***Blockchain***

Project

# **PROBLEM STATEMENT**

**Problem**: Currently, we use the central authority to transfer the property ownership. This makes it time consuming and attracts a lot of extra expenses too with an additional burden of document management. Also, since the system is centralized, there is always a possibility of fraudulent.

**Possible Solution**: One technology that immediately comes to mind is the blockchain and the elegant way in which it uses distributed ledgers. If we use blockchain to record land transfers, we can have an immutable history of every property transaction that can be viewed by everyone and tampered by no one.

This brings us to a point that how the government will play its role when people shall try to transfer the property on their own? Here govt can initiate this service as a part of e-governance & cut down delays

**Conditions to meet:**

* Insert some dummy properties to replicate the real world. These properties shall have basic characteristics e.g. address, location, floors & etc.

Make sure you create demo entries from a single owner. This single owner shall assign the properties to you.

If you try to replicate it in real world, generally the government shall be signing the document which says that now you’re the owner of this property. So here in this case, think that you’re building this solution for a government. Government shall be assigning the properties after verifying your original documents. Hence, make sure that you generate the dummy properties by a single owner & this shall be distributing it to multiple people (addresses).

* Any address can send the property from his account to other owner. Check the ownership of property before transferring it to the other owner.
* Try to include as many general parameters as possible to keep the smart contract almost replicating the real-world transaction
* After the successful transfer of the property, you need to be sure that the old owner is not able to send the property again (encountering double spending).
* Finally, the new owner must be able to send the freshly received property to any owner(address), he/she wishes to.

