

Limited Edition Store

First Name: Sai Kumar
Last Name: Madhadi
Email: smadhadi@buffalo.edu
UB ID: 50419505

First Name: Rajesh Reddy
Last Name: Mekala
Email: rajeshre@buffalo.edu
UB ID: 50415674

Issue addressed:

Limited edition products are uniquely branded items that are created in small quantities and only sold for a certain time period in the market. People buy these to highlight their own individuality and social status by owning an exclusive item. There are numerous companies around the world who sell these items like Bugatti, Rolex, etc. However, there is no track of who owns these limited edition items. This decentralized application assists companies in selling their limited edition items to consumers and also helps to keep track of the proud owners of these items for the benefit of companies and future generations.

Abstract:

This project's goal is to enable consumers to purchase limited edition products using a token. We are going to take the product details like product name, cost of the product, and the number of items manufactured from the companies that are registered on the application. Users can buy these limited edition items and the tokens will be credited to the company as per the cost. As the limited edition items are rare and product manufacturing ceases after a certain time. This application can be used to keep track of those owners for both companies and future generations.

Symbol:

LEC

LimitedEditionCoin

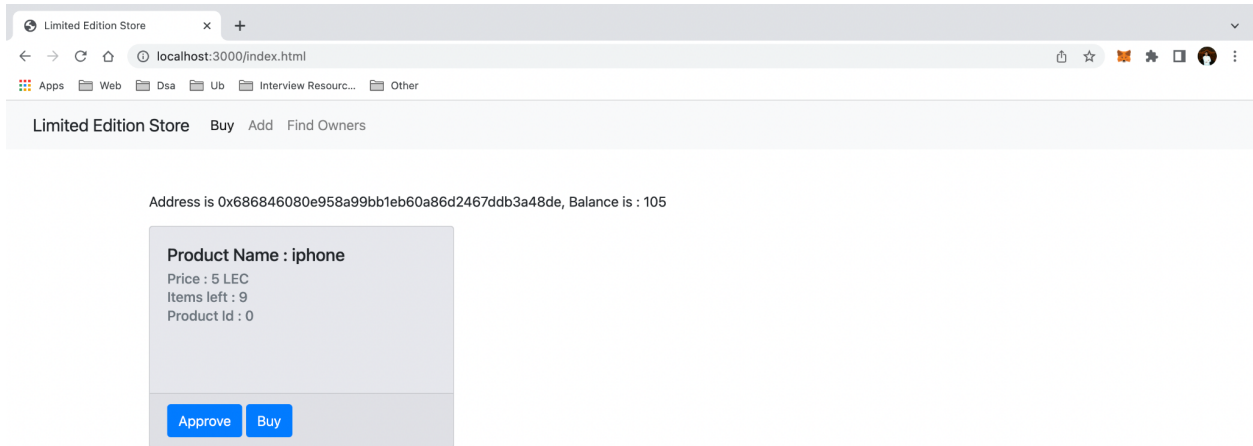
Instructions to deploy, test, and interact:

1. We have to use Infura to deploy our contract on a ropsten test network using truffle.
2. For that we need to create an account on infura, set up a project, use the project key and the deployer mnemonic and add it in the "les-Dapp/les-contract/migrations/2_deploy_contracts.js" file at the specified position.
3. Install truffle-hd-wallet provider using the command
npm install --save truffle-hdwallet-provider
4. In the "les-Dapp/les-contract" folder, use the command prompt and enter "truffle compile".
5. After compiling without any errors, enter "truffle deploy --network ropsten --reset". (use sudo if needed).
6. Copy the LimitedEditionStore contract address and paste it into the App.address property in the les-Dapp/les-app/src/js/app.js file.
7. In the "les-Dapp/les-app/" folder, enter "npm install" to download node modules.
8. Now in the "les-Dapp/les-app" folder, enter "npm start".
9. The server will start on port 3000. You should be able to access it in the browser using localhost:3000.
10. To import tokens open metamask in the browser, under assets click import tokens and enter the ERC20 contract address. This will fetch the tokens.

The marketplace has 3 pages, as shown in the screenshots below.

User Interface of the marketplace:

Page1: To Buy Limited Edition Products.



Page 2: For Companies to add new products / For Admin to Register & Unregister Companies / For Admin to transfer tokens to users.

Limited edition store

localhost:3000/add.html

AppsWebDsaUbInterview Resourc...Other

Limited Edition StoreBuyAddFind Owners

Address is 0x686846080e958a99bb1eb60a86d2467ddb3a48de, Balance is : 105

Add Product

Name

Price(in LEC)

No. of Units Manufactured

Add Product

Register Company

Name

Address

Register

Unregister Company

Address

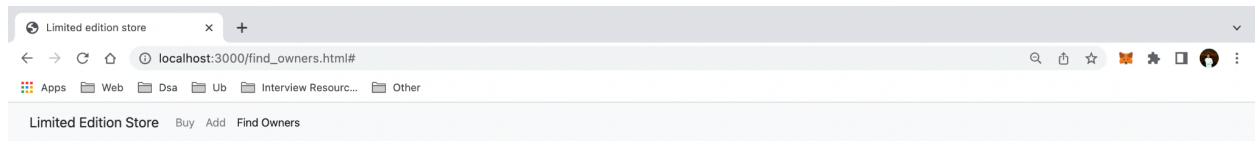
Unregister

Send Tokens

Address

Send Tokens

Page 3: To find owners of limited edition products.



▼

LIMITEDEDITIONSTORE AT 0XD8B...3

×

addProduct	bytes32 productName, uint	▼
approve	uint256 price	▼
buyProduct	uint256 productId1	▼
registerCompa...	bytes32 companyName, ac	▼
sendTokens	address address1	▼
unRegisterCo...	address companyAddress	▼
getBalance	address user	▼
getOwners	uint256 productId2	▼
getProducts		

Deployed Contracts

▼

ERC20MYN AT 0XD91...39138 (MEMC

×

approve

address owner, address sp

▼

close

transfer

address receiver, uint256 n

▼

transferFrom

address user, address com

▼

allowance

address owner, address sp

▼

balanceOf

address tokenOwner

▼

decimals

name

symbol

totalSupply

About Marketplace:

Limited edition store is a marketplace for companies to add(sell) their limited edition products and for users to buy them using LEC tokens. Marketplace has a special feature of finding the proud owners of these limited edition products. The process begins with admin registering the companies in the marketplace using company name and company address. Admin can also unregister the companies. Upon registration companies can add their products with parameters like product name, price(in LEC tokens), and number of units manufactured. Now users can see the products and buy them. The cost of the product is directly transferred from the user to the companies. Admin can transfer LEC tokens to users as needed. The future generations and companies can check the proud owners of a product by using productId. This can also prevent existent fraud of selling wrong products(claiming to be original) to users, as the companies are only registered by the admin.

Data structures used in the marketplace:

1. struct Company{}: This is used to store the details of the company like companyId, companyName, isActive, companyAddress (160-bit address).
2. struct Product{}: This is used to store the details of a product like a productId, productName, noOfUnitsManufactured, price, and companyId(which added the product into the marketplace).
3. Company[] companies: This Array of Company structure is used to store the details of all the companies.
4. Product[] products: This Array of Product structure is used to store the details of all the products.
5. mapping(address => Company): This is used to get the details of the company by address.
6. mapping(uint => address[]) owners: This is used to store the details of owners of the products by productId.
7. uint productId: This variable is used to assign productId to products by incrementing it.
8. uint companyId: This variable is used to assign companyId to companies by incrementing it.
9. address contract_addr: This is used to store the marketplace contract address.
10. address ERC20contract_Address: This is used to store the ERC20 contract address.

Modifiers used in the marketplace:

1. onlyAdmin: This modifier is used to restrict certain functions to be invoked only by the admin.
2. onlyCompanies: This modifier is used to restrict certain functions to be invoked only by the companies.

Functions used in the marketplace:

1. registerCompany(bytes32 companyName, address payable companyAddress): This function is used to register companies and is only accessible to admin. It takes the company name and company address as parameters. Companies are sent 100 tokens on registration by using transfer() method from ERC20 contract.
2. unRegisterCompany(address payable companyAddress): This function is used to unregister companies and is only accessible to admin. It takes the company address as a parameter.
3. sendTokens(address payable address1): This function is used to send tokens to users using transfer() method of ERC20 contract. It takes address as a parameter and only the admin can send tokens.
4. getBalance(address user): This is a view, which calls the balanceOf() method of ERC20 contract to find the balance of an account. This takes the address of the account as a parameter.
5. approve(uint price): This function calls the approve() method of ERC20 contract to allow the contract to spend tokens of worth 'price' from the user's account.
6. addProduct(bytes32 productName, uint noOfUnitsManufactured, uint price): This function is used to add products to the market place and is only accessible to companies. It takes the product name, the number of units manufactured, and the price of the product as parameters.
7. buyProduct(uint productId1): This function is used to buy products from the marketplace by users. It takes the productId of the product as a parameter. It calls transferFrom() method of the ERC20 contract, to transfer the tokens from user account to company account.
8. getOwners(uint productId2): This is a view, it used to find the owners of limited edition products by entering the productId.
9. getProducts(): This is a View, it is used to return the product details to UI.

About ERC20:

Functions used in the ERC20:

1. balanceOf(address tokenOwner) : This a view. This function is used to get the balance of a given address and return it.
2. approve(address owner, address spender, uint numTokens) : This function is used by the owner of the tokens to give the spender the permission to spend `numTokens` amount of tokens for any transactions.
3. transfer(address receiver, uint numTokens) : This function is used to transfer the `numToken` amount of tokens from the ERC contract deployer to the `receiver` passed in parameter.
4. transferFrom(address sender, address receiver, uint numTokens) : This function is used to transfer `numTokens` amount of tokens from one address (sender) to another address (receiver).

5. allowance(address owner, address spender) : This function is used to check the amount of tokens which the spender is allowed to spend from the owner's account on behalf of him.

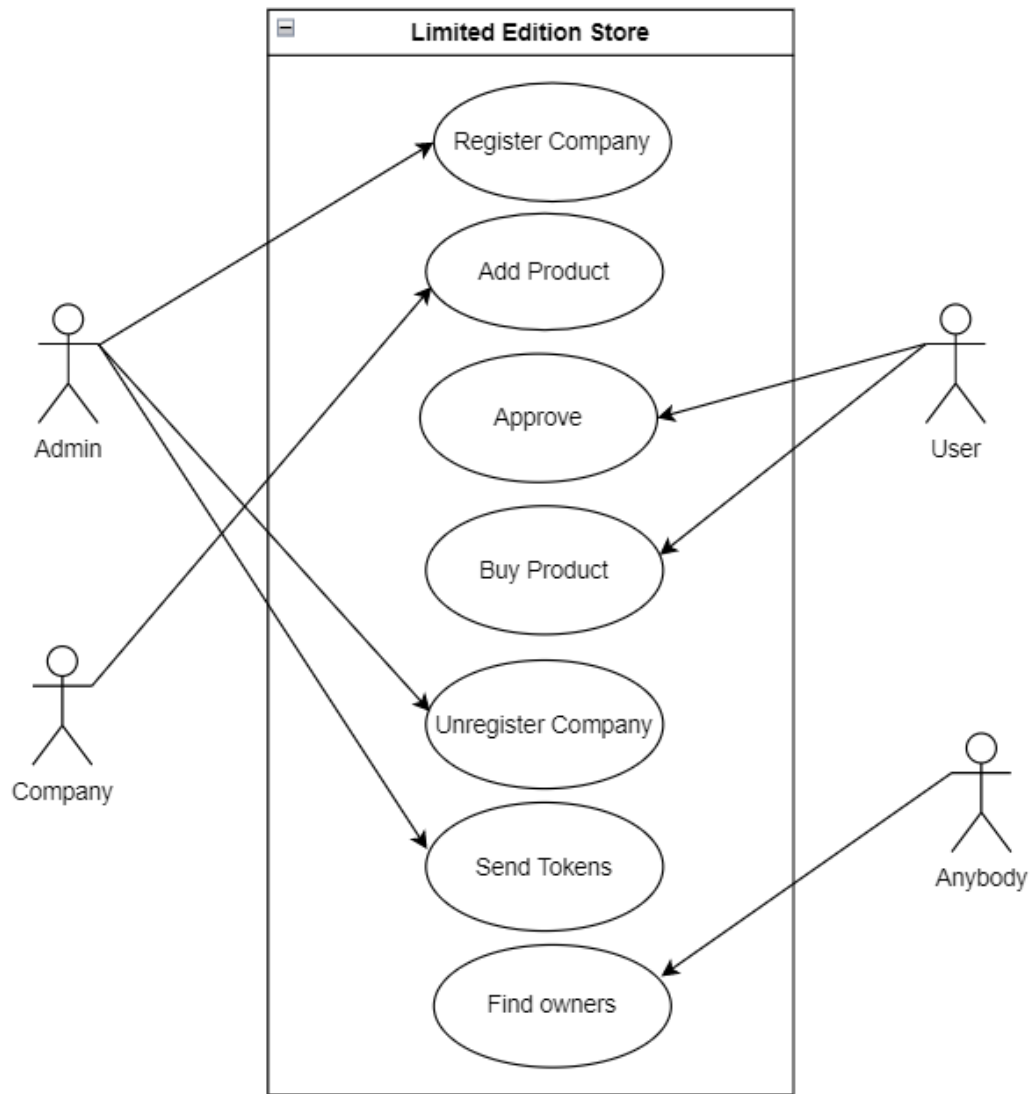
ERC20 contract connection with marketplace contract:

Since we are using two contracts now, we need to connect both of them for completing the transactions. So First we deploy the ERC20 smart contract, After deploying it we will get its contract address, Now we pass this contract address in the constructor to our marketplace contract which is LimitedEditionStore. In 2_deploy_contracts.js we achieve this task by the following code. After this we can use the functionality of the ERC20 contract from the LimitedEditionStore contract.


```
var les = artifacts.require("LimitedEditionStore");
var erc = artifacts.require("ERC20MYN");

module.exports = async function(deployer) {
    await deployer.deploy(erc, "10000");
    const erc_contract = await erc.deployed();
    await deployer.deploy(les, erc_contract.address);
}
```

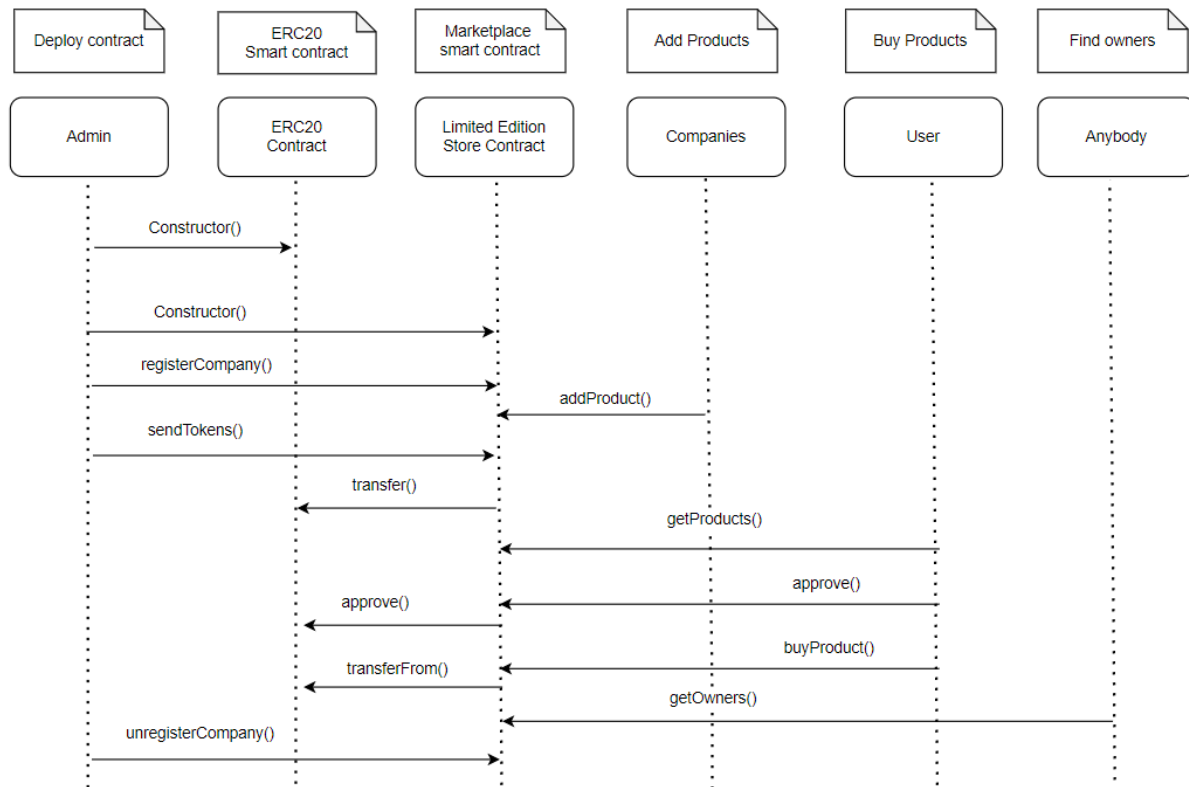
Use case diagram:



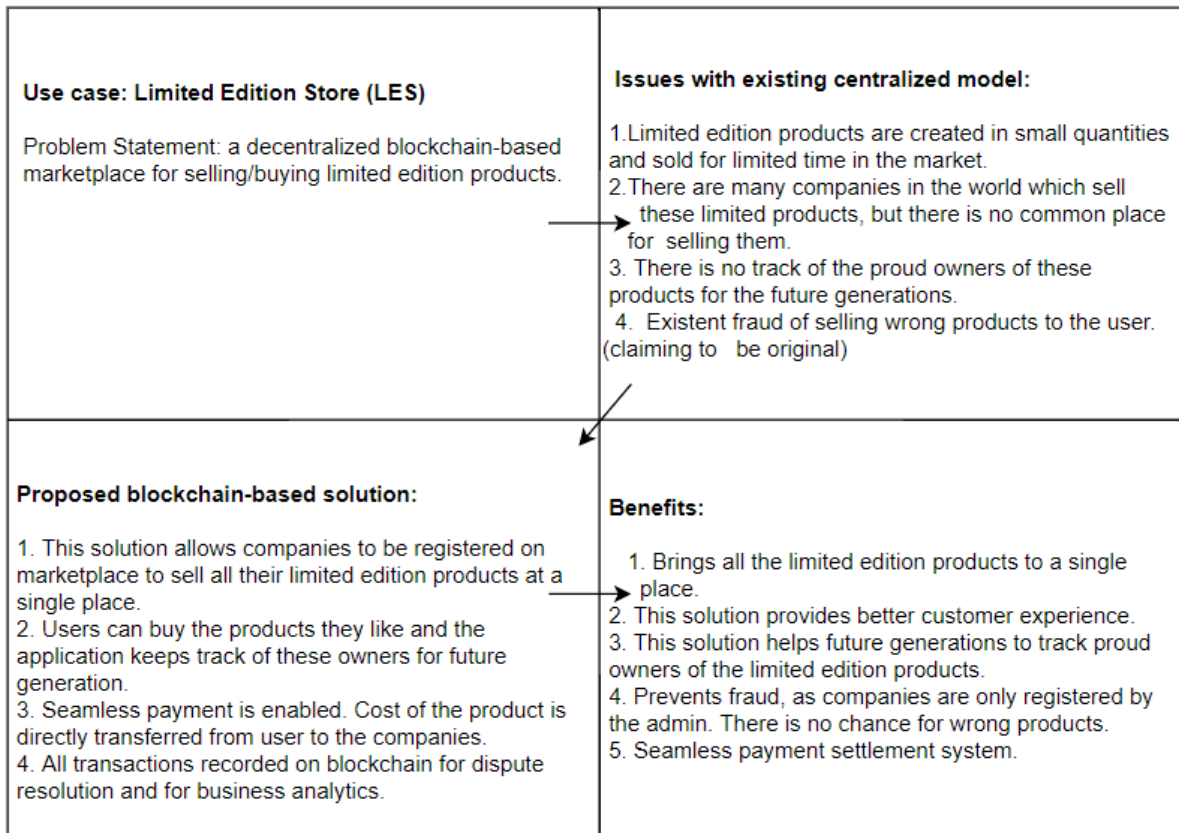
Contract Diagram:

 Limited Edition Store
<pre>Struct Product{} Struct Company{} address admin; address ERC20contract_Address; address contract_addr; mapping(address => Company) addressToCompanyMap; mapping(uint => address[]) owners; Product[] products; Company[] companies; uint productId = 0; uint companyId = 0;</pre>
<pre>modifier onlyAdmin(){ require(msg.sender == admin); _; } modifier onlyCompanies(){ require(addressToCompanyMap[msg.sender].isActive== true); _; }</pre>
<pre>function registerCompany(bytes32 companyName,address payable companyAddress) public onlyAdmin function unRegisterCompany(address companyAddress) public onlyAdmin function addProduct(bytes32 productName,uint noOfUnitsManufactured, uint price) public onlyCompanies function buyProduct(uint productId1) public payable function getOwners(uint productId2) view public returns(address[] memory) function getProducts() view public returns(Product[] memory) function sendTokens(address payable address1) payable public onlyAdmin function getBalance(address user) view public returns (uint) function approve(uint price) public</pre>

Sequence Diagram:



Quad chart Diagram:



Architectural Diagram Showing interactions between LimitedEditionStore Dapp, LimitedEditionStore Smart Contract, ERC20 Smart Contract:

