step),'"', ",",'"',"HighestBidderAddress",'"',
":", '"',Strings.toHexString(uint160(tempA.highestBidder), 20),'"',
",",'"',"HighestBidderValue",'"', ":",
'"',Strings.toString(tempA.highestBidderValue),'"', "}");
                if(j+1 != temp.allTokens.length){
                    res = string.concat(res, ",");
                }
            }
            res = string.concat(res, "}");
            if(i+1 != allUsersAdresses.length){
                res = string.concat(res, ",");
            }
        }
        res = string.concat(res, "},",'"',"Listings",'"', ":{");
        for(uint i=0; i<allUsersAdresses.length; i++){
            user storage temp = allUsers[allUsersAdresses[i]];
            res = string.concat(res,
'"',Strings.toHexString(uint160(allUsersAdresses[i]), 20),'"', ": {");
            for(uint j=0; j<temp.allTokensListing.length; j++){
                Listing storage tempA =
temp.allListings[temp.allTokensListing[j]];
                res = string.concat(res,
'"',Strings.toString(temp.allTokensListing[j]),'"', ":{",'"',"State",'"',
":", '"',Strings.toString(tempA.state),'"', ",",'"',"ListingValue",'"',
":", '"',Strings.toString(tempA.listingValue),'"', "}");
                if(j+1 != temp.allTokensListing.length){
                    res = string.concat(res, ",");
                }
            }
            res = string.concat(res, "}");
            if(i+1 != allUsersAdresses.length){
                res = string.concat(res, ",");
            }
        }
        res = string.concat(res, "} }");
        return res;
    }
}
```

Implementation

Working Application Demo



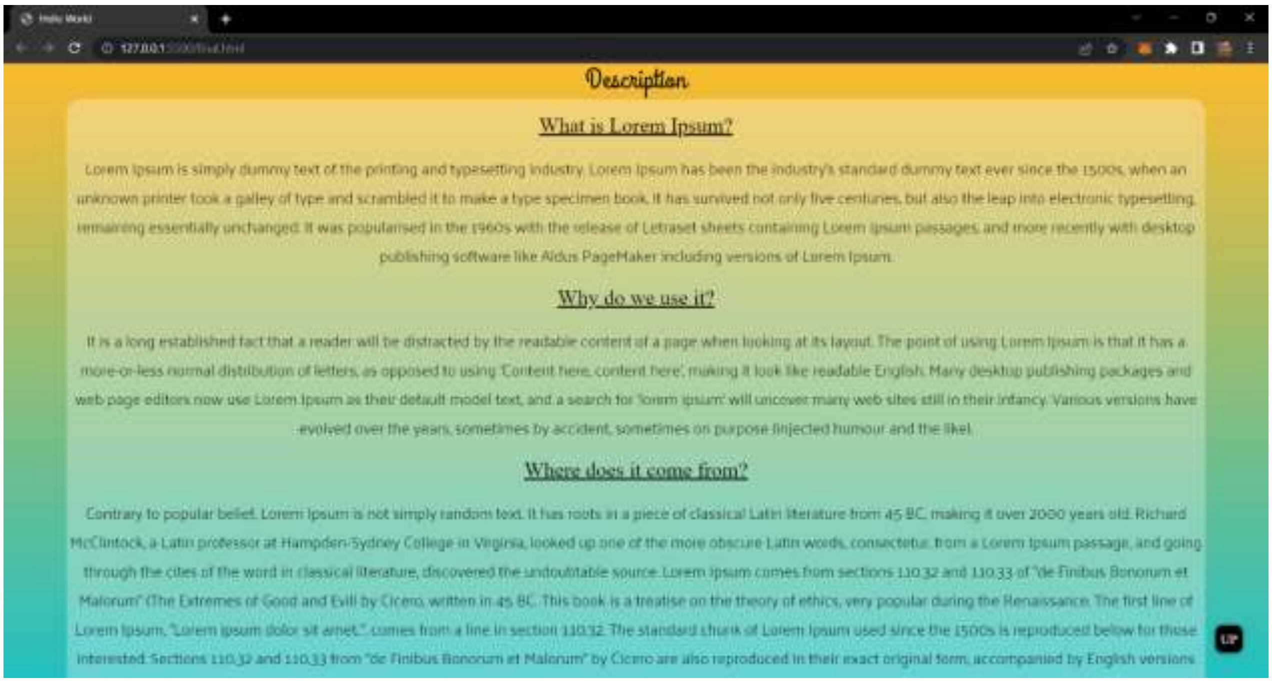