# **Solution**

# THE CONTRACT OT BE DEPLOYED ON REMIX

1. pragma solidity ^ 0.4.11;
2. contract PropertyTransfer {
3. address public DA;
4. uint256 public totalNoOfProperty;
6. function PropertyTransfer() {
7. DA = msg.sender;
8. }
9. modifier onlyOwner() {
10. require(msg.sender == DA);
11. \_;
12. } // To represent type of property i.e land or flat or bungalow or house
13. enum TypeOfProperty {
14. LAND, FLAT, BUNGALOW, HOUSE
15. }
16. struct Property {
17. string name;
18. bool isSold;
19. TypeOfProperty propertytype;
20. string postaladdress;
21. uint256 surfacearea; // in square ft
22. string ownername;
23. string owneraddress;
24. string ownerphonenumber;
25. uint propertycost; // It contains at how much cost property was sold.
26. }
27. mapping(address => mapping(uint256 => Property)) public propertiesOwner;
28. mapping(address => uint256) individualCountOfPropertyPerOwner;
29. event PropertyAlloted(address indexed \_verifiedOwner, uint256 indexed \_totalNoOfPropertyCurrently, string \_nameOfProperty, string \_msg);
30. event PropertyTransfered(address indexed \_from, address indexed \_to, string \_propertyName, string \_msg); // a getter function for propertiesOwner
31. function getpropertiesOwner(address \_ownerAddress, uint256 \_inxed) returns(string) {
32. string propertyname = propertiesOwner[\_ownerAddress][\_inxed].name;
33. return propertyname;
34. }
36. function getPropertyCountOfAnyAddress(address \_ownerAddress) view returns(uint256) {
37. uint count = 0;
38. for (uint i = 1; i <= individualCountOfPropertyPerOwner[\_ownerAddress]; i++) {
39. if (propertiesOwner[\_ownerAddress][i].isSold != true) {
40. count += 1;
41. }
42. }
43. return count;
44. }
45. /\*allotProperty is the function which is responsible of assigning property to anyone,    this function is only accessible by one owner which is here the Government who deploy the    contract.\*/
46. function allotProperty(address \_verifiedOwner, string \_propertyName, string \_propertytype, string \_postaladdress, uint256 \_surfacearea, string \_ownername, string \_owneradd, string \_ownerphonenumber, uint256 \_propertycost) onlyOwner returns(bool) {
47. individualCountOfPropertyPerOwner[\_verifiedOwner] += 1;
48. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].name = \_propertyName; // assign the type of property
49. if (stringsEqual(\_propertytype, "HOUSE")) {
50. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].propertytype = TypeOfProperty.HOUSE;
51. } else if (stringsEqual(\_propertytype, "LAND")) {
52. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].propertytype = TypeOfProperty.LAND;
53. } else if (stringsEqual(\_propertytype, "BUNGALOW")) {
54. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].propertytype = TypeOfProperty.BUNGALOW;
55. } else if (stringsEqual(\_propertytype, "FLAT")) {
56. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].propertytype = TypeOfProperty.FLAT;
57. }
58. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].postaladdress = \_postaladdress;
59. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].surfacearea = \_surfacearea;
60. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].ownername = \_ownername;
61. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].owneraddress = \_owneradd;
62. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].ownerphonenumber = \_ownerphonenumber;
63. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].propertycost = \_propertycost;
64. totalNoOfProperty++;
65. propertiesOwner[\_verifiedOwner][individualCountOfPropertyPerOwner[\_verifiedOwner]].isSold = false;
66. PropertyAlloted(\_verifiedOwner, individualCountOfPropertyPerOwner[\_verifiedOwner], \_propertyName, "property allotted successfully");
67. }
69. function stringsEqual(string al, string a2) view returns(bool) {
70. return sha3(al) == sha3(a2) ? true : false;
71. }
73. function isOwner(address \_checkOwnerAddress, string \_propertyName) view returns(uint) {
74. uint i;
75. bool flag = false;
76. for (i = 1; i <= individualCountOfPropertyPerOwner[\_checkOwnerAddress]; i++) {
77. if (stringsEqual(propertiesOwner[\_checkOwnerAddress][i].name, \_propertyName) == true && propertiesOwner[\_checkOwnerAddress][i].isSold == false) {
78. flag = true;
79. break;
80. }
81. }
82. if (flag == true) {
83. return i;
84. } else {
85. return 999999999;
86. }
87. } /\*Function to transfer property from one owner to another owner\*/ // propertycost to set at which price the property is sold.
88. function transferProperty(address \_to, string \_propertyName, uint256 \_propertycost, string \_newownername, string \_newowneraddress, string \_newownerphone) returns(uint, uint) {
89. uint256 checkOwner = isOwner(msg.sender, \_propertyName); // checking the ownership of property
90. if (checkOwner != 999999999 && propertiesOwner[msg.sender][checkOwner].isSold == false) { /// step 1 . remove the property from the current owner and decrase the counter. /// step 2 . assign the property to the new owner and increase the counter
91. propertiesOwner[msg.sender][checkOwner].isSold = true;
92. propertiesOwner[msg.sender][checkOwner].name = "Sold"; // really nice finding. we can't put empty string
93. individualCountOfPropertyPerOwner[\_to] += 1;
94. propertiesOwner[\_to][individualCountOfPropertyPerOwner[\_to]].name = \_propertyName;
95. propertiesOwner[\_to][individualCountOfPropertyPerOwner[\_to]].propertytype = propertiesOwner[msg.sender][checkOwner].propertytype;
96. propertiesOwner[\_to][individualCountOfPropertyPerOwner[\_to]].postaladdress = propertiesOwner[msg.sender][checkOwner].postaladdress;
97. propertiesOwner[\_to][individualCountOfPropertyPerOwner[\_to]].surfacearea = propertiesOwner[msg.sender][checkOwner].surfacearea;
98. propertiesOwner[\_to][individualCountOfPropertyPerOwner[\_to]].ownername = \_newownername;
99. propertiesOwner[\_to][individualCountOfPropertyPerOwner[\_to]].owneraddress = \_newowneraddress;
100. propertiesOwner[\_to][individualCountOfPropertyPerOwner[\_to]].ownerphonenumber = \_newownerphone;
101. propertiesOwner[\_to][individualCountOfPropertyPerOwner[\_to]].propertycost = \_propertycost;
102. propertiesOwner[\_to][individualCountOfPropertyPerOwner[\_to]].isSold = false;
103. PropertyTransfered(msg.sender, \_to, \_propertyName, "Owner has been changed.");
104. return (10, checkOwner);
105. } else {
106. revert('Something bad happened');
107. }
108. }
109. }

# set compiler version on remix

Settings 🡪 Select new compiler version.

