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 coins 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

Limited Edition Coin

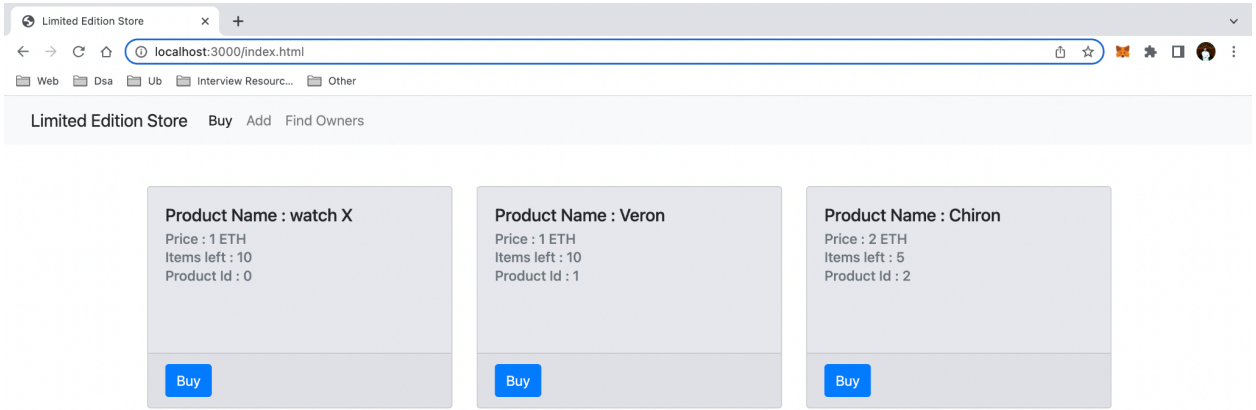
Instructions to deploy, test, and interact:

1. Open Ganache, and click “quick start”. Copy the mnemonic.
2. In the browser, open metamask and click on “ import using Secret Recovery Phrase”. Use the mnemonic and set the new password, and add the accounts.
3. Now in the “les-Dapp/les-contract” folder, use the command prompt and enter “truffle compile”.
4. After compiling without any errors, now enter “truffle migrate –reset”. (use sudo if needed).
5. Copy the contract address and paste it into the App.address property in the les-Dapp/les-app/src/js/app.js file.
6. In the “les-Dapp/les-app/” folder, enter “npm install” to download node modules.
7. Now in the “les-Dapp/les-app” folder, enter “npm start”.
8. The server will start on port 3000. You should be able to access it in the browser using localhost:3000.

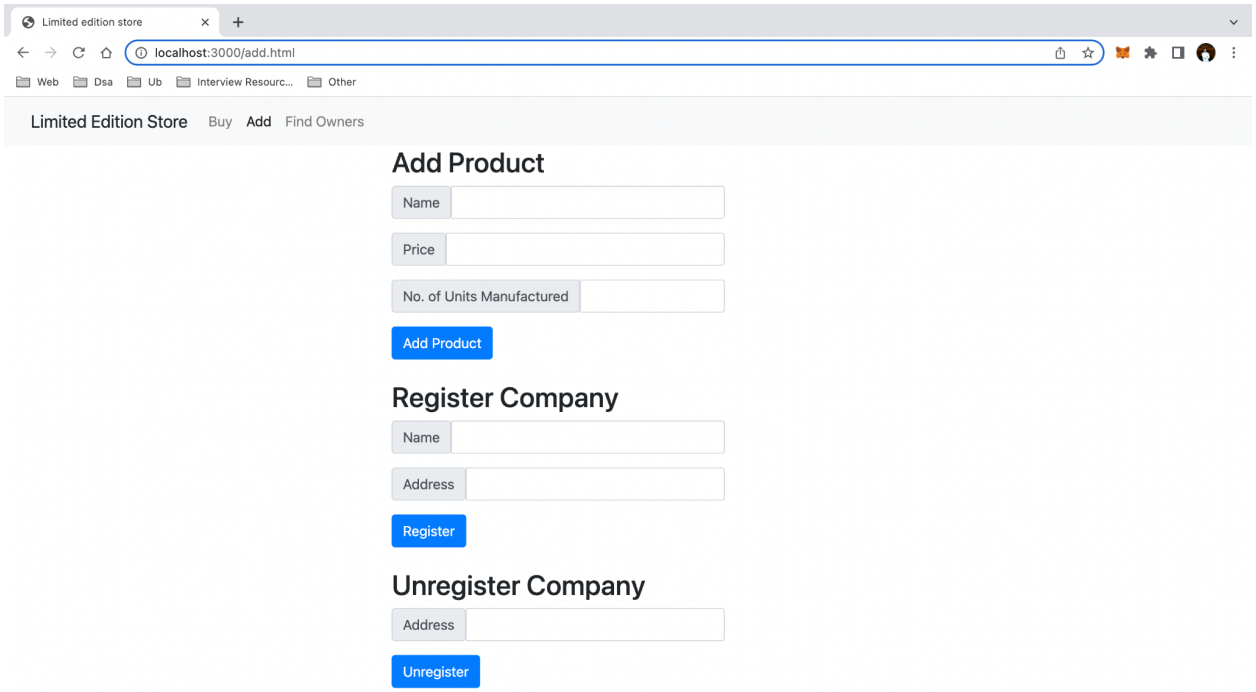
The marketplace has 3 pages, as shown in the screenshots below.