Description

What is Lorem Ipsum?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Why do we use it?

Home Page



Description

What is Lorem Ipsum?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

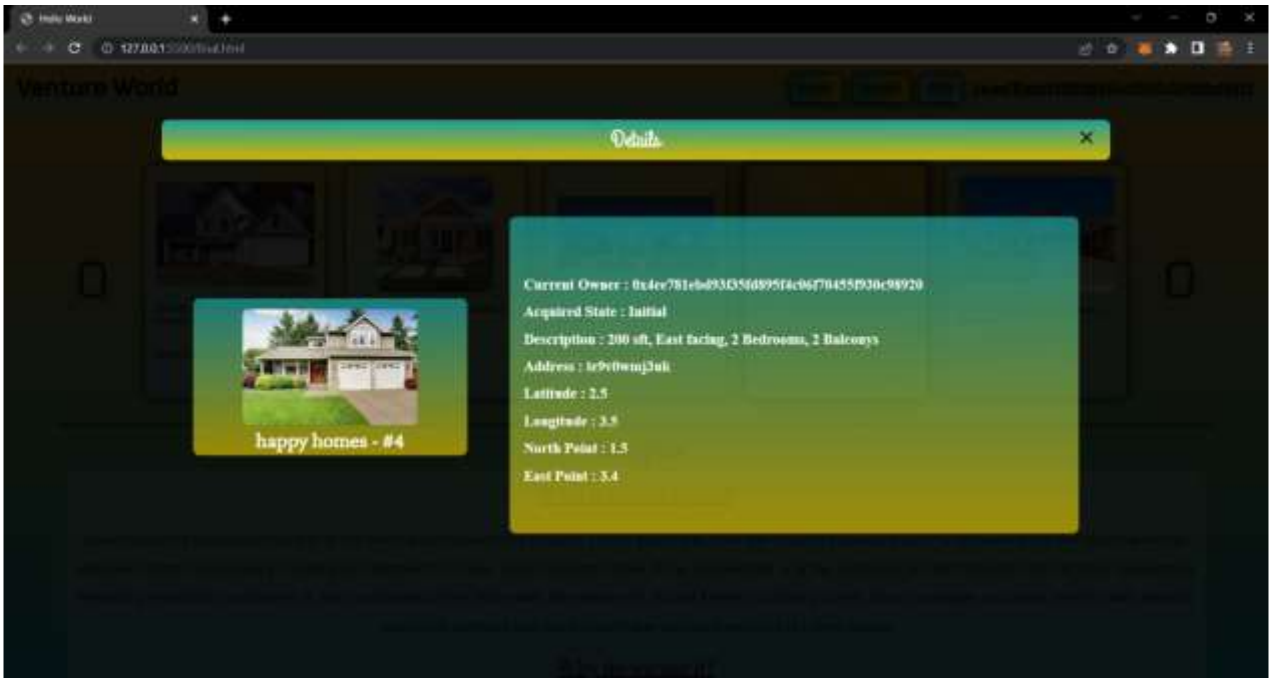
Why do we use it?

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Where does it come from?

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32. The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions

Description



Current Owner : 0x4ee781ebd93f35fd895f4c06f70455f930c98920

Acquired State : Initial

Description : 200 sft, East facing, 2 Bedrooms, 2 Balconys

Address : ie9vt0wmj3uk

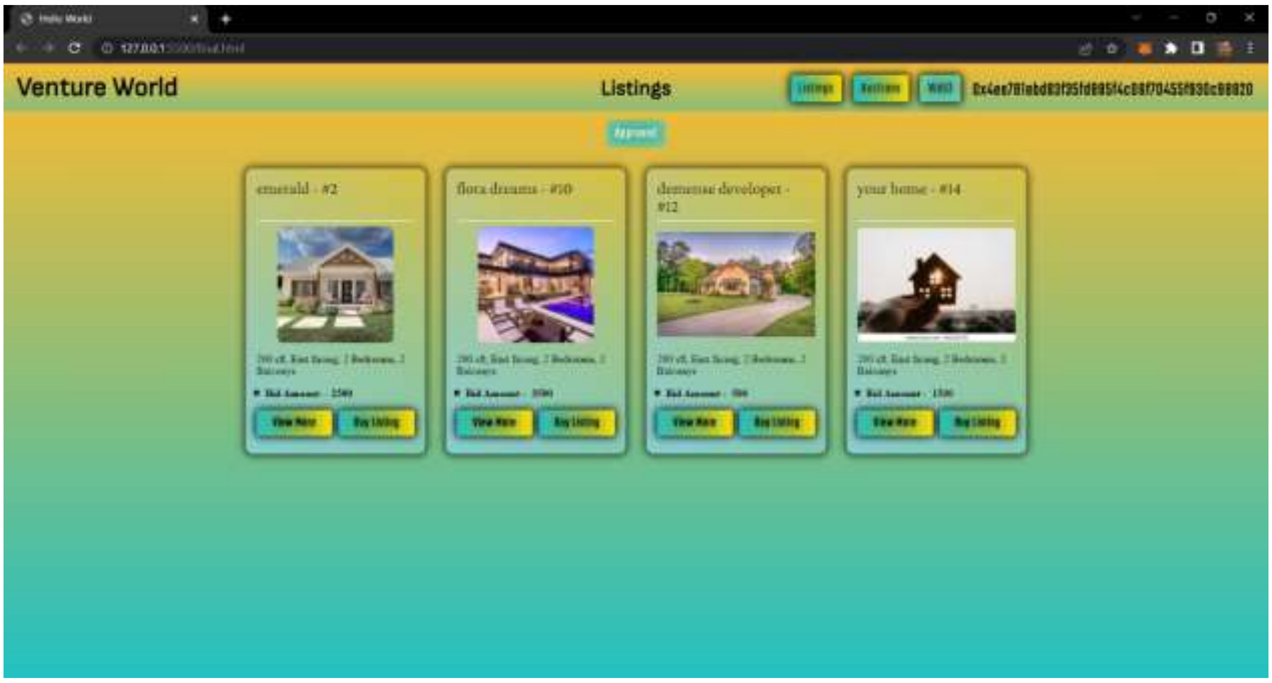
Latitude : 2.5

Longitude : 3.5

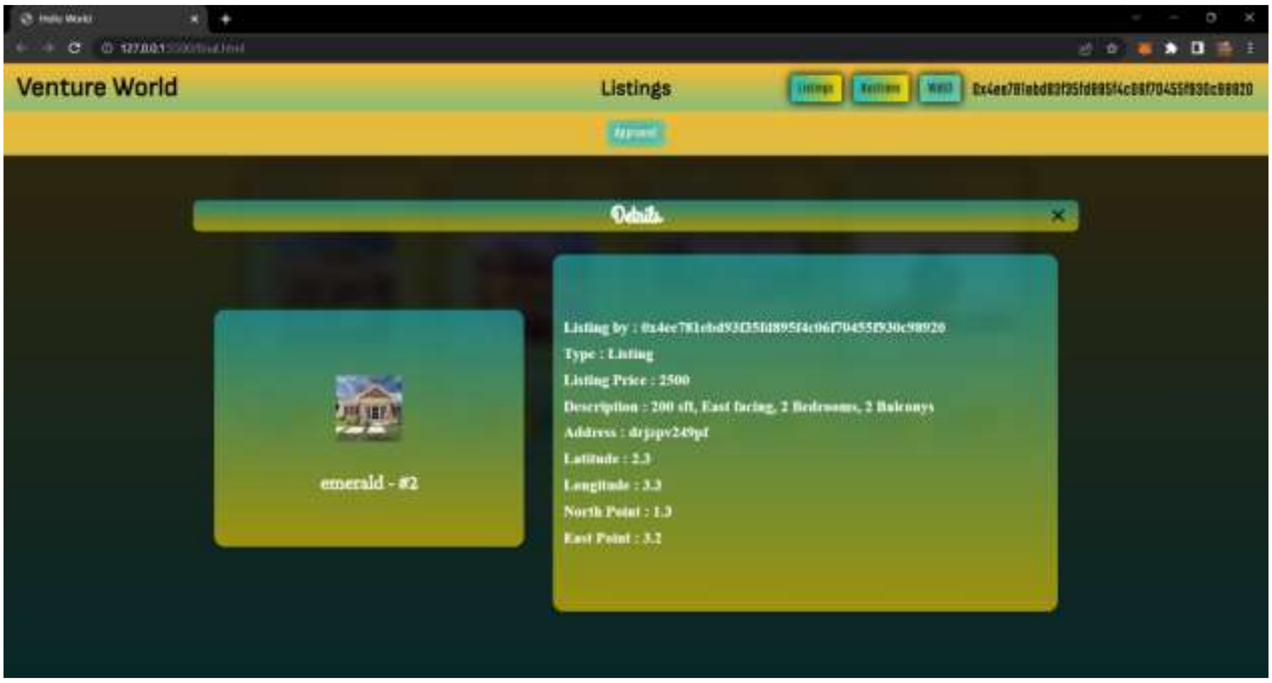
North Point : 1.5

East Point : 3.4

View Details(Home)

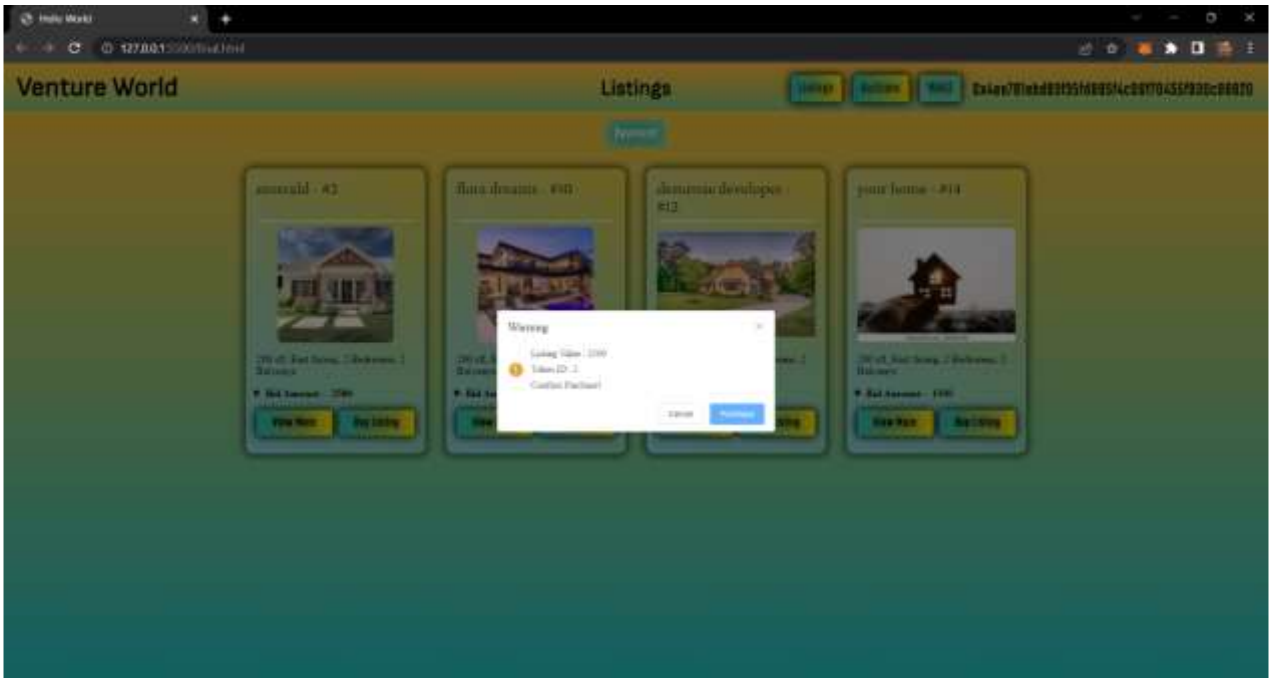


All Live Listings



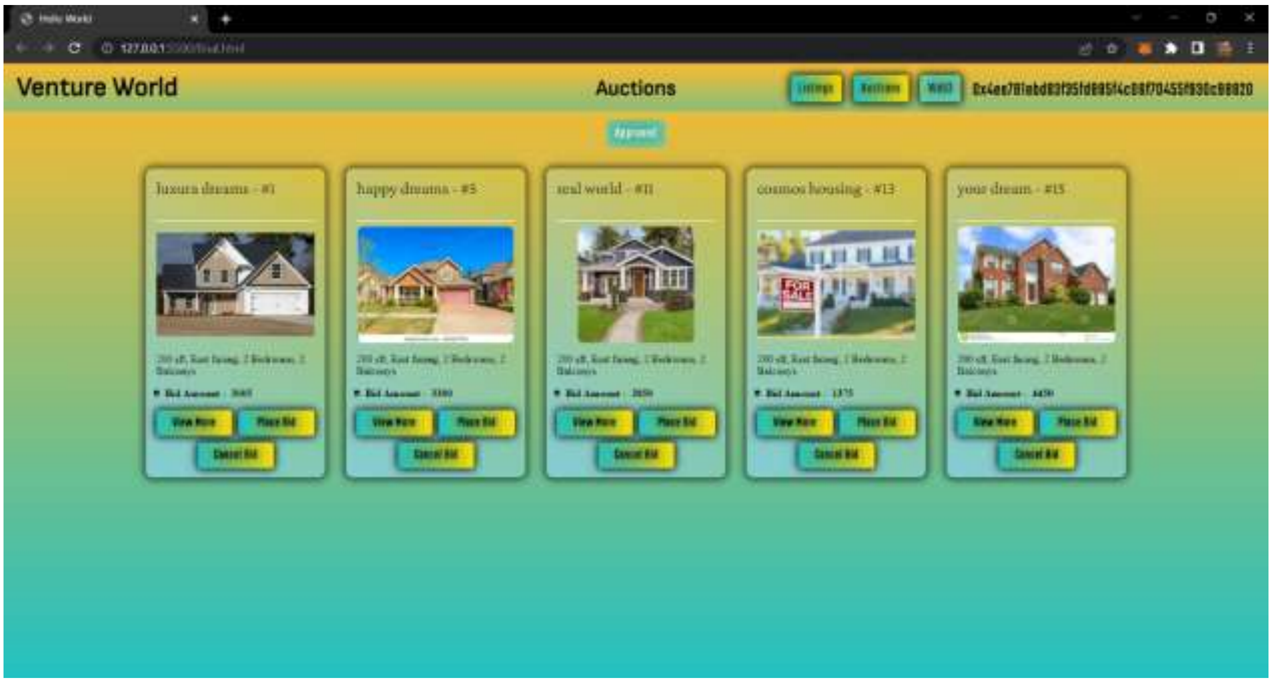
Listing by : 0x4ee781ebd93f35fd895f4c06f70455f930c98920
Type : Listing
Listing Price : 2500
Description : 200 sft, East facing, 2 Bedrooms, 2 Balconys
Address : drjapv249pf
Latitude : 2.3
Longitude : 3.3
North Point : 1.3
East Point : 3.2

View Details(Listings)

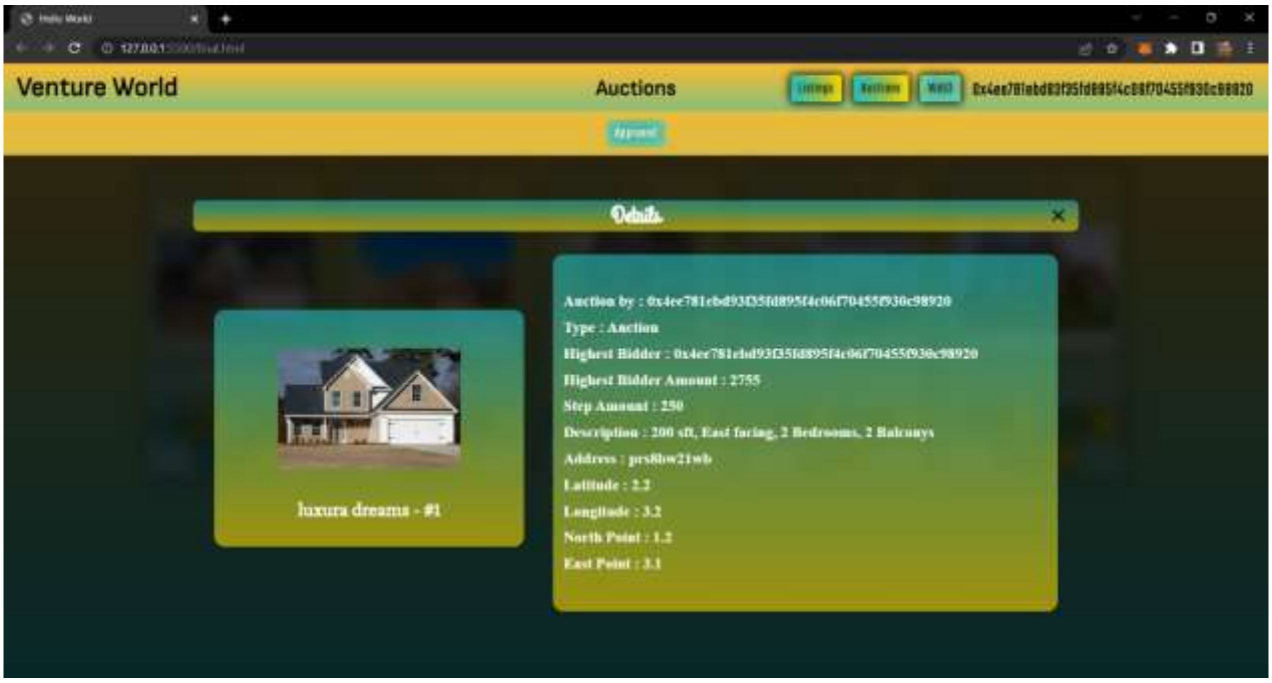


Warning
Listing Value : 2500
Token ID : 2
Confirm Purchase!
Cancel
Purchase

Buy Listing

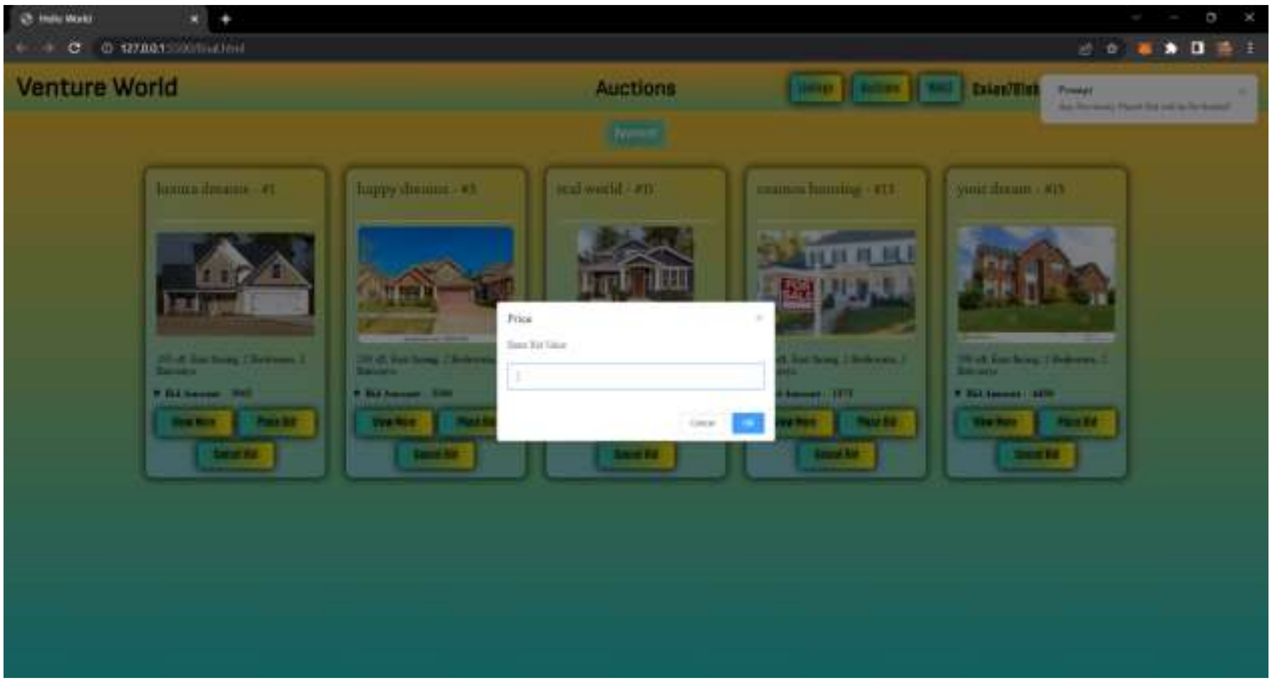


All Live Auctions

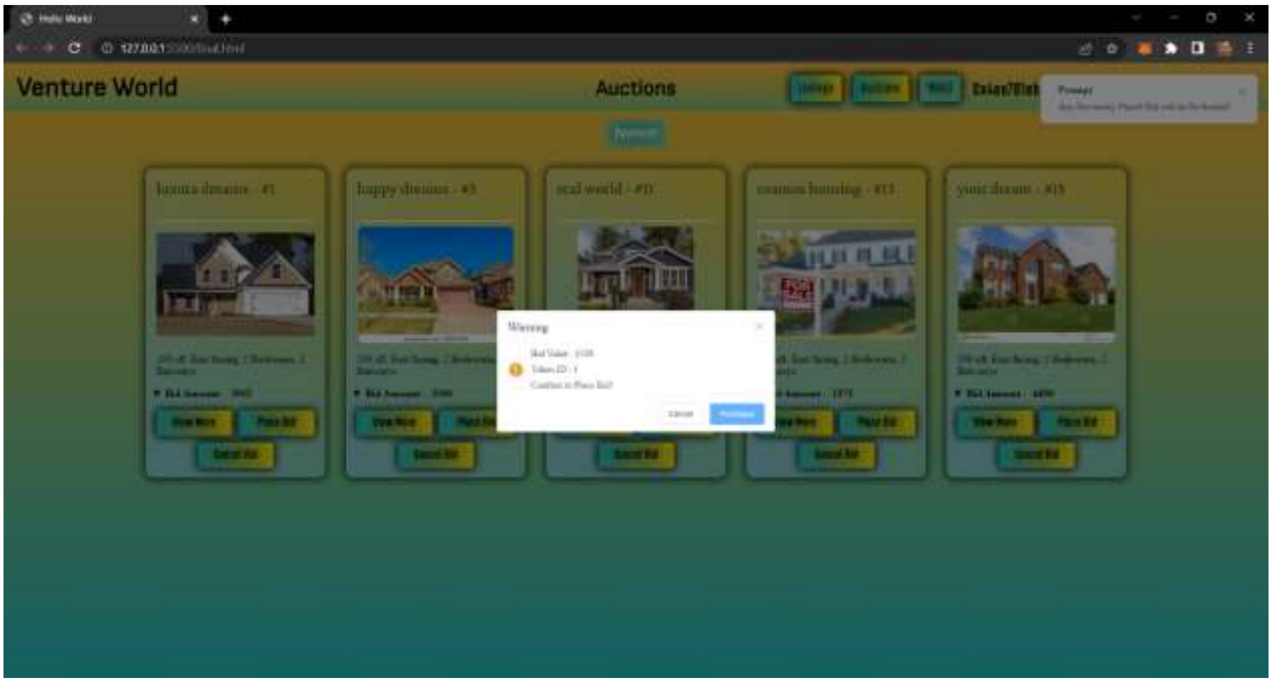


Auction by : 0x4ee781ebd93f35fd895f4c06f70455f930c98920
Type : Auction
Highest Bidder : 0x4ee781ebd93f35fd895f4c06f70455f930c98920
Highest Bidder Amount : 2755
Step Amount : 250
Description : 200 sft, East facing, 2 Bedrooms, 2 Balcunys
Address : prs8bw21wb
Latitude : 2.2
Longitude : 3.2
North Point : 1.2
East Point : 3.1

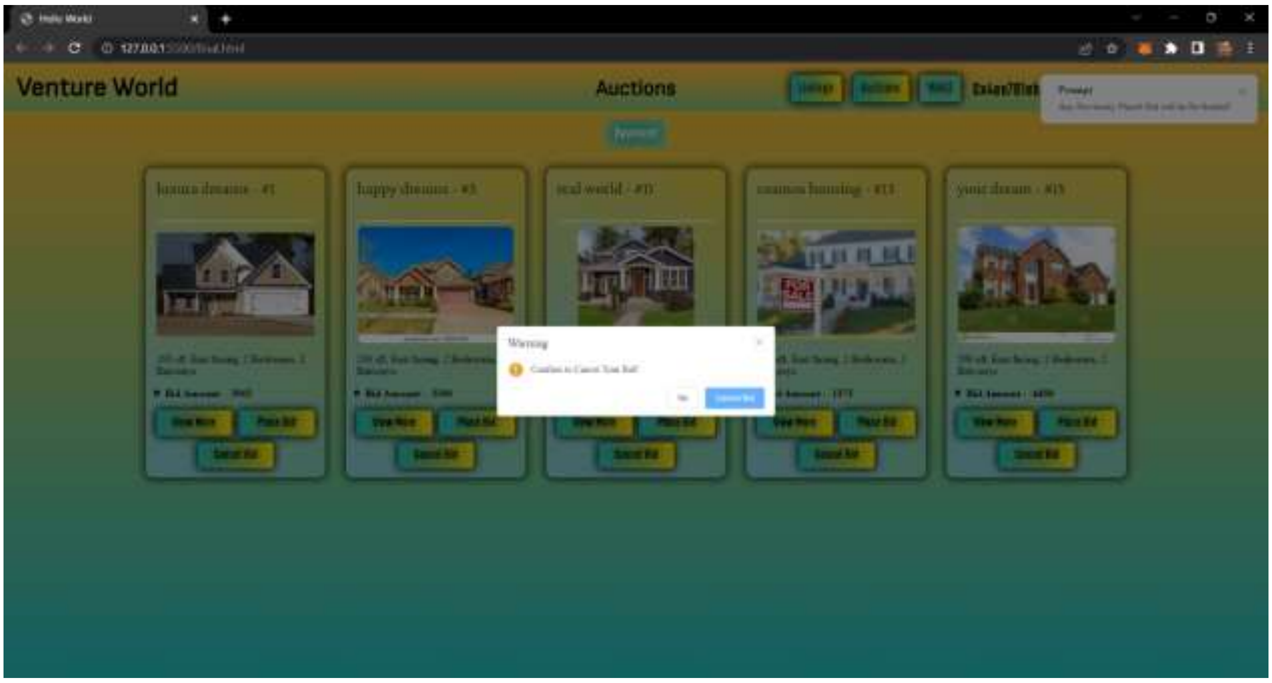
View Details(Auctions)



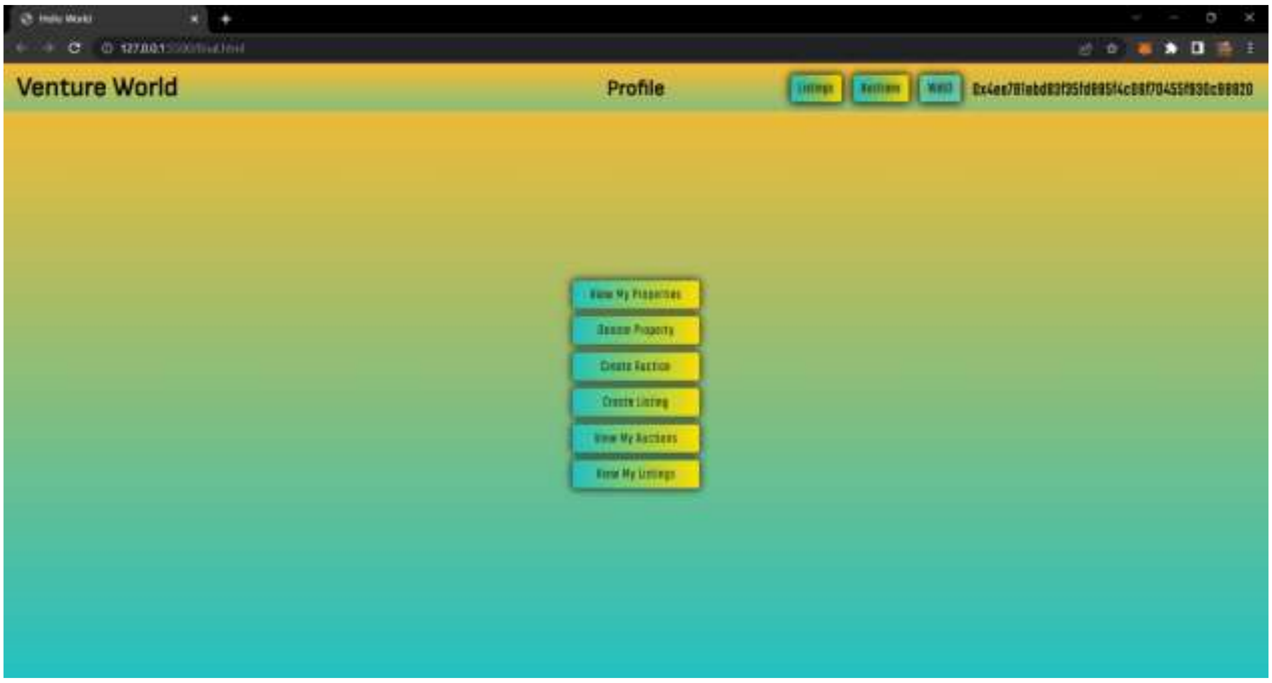
Place Bid Amount



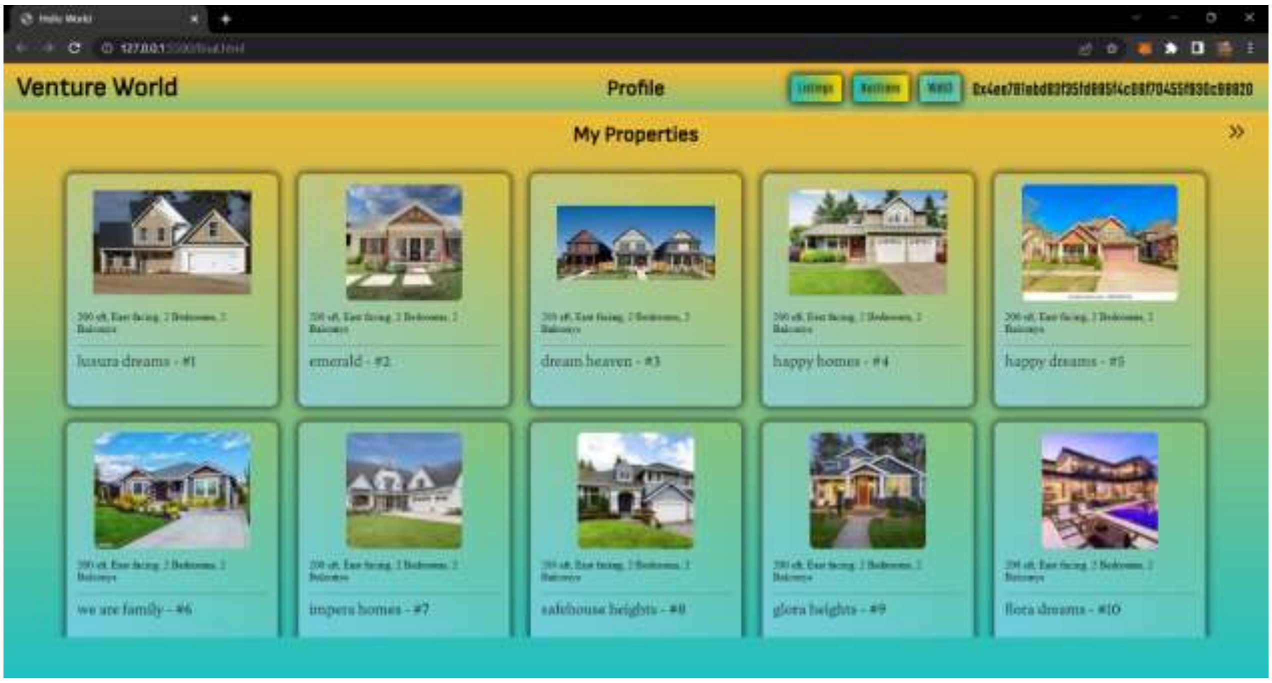
Confirm Bid



Cancel Bid



Profile



200 sft, East facing, 2 Bedrooms, 2 Balconys
kusura dreams - #1

200 sft, East facing, 2 Bedrooms, 2 Balconys
emerald - #2

200 sft, East facing, 2 Bedrooms, 2 Balconys
dream heaven - #3

200 sft, East facing, 2 Bedrooms, 2 Balconys
happy homes - #4

200 sft, East facing, 2 Bedrooms, 2 Balconys
happy dreams - #5

200 sft, East facing, 2 Bedrooms, 2 Balconys
we are family - #6

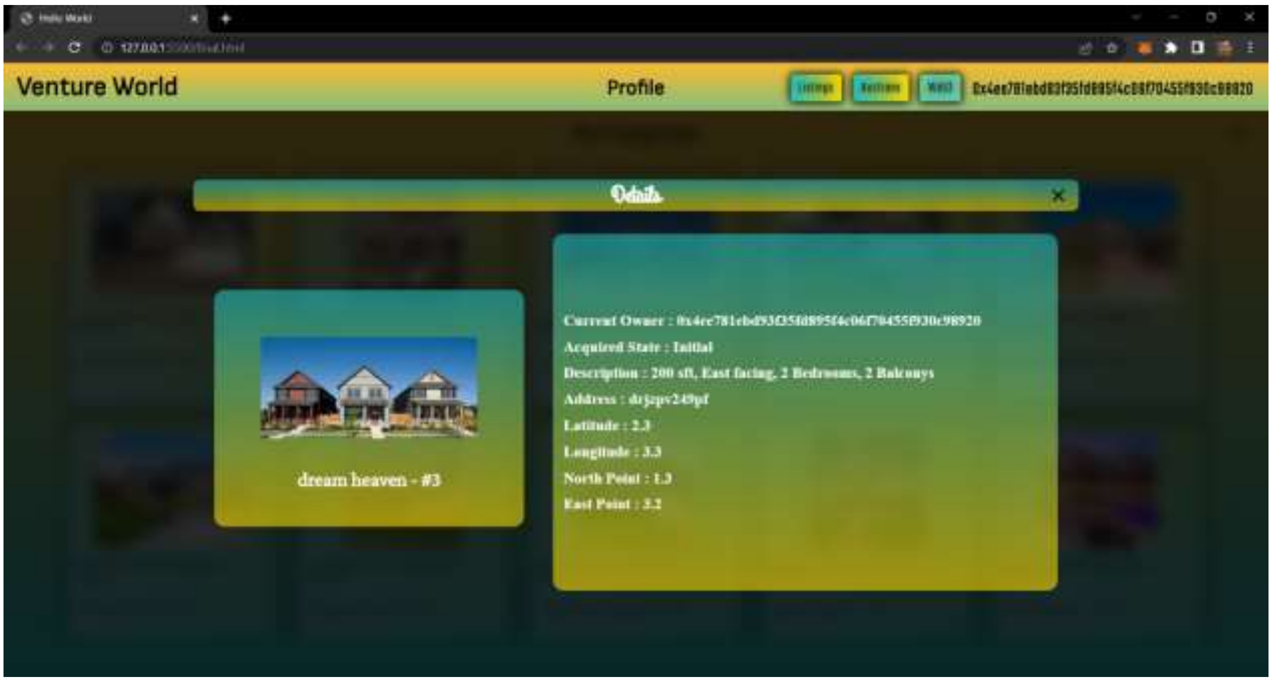
200 sft, East facing, 2 Bedrooms, 2 Balconys
impera homes - #7

200 sft, East facing, 2 Bedrooms, 2 Balconys
safehouse heights - #8

200 sft, East facing, 2 Bedrooms, 2 Balconys
glora heights - #9

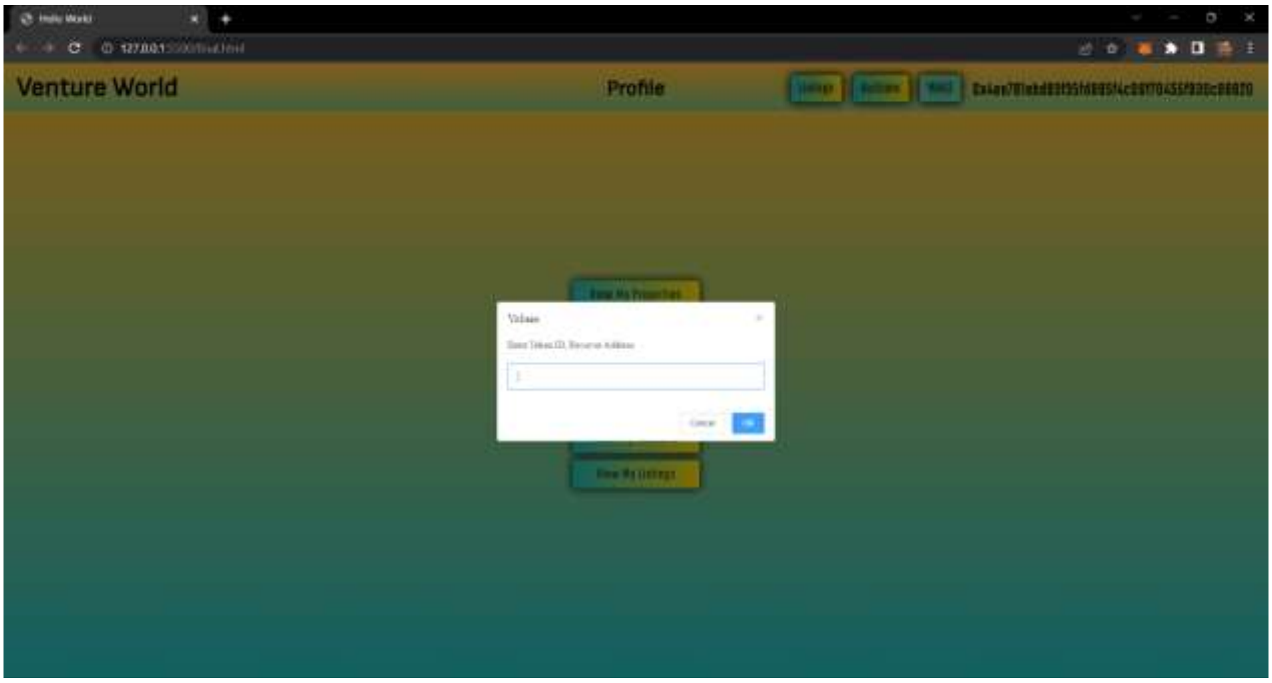
200 sft, East facing, 2 Bedrooms, 2 Balconys
flora dreams - #10

User Owned Properties

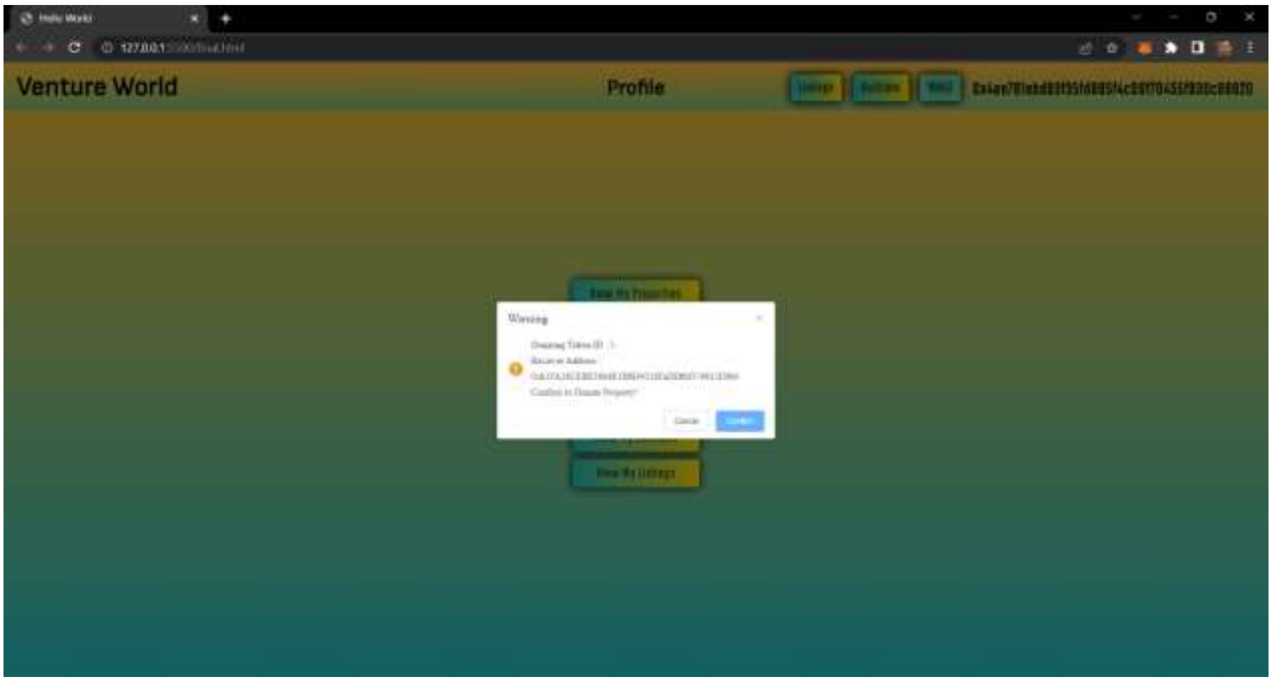


Current Owner : 0x4ee781ebd93f35fd895f4c06f70455f930c98920
Acquired State : Initial
Description : 200 sft, East facing, 2 Bedrooms, 2 Balconys
Address : drjzpv249pf
Latitude : 2.3
Longitude : 3.3
North Point : 1.3
East Point : 3.2

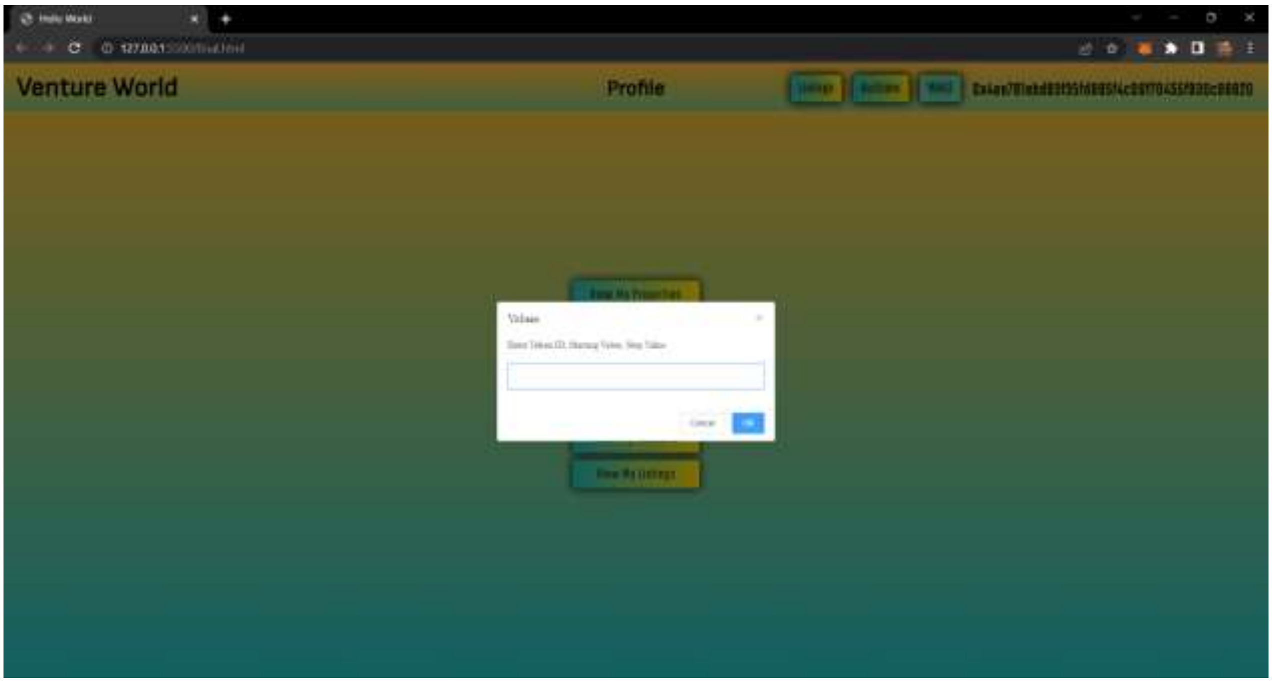
View Details(User Owned Properties)



Donate Property Details



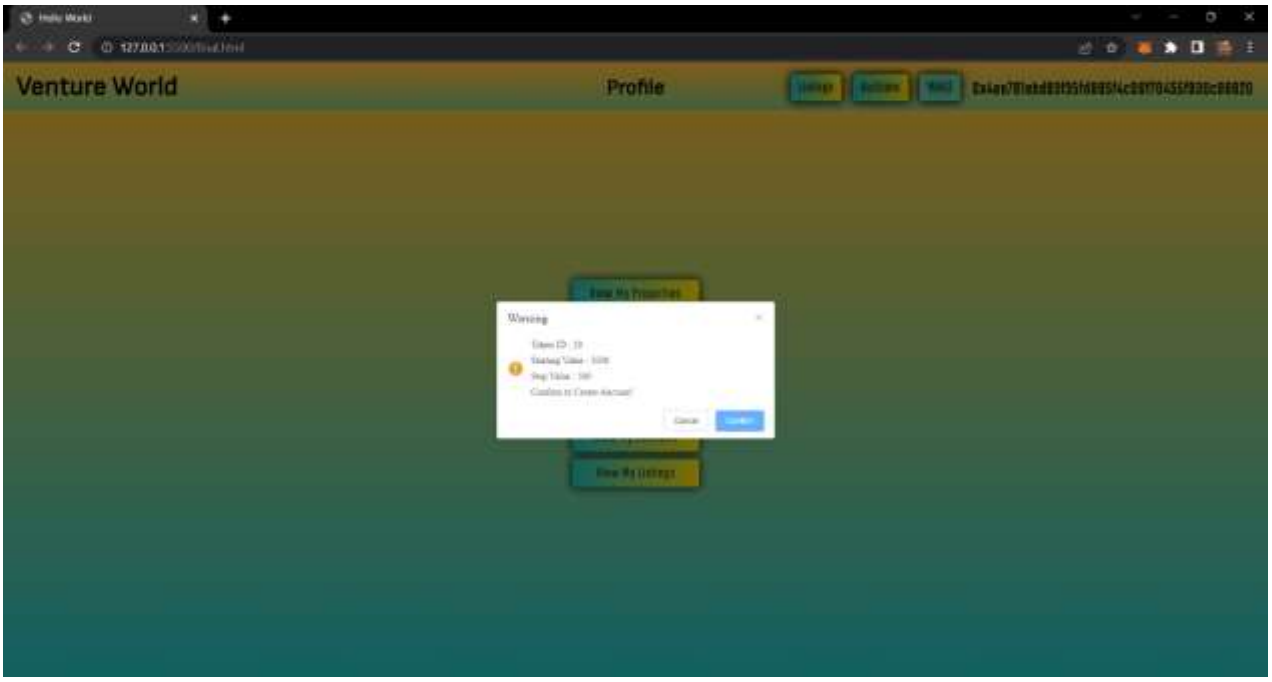
Confirm Donate Property



Values

Enter Token ID, Starting Value, Step Value

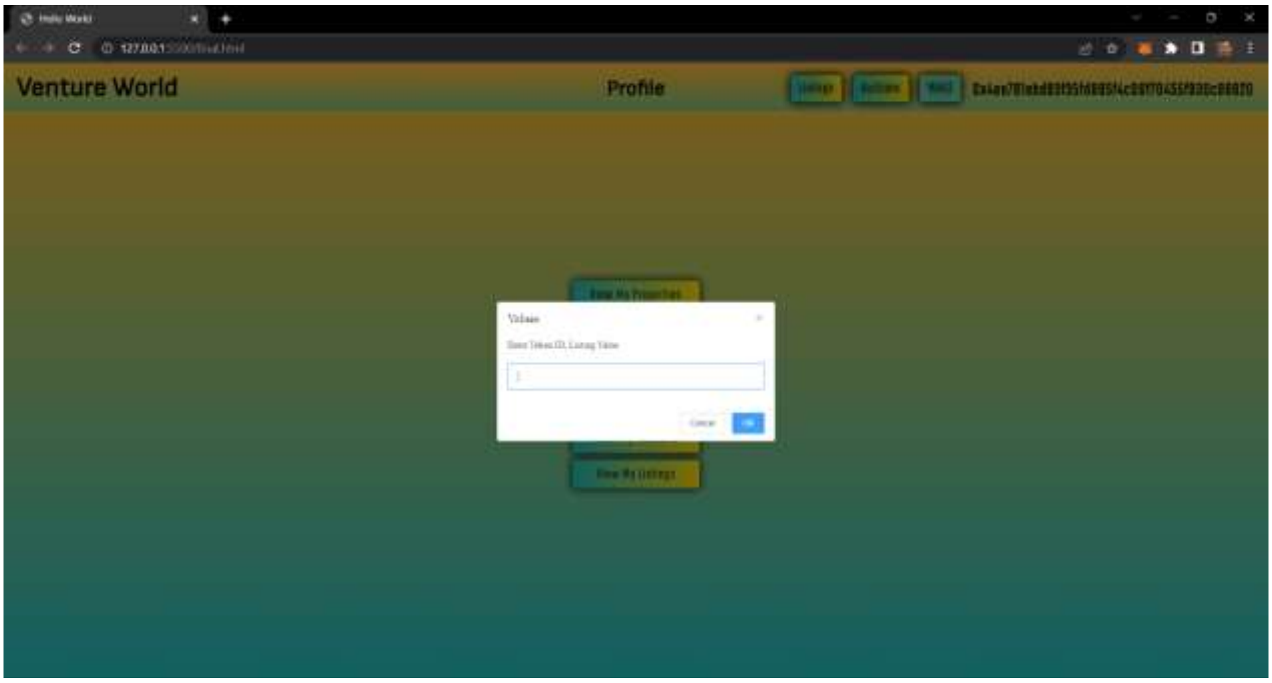
Create Auction Details



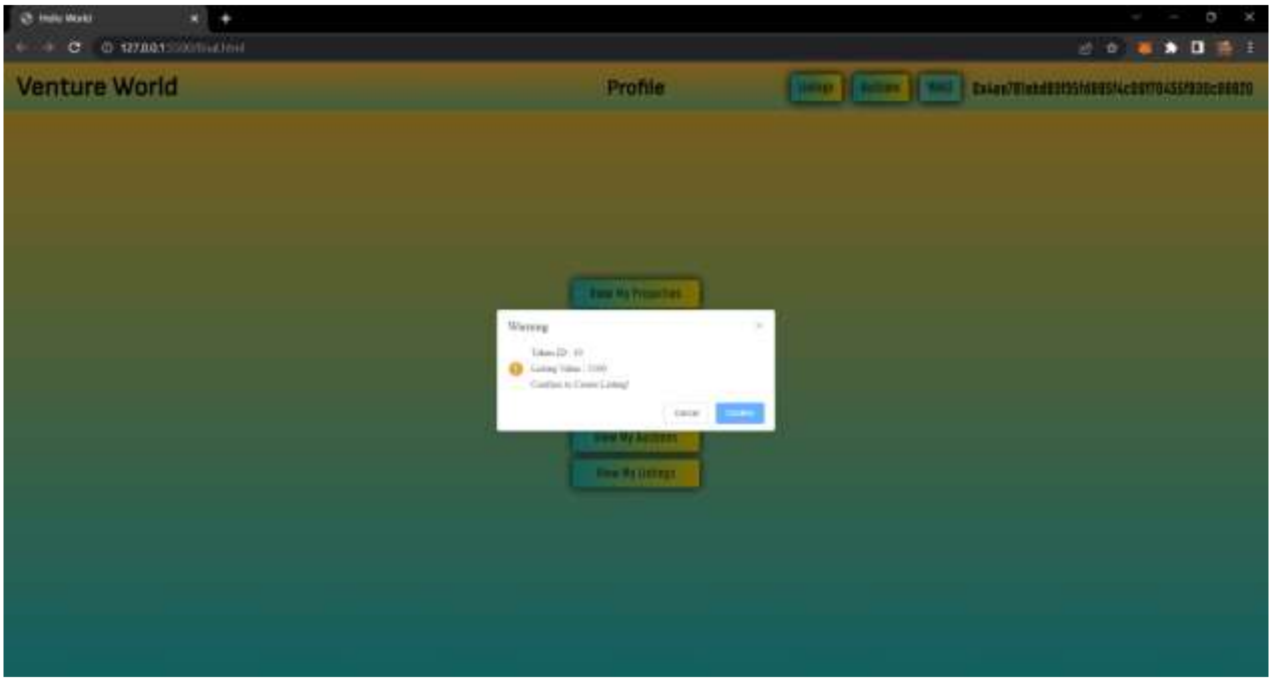
