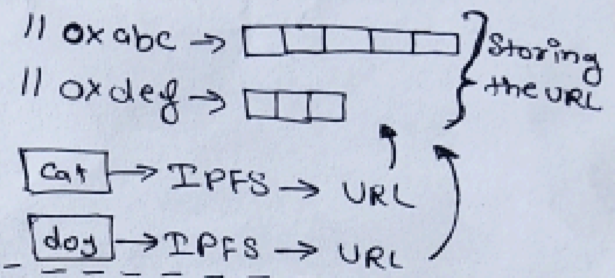
```
Struct Access {
        address user;
        bool access;  // This is for accessing the value is true or false
}
```

// Mapping With Array

* mapping (address => string[]) value;

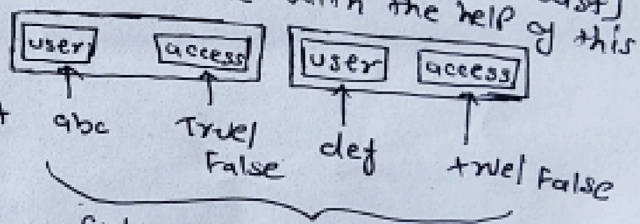


// Mapping

* mapping (address => Access[]) accessList;

// after sharing the image, their will be form (You recently accessing ···> [Form a list]
it will be done with the help of this.

after giving access it will be storing this list

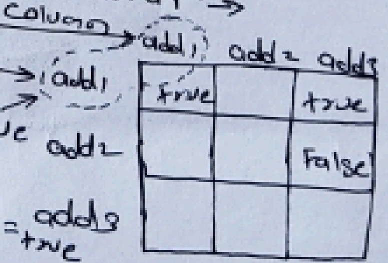


after Fetching the address of their list, [from smart contract]
it will be see how many user You giving the address.

// Nested Mapping

* mapping (address => mapping (address => bool)) ownership;

"2d array" →
column
Row



Ownership [add1] [add2] = true
or
Ownership[add 3] [add 1] = true

Ownership [add 2] [add 5] = False

* mapping (address => mapping (address => bool)) PriviousData;

// For storing the privious data of a server

// bec. We totally dependent on "blockchain"
So, We take these information on "blockchain"
We can't used to stored our information on server.

```
function add(address _user, String memory url) external {
    Value [_user].push(url);
}
                            // For adding information [images or some
                               Other video data] it will be given as
                               a URL.
                            // And it is dynamic array so, we can
                               easily push on that.
```

```
//Function for allowed
function allow(address user) external {
    ownership [msg.sender] [user] = true;
    // accessList [msg.sender].
       push(Access (user, true));
}
// this will be override
   so, it will not used
                            // Whatever user can access it will
                               be make msg.sender
                            // And whatever user can taking the
                               address it will be make second add.
                            // This function help to push these
                               above data to the Access List [array]

// Function for disallow
Function disallow (address user) public {
    ownership [msg. sender ] [user] = false;     // Sometime you want to
                                                    Revoke the user
    For (uint i=0; i<accessList [msg.sender].length;   // but you want to maintain
                                             i++){         the data or a list.
      if (accessList [msg.sender][i]. user == user)    // So thats why we run an
      {                                                   array one by one
                                       // this will be variable, 0,1,2,3
                                                                 List number
          accessList [msg.sender [i]. access = False;
                                       To the user(.user) of that list
      }
    }
}
                                       // it will not delete, we are just
                                          changing their data point.

For maintaining the Privious data

if (PreviousData [msg.sender][user]){     // For first time user want to
    For (uint i=0; i<accessList [msg.sender].length ; i++){   interact with it will be
      if (accessList [msg.sender [i].user == user) {          initially
        accessList [msg.sender]. ~~push (Access (user, true))~~;   False,
                                  access = true ;       [bec. of bool value
    }                                                    is initially 'False'.
}                                                        So that in that
else{                                                    condition the
    accessList [msg.sender]. push(Access(user, true));   else part will
    PreviousData [msg.sender][user] = true;              be run, and
                                                       // to make it data
                                                          true & their
                                                       // Privious data also
                                                          true.
Function display(address_User) external view returns (string[] memory){
        require(_user == msg.sender || ownership[_user][msg.sender],
                                                          ," You don't have an access");
        return value[_user];                               // images or data for displaying
  }                                                           Purpose.
                      // return the                // for checking Access, whether
                         data.                        the user have an access or
                                                          not.
                                                       // And initially their will be
                                                          have bool value. [true or False].
  Function shareAccess() Public view return (Access[] memory){
        return accessList[msg.sender];          // When we are sharing our
  }                                                       data to the other one,
}                                                      // So it will be form a list
                   // Return the accessList. Which user You sharing their
                                                 ~~ownership~~ or data recently
                                                 it will be form a list.
```