User Interface of the marketplace:

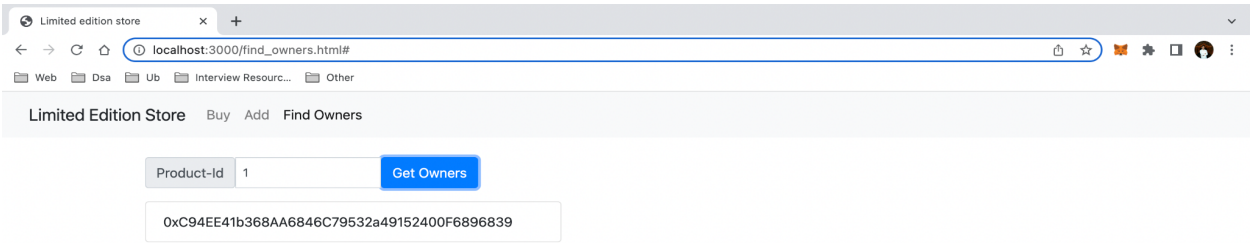
Page1: To Buy Limited Edition Products

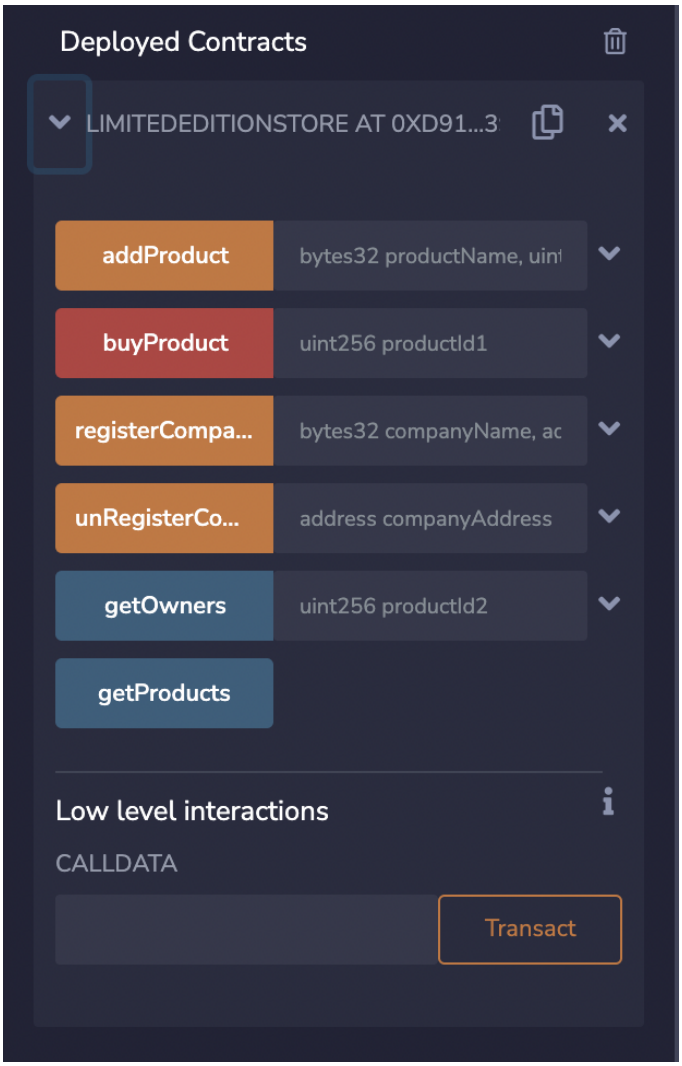


Page 2: For Companies to add new products / For Admin to Register & Unregister Companies.



Page 3: To find owners of limited edition products.





About Marketplace:

Limited edition store is a marketplace for companies to add(sell) their limited edition products and for users to buy them. 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, 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. 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, noOfUnitsManfactured, 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.

Modifiers used in the marketplace:

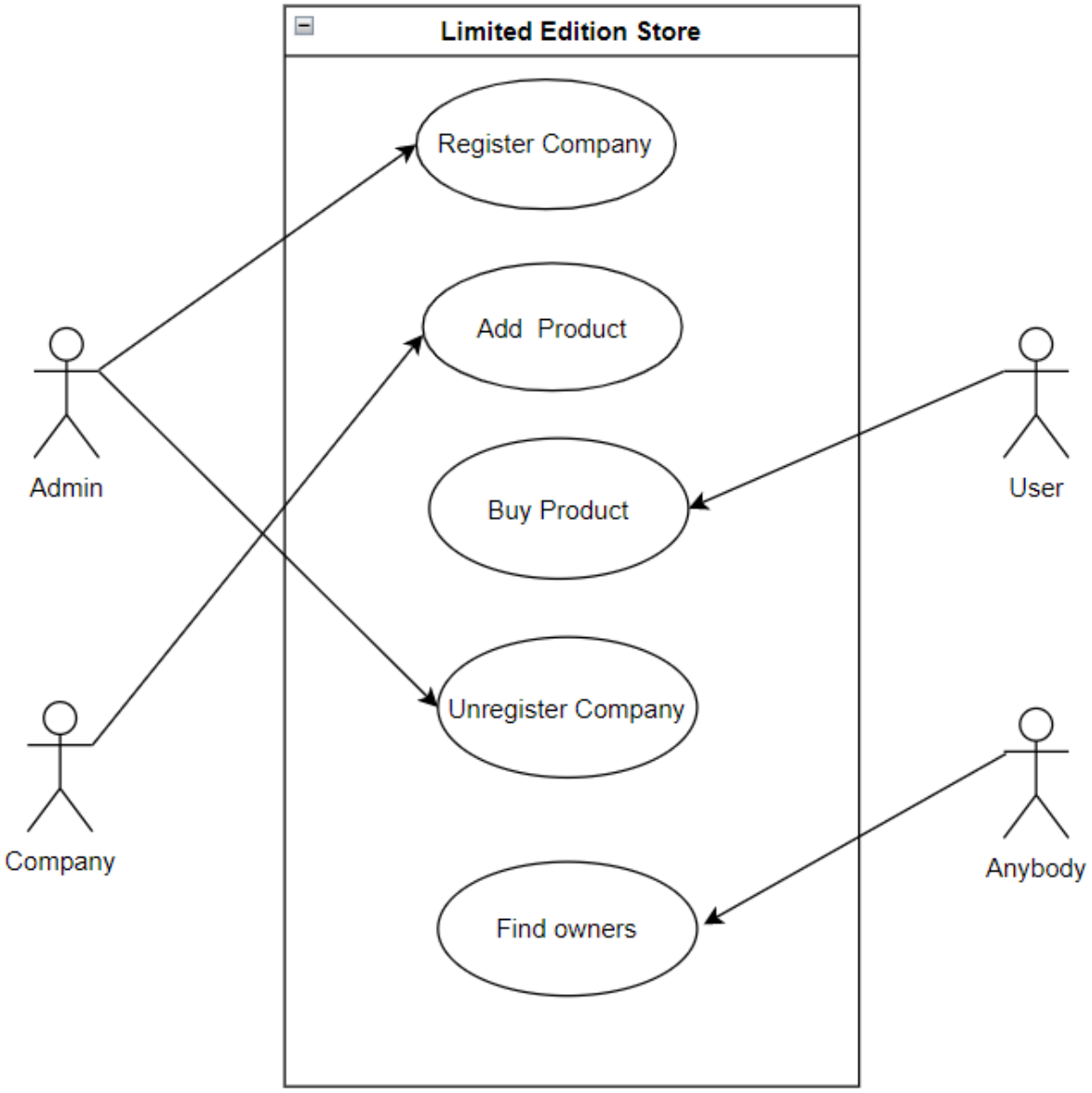
1. onlyAdmin: This modifier is used to restrict certain functions to be invoked only by the admin.
2. onlyCompanies: This modifier is used to restrict certain functions to be invoked only by the companies.

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.
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. 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.
4. 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 directly transfers the amount of ethers which are worth the product price from the user to the company.
5. getOwners(uint productId2): This is a view, it used to find the owners of limited edition products by entering the productId.

6. getProducts(): This is a View, it is used to return the product details to UI.

Use case diagram:



Contract Diagram:

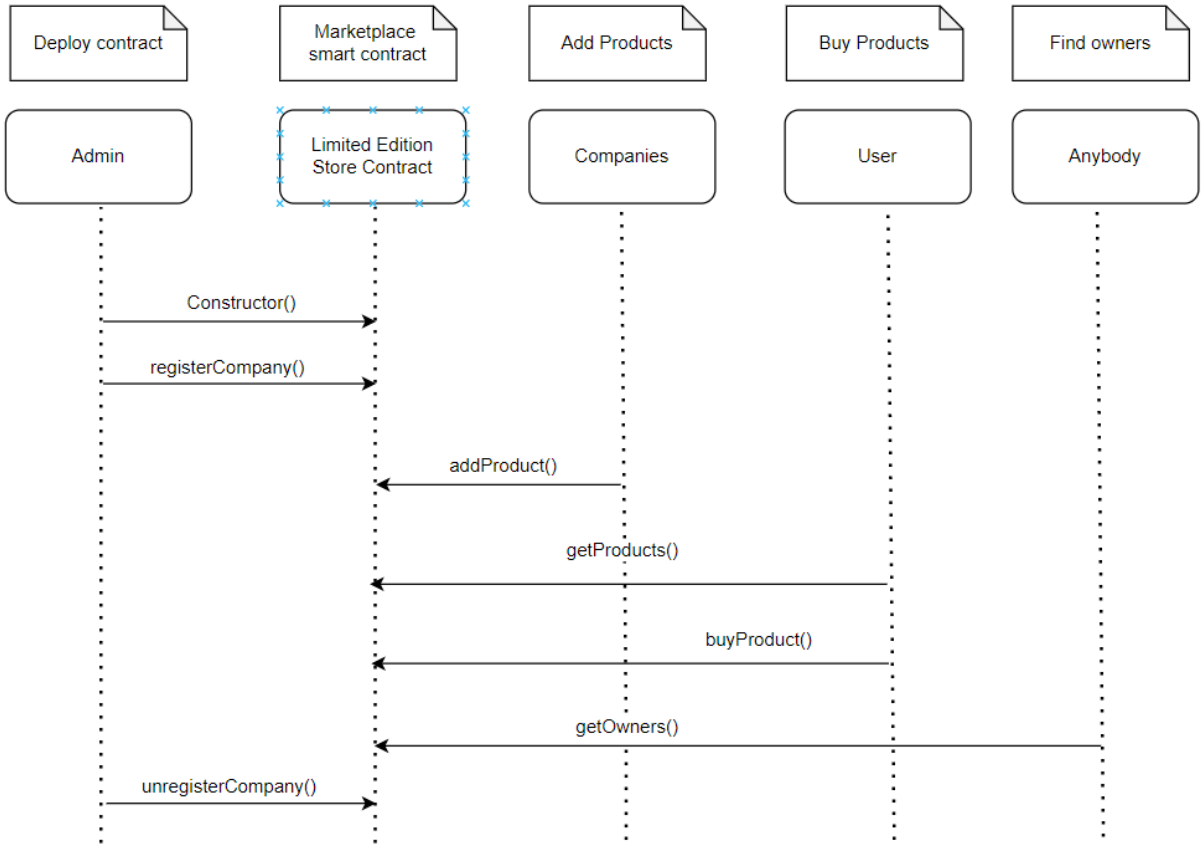
Limited Edition Store

```
Struct Product{}
Struct Company{}
address admin;
mapping(address => Company) addressToCompanyMap;
mapping(uint => address[]) owners;
Product[] products;
Company[] companies;
uint productId = 0;
uint companyId = 0;
```

```
modifier onlyAdmin(){
        require(msg.sender == admin);
        _;
 }
modifier onlyCompanies(){
       require(addressToCompanyMap[msg.sender].isActive== true);
       _;
 }
```

```
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)
```

Sequence Diagram:



Quad chart Diagram:

Use case: Limited Edition Store (LES) Problem Statement: a decentralized blockchain-based marketplace for selling/buying limited edition products.	**Issues with existing centralized model:** 1.Limited edition products are created in small quantities and sold for limited time in the market. 2.There are many companies in the world which sell these limited products, but there is no common place for selling them. 3. There is no track of the proud owners of these products for the future generations. 4. Existent fraud of selling wrong products to the user. (claiming to be original)
Proposed blockchain-based solution: 1. This solution allows companies to be registered on marketplace to sell all their limited edition products at a single place. 2. Users can buy the products they like and the application keeps track of these owners for future generation. 3. Seamless payment is enabled. Cost of the product is directly transferred from user to the companies. 4. All transactions recorded on blockchain for dispute resolution and for business analytics.	**Benefits:** 1. Brings all the limited edition products to a single place. 2. This solution provides better customer experience. 3. This solution helps future generations to track proud owners of the limited edition products. 4. Prevents fraud, as companies are only registered by the admin. There is no chance for wrong products. 5. Seamless payment settlement system.