Warning

Token ID : 13
Starting Value : 1000
Step Value : 100
Confirm to Create Auction?

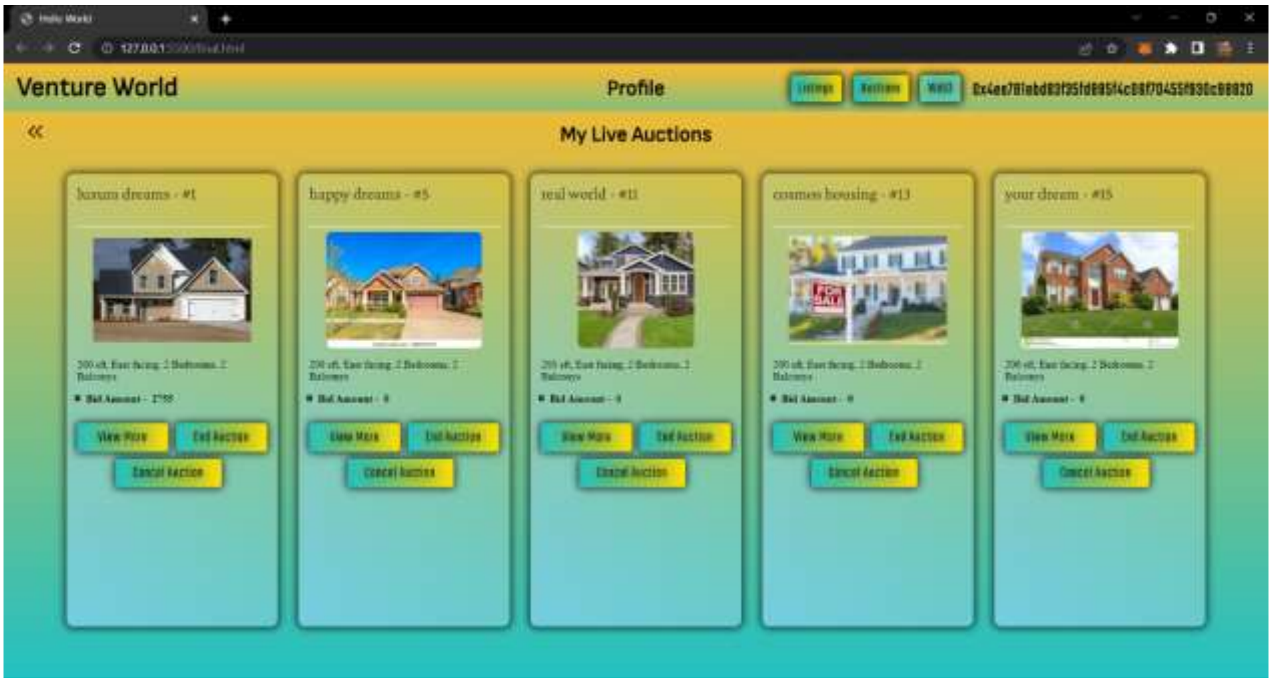
Confirm Create Auction



Create Listing Details



Confirm Create Listing



luxum dreams - #1
200 sft, East facing, 2 Bedrooms, 2 Balconys
- Bid Amount - 2755

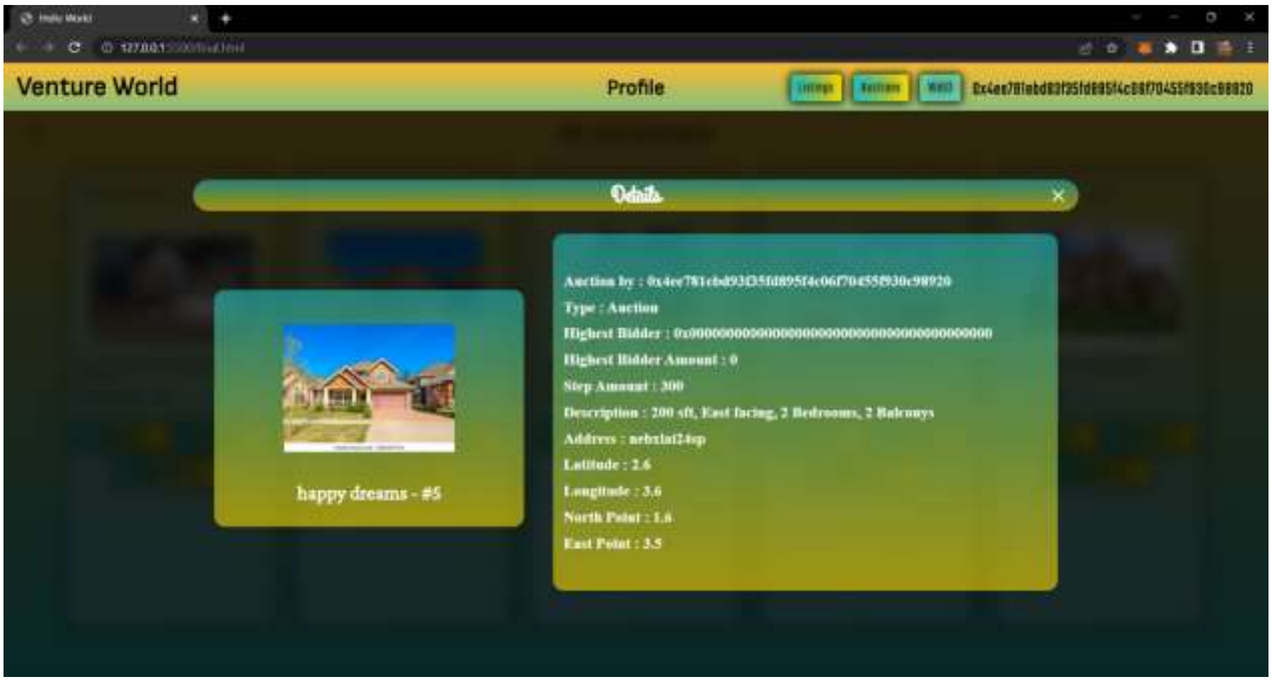
happy dreams - #5
200 sft, East facing, 2 Bedrooms, 2 Balconys
- Bid Amount - 0

real world - #11
200 sft, East facing, 2 Bedrooms, 2 Balconys
- Bid Amount - 0

cosmos housing - #13
200 sft, East facing, 2 Bedrooms, 2 Balconys
- Bid Amount - 0

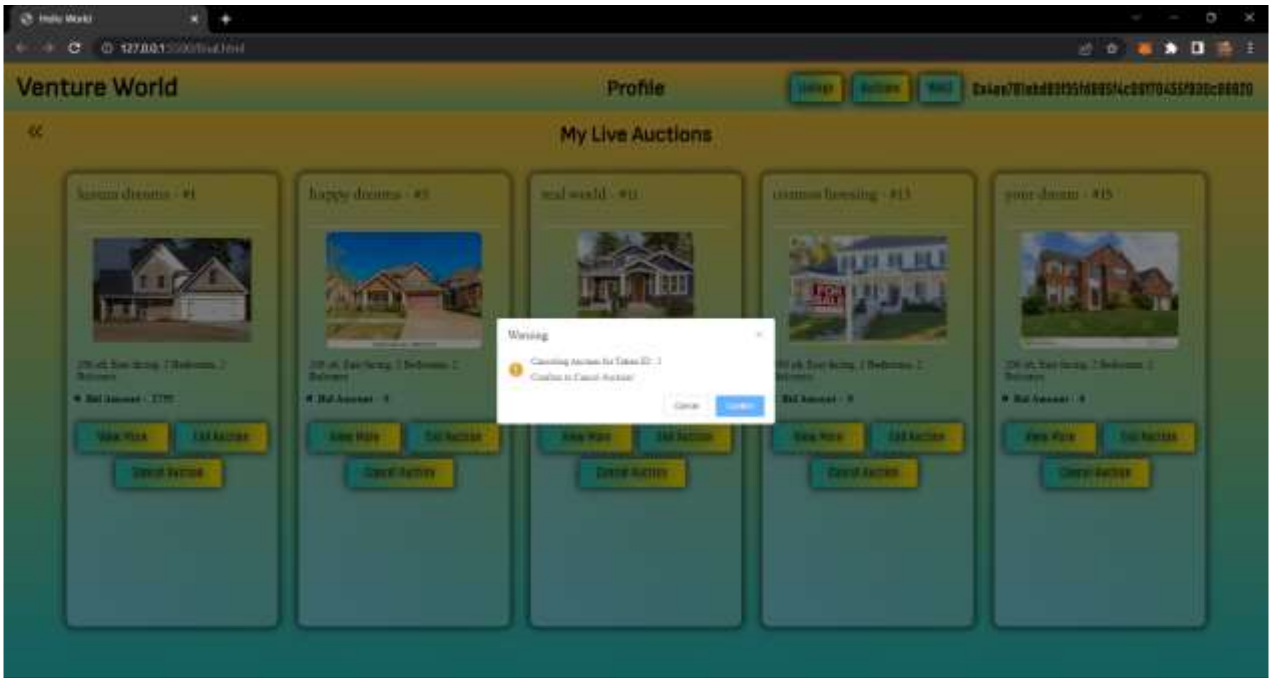
your dream - #15
200 sft, East facing, 2 Bedrooms, 2 Balconys
- Bid Amount - 0

User Created Live Auctions



Auction by : 0x4ee781ebd93f35fd895f4c06f70455f930c98920
Type : Auction
Highest Bidder : 0x0000000000000000000000000000000000000000
Highest Bidder Amount : 0
Step Amount : 300
Description : 200 sft, East facing, 2 Bedrooms, 2 Balconys
Address : nebxlal24sp
Latitude : 2.6
Longitude : 3.6
North Point : 1.6
East Point : 3.5

View Details(User Created Live Auctions)



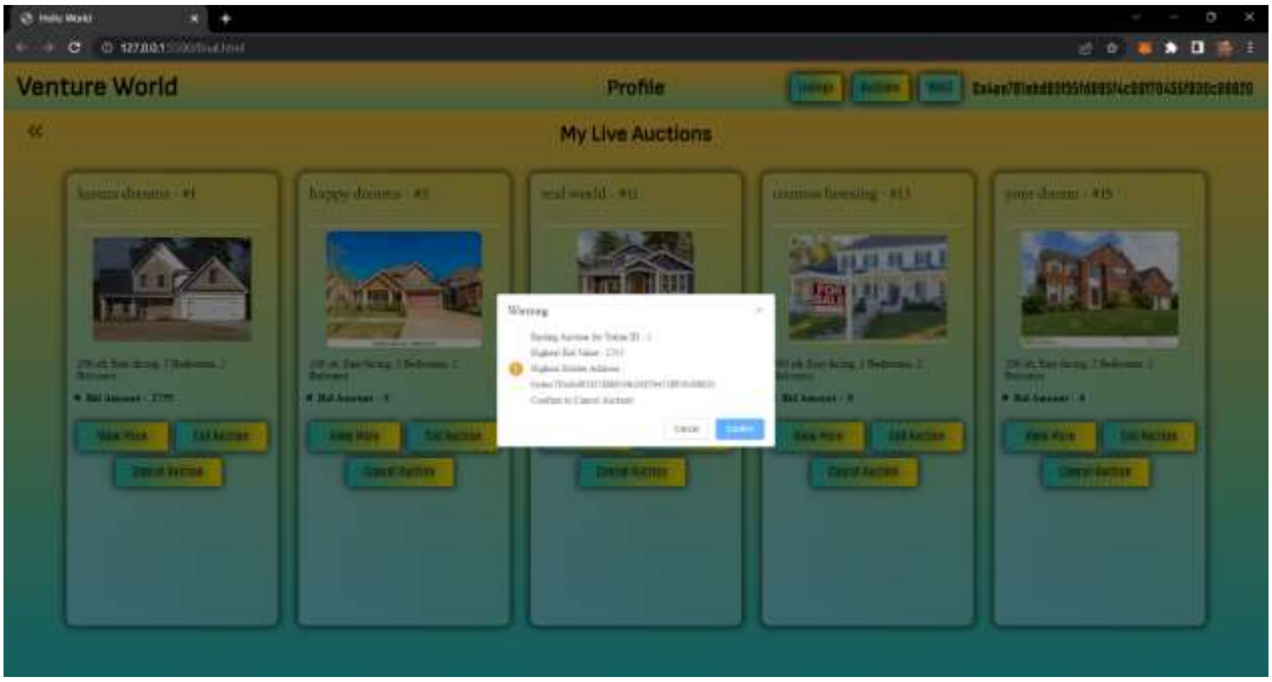
Warning

Canceling Auction for Token ID : 1

Confirm to Cancel Auction!

Cancel Confirm

Cancel Auction



Warning

Ending Auction for Token ID : 1

Highest Bid Value : 2711

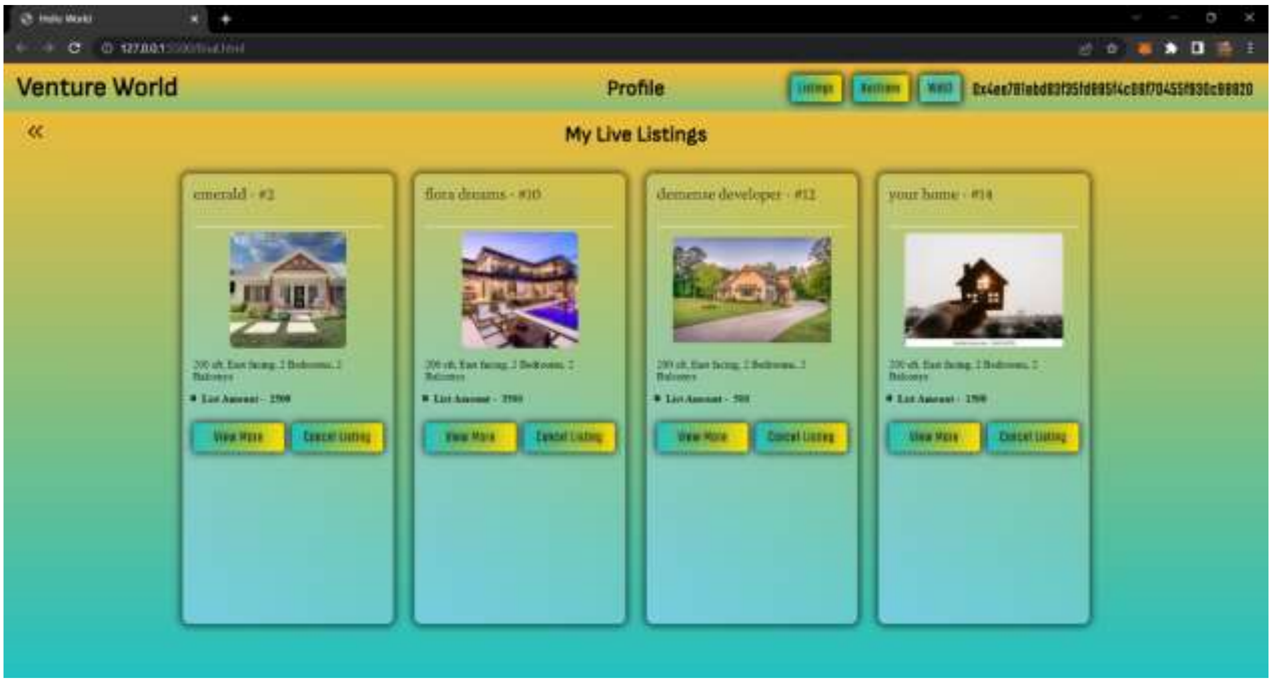
Highest Bidder Address :

[illegible]

Confirm to Cancel Auction!

Cancel Confirm

End Auction



Venture World

Profile

0x4ee781ebd83f35fd895f4c06f70455f930c98920

My Live Listings

emerald - #2

200 sft, East facing, 2 Bedrooms, 2 Balconys

- List Amount - 2500

flora dreams - #10

200 sft, East facing, 2 Bedrooms, 2 Balconys

- List Amount - 3500

demense developer - #12

200 sft, East facing, 2 Bedrooms, 2 Balconys

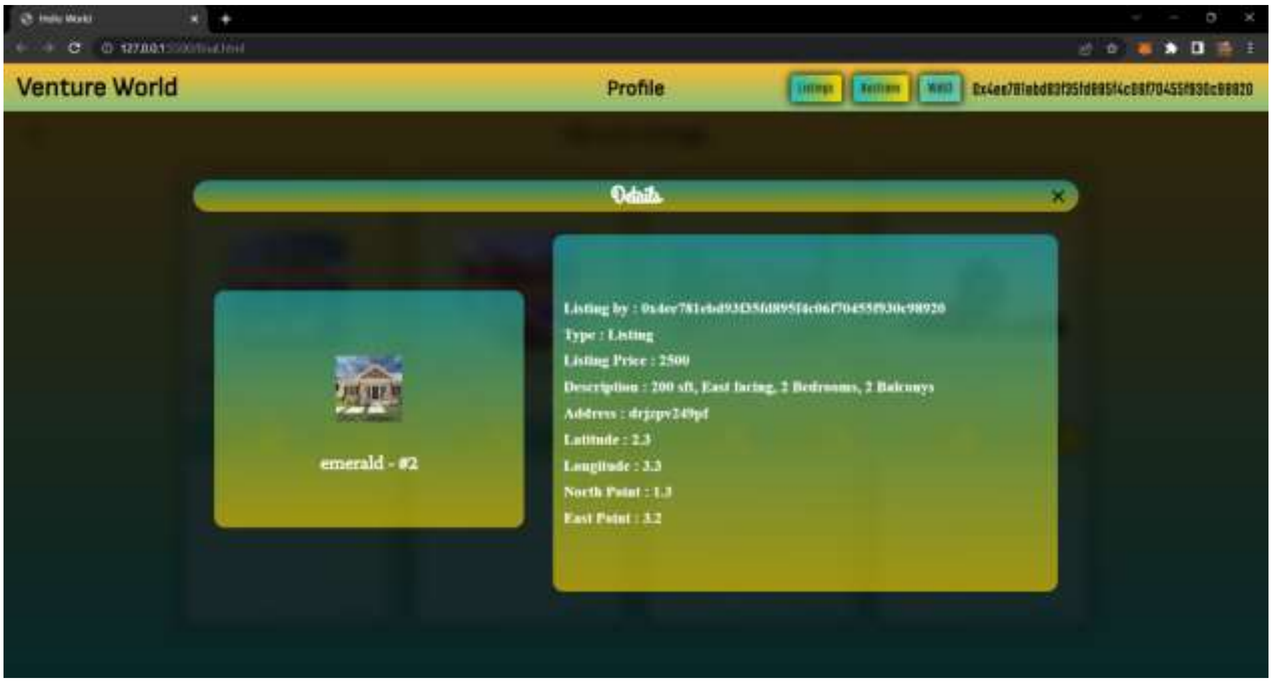
- List Amount - 500

your home - #14

200 sft, East facing, 2 Bedrooms, 2 Balconys

- List Amount - 1500

User Create Live Listings



Venture World

Profile

0x4ee781ebd83f35fd895f4c06f70455f930c98920

Details

emerald - #2

Listing by : 0x4ee781ebd93f35fd895f4c06f70455f930c98920

Type : Listing

Listing Price : 2500

Description : 200 sft, East facing, 2 Bedrooms, 2 Balconys

Address : drjzpv249pf

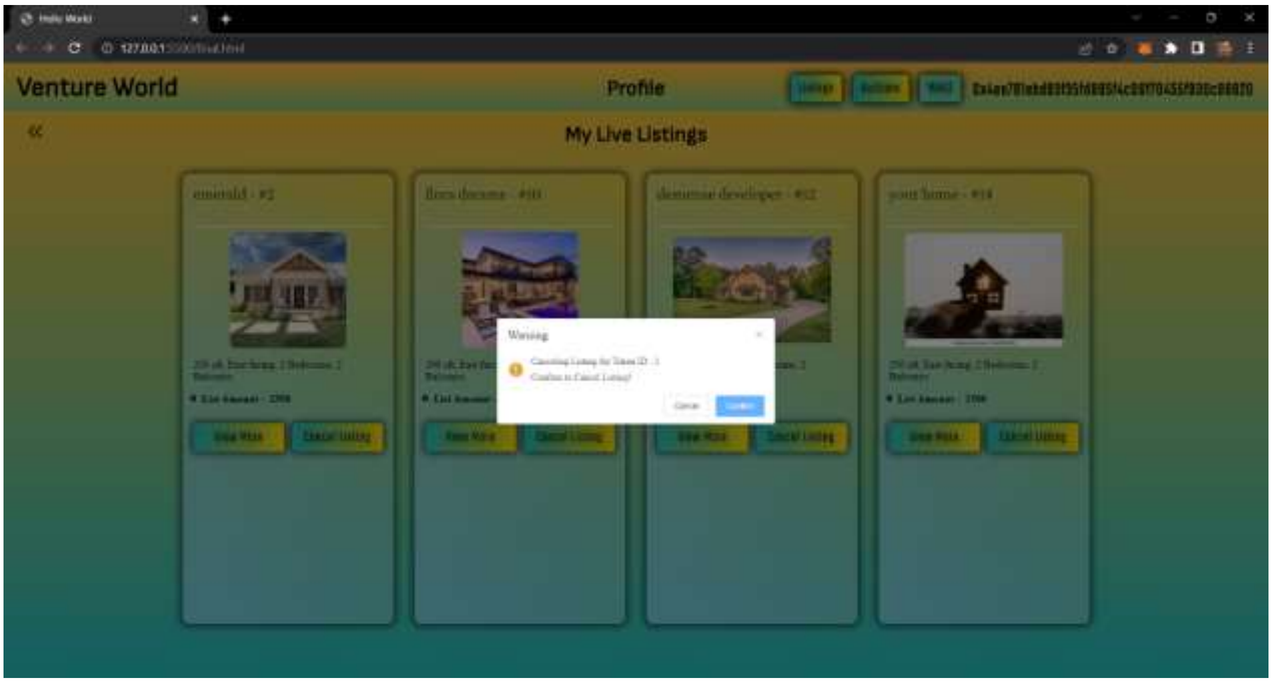
Latitude : 2.3

Longitude : 3.3

North Point : 1.3

East Point : 3.2

View Details(User Created Live Listings)



Warning

Canceling Listing for Token ID : 1
Confirm to Cancel Listing?

Cancel Listing

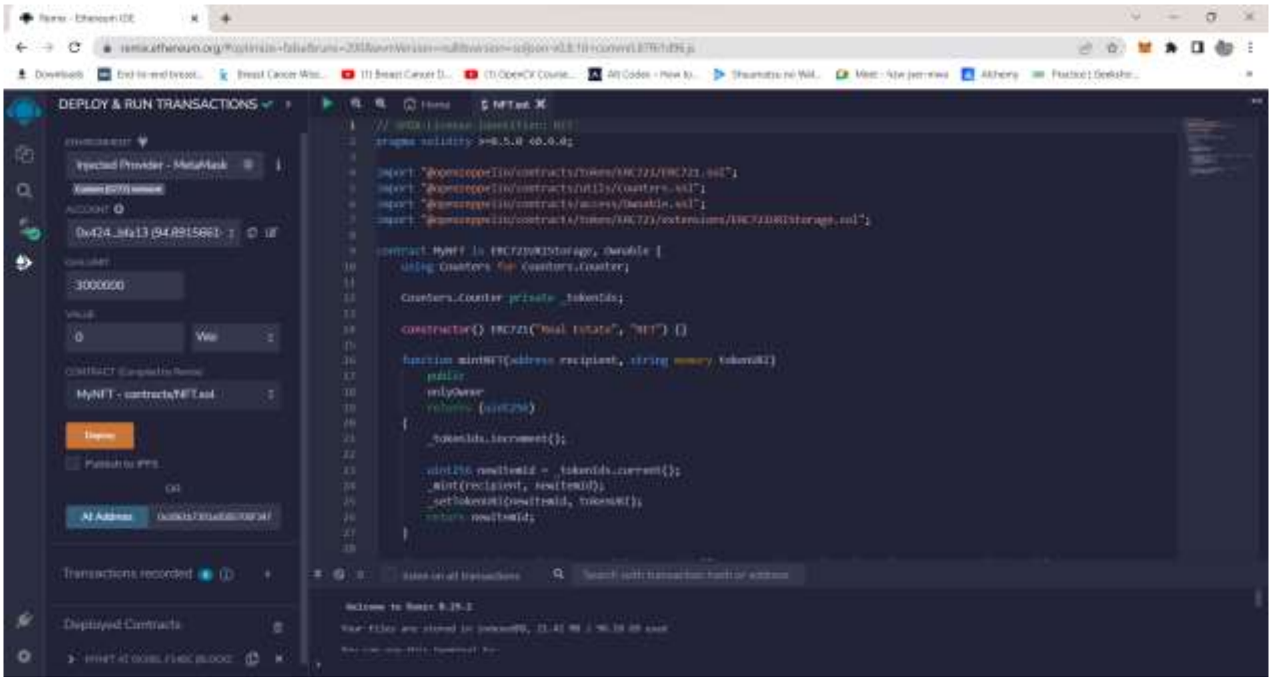
Running the application

- **Setting up NFTs**

NFTs can be minted one after the other or in a batch after the smart contract for NFTs is deployed. Each NFT will have a string URI which is the meta data of that particular NFT. Instead of saving all the data related to a property directly as the string URI which is not cost efficient, IPFS is used which is a de-centralized file distribution network to store the data of each individual property and it provides a CID which is used to point to the data in the network and this CID can be used as the string URI for each NFT which is cost efficient.