Current Version: 0.4.25-nightly.2018.5.30+commit.3f3d6df2.Emscripten.clang

(The version above it can cause syntax error.)

# Deploying smart contract

Deploy the smart contract using first address, because this address is set as default address for transactions.

# How to test

* 1. First run the testrpc on the terminal.
  2. Set Environment to: Web3 provider
  3. After deploying the contract, replace the contract address in index.html page with current contract address.
  4. After that, open the index.html file.

(If dummy transactions are not visible then in such case refresh the page)

***NOTE: Use only those addresses which are displayed on index.html page.***

# The dummy transactions present

***(The accounts are handled using Web3.eth.accounts[] that’s why the accounts numbers are represented by indexes)***

**Address (2): // i.e 2nd address from the list displayed on index.html page**

Property Name : property1

Property Sold : false

Property type : 3

Property Address : 138 , Oppshops, Apollotower

Property Area : 900

Property Owner : Tanmay Pathak

Property Owner :Address A-12 Girraj Colony,Gwalior,(M.P)

Property Owner :Contact number 9852365478

Property Current Rate : 2500000

**Address (2):**

Property Name : property2

Property Sold : false

Property type : 3

Property Address : A-12 Millennium Apartment,Gwalior,(M.P)

Property Area : 1200

Property Owner : Tanmay Pathak

Property Owner :Address A-12 Girraj Colony,Gwalior,(M.P)

Property Owner :Contact number 9852365478

Property Current Rate : 3500000

**Address (3):**

Property Name : property3

Property Sold : false

Property type : 0

Property Address : A-12 Atul Apartment,Indore,(M.P)

Property Area : 760

Property Owner : Rajesh Kumar

Property Owner :Address Millennium Corridor, Super Corridor Rd, Palakhedi, Madhya Pradesh 452005

Property Owner :Contact number 9852365858

Property Current Rate : 3500000

**Address (4):**

Property Name : property4

Property Sold : false

Property type : 2

Property Address : Bypass Rd, Near Silver Spring Township, Phase II, Nayta Mundla, Indore, Madhya Pradesh 452001

Property Area : 1300

Property Owner : Vipin Kumar Sharma

Property Owner :Address Millennium Corridor, Super Corridor Rd, Palakhedi, Madhya Pradesh 452005

Property Owner :Contact number 985233652

Property Current Rate : 3500000

**Address (4):**

Property Name : property5

Property Sold : false

Property type : 1

Property Address : 131,1st Floor,Trade Center, 18 South Tukoganj, Indore, Madhya Pradesh 452001

Property Area : 750

Property Owner : Vipin Kumar Sharma

Property Owner :Address Millennium Corridor, Super Corridor Rd, Palakhedi, Madhya Pradesh 452005

Property Owner :Contact number 985233652

Property Current Rate : 350000

**Address (4):**

Property Name : property6

Property Sold : false

Property type : 3

Property Address : 620-621, Sector A, Mahalaxmi Nagar, Indore, Madhya Pradesh 452010

Property Area : 1100

Property Owner : Vipin Kumar Sharma

Property Owner :Address Millennium Corridor, Super Corridor Rd, Palakhedi, Madhya Pradesh 452005

Property Owner :Contact number 985233652

Property Current Rate : 3500000

# Code files backup copy





**(A copy present in codefiles folder)**