Project : **MarketPlace** (I have named it as Affordable Marketplace).

Marketplace Website Link : https://avulaudaykumarreddy.github.io/BCMarketPlace/

Code Repo: https://github.com/AvulaUdayKumarReddy/BCMarketPlace

This project is a decentralized app, where any user can list the items and any user can buy the items by connecting to the wallet, here we are using the metamask as the wallet. The App is built using the solidity smart contract on Remix IDE and the front end is built using HTML,CSS and Javascript.(as well as web3.js).

This document briefly explains the Project Design, Challenges and Issues faced, how to access the website, explaining the code for smart contract for each of the functions listed.

Project Design

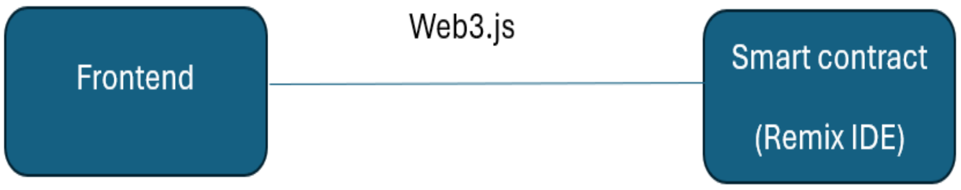
To get an understanding of the marketplaces, I have seen many youtube videos and read many blocks to get a clear understanding of the DApp marketplaces.Through this understanding I came up with a design to develop this marketplace.I took help from chatGPT to take my decisions and also to code.

I have divided the project into two modules.

1) Solidity smart contract.
2) Frontend with HTML, CSS, Javascript.

These both are integrated with web3.js using Javascript.

The solidity smart contract is compiled on Remix IDE using Inject provider- metamask as environment.we will see solidity smart contract role detailly in the further sections.



Challenges and Issues:

I have mainly faced challenges integrating the front end with the smart contract.

Implementing Buy:

The main implementation challenge here is to integrate the buy button with every listing, so that every buy functionality can be integrated individually. Apart from this I also need to ensure that the items that are listed by the seller should not be bought by the same seller. I have taken chatGPT help to integrate my buy button with the item listed.

The items listed will show the buy button for all the listed items, but if a seller clicks on the buy button of his own listed item, a alert will be generated saying that “**you cannot buy your item**” which avoids the item buying

Transaction Failed Issue for Buy :

I faced two issues while buying the items , very rarely I used to get the message from metamask showing this transaction likely going to fail and also charges more for transaction fees, but when reconnected to the site again this message is not displayed and transaction is done smoothly.

Another issue, after clicking confirm on the metamask wallet while buying, very rarely I got the **Transaction failed, already known** error. I didn’t find the reason for this error,but it happened only two times in vast testing.

Issue related to Refreshing items:

After doing the transactions, to see the transactional changes we need to press the display button. I have tried refreshing code but I noticed that site flickering is heavy, so opted to remove the refreshing the page automatically.

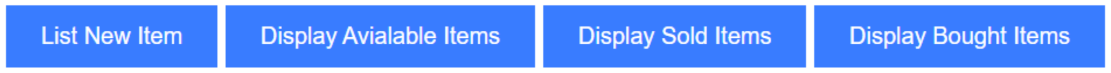
Issue when the contract is not deployed using inject provider- Metamask

I have unknowingly deployed the smart contract using the testnet - sepolia, I have copied the contract address and ABI, I have seen that website console which asked me check the ABI and contract address are matching , then I noticed that contract is not deployed using metamask, but we are connecting to metamask while doing the listing and buying.

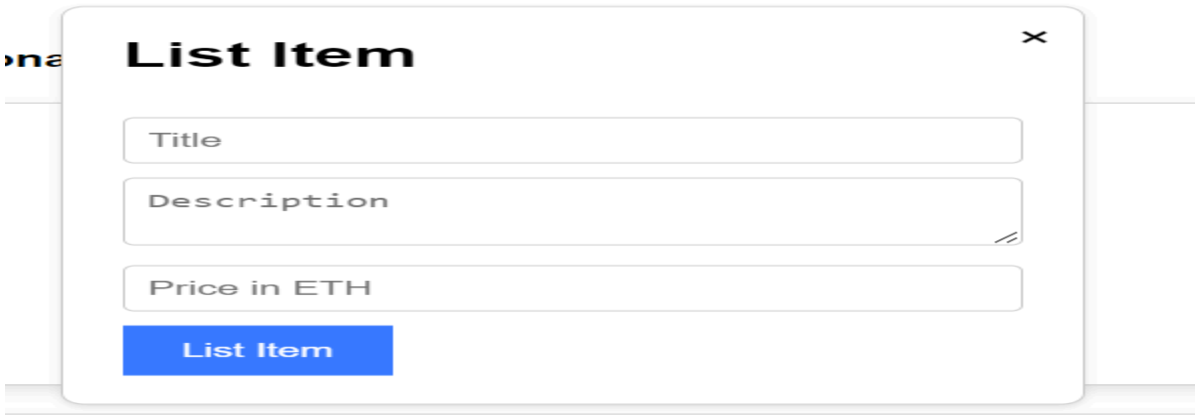
Accessing the website:

Go to the website : https://avulaudaykumarreddy.github.io/BCMarketPlace/

Click on the connect wallet button on the home page, this enables you to connect to the wallet and this will redirect to the page that can help to list the items and buy the items.



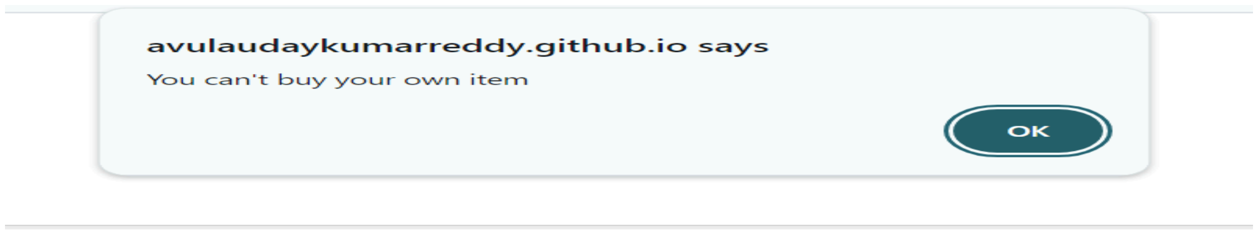
You can see the above figure, please click on the display available items to list all the items. To list the new item click on the “**List New Item**” button which will open a pop up button to list the new item.



Enter the details and click on the button to list the item, the transactional hash is displayed on the page which can be verified on the etherscan.

To buy an item, press the button the “Display all available items”, where you will get a list of the items, you can buy an item by clicking on the buy.

In the first attempt, I will try buying my own item.I will get this error as I can’t buy my own item.



avulaudaykumarreddy.github.io says

You can't buy your own item

In the second attempt, I will try to buy the listed item through another site.

The metamask will open and ask for confirmation, doing this our transaction gets submitted and we will get confirmation transaction hash which is verifiable on etherscan.

Solidity Smart contract code:

Listing New Item:

This section describes how to list a new item to the marketplace.

Smart contract Code for Listing New Item in marketplace

```
struct Item {
        int IID;
        ItemStatus status;
        address seller;
        string title;
        string description;
        uint price;
    }

```

```
function listItems(string memory title, string memory description,uint
price) external {
   c++;
 Item memory item =Item(
    c,
    ItemStatus.Active,
    msg.sender,
    title,
    description,
    price
 );

 _items[_itemId]=item;
  _itemId++;

 emit Itemized(
    _itemId,
    msg.sender,
    title,
    price
 );
}
```

The above code shows the structure of the Item that is captured, which contains an ID, title, seller(or owner), description and price, the item is initialized with the values and this item is maintained in _items array, which is used in javascript file to display the items.Here Itemized is event that can be emitted after the listing.

Buying item code:

```
function buy(uint itemId) external payable  {
    Item storage item = _items[itemId];
    require (item.status == ItemStatus.Active,"Listing is not active");
    require (msg.sender!=item.seller, "Seller cannot by their own
things");

    require(msg.value >= item.price, "Insufficient amount");

    item.status = ItemStatus.Sold;
    payable(item.seller).transfer(item.price);
    item.seller = msg.sender;
    emit Sale(
    _itemId,
    msg.sender,
    item.price
 );

}
```

The above code shows the buy function, which takes the parameter as Item ID to initiate the process.
The item is retrieved from the items array and checks whether the listing is active or not, the code proceeds forward if and only if the listing is active and secondly this code also checks if the buyer is seller, if buyer is seller then this code will stop and mentions that seller can’t buy their own things.

After these checks, the code also checks whether the buyer has enough funds to buy the item and subsequently it transfers the ownership of the item to buyer and price to the seller.

getAllItems:

```
function getAllItems() public view returns (Item[] memory) {
        Item[] memory allItems = new Item[](_itemId);
        for (uint256 i = 0; i < _itemId; i++) {
            allItems[i] = _items[i];
        }
        return allItems;
    }
```

This function is used for returning the items to our web page through web3.js and will be helpful incase of listing the items.This function creates an array of item object and copies items from _item array and returns it to the calling entity.