Now, to store the meta data in IPFS and get the CID a python script is used which takes a folder containing images for the NFTs and all the meta data related to each NFT from the client as inputs and stores them in the IPFS network through the help of a website called Pinata and returns a list of strings where each string is a CID which is pointing towards specific meta data for each NFT.

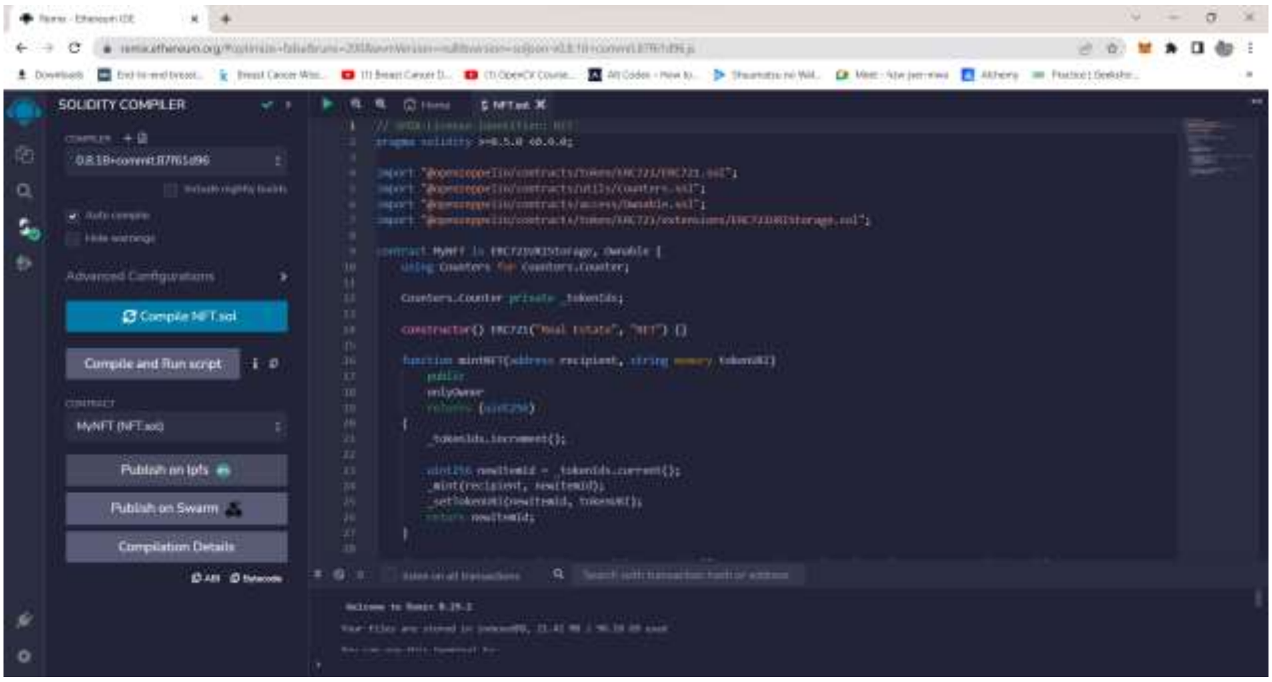
Remix IDE is used to deploy the NFT smart contract which has three functions mintNFT, batchMintNFT and viewBatchURI. After successful deployment, mintNFT function can be used multiple times to mint NFTs one at a time but for this demonstration the batchMintNFT function is used which will take a reciever address which is the address to which the NFTs are minted and it also takes a list of string URIs as Input and mints all the required NFTs in one go. Now, the NFTs representing each property are minted to the client's crypto wallet address.


DEPLOY & RUN TRANSACTIONS
Injected Provider - MetaMask
3000000
0
Wei
MyNFT - contracts/NFT.sol
Deploy
At Address
Transactions recorded
Deployed Contracts
NFT.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.5.0 <0.9.0;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

contract MyNFT is ERC721URIStorage, Ownable {
    using Counters for Counters.Counter;

    Counters.Counter private _tokenIds;

    constructor() ERC721("Real Estate", "NFT") {}

    function mintNFT(address recipient, string memory tokenURI)
        public
        onlyOwner
        returns (uint256)
    {
        _tokenIds.increment();

        uint256 newItemId = _tokenIds.current();
        _mint(recipient, newItemId);
        _setTokenURI(newItemId, tokenURI);
        return newItemId;
    }
```

```
Welcome to Remix 0.29.2
Your files are stored in indexedDB, [illegible]
```

After the deployment of the NFT smart contract, the Contract Address must be saved which can be seen below the deploy button as it is required at a later step. Then going to the solidity compile slide to view the compilation of the smart contract and at the bottom of the compilation information, it shows the ABI and Bytecode of the smart contract, the ABI for the smart contract is also saved as it is required at a later step.



```
// SPDX-License-Identifier: MIT
pragma solidity >=0.5.0 <0.9.0;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

contract MyNFT is ERC721URIStorage, Ownable {
    using Counters for Counters.Counter;

    Counters.Counter private _tokenIds;

    constructor() ERC721("Real Estate", "NFT") {}

    function mintNFT(address recipient, string memory tokenURI)
        public
        onlyOwner
        returns (uint256)
    {
        _tokenIds.increment();

        uint256 newItemId = _tokenIds.current();
        _mint(recipient, newItemId);
        _setTokenURI(newItemId, tokenURI);
        return newItemId;
    }
```

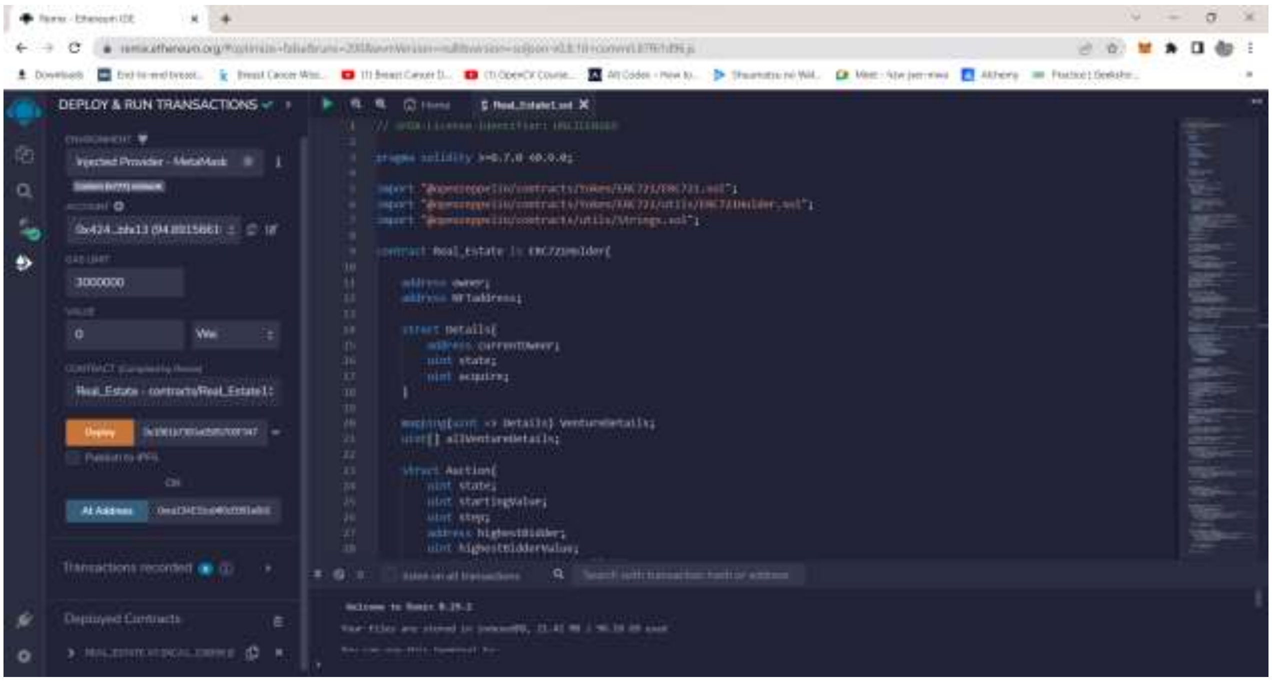
```
Welcome to Remix 0.29.1
Your files are stored in indexedDB, [illegible]
[illegible]
```

- **Setting up Real Estate Contract**

Now, the Real Estate smart contract is deployed which is the heart of our project as it contains the entire logic for the project. The constructor for the smart contract requires two parameters to deploy successfully.

First parameter requires the contract address of the previously deployed NFT smart contract as it has to link up with it to perform transactions with the NFTs minted from it, which is done under the supervision of the user as the user has to approve the Real Estate smart contract to be able to transact with his/her NFTs and can revoke permission in the future if he/she wants to.

Second parameter requires an array of integers representing the Token IDs minted from the NFT smart contract. This parameter is used to give the client an option to choose which NFTs are to be initiated for transactions within the venture at the time of deployment. The Client can later add more properties to the Venture if he/she wishes to by invoking the addProperties function.



DEPLOY & RUN TRANSACTIONS

Injected Provider - MetaMask

ACCOUNT: 0x424...a13 (94.8815661

GAS LIMIT: 3000000

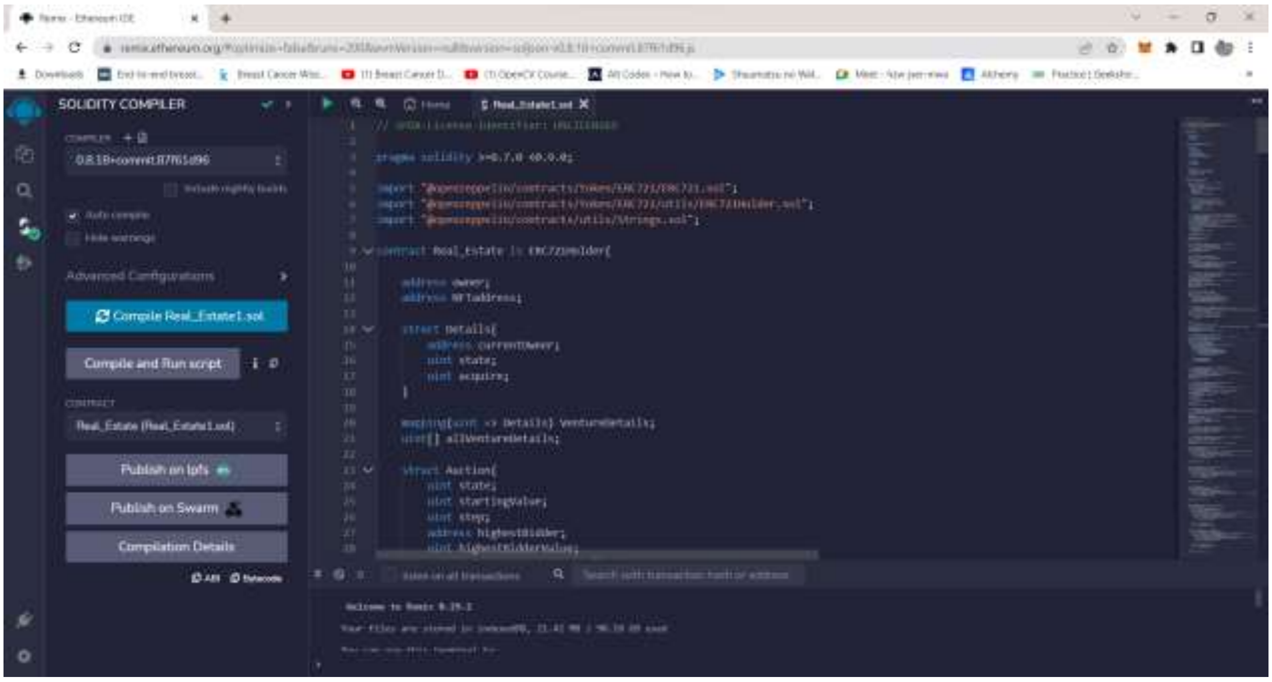
VALUE: 0 Wei

CONTRACT: Real_Estate - contracts/Real_Estate1

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

pragma solidity >=0.7.0 <0.9.0;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol";
import "@openzeppelin/contracts/utils/Strings.sol";

contract Real_Estate is ERC721Holder{

    address owner;
    address NFTaddress;

    struct Details{
        address currentOwner;
        uint state;
        uint acquire;
    }

    mapping(uint => Details) ventureDetails;
    uint[] allVentureDetails;

    struct Auction{
        uint state;
        uint startingValue;
        uint step;
        address highestBidder;
        uint highestBidderValue;
```

```
Welcome to Remix 0.29.1
```

Once the Real Estate smart contract is also deployed, the Contract Address must be saved which can be seen below the deploy button as it is required at in the next step. Then going to the solidity compile slide to view the compilation of the smart contract and at the bottom of the compilation information, it shows the ABI and Bytecode of the smart contract, the ABI for the smart contract is also saved as it is required in the next step.



SOLIDITY COMPILER

COMPILER: 0.8.18+commit.87f61d96

✓ Auto compile

Compile Real_Estate1.sol

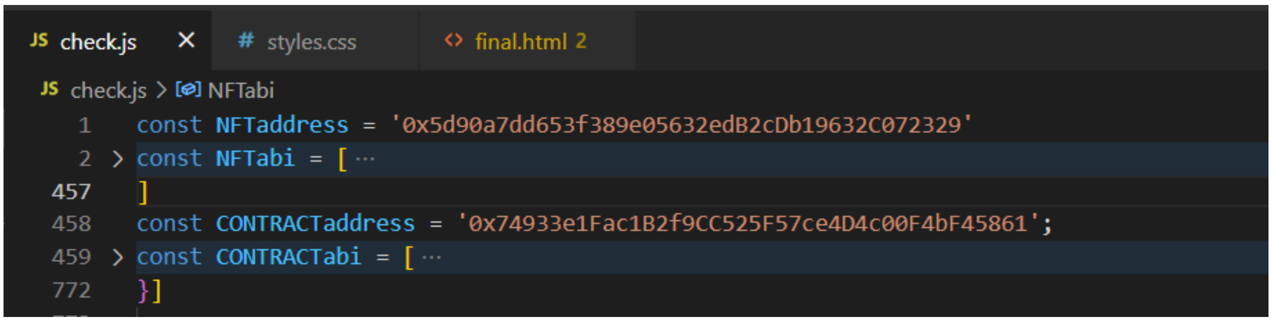
CONTRACT: Real_Estate (Real_Estate1.sol)

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

pragma solidity >=0.7.0 <0.9.0;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol";
import "@openzeppelin/contracts/utils/Strings.sol";

contract Real_Estate is ERC721Holder{

    address owner;
    address NFTaddress;

    struct Details{
        address currentOwner;
        uint state;
        uint acquire;
    }

    mapping(uint => Details) ventureDetails;
    uint[] allVentureDetails;

    struct Auction{
        uint state;
        uint startingValue;
        uint step;
        address highestBidder;
        uint highestBidderValue;
```

- **Setting up Front End**

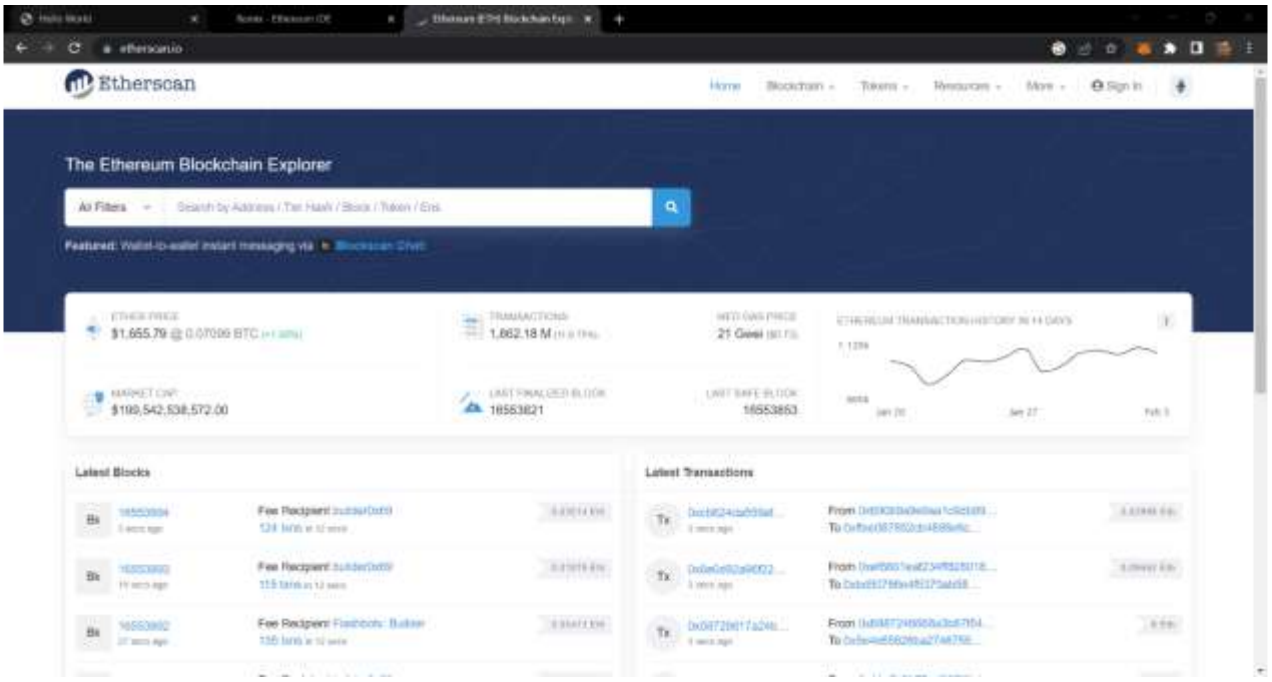
Finally to setup the front end, few changes to the initial lines in the JavaScript part of the code is needed. The data saved in the previous steps is used to make the following changes.

The first line will be adding the NFT smart contract address to the NFTaddress variable as a string. The second line will be the ABI for the NFT smart contract assigned to the variable called NFTabi. The third line will be the contract address for the Real Estate smart contract assigned to the variable CONTRACTaddress, and lastly the ABI for the Real Estate smart contract is assigned to the variable CONTRACTabi.



```
const NFTaddress = '0x5d90a7dd653f389e05632edB2cDb19632C072329'
const NFTabi = [ ···
]
const CONTRACTaddress = '0x74933e1Fac1B2f9CC525F57ce4D4c00F4bF45861';
const CONTRACTabi = [ ···
}]
```

The client can also view all the details related to the smart contracts after deployment using their contract address in a website called etherscan.io where all the transactions, costs, timestamps, history, etc. related to that smart contract can be viewed.



Workflow of the transactions

There are three possible transactions in the product i.e., a Listing, an Auction or a Donation. The NFT in the venture can be traded using these three ways of transactions and the Real Estate smart contract helps to make the transactions more secure and safe for both the parties involved. A user can List, Auction, Donate their property which once done can be seen by the everyone.

- If the user wants to create a listing, he/she need to specify a value to list the NFT for and can also cancel the listing in the future before its sold. The NFT is transferred to the Real Estate smart contract and stored there until it is bought or the listing is canceled. If the listing is bought, then the money is transferred to the listing owner and the NFT is transferred to the buyer automatically by the Real Estate smart contract. If the listing is canceled, the NFT is transferred back to the listing owner and the listing is taken off the Live page. After any successful transaction the details for the particular property is updated.
- If the user wants to create an Auction, he/she need to specify the starting price for the bidding and the step price which acts as the difference between each bid. The NFT is transferred to the Real Estate smart contract and stored there until the auction owner ends the auction in which case the highest bidder’s amount is transferred to the auction owner and the NFT is transferred to the highest bidder and all the other bids are refunded, or until the auction owner cancels the auction in which case all the bids are refunded and the NFT is transferred back to the auction owner. After any successful transaction the details for the particular property is updated.
- The user can also choose to donate the property in which case the user needs to specify the receiver’s address and that NFT is directly transferred from the sender address to the reciever address. After any successful transaction the details for the particular property is updated.

Additional Resources

https://medium.com/predict/ethereum-create-a-smart-contract-in-5-minutes-9bfc449c7ae5

https://medium.com/coinmonks/hello-world-smart-contract-using-ethers-js-e33b5bf50c19

https://medium.com/coinmonks/introduction-to-solidity-programming-and-smart-contracts-for-complete-beginners-eb46472058cf

https://medium.com/coinmonks/how-to-interact-with-blockchain-using-ethers-js-7d0be4a8a6e8#:~:text=js-,Ethers.,Alchemy%2C%20Cloudflare%2C%20or%20Metamask.

The Team

The Real Estate Application using Ethereum was in development for 12 weeks. The following have worked on the development of the application —

Gadiraju Prajwal Kumar	Addanki Tirumala Vishnu Vardhan
Vineet karwa	Pavan Kumar K N
Anirudh Gadekar	Mustafa Ali Baig
Alishala Sai Ujwala	Anusree Bejugam