



THE UNIVERSITY OF  
**MELBOURNE**

School of Computing and Information Systems  
University of Melbourne

Software development project

**The Prototype of**  
**Order Tracking & Certification Application**  
**Leveraging Blockchain Technology**

Kaiquan Shi, 906555, [kaiquans@student.unimelb.edu.au](mailto:kaiquans@student.unimelb.edu.au)

Tao Jin, 827872, [tjin2@student.unimelb.edu.au](mailto:tjin2@student.unimelb.edu.au)

Yubing Yang, 883877, [yubingy@student.unimelb.edu.au](mailto:yubingy@student.unimelb.edu.au)

Supervisor: Prof. Richard Sinnott, [rsinnott@unimelb.edu.au](mailto:rsinnott@unimelb.edu.au)  
COMP90055 2019 SM1(25 Credit Points)

# Abstract

The task of this project is to build a decentralized application that can help customers that allows customers to check the source of the goods they received. Traditional web applications are vulnerable to attacks, so that the authenticity and integrity of data cannot be effectively guaranteed. The introduction of blockchain technology can solve this problem because the blockchain effectively prevents data from being deleted, and the editing of data is recorded by each node in the peer-to-peer structure. In this project, we implemented a decentralized application using product data from Capral Ltd. These data include product title, short description, categories, product certificates, etc. Manufacturers distributed in different locations around the world will send the corresponding data input blockchain when order is shipped and update the logistics information of the product in the blockchain. Whenever a new entry is generated, the corresponding link and QR code are also generated, which helps the customer to track the source of the product.

# Declaration

This thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.

We have received clearance for this research from the University's Ethics Committee and have submitted all required data to the Department.

# Acknowledgement

This paper was completed under the direction of Dr Richard Sinnott. Dr Richard Sinnott's profound expertise and rigorous academic spirit have had a profound impact on us. From the selection of the subject to the final completion of the project, Dr Richard Sinnott has always given us careful guidance. Here, we would like to express our sincere gratitude to Dr Richard Sinnott.

We also received great care and help from friends and family in life and learning, thank you!

# Table of Contents

|  |               |
|--|---------------|
| <b>Abstract .....</b>  | <b>I</b>      |
| <b>Declaration .....</b>   | <b>II</b>     |
| <b>Acknowledgement.....</b>  | <b>III</b>    |
| <b>Table of Contents .....</b>   | <b>IV</b>     |
| <b>Table of figures .....</b>  | <b>VI</b>     |
| <b>1. Introduction .....</b>   | <b>- 1 -</b>  |
| <b>2. Background.....</b>  | <b>- 2 -</b>  |
| <b>3. Technology.....</b>  | <b>- 3 -</b>  |
| <b>3.1. Blockchain .....</b>   | <b>- 3 -</b>  |
| <b>3.2. Smart contract .....</b>                                       | <b>- 5 -</b>  |
| 3.2.1. The uniqueness of smart contracts.....                          | - 5 -         |
| 3.2.1.1. Whole alloy flow is easy.....                                 | - 5 -         |
| 3.2.1.2. Additional costs for deployment and subsequent writes .....   | - 5 -         |
| 3.2.1.3. The cost of storing data is higher.....                       | - 5 -         |
| 3.2.2. How to write a smart contract? .....                            | - 6 -         |
| <b>4. Related work.....</b>  | <b>- 8 -</b>  |
| <b>4.1. Blockchain product examples .....</b>                          | <b>- 8 -</b>  |
| <b>4.2. Differences between traditional bitcoin and Ethereum .....</b> | <b>- 9 -</b>  |
| <b>5. Requirements analysis .....</b>                                  | <b>- 10 -</b> |
| <b>5.1. Data analysis .....</b>  | <b>- 10 -</b> |
| <b>5.2. Requirement .....</b>  | <b>- 10 -</b> |
| <b>5.3. Design schedule.....</b>                                       | <b>- 10 -</b> |
| <b>6. Architecture Design.....</b>                                     | <b>- 11 -</b> |
| <b>6.1. Overall Architecture.....</b>                                  | <b>- 11 -</b> |
| 6.1.1. Process for Decentralized App Development.....                  | - 12 -        |
| 6.1.2. Design of Smart Contracts.....                                  | - 12 -        |

