NFT EXPLANATION DOCUMENT.

Prepared by: Renee Krom.

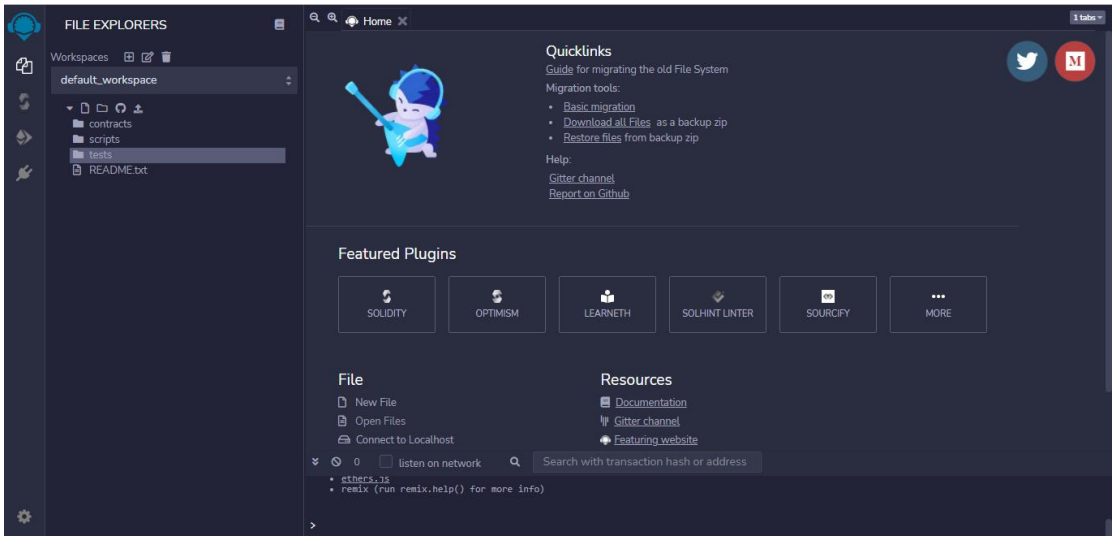
To: Irfan Khurshid

How to build and deploy smart contract

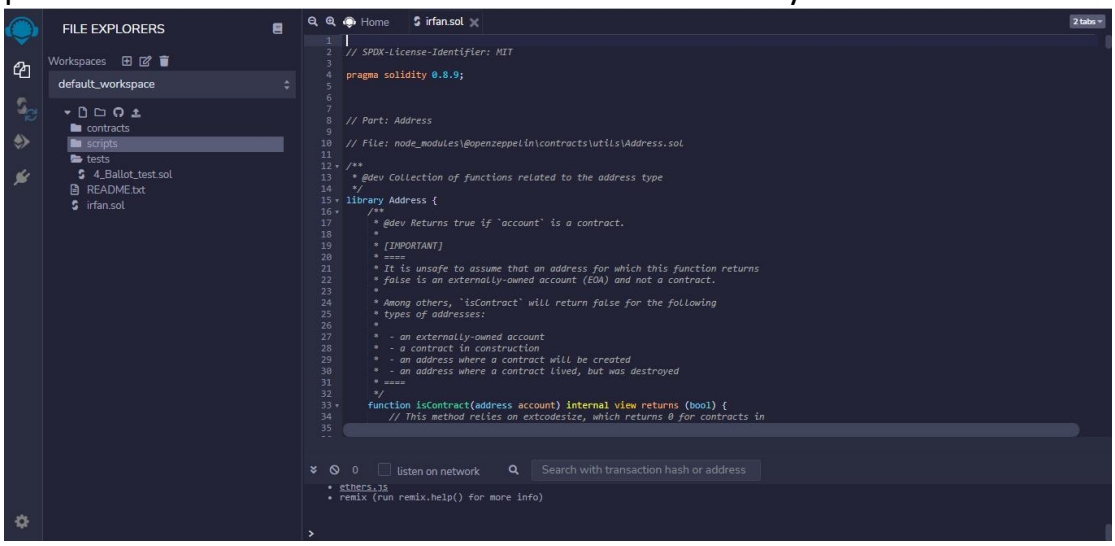
1. First we will use remix to deploy the smart contract.

