

# Renewable Energy Trading with Blockchain and NFTs

## Team Members:

1. Sandeep Kunusoth - [skunusot@buffalo.edu](mailto:skunusot@buffalo.edu) - 50465621
2. Ashwin Ashok – [aashok3@buffalo.edu](mailto:aashok3@buffalo.edu) – 50478913

## Phase1:

### Abstract:

The following possible problems with conventional ways of selling renewable energy certificates (RECs) might reduce their efficacy in encouraging renewable energy:

1. Lack of market transparency: One problem with conventional REC sales is the lack of market openness. It might be challenging for purchasers to confirm the legitimacy and caliber of the RECs they are buying. This may result in a lack of faith and trust in the system, which may eventually deter customers from taking part.
2. Restricted geographic reach: Conventional REC sales frequently concentrate on particular geographic areas, which may restrict the market for renewable energy. Due to this, accessing RECs may be challenging for purchasers in regions with weak renewable energy markets.
3. Absence of standardization: The issuance and tracking of RECs are not yet standardized. This might make it challenging for purchasers to compare various RECs and assess how well they promote renewable energy.

The market for renewable energy is expanding quickly, and Renewable Energy Certificates (RECs) are a crucial instrument for businesses and utilities to satisfy their objectives for renewable energy and legal obligations. Unfortunately, the existing tracking and trading mechanism for RECs is convoluted and vulnerable to fraud, which might damage the market's credibility and transparency. With the help of blockchain technology and smart contracts, this project suggests using Non-Fungible Tokens (NFTs) as a more efficient, transparent, and safe way to track and exchange RECs. By doing so, the market's liquidity may improve, and transaction costs may be decreased.

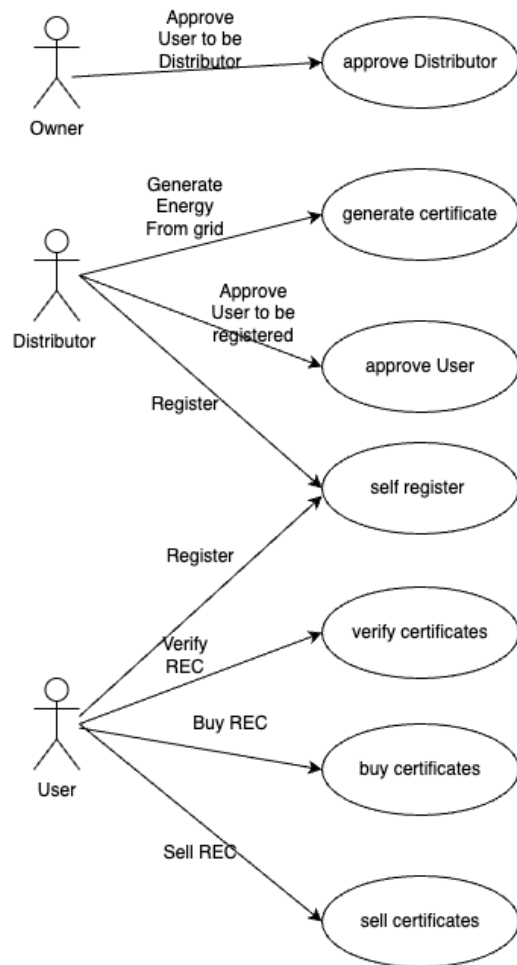
### Digital Asset, Tokenization and Reasoning:

Digital Asset: Renewable Energy Certificate (REC)

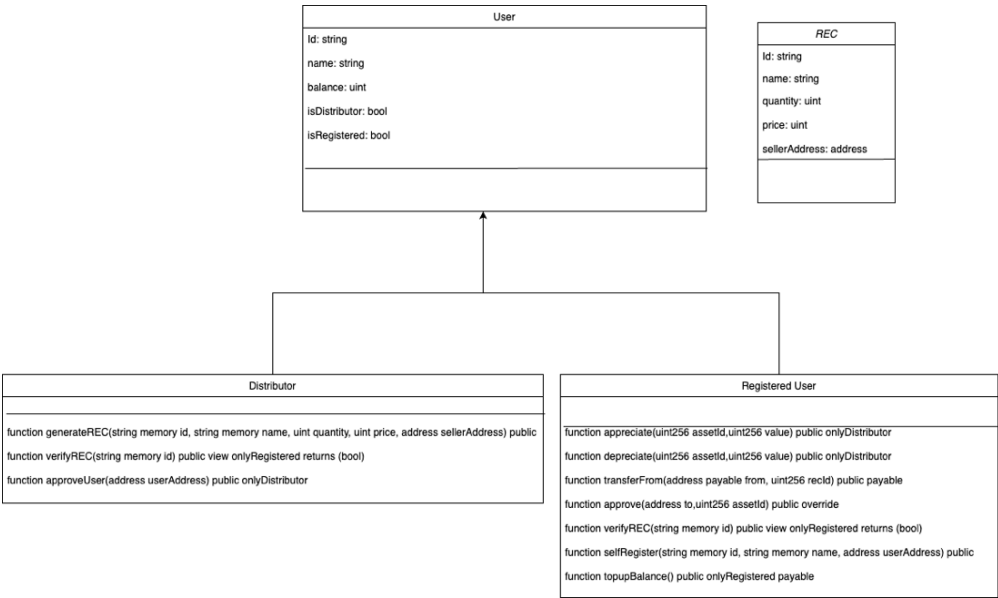
Tokenization: ERC721 ethereum token standard

Reasoning: ERC721: ERC721 is a non-fungible token standard on the Ethereum network. Unlike ERC20 tokens, each ERC721 token is unique and cannot be replaced by another token. This makes ERC721 tokens suitable for applications such as digital collectibles, gaming items, and other assets that have unique properties. ERC721 tokens are often used to represent assets that have real-world value and can be traded in digital marketplaces.

## Use Case diagram:



ER Diagram:



Contract diagram:

Renewable Energy Token
<pre> struct User struct REC mapping (address -&gt; User) users uint public recCount mapping (string -&gt; REC) recs mapping (uint =&gt; uint256) private recOwner mapping(address =&gt; uint256) private ownedAssestsCount mapping (uint256 =&gt; address) public assestApprovals address[] addressList address owner enum Status {Available, Sold} </pre>
<pre> onlyOwner onlyDistributor onlyRegistered </pre>
<pre> function getCertificate(string memory id) public view onlyRegistered returns (string memory, string memory, uint, uint, address, Status) function verifyREC(string memory id) public view onlyRegistered returns (bool) function generateREC(string memory id, string memory name, uint quantity, uint price, address sellerAddress) public onlyDistributor function appreciate(uint256 assetId,uint256 value) public onlyDistributor function depreciate(uint256 assetId,uint256 value) public onlyDistributor function transferFrom(address payable from, uint256 recId) public payable function approve(address to,uint256 assetId) public override function approveDistributor(address userAddress) public onlyOwner function approveUser(address userAddress) public onlyDistributor function selfRegister(string memory id, string memory name, address userAddress) public </pre>

## UI Wireframe:

Home page

Search

Distributor

RegisteredUser

User

List of Renewable Energy Certificates

Renwable Enerergy Certificate 1

Renwable Enerergy Certificate 2

Renwable Enerergy Certificate 3

Homepage: The main page of the DApp which will be available for an end user

Page 1

https://www.default.com

Logged in as Distributor

Create REC

id:

name:

address:

quantity:

price:

Generate REC

Approve User

Address:

Approve User

The webpage when the user is logged in as Distributor

Page 1

https://www.default.com

Logged in as Registered User

Get Certificates of User

address:

Get Certificates of user

List of Renewable Energy Certificates of the user

Renwable Engergy Certificate 1

Renwable Engergy Certificate 2

Renwable Engergy Certificate 3

Get All Certificates

Get All Certificates

List of Renewable Energy Certificates of all users

Renwable Engergy Certificate 1

Renwable Engergy Certificate 2

Renwable Engergy Certificate 3

Get Certificate

Id:

Get Certificate

Id: RECid

name: RECname

seller address: REC seller address

quantity: RECquantity

price: REC price

Verify REC

Id:

Verify REC

Sell REC

Id:

Sell REC

Buy REC

Id:

Buy REC

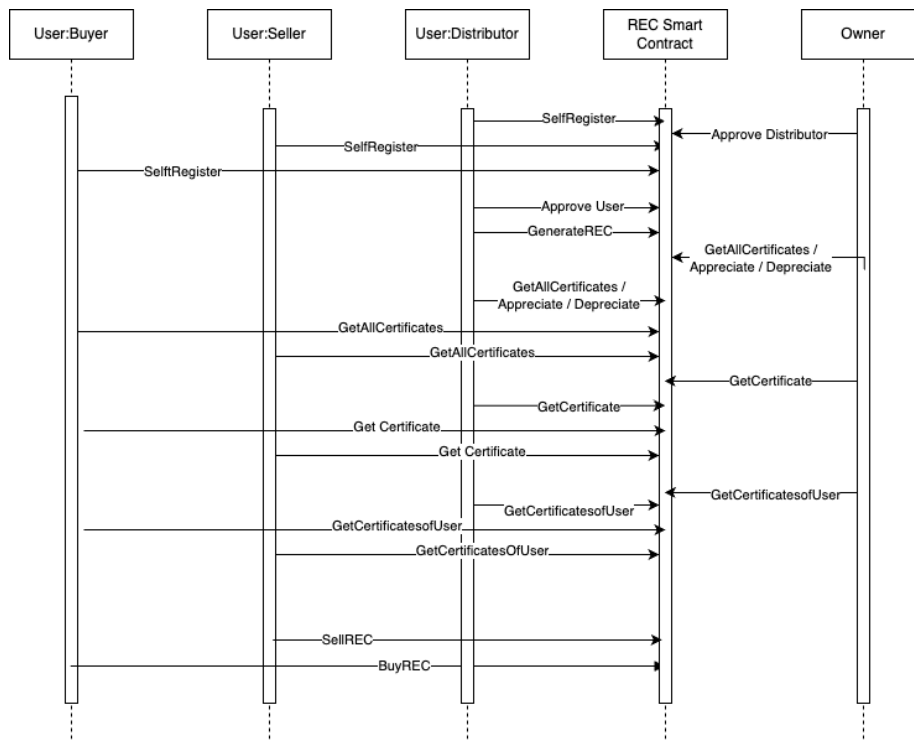
Top up Balance

Top Up

Webpage is shown as above when user is logged in as Normal user role.

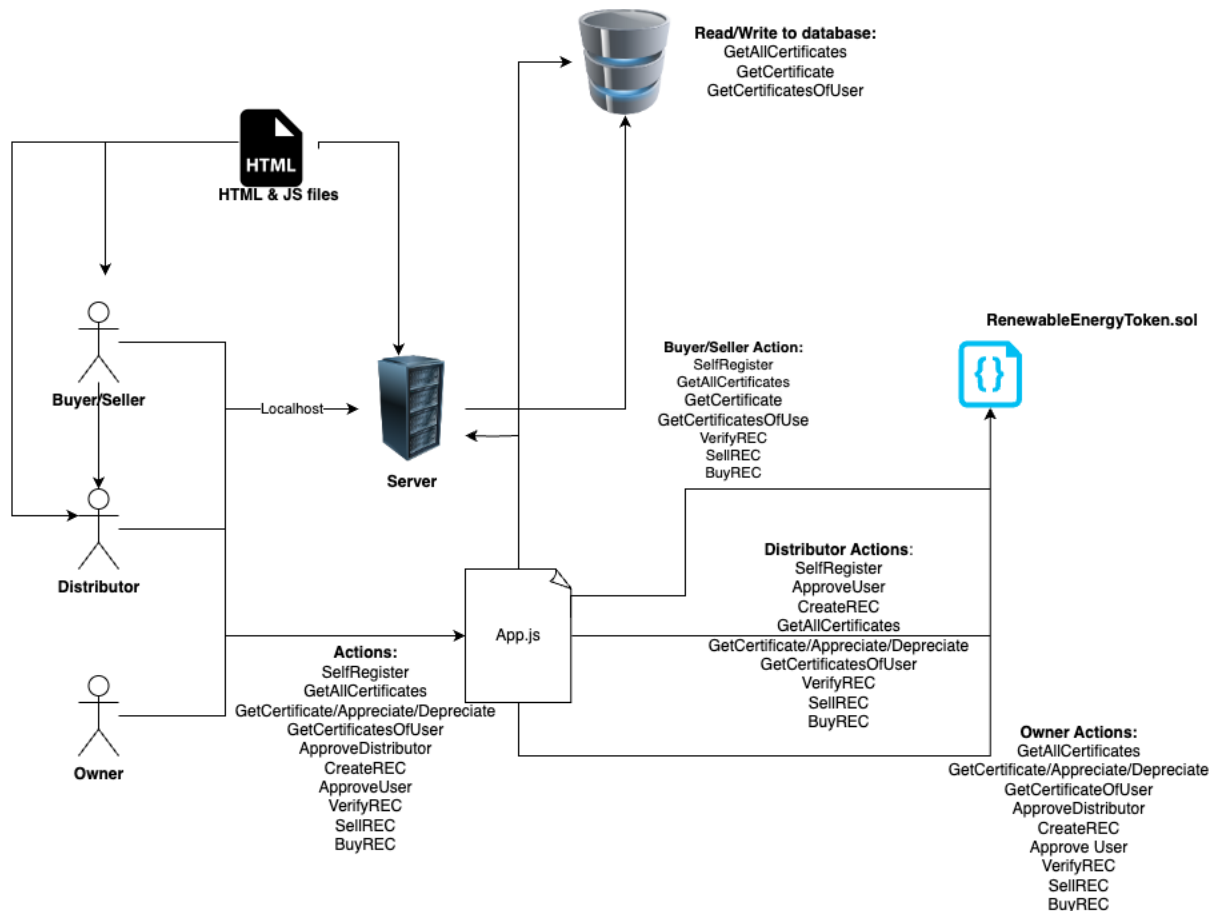
## Phase 2:

### Sequence Diagram:



1. The Owner of the smart contract deploys the contract for the users to interact with the contract.
2. The users register to interact with the contract. (SelfRegister)
3. The owner then approves a user to act as a distributor (Approve Distributor)
4. All other users are approved by the distributor to become a registered user (ApproveUser)
5. The distributor can create REC (GenerateREC)
6. Any registered user can query to get all certificates, get a particular certificate and get certificates of user. (GetAllCertificates, GetCertificate, GetCertificateofUser)
7. The registered users can verify, sell or buy tokens. (VerifyREC, SellREC, BuyREC)
8. The distributor has the permission to appreciate or depreciate the value of the tokens (Appreciate/Depreciate)
9. Here, the owner can perform all the operations that any registered user can.

## Architecture Diagram:



1. The owner deploys the smart contract and Dapp to a server (here, localhost)
2. The users will register themselves to the application with the wallet.
3. The owner approves the distributor and the distributor approved all the users who have registered
4. The actors then interact with the contract through the UI where the App.js file clues the front-end with the smart contract with the functions available for its role
5. The server code (app.js) interacts with the offchain data to fetch the more details of the users/certificates and populate the UI with the tokens

## OffChain/OnChain Data:

The Users will fetch offchain data like user details like username, certificate details like quantity, certificatename from external datasource by using identifiers fetched from blockchain like certificateId, userAddress. The external datasource contains mapping from userAddress to userDetails and certificateId to certificateDetails.

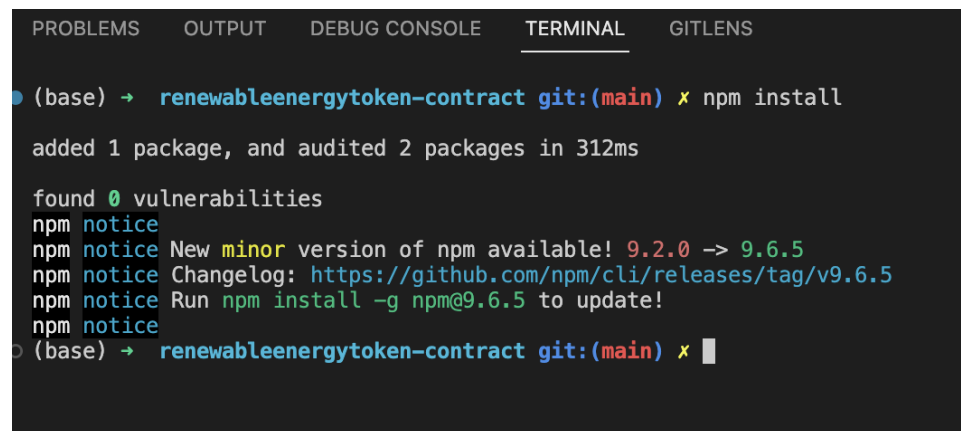


### Aprecciation/ Depreciation of Assets:

Renewable energy certificates (RECs) can appreciate or depreciate in value based on various factors, including market demand, renewable energy generation levels, regulatory changes, and public perception. Distributors can appreciate, depreciate certificate value based on these factors.

### Steps of deployment:

1. **Tools:** Install the required tools for the development of the Dapp: solidity, node, truffle, javascript, Ganache, digital wallet (ex: Metamask),
2. **Develop:** First, you need to develop and test the smart contract in remix to ensure it functions properly and developing the front-end interface and integrating any necessary third-party tools or services.
3. **Test framework:** Choose a test Blockchain test platform. Once the dapp is ready, choose Ganache test platform to deploy and test.
4. Install npm packages in both folders **renewableenergytoken-app** and **renewableenergytoken-contract** like below.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

(base) → renewableenergytoken-contract git:(main) x npm install

added 1 package, and audited 2 packages in 312ms

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 9.2.0 -> 9.6.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.5
npm notice Run npm install -g npm@9.6.5 to update!
npm notice
(base) → renewableenergytoken-contract git:(main) x
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  GITLENS

● (base) → renewableenergytoken-app git:(main) x npm install

added 58 packages, and audited 59 packages in 1s

7 packages are looking for funding
  run `npm fund` for details

1 high severity vulnerability

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New minor version of npm available! 9.2.0 -> 9.6.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.5
npm notice Run npm install -g npm@9.6.5 to update!
npm notice
○ (base) → renewableenergytoken-app git:(main) x █
```

## 5. Contract deployment by executing truffle migrate --reset in renewableenergytoken-contract folder:

```
renewableenergytoken-contract git:(main) x truffle migrate --reset

Compiling your contracts...
There was an error attempting to save the compiler to disk. The current user likely does not have sufficient permissions to write to disk in Truffle's compiler cache directory. See the errorpranted below for more information about this directory.
Error: EACCES: permission denied, open "/home/chaosim/.truffle/compilers/node_modules/solidity-compiler-wasm2-0.8.10/compiler/40d4d44.js"
There was an error attempting to save the compiler to disk. The current user likely does not have sufficient permissions to write to disk in Truffle's compiler cache directory. See the errorpranted below for more information about this directory.
Error: EACCES: permission denied, open "/home/chaosim/.truffle/compilers/node_modules/solidity-compiler-wasm2-0.8.10/compiler/40d4d44.js"
Everything is up to date, there is nothing to compile.
Fetching solc version list from solc-bin. Attempt #1
Starting migration... Attempt #1.
> Network name: 'ganache'
> Network id: 5777
> Block gas limit: 872375 (0x6d93d7)

1. Initial migration.js
  Fetching solc version list from solc-bin. Attempt #1
  Replacing 'ganache' target #1.
  > Transaction hash: 0x3fe25f626a568720968fedc3fa9027b058c7e7b0d3726c299400077c4e0
  > Block: 0
  > contract address: 0x3fe25f626a568720968fedc3fa9027b058c7e7b0d3726c299400077c4e0
  > block number: 1
  > block timestamp: 1602981476
  > account: 0x0000000000000000000000000000000000000000
  > balance: 99.999974925
  > gas used: 254076 (0x3e7317)
  > gas price: 3.275 gwei
  > value sent: 0 ETH
  > total cost: 0.000258065 ETH
  > Saving migration to chaos.im solc-bin. Attempt #1
  > Saving artifact, attempt #1.
  > Total cost: 0.000258065 ETH

2. deploy_contracts.js
  Fetching solc version list from solc-bin. Attempt #1
  Replacing 'renewableenergytoken' target #1.
  > Transaction hash: 0x0804055858b0e5a275e0bcf1b377254558233f081c43b794083773b0ec0
  > Block: 0
  > contract address: 0x0804055858b0e5a275e0bcf1b377254558233f081c43b794083773b0ec0
  > block number: 1
  > block timestamp: 1602981476
  > account: 0x0000000000000000000000000000000000000000
  > balance: 99.999986247783193
  > gas used: 464743 (0x707317)
  > gas price: 3.17949333 gwei
  > value sent: 0 ETH
  > total cost: 0.014837942542584195 ETH
  > Saving migration to chaos.im solc-bin. Attempt #1
  > Saving artifact, attempt #1.
  > Total cost: 0.014837942542584195 ETH

Summary
> Total deployments: 2
> Final cost: 0.015762949042584195 ETH
```

## 6. Running Application by executing npm run start in renewableenergytoken-app folder:

```
○ (base) → renewableenergytoken-app git:(main) x npm run start

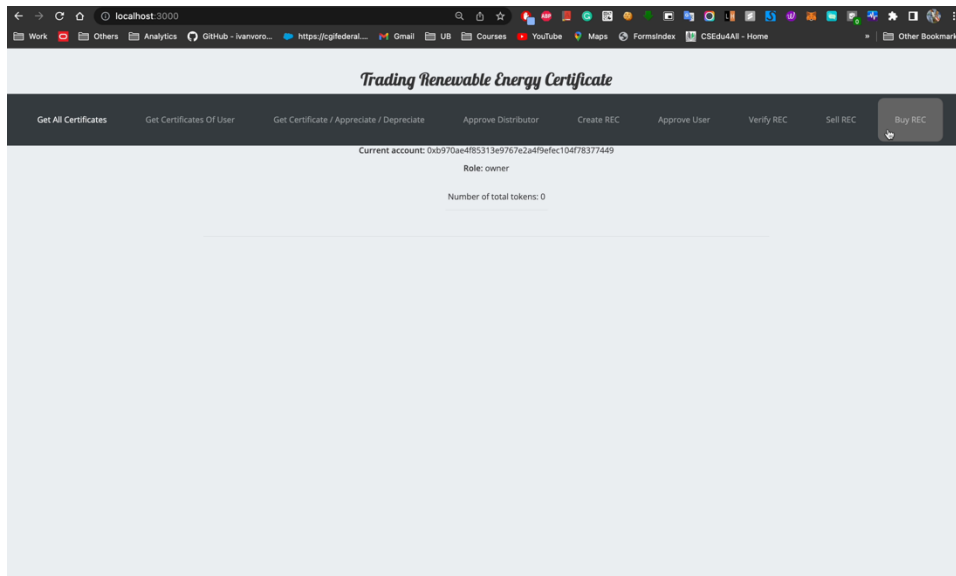
> renewableenergytoken-app@1.0.0 start
> node index.js

Renewable Energy Token Dapp listening on port 3000!
█
```

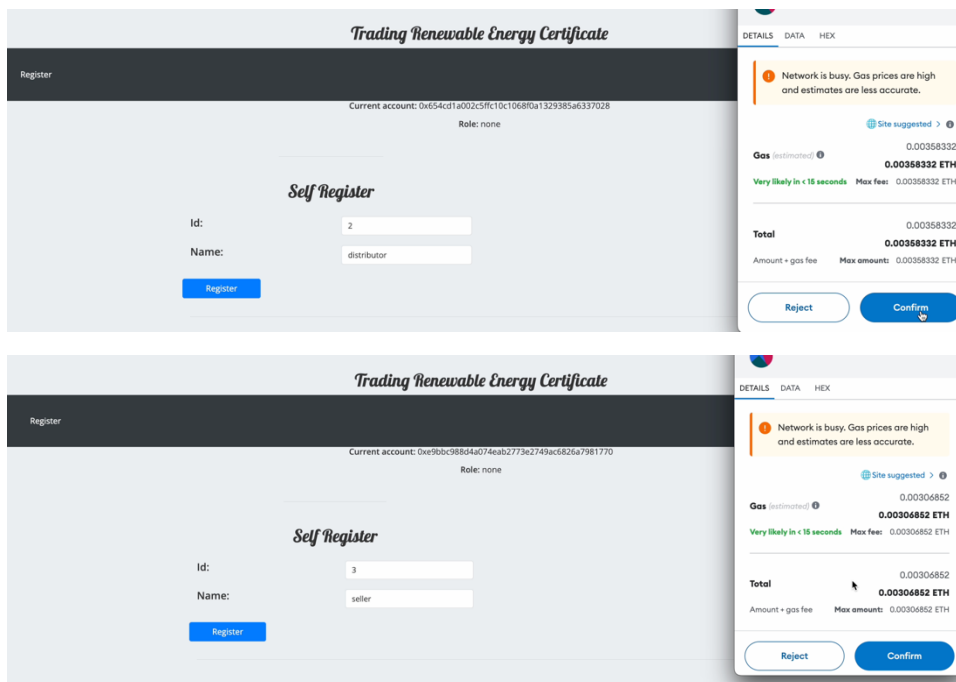
## Working:

We have attached the working screenshots attached below.

Initial view as viewed by the Owner:



Self register Flow for Distributor, Seller, Buyer:



Trading Renewable Energy Certificate

Register

Current account: 0xa774f0db496323fef7d80971885b0a53018fb97

Role: none

Self Register

Id:

4

Name:

buyer

Register

DETAILS DATA HEX

1

Network is busy. Gas prices are high and estimates are less accurate.

Site suggested >

Gas (estimated)

0.00306816

Very likely in < 15 seconds

Max fee: 0.00306816 ETH

Total

0.00306816

Amount + gas fee

Max amount: 0.00306816 ETH

Reject

Confirm

After all users self register, request is sent to owner, distributor for approval. All users will not be able to perform any actions until owner or distributor approves their request. They will see screen as below.

Trading Renewable Energy Certificate

Current account: 0xe9bbc988d4a074eab2773e2749ac6826a7981770

Role: not yet verified

Owner Approving Distributor:

Trading Renewable Energy Certificate

Get All Certificates

Get Certificates Of User

Get Certificate / Appreciate / Depreciate

Approve Distributor

Create REC

Approve User

Current account: 0xb970ae4f85313e9767e2a4f9efec104f78377449

Role: owner

Approve Distributor

Address:

0x654cd1A002c5fC10C1068

Approve Distributor

DETAILS DATA HEX

1

Network is busy. Gas prices are high and estimates are less accurate.

Site suggested >

Gas (estimated)

0.00154064

Very likely in < 15 seconds

Max fee: 0.00154064 ETH

Total

0.00154064

Amount + gas fee

Max amount: 0.00154064 ETH

Reject

Confirm

Distributor approving Seller and buyer:

Trading Renewable Energy Certificate

Get All Certificates

Get Certificates Of User

Get Certificate / Appreciate / Depreciate

Create REC

Approve User

Verify REC

Sell REC

Current account: 0x654cd1a002c5fC10C10680a1329385ae6337028

Role: distributor

Approve User

Address:

0xE9b8c988d4A074eAB2773

Approve User

DETAILS DATA HEX

1

Network is busy. Gas prices are high and estimates are less accurate.

Site suggested >

Gas (estimated)

0.00163628

Very likely in < 15 seconds

Max fee: 0.00163628 ETH

Total

0.00163628

Amount + gas fee

Max amount: 0.00163628 ETH

Reject

Confirm

Distributor will be able to generate Renewable energy certificate:

Trading Renewable Energy Certificate

Get All Certificates

Get Certificates Of User

Get Certificate / Appreciate / Depreciate

Create REC

Approve User

Verify REC

Sell REC

Current account: 0x54cd1a002c3f10c10680a1329385a6337028

Role: distributor

Create REC

Name:

NFT 0

Address:

0xE9bcC988d4A074eAb2773E27

Quantity:

1

Price:

10

Generate REC

DETAILS DATA HEX

Network is busy. Gas prices are high and estimates are less accurate.

Site suggested

Gas estimated

0.00508626

0.00508626 ETH

Very likely in < 15 seconds

Max fee: 0.00508626 ETH

Total

10.00508626

10.00508626 ETH

Amount + gas fee

Max amount: 10.00508626 ETH

Reject

Confirm

Seller view of Get certificates of user where he can see the certificates of him assigned by the distributor. Certificates contain information about asset name, id, owner and approved buyer addresses.

Trading Renewable Energy Certificate

Get All Certificates

Get Certificates Of User

Get Certificate

Verify REC

Sell REC

Buy REC

Current account: 0xe9bbc988d4a074eab2773e2749ac6826a7981770

Role: registered

Number of token owned by the current account: 2

NFT 0

Asset # 0

Price: 10 ETH

Owner: 0xe9bbc988d4A074eAb2773E27

49ac6826A7981770

Approved: None

NFT 1

Asset # 1

Price: 20 ETH

Owner: 0xe9bbc988d4A074eAb2773E27

49ac6826A7981770

Approved: None

By using Sell Renewable Energy Certificate functionality Seller will be able to make Buyer approved Buyer to Asset 0 as shown below.

Trading Renewable Energy Certificate

Get All Certificates

Get Certificates Of User

Get Certificate

Verify REC

Sell REC

Buy REC

Current account: 0xe9bbc988d4a074eab2773e2749ac6826a7981770

Role: registered

Sell REC

To:

0xA774f0Db496323eF7D809

Id:

0

Sell REC

Allow access to and transfer of your RenewableEnergyToken (#0)?

This allows a third party to access and transfer the following NFTs without further notice until you revoke its access.

Verify contract details

Site suggested

Gas estimated

0.02

0.02 ETH

Very likely in < 15 seconds

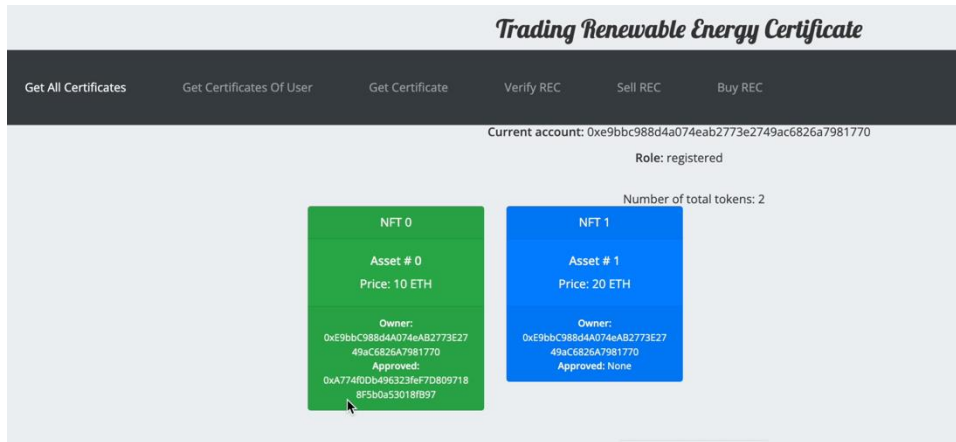
Max fee: 0.02 ETH

View full transaction details

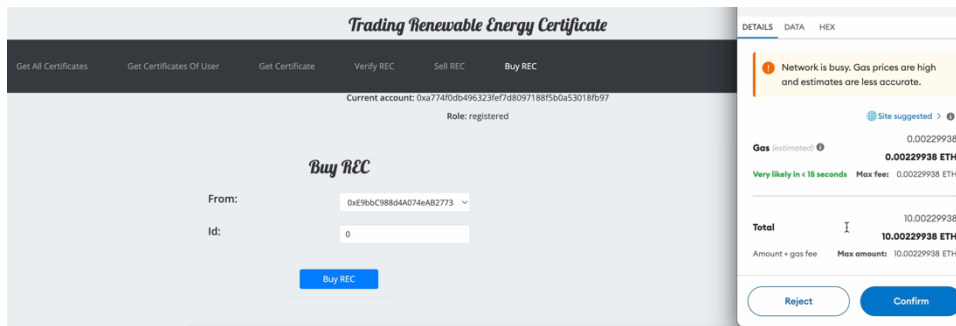
Reject

Confirm

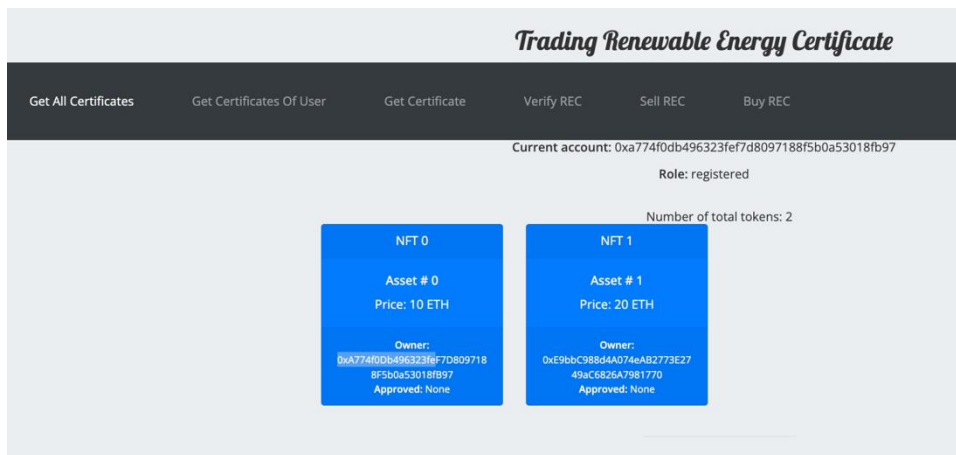
After above transaction, buyer can buy asset 0 as buyer address is approved by seller of asset 0.



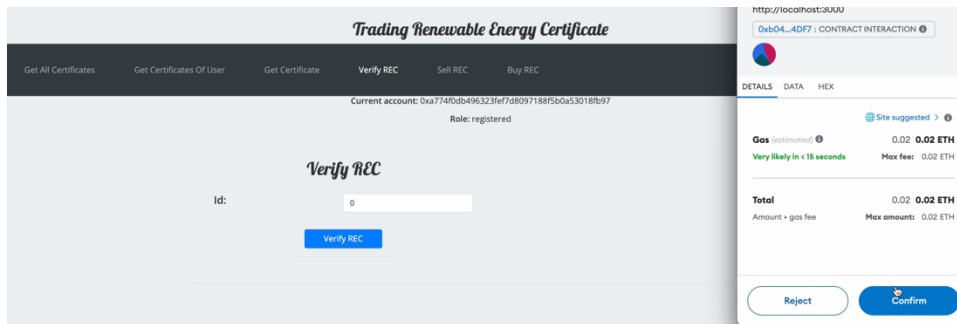
Buyer can use Buy Renewable Energy Certificate page to buy certificate from seller as shown below.



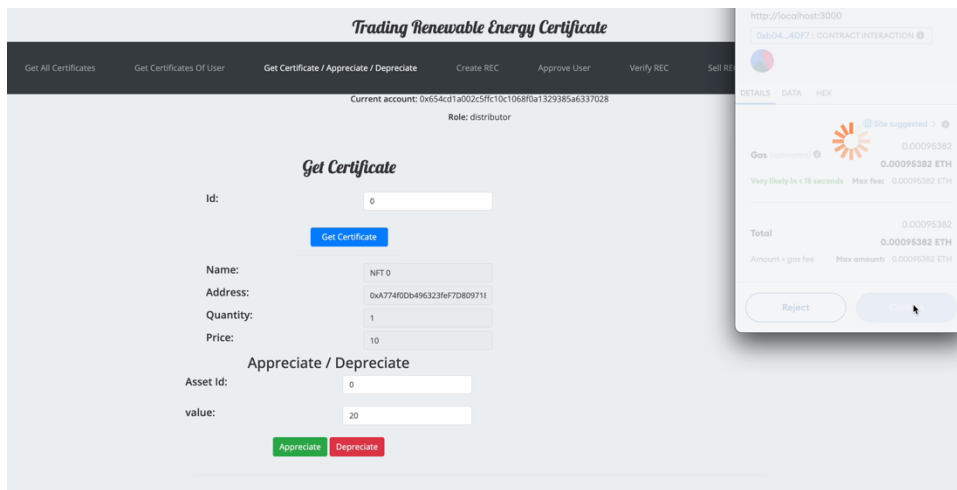
After Buyer buys Asset 0, owner id of asset 0 will be changed to address of buyer as shown below.



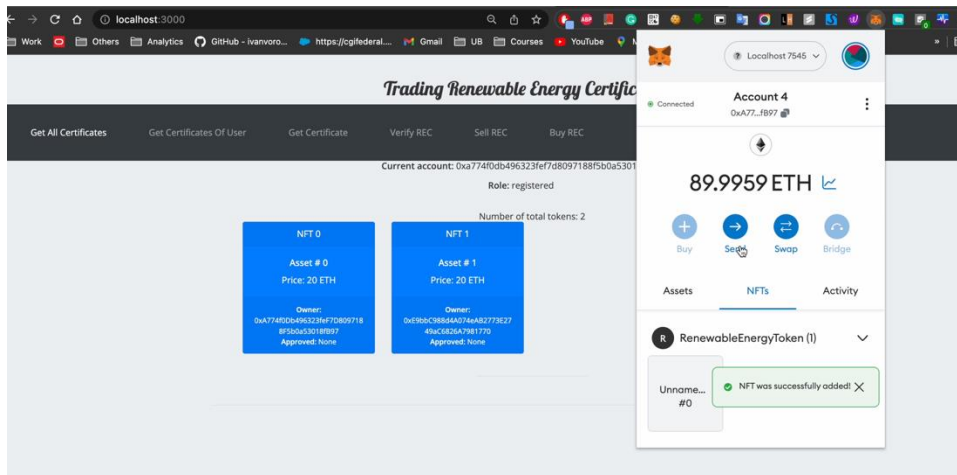
Buyer can verify certificate by using very certificate page.



Distributor can appreciate/ depreciate value of asset by using appreciate depreciate certificate page.



Screenshot of NFT in Metamask wallet:



## Phase 3:

### Contract and Dapp deployment:

We will be using truffle using in the previous phase to deploy the contract,

(We make sure to have truffle installed properly and have enough ethers in the digital wallet to deploy the contract)

### Steps to deploy the Smart Contract on Infura (blockchain as a node service):

1. We create an Ethereum project and get a API Key on Infura
2. We install hdwallet-provider  
Run the following command to install hdwallet-provider:  
**npm install @truffle/hdwallet-provider**

3. Updated **truffle-config.js** file in the **renewableenergytoken-contract** folder
  - a. We add an import and the following constants to the file

```
const HDWalletProvider = require('@truffle/hdwallet-provider');  
  
INFURA_API_KEY = "https://sepolia.infura.io/v3/< API-Key>"  
MNEMONIC = "<The-MetaMask-Secret-Recovery-Phrase>"
```

- b. We add the network to where we deploy, here we deployed on Sepolia network

```
module.exports = {  
  networks: {  
    development: {  
      host: "127.0.0.1",  
      port: 8545,  
      network_id: "*"  
    },  
    sepolia: {  
      provider: () => new HDWalletProvider(mnemonic,  
        `https://sepolia.infura.io/v3/${infuraKey}`),  
      network_id: 11155111,      // sepolia's network id  
      confirmations: 2,  
      timeoutBlocks: 200,  
      skipDryRun: true  
    }  
  }  
};
```

4. Compile and deploying the smart contract:
  - a. Compiling the smart contract:



Run **truffle compile** command from the **renewableenergytoken-contract** folder

- b. Deploying the smart contract:

Execute **truffle migrate --network sepolia** command to deploy the smart contract on the blockchain

```
> total cost:          0.000679410001902348 ETH

Pausing for 2 confirmations...

=====
> confirmation number: 1 (block: 3454050)Attempt #1
> confirmation number: 2 (block: 3454051)Attempt #1
> Saving migration to chain.om solc-bin. Attempt #1
> Saving artifactsr. Attempt #1.
=====
> Total cost:          0.000679410001902348 ETH

:: Fetching solc version list from solc-bin. Attempt #1
2_deploy_contracts.jsr. Attempt #1.
=====
:: Fetching solc version list from solc-bin. Attempt #1
Replacing 'RenewableEnergyToken'

=====
> transaction hash:    0x3ae50d90dbde1934a8bc4b5b7c0b03923326651fd8b937c743503af3df268585
> Blocks: 0           Seconds: 81c-bin. Attempt #1
> contract address:   0xE95B50500a0E6BEB53Aa0CC5ACecc51C054bbc91
> block number:       3454054: 8
> block timestamp:    1683682980
> account:            0x1261Daa4EB46aC67A9AF5cA626e7F63cc9A6D90d
> balance:            1.369296468963271356
> gas used:           4750896 (0x487e30)
> gas price:          2.500000007 gwei
> value sent:         0 ETH
> total cost:         0.011877240033256272 ETH

Pausing for 2 confirmations...

=====
> confirmation number: 1 (block: 3454055)Attempt #1
> confirmation number: 2 (block: 3454056)Attempt #1
> Saving migration to chain.om solc-bin. Attempt #1
> Saving artifactsr. Attempt #1.
=====
> Total cost:          0.011877240033256272 ETH

Summary
=====
> Total deployments:   2
> Final cost:          0.01255665003515862 ETH
```

We get the **deployed smart contract address** in the output when the command executed successfully.

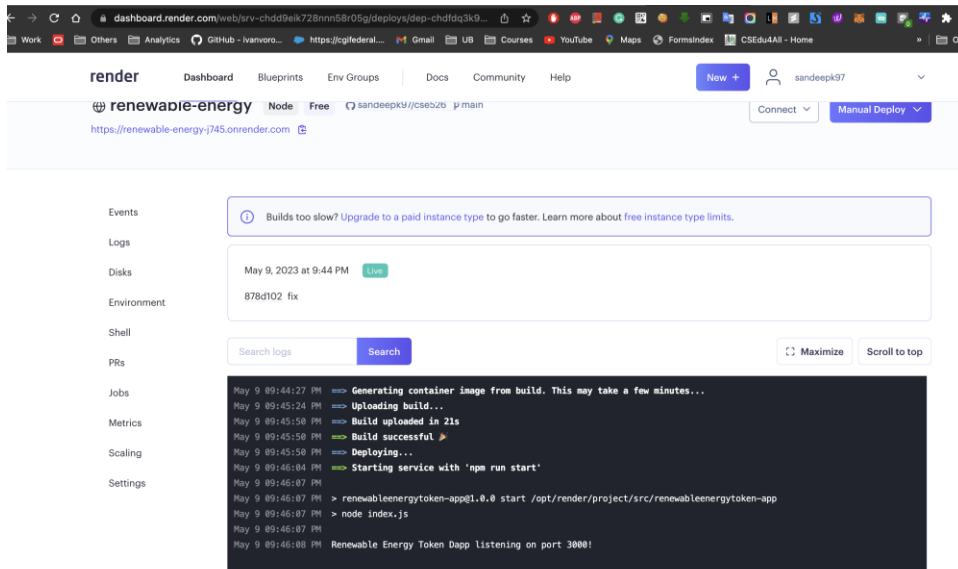
5. We update **app.js** file in the **renewableenergytoken-app** folder  
The app parameter **url**, **address** are to be updated with the smart contract address and url is set to the network endpoint

```
address: '<smart contract address>',
url: 'https://sepolia.infura.io/v3/<infura-key>',
```

### Steps to deploy the Dapp on Render (Cloud application):

1. Create a Render account: First, you need to create a Render account by going to <https://dashboard.render.com/signup>. Once you create an account, log in to Render.

2. Create a new service: In the Render dashboard, click on the "Create a New Service" button. Choose the appropriate language or framework that you want to use to create the web service.
3. Configure the service: After selecting language or framework, need to configure the service. Provide a name for the service and choose the deployment environment. Specify the amount of CPU, memory, and disk space you want to allocate to the service.
4. Set up the code: Next, you need to set up the code. You can either upload the code or connect to a Git repository. If you're uploading the code, you can do so by dragging and dropping the files into the Render dashboard. If you're connecting to a Git repository, you can provide the repository URL and the branch you want to deploy.
5. Deploy the service: Once you've configured everything, you can deploy the service by clicking on the "Create Service" button. Render will start building and deploying the service.
6. Test the service: Once the service is deployed, test it by accessing the URL provided in the Render dashboard. You can also use the Render Logs to see any errors or issues with the service.



**The address of the Renewable Energy Token Application - [Renewable Energy Token Dapp](https://renewable-energy-j745.onrender.com)**

## References:

1. "Blockchain In Action" Textbook –
  - a. Chapters 1 - 4 for Smart Contract
  - b. Chapters 5 – 9 for Decentralized Application Development
2. [Renewable Energy Certificates \(RECs\) | US UPA](#)

3. [ERC721 vs ERC1155](#)
4. [Openzeppelin ERC721](#)

**Project Idea Approved By:**

Baibhav Thapa ([baibhavg@buffalo.edu](mailto:baibhavg@buffalo.edu))