|           |   |               |
|-----------|---|---------------|
| 6.1.3.    | Compilation and Migration of Smart Contracts .....        | - 13 -        |
| 6.1.4.    | Design of Front-end Side .....                            | - 13 -        |
| <b>7.</b> | <b><i>Implementation .....</i></b>                        | <b>- 14 -</b> |
| 7.1.      | Add Products and Certifications .....                     | - 14 -        |
| 7.2.      | View created products .....                               | - 16 -        |
| 7.3.      | Update information about the product .....                | - 17 -        |
| 7.4.      | Search and View .....                                     | - 18 -        |
| <b>8.</b> | <b><i>Evaluation and Future Work—Improvement.....</i></b> | <b>- 20 -</b> |
| 8.1.      | Critical Evaluation .....                                 | - 20 -        |
| 8.2.      | Improvement .....   | - 20 -        |
| <b>9.</b> | <b><i>Conclusion.....</i></b>                             | <b>- 22 -</b> |
|           | <b><i>Reference.....</i></b>                              | <b>- 23 -</b> |
|           | <b><i>Appendix.....</i></b>                               | <b>- 24 -</b> |

# Table of figures

|  |        |
|--|--------|
| Figure 3-1 Structure of Blockchain .....             | - 3 -  |
| Figure 4-1 Deploy smart contract .....               | - 6 -  |
| Figure 4-2 Smart contract sample .....               | - 7 -  |
| Figure 6-1 Architecture Overview .....               | - 11 - |
| Figure 6-2 Smart Contract Design.....                | - 12 - |
| Figure 6-3 Basic Functions for Different Roles ..... | - 13 - |
| Figure 7-1 Interaction with the Dapp .....           | - 14 - |
| Figure 7-2 Main page.....                            | - 14 - |
| Figure 7-3 Create Certification(s) .....             | - 15 - |
| Figure 7-4 Create Product(s) .....                   | - 16 - |
| Figure 7-5 Main Page after Creation .....            | - 16 - |
| Figure 7-6 Tracking Info of the product .....        | - 17 - |
| Figure 7-7 Update the Product .....                  | - 18 - |
| Figure 7-8 Simplified Search Page for Customer ..... | - 18 - |
| Figure 7-9 Details of the Searching Product .....    | - 19 - |

# 1.Introduction

Blockchain is a very popular emerging technology. It is a combination of computer technology such as distributed data storage, point-to-point transmission, consensus mechanism, and encryption algorithm. (Economist, 2015)

A blockchain is essentially a decentralized database. It is a chained data structure in which data blocks are sequentially connected in chronological order. Blockchains are cryptographically guaranteed distributed ledgers which cannot be overwritten. Each block contains information that needs to be protected, associated information with previous blocks, and information that supports data validation.



## 2. Background

We live in a digital age. Many information needs to be transmitted and stored in the cloud through the network. This promotes information sharing and production automation. But on the other hand, it also brings hidden dangers of information security.

Some individuals and organizations add, modify, or delete relevant information in a network-based shared database for the benefit of themselves. For example, in the product supply chain, some distributors falsify the logistics information of the products in order to obtain greater benefits, they confused the identity of the upstream manufacturers. This makes it impossible for the customer to confirm whether the product in hand is from a manufacturer that is consistent with the order, or whether the product is in line with expectations from production to processing.

In this model, the main reason for the database to be attacked is that it adopts a centralized structure, and the original data will be overwritten after the data is changed, so it cannot be verified.

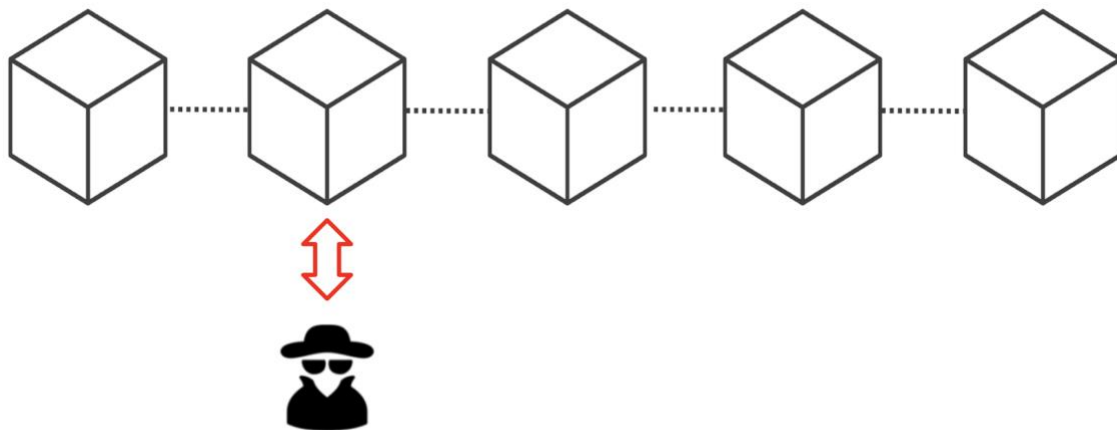
Blockchain technology just happens to avoid such threats. First, the blockchain uses a peer-to-peer network structure, and each user who joins the blockchain has a complete account book. Second, the data that has been saved in the blockchain cannot be changed, which makes the history information of each data update completely recorded.

## 3. Technology

### 3.1. Blockchain

In a valid blockchain, each block contains the hash value of previous block, some data, and the hash value of the current block. Bitcoin is an important application of the blockchain, in which each block stores transaction information. However, the data that can be saved in the blockchain is not limited to this. Similar to traditional databases, blockchains can hold different types of data under different requirements. For example, in a blockchain used for supply chain management, the data in the blocks should be some information that supports order validation. The hash values are calculated by hash functions it's always unique, so it can be used to identify a block just as a fingerprint.

Once the data inside the block has been changed, its hash value will also be changed. As a consequence, the hashing scheme can help us to detect modifications.



*Figure 3-1 Structure of Blockchain*

Let's take an example in the Figure 3-1. Here we have a chain of 5 blocks, each block has a hash and the hash of the previous block. Tampering with the second block causes the hash of the block to change as well. When the hash value in a block is modified, the previous hash value stored in the subsequent block will no longer match it, which will cause the subsequent block to become invalid. In turn, all the block after the medicated block will be invalid.

However, the hash function is not sufficient to make the blockchain completely reliable. Computers nowadays are very powerful and can process massive data in a short time frame. If the attacker modifies the information in a block and recalculates the hash values of all the subsequent blocks, the blockchain will remain valid. In order to avoid this kind of attack, the blockchain also introduces a proof-of-work mechanism, which makes the calculation process of hash value significantly increase, thus slowing down the speed of generating new blocks. In Bitcoins blockchain for example: when you want to add a new block, it will take you about ten minutes to calculate the have value because of the proof-of-work mechanism.( Brito & Castillo, 2013) The chain of hash values and the scheme of proof-of-work makes blockchain very safe. Because when an attacker wants to change the data in a block, it takes a lot of time to recalculate the hash values in subsequent blocks.

In addition, unlike the centralized database, the blockchain uses a peer-to-peer(P2P) network structure, which makes the blockchain more secure. The blockchain is open to all users, which means that when a new user joins the network, he will get a full copy of the data already in all the blockchains. Each update of the blockchain will be processed through the smart contract mechanism and stored in each node of the P2P network.

So, if an attacker wants to change the data in the blockchain, he needs to get access to at least 50% of the nodes in the P2P network, recalculate the hash function with the delay of proof-of-work mechanism. Only then the modified data will be valid in the tampered blockchain, but this is a task that is almost impossible to accomplish.

In this project, we developed a decentralised application (Dapp) to track the product information. A Dapp is a web application that utilizes blockchain technology. The front end of Dapp uses the same technology as traditional web applications. The main difference exists in the back-end structure. Instead of Application programming interfaces (APIs) connecting to databases in traditional web application, there are Smart Contracts connecting to blockchains in Dapp. (Raval, 2016)

## 3.2. Smart contract

### 3.2.1. The uniqueness of smart contracts

#### 3.2.1.1. Whole alloy flow is easy

It's not easy for a typical application to integrate the golden stream. Smart contracts are extremely easy to integrate with the Golden Stream system (using Ethereum or a new token contract created by itself).

#### 3.2.1.2. Additional costs for deployment and subsequent writes

The general application needs to provide the URL for the user to download. The general web application also needs to run on the server. The developer needs to maintain the operation of the server to provide the service, which requires continuous cost (even if it is a free server or the web space is also absorbed by the manufacturer itself. After the program starts running, there is no additional cost other than maintaining the cost.

Smart contracts require a fee for deployment, which is distributed to those involved in transaction verification (mining). After the contract is successfully deployed, the contract is distributed as part of the unchangeable blockchain and is stored decentralized at nodes in Ethereum around the world. Therefore, after the smart contract is deployed, there is no need to provide maintenance costs on a regular basis, and there is no charge when querying static data that has been written into the blockchain. Only a small transaction fee is required each time a calculation is written or read through a smart contract.

#### 3.2.1.3. The cost of storing data is higher

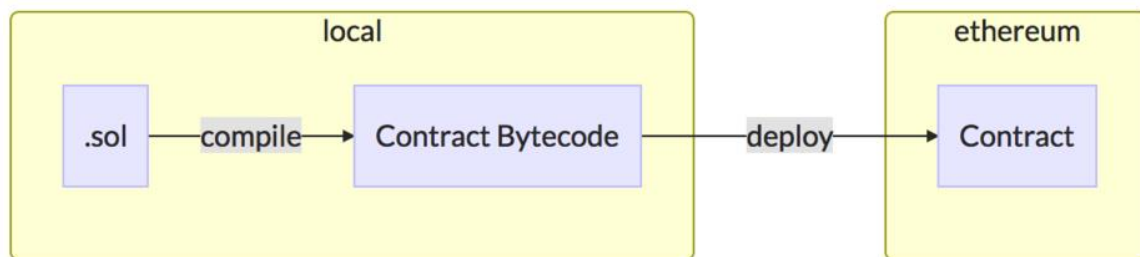
A general application stores data on the local machine or server. When the data is needed, it is read from the local machine or the server. The smart contract stores the data in the blockchain. The time and cost required to store the data are relative to the cost. expensive. Fourth, after the deployment cannot be changed, the general application can be modified by installing a new version of the program, the web application can also be achieved by deploying a new version of the program, and the smart contract cannot be modified after being applied. Of course, smart developers

have had the means to bypass the limitations of smart contract deployments that can't be changed by adding additional smart contracts.

### 3.2.2. How to write a smart contract?

Smart contracts on Ethereum need to be written in solidity language.

After writing the solidity code (.sol), we need to compile (compile) the program code into the binary contiguous Contract Bytecode that EVM (Ethereum Virtual Machine) can read before deploying to Ethereum. Executed on the blockchain. Contracts deployed on the blockchain will have a Contract Address in the same format as the wallet address (address).



*Figure 3-2 Deploy smart contract*

Smart contracts can be executed automatically after deployment. When a subsequent smart contract is called, the user can use the wallet address (owner account) of the deployment contract, or allow other wallet addresses to call this smart contract by the smart contract conditions written. Calling a smart contract is actually a transaction to this contract address, but the transaction is not just a token, but a call method provided by a smart contract.

The image shows a web-based IDE for developing smart contracts. On the left, a file explorer shows a project named 'browser' containing a file 'ballot.sol'. The main editor displays the Solidity code for the 'PushButton' contract. The code includes a pragma statement for Solidity version ^0.4.13, a contract definition, a public event 'ButtonPushed', a modifier 'isTimeout', a public function 'PushButton', a public function 'push' with an 'isTimeout' modifier, a public function 'getBlock', and a private function 'checkTitle'.

```

1 pragma solidity ^0.4.13;
2
3 contract PushButton {
4
5     uint public startBlock;
6     uint public interval = 108 * 60 / 4;
7     uint public nextTimeoutBlock;
8     uint public totalPush;
9     string public title;
10
11     event ButtonPushed(address indexed _address, uint _totalPush, uint _nextTimeoutBlock);
12
13     modifier isTimeout() {
14         require( getBlock() <= nextTimeoutBlock );
15         _;
16     }
17
18     function PushButton() {
19         startBlock = block.number;
20         nextTimeoutBlock = startBlock + interval;
21         totalPush = 0;
22     }
23
24     function push() isTimeout() returns (bool) {
25         totalPush += 1;
26         nextTimeoutBlock = getBlock() + interval;
27         checkTitle();
28         ButtonPushed(msg.sender, totalPush, nextTimeoutBlock);
29         return true;
30     }
31
32     function getBlock() constant returns (uint) {
33         return block.number;
34     }
35
36     function checkTitle() internal {
37         if (totalPush < 10) {
38             title = "newbie";
39         } else if (totalPush < 1000) {
40             title = "apprentice";
41         } else if (totalPush < 10000) {
42             title = "master";
43         } else {
44             title = "believer";
45         }
46     }
47
48 }
49

```

On the right side, there is a sidebar with tabs for 'Contract', 'Settings', 'Files', 'Debugger', 'Analysis', and 'Docs'. The 'Settings' tab is active, showing deployment configuration for a 'JavaScript VM' environment. The 'Account' field displays '0xca3...a733c (8901850771803170616147943.4244)'. The 'Gas limit' is set to '3000000' and the 'Value' is '0'. Below these settings, there are buttons for 'Publish', 'Attach', 'Transact', 'Transact(Payable)', and 'Call'. A dropdown menu shows the selected contract 'browser/ballot.sol:PushButton' with a size of '15 by'. At the bottom of the sidebar, there are buttons for 'Publish', 'At Address', and 'Create', along with a link to 'Contract details (bytecode, interface, etc.)'.

Figure 3-3 Smart contract sample (Hajimirza, 2019)

## 4.Related work

### 4.1.Blockchain product examples

Steem is an encrypted currency that circulates on the Steemit platform. Steemit is a decentralized social media platform that motivates users to participate in creation through micropayments. Users also encourage creators by using Steem currency for micropayments, which allows content creators to publish articles on the blockchain platform to get direct benefits into reality (Alon, 2019).

Brave was created by JavaScript inventor Brendan Eich. People can use the Brave browser to watch ads and get BAT coins. The BAT coin is called the Basic Attention Token, which is equivalent to the bonus point that the browser member can sign in to sign in. However, it increases the liquidity. The Brave browser development team introduces very attractive incentives for users to watch ads that are 70% of total advertising revenue, which is billed monthly through BAT tokens. Brave's Token is used to motivate users to choose the right ads for them, and advertisers provide better advertising.

Bancor is a decentralized mobility network that allows users to use a simple web wallet, hold any tokens and convert them to any other token in the network, without the other party, to automatically calculate the price.

Bancor-compatible tokens are a new ERC20-compatible token on the Ethereum blockchain, thanks to its built-in features, making them an essentially tradable token. This feature is simple but far-reaching: you can have one token endorsed by another token or tokens, holding these endorsement tokens as a reserve at a ratio of 0 to 100%. In this way, the price of this token will be automatically set according to its supply, reserves, and ratio. The transaction does not require the participation of the second party.

Golem is one of the earliest generations of Ethernet applications. The token is GNT, which is a decentralized computer computing rental platform built on the Ethereum platform. Through the Golem platform, any user can become a power seller and renter.

Whether you have an idle home computer or a few large data centres, you can join the Golem platform. The trading system based on Ethereum is applied to the Golem platform to settle the revenue of the computing provider and the cost of the computing user.

## 4.2. Differences between traditional bitcoin and Ethereum

Bitcoin is a value storage tool designed to be a global, peer-to-peer, distributed, and transparent digital currency. In contrast, Ethereum's design provides an ecosystem for distributed applications and decentralized autonomous organizations (DAOs). Unlike most people's perceptions, Ethereum and Bitcoin have different goals. Ethereum focuses on making the blockchain more attractive to users, while Bitcoin is committed to changing the finance industry.

Both Bitcoin and Ethereum are based on decentralized blockchain technology, but there are still many differences in the deep technical field. For the specific performance of decentralization, Bitcoin is mainly divided into three aspects: complete node decentralization, computational decentralization, and development decentralization. In contrast, the development process of Ethereum is completely centralized, although it can greatly improve efficiency, but it also makes it impossible to guarantee the security of its rules and is vulnerable to attack.

Bitcoin was originally designed as a decentralized cryptographic currency network for trading currency values. The main purpose of the Bitcoin blockchain is to provide trust support for these financial transactions.

Contrary to Bitcoin, Ethereum has been conceived as a decentralized application development platform since its first day, and its blockchain is designed to support the operation of decentralized applications.

Therefore, the design of Ethereum learned from the experience of Bitcoin and improved the shortcomings of Bitcoin. The difference between the two systems is that Ethereum's data processing is faster than Bitcoin, because the Ethereum system automatically applies to the terms and conditions of the contract once agreed (Munro,2019).



## 5. Requirements analysis

### 5.1. Data analysis

The given excel table shows the basic information of Capral products, including title, picture file number, short description, all\_product\_categories, created/updated time and so on.

### 5.2. Requirement

The goal of our project is to design a web app to meet the following requirements. The first one is that customers can search for any information they want to know about the products they have bought. For example, customers can find the detailed materials and created time in the app, even whether the product meets the Australian standard. Moreover, they can track the current location of the product and the time when they can get it.

### 5.3. Design schedule

Feb: Collect the knowledge and background about blockchain together. Analyse the data and requirement.

March: Build the basic architecture including blockchain, smart contract and so on.

April: Complete front-to-end system

May: Implementation of the web app and fix bugs.

## 6. Architecture Design

Before presenting the general design, it is quite significant to explain the reasons for using truffle (<https://www.trufflesuite.com/>) to develop this decentralized application. The truffle framework is fully integrated at the client side since it includes the built-in compiler and network management so that beginners for developing Dapp based on Ethereum are easy to deploy. In addition, the console is provided for testing automatically thus developers could directly use different commands to get the output after framework built. More importantly, there are various boxes which already contain the basic templates for interacting UI with smart contract to implement the details of Dapp.

### 6.1. Overall Architecture

As can be seen from the below diagram, this project is mainly based on truffle which combines different smart contracts with the Ethereum client named web3js for interaction with blockchain at the front-end side and the partial model is inspired by a project called trace (Vaillancourt, 2018). The development environment of Dapp could be simply to configure if choosing truffle as the framework. In the following sections, more related details about other parts will be discussed.