-The sample smart contract can be found at https://rinkeby.etherscan.io/address/0x6E632C864b80DC76074d7c6443B9Df32578E53a9

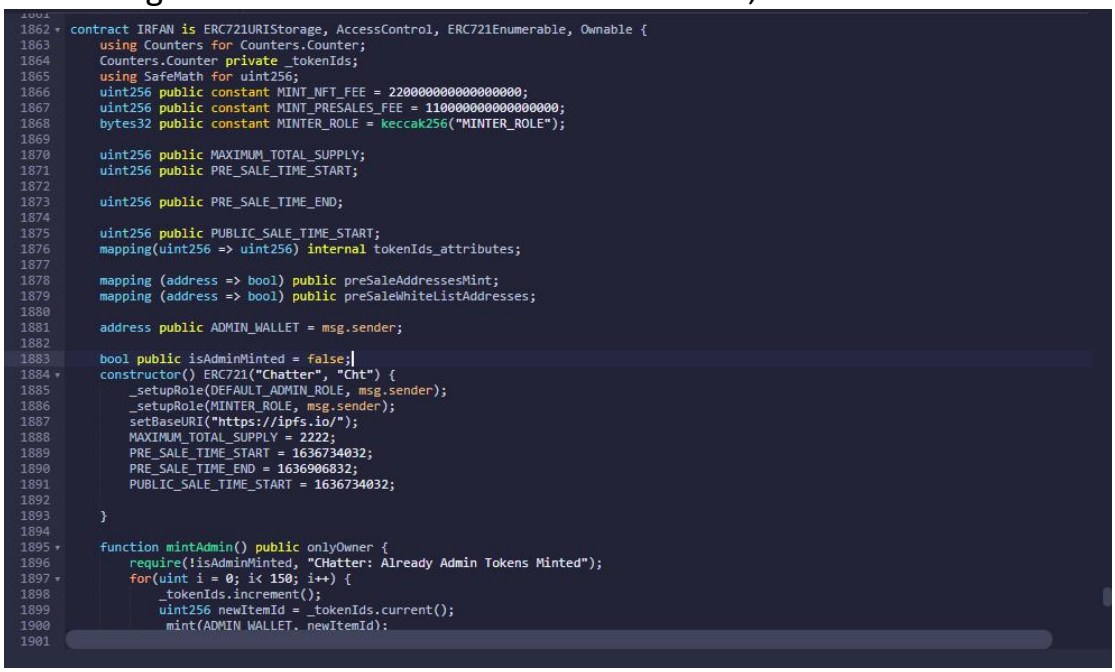
Open remix at http://remix.ethereum.org. and Follow the step as follows.



From the above screenshot, you go to the file explorer section and in default workspace bar you create a new file, lets call it irfan.sol and then paste all of our code from the smart contract I sent you.



From the above screenshot, We can see that our code is there in the remix IDE.
The next step is to configure neccessary details about the smart contract.
Now navigate to the line in the screenshot below,



```
contract IRFAN is ERC721URIStorage, AccessControl, ERC721Enumerable, Ownable {
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;
    using SafeMath for uint256;
    uint256 public constant MINT_NFT_FEE = 22000000000000000;
    uint256 public constant MINT_PRESALES_FEE = 110000000000000000;
    bytes32 public constant MINTER_ROLE = keccak256("MINTER_ROLE");

    uint256 public MAXIMUM_TOTAL_SUPPLY;
    uint256 public PRE_SALE_TIME_START;

    uint256 public PRE_SALE_TIME_END;

    uint256 public PUBLIC_SALE_TIME_START;
    mapping(uint256 => uint256) internal tokenIds_attributes;

    mapping (address => bool) public preSaleAddressesMint;
    mapping (address => bool) public preSaleWhiteListAddresses;

    address public ADMIN_WALLET = msg.sender;

    bool public isAdminMinted = false;
    constructor() ERC721("Chatter", "Cht") {
        _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
        _setupRole(MINTER_ROLE, msg.sender);
        setBaseURI("https://ipfs.io/");
        MAXIMUM_TOTAL_SUPPLY = 2222;
        PRE_SALE_TIME_START = 1636734032;
        PRE_SALE_TIME_END = 1636906832;
        PUBLIC_SALE_TIME_START = 1636734032;

    }

    function mintAdmin() public onlyOwner {
        require(!isAdminMinted, "CHatter: Already Admin Tokens Minted");
        for(uint i = 0; i< 150; i++) {
            _tokenIds.increment();
            uint256 newItemId = _tokenIds.current();
             mint(ADMIN_WALLET, newItemId);
```

From above we need to set up our own parameter, you need to change NFT_MINT_FEE and put the fee for public sales, and MINT_PRESALE_FEE for presale price.
In the constructor arguments, you need to change the name and the symbol as in “chatter”, “cht”, change your baseURI and input yours, change maximum supply to accommodate what you want.
Also we would like to input time for presale to start and end. We use unix timestamps to calculate that. So we need to change PRE_SALE_TIME_start to our own time start, and head on to https://www.unixtimestamp.com/ and convert normal date to unix timestamp and input the result to that, same goes to PRESALE_END_TIME and PUBLIC_SALE. And thats how you set time.

So from the above screenshot you have viewed certain details,
ADMIN_WALLET stands as owners address,
isAdminMinted set states to false because admin didn’t yet mint the token.

Those are only configurable/changed items in the contract before deployment. Now lets explain one function after another.

FUNCTION 1: mintAdmin

```
function mintAdmin() public onlyOwner {
    require(!isAdminMinted, "CHatter: Already Admin Tokens Minted");
    for(uint i = 0; i< 150; i++) {
        _tokenIds.increment();
        uint256 newItemId = _tokenIds.current();
        _mint(ADMIN_WALLET, newItemId);
    }
    isAdminMinted = true;
}
```

The above function mints 150 tokens to the admins wallet, this value can be changed above to accommodate what you want. And it mints all of the token to the admin wallet, and then after the function is executed it change the state of isAdminMinted to true,and admin cant mint again. This function is as proposed by you.

FUNCTION 2: preSaleMint

```
function preSaleMint(
  address _mintTo,
uint _maxMintCount
)
external payable
returns (bool)
{
    require(preSaleWhiteListAddresses[msg.sender], "CHATTER, You are not whitelisted to mint pre sale token");
    require(!preSaleAddressesMint[msg.sender], "CHATTER, You have already minted pre sale token");
    require(block.timestamp >= PRE_SALE_TIME_START && block.timestamp <= PRE_SALE_TIME_END, "CHATTER: This is not pre sale time for mi
    require(totalSupply() + 1 <= MAXIMUM_TOTAL_SUPPLY, "CHATTER: Maximum Supply Minted");
    require(_maxMintCount >= 0 && _maxMintCount <= 1, "CHATTER: Invalid Max Mint Count");
    require(msg.value == MINT_PRESALES_FEE, "CHATTER: Invalid Minting Fee");
    payable(ADMIN_WALLET).transfer(msg.value);
    _tokenIds.increment();
    uint256 newItemId = _tokenIds.current();
    _mint(_mintTo, newItemId);
    preSaleAddressesMint[msg.sender] = true;
    return true;
}
```

From above screenshot, this function allows whitelisted users to mint NFTs at a pre-defined time. This function execute only if all conditions are met. From is it whitelisted?, requiring the time to be in that pre-sales ongoing time and mint fee.

FUNCTION 3: bachMint

```
function bachMint(
address _mintTo
, uint256 _maxMintCount
)
external payable
returns (bool)
{
    require(block.timestamp >= PUBLIC_SALE_TIME_START, "CHATTER: Public Sale not yet started");
    require(_maxMintCount >= 0 && _maxMintCount <= 1, "CHATTER: Invalid Max Mint Count");
    require(totalSupply() + _maxMintCount <= MAXIMUM_TOTAL_SUPPLY, "CHATTER: Maximum Supply Minted");
    require(msg.value == MINT_NFT_FEE.mul(_maxMintCount), "CHATTER: Invalid Minting Fee");
    payable(ADMIN_WALLET).transfer(msg.value);
    for(uint i = 0; i < _maxMintCount; i++ ) {
        _tokenIds.increment();
        uint256 newItemId = _tokenIds.current();
        _mint(_mintTo, newItemId);
    }

    return true;
}
```

From above screenshot, we only allow minting for public sales, the function executes only if all conditions are met.

FUNCTION 4: setPreSaleWhiteListAddress

```
function setPreSaleWhiteListAddress(address[] memory addresses, bool[] memory statuses) public onlyOwner() {
    for (uint i = 0; i < addresses.length; i++) {
        preSaleWhiteListAddresses[addresses[i]] = statuses[i];
    }
}
```

The above function, help us to add an address to whitelist.

HOW TO DEPLOY THE UI.
The UI is in react, just use any tool that allow the deployment of React.

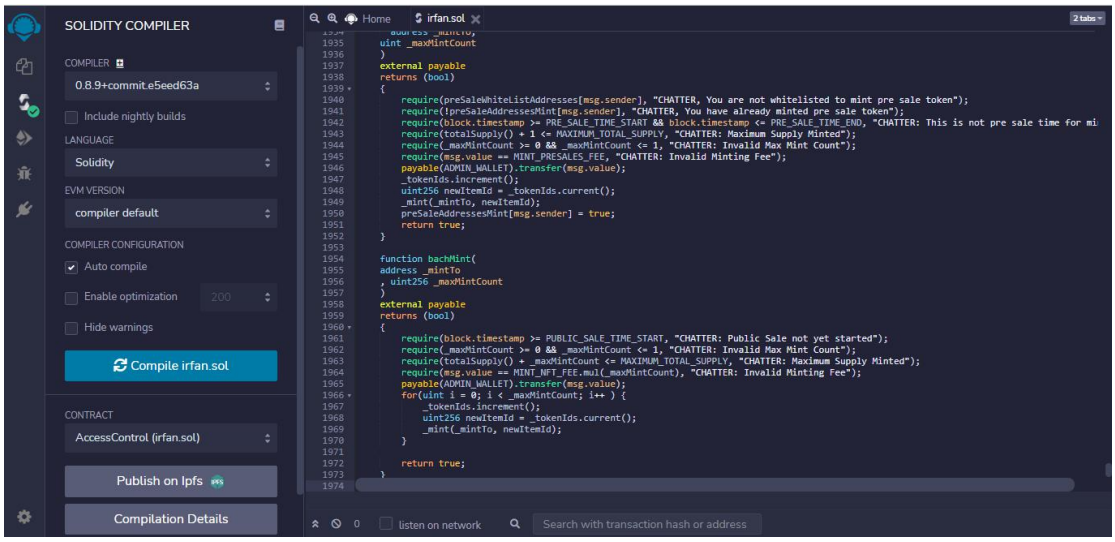
HOW TO ADD ADDRESS TO WHITELIST.
As in the function of setPreSaleWhiteListAddress it help us to add address, we may use etherscan to add these address, or have simple customized api call that will handle the addition to the smart contract, this will depend after you deploy the smart contract.

HOW TO MINT.
The site which is in react, it has a function that is already connected to the web3, We will just call the mint function from the the smart contract, Now this will depend with your final UI for the sales page.

So there is many thing that will need support or update, I will be there providing that even if the contract is ended.

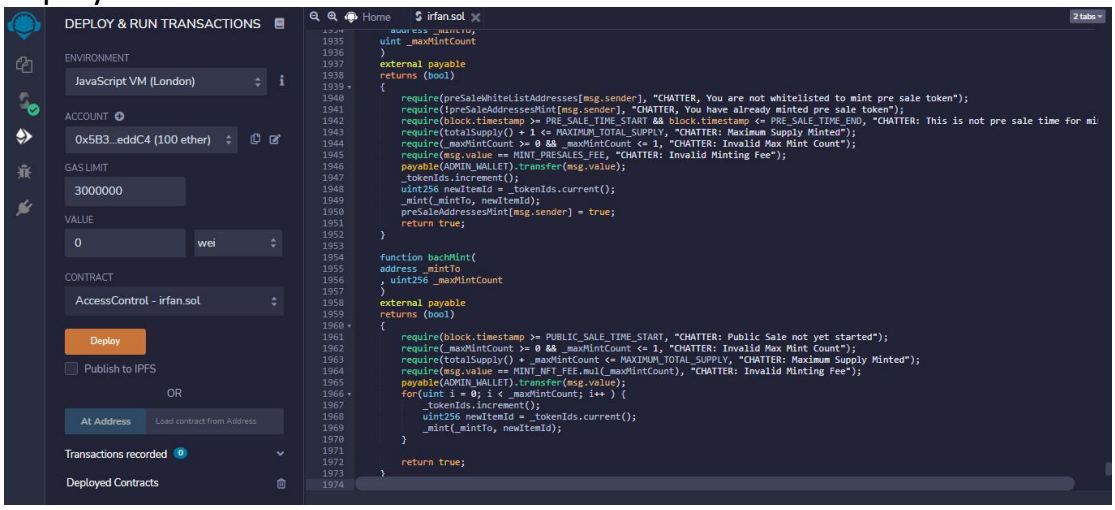
DEPLOYING THE SMART CONTRACT TO THE NETWORK.

Make sure you have metamask installed in your browser.
After that head in to remix and as follows.



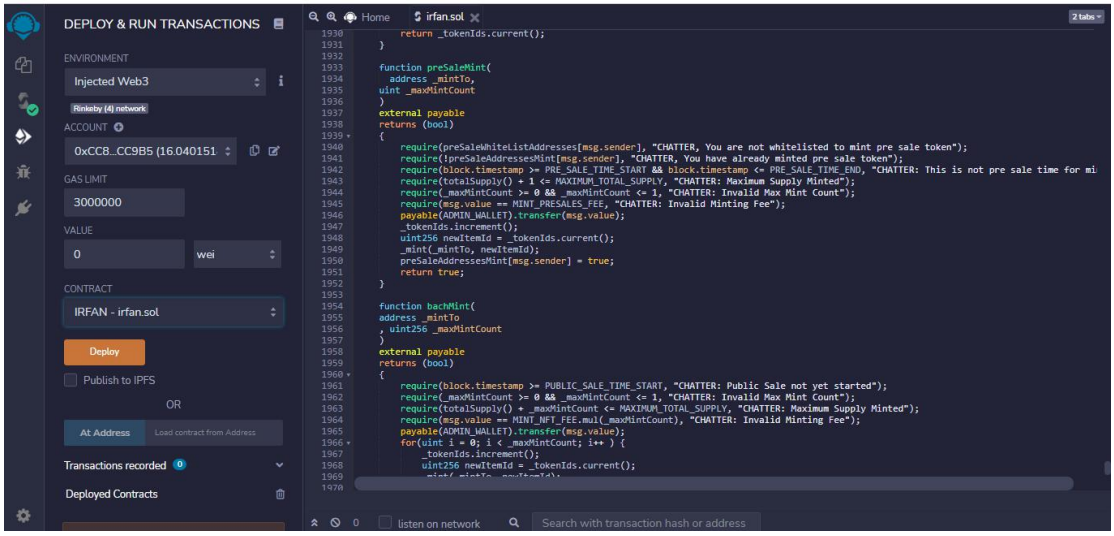
Go to the left handside of remix and choose as above. Meaning save the information like the compiler version and optimization status, These will be used later when verifying smart contract.

After that means the contract is compiled successfully and its time to deploy it. Head as in below.



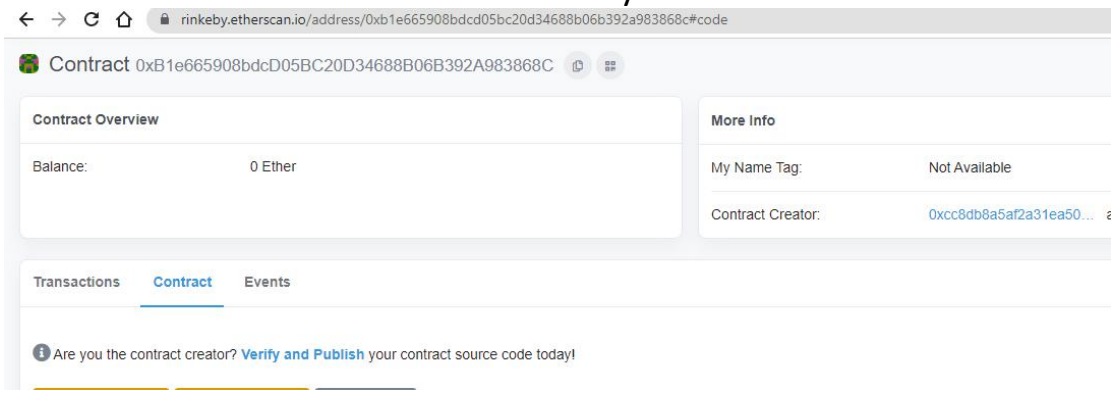
When you open it you will see exactly like that, and change to as follow, make sure you choose the right network in your metamask

MAKE SURE IN THE CONTRACT SECTION, THERE IS A DROPDOWN AND CHOOSE THE NAME OF THE CONTRACT AS IN OUR CASE ITS CALLED IRFAN.

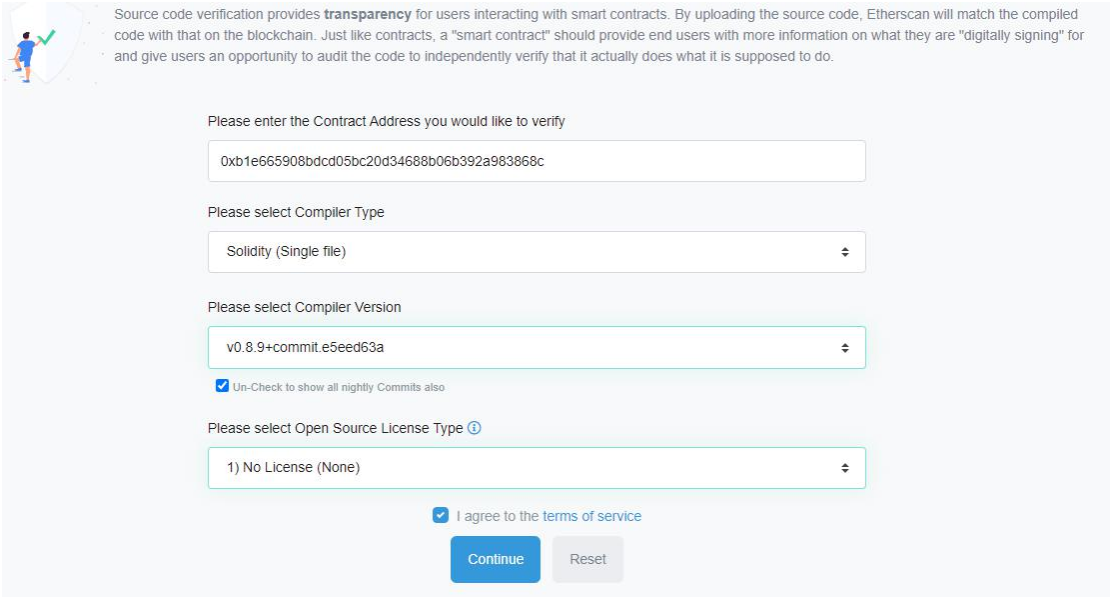


You must have as below, the only different will be you will be connected to main net.
After that fire the deploy button and it will be done.

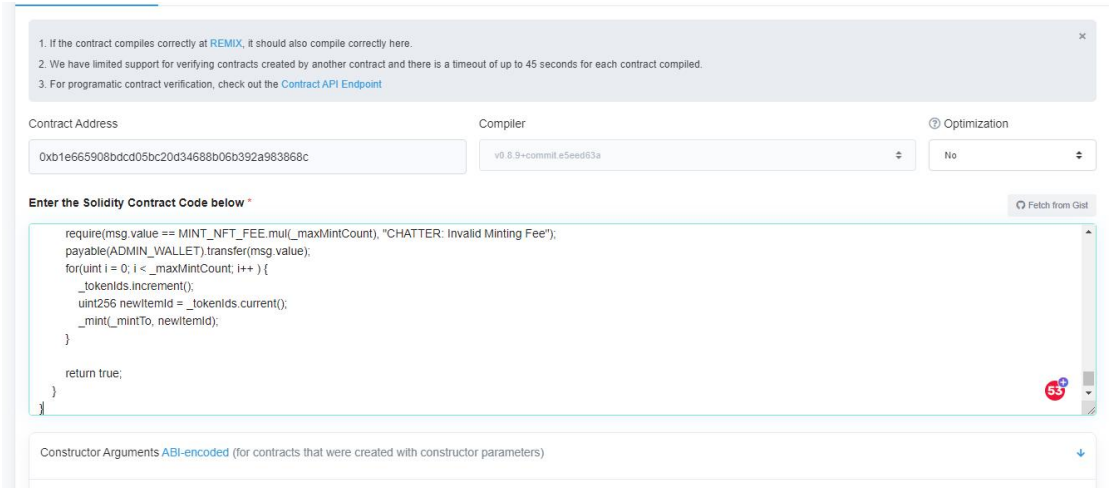
AFTER THAT IT WILL BE TIME TO VERIFY THE SMART CONTRACT.
Head In to etherscan and choose to verify the smart contract



Choose verify and publish.
After that please choose as indicated below,



Those are values that we took from remix when deploying.
After that hit continue button



Paste the code as instructed, paste the code from remix don't change anything after that.
And then hit verify and publish. And that will be done and the contract will done.
Please use the code which is here and modify as the document instruct, the code can be found here
https://rinkeby.etherscan.io/address/0x6E632C864b80DC76074d7c6443B9Df32578E53a9

Thanks.
Renee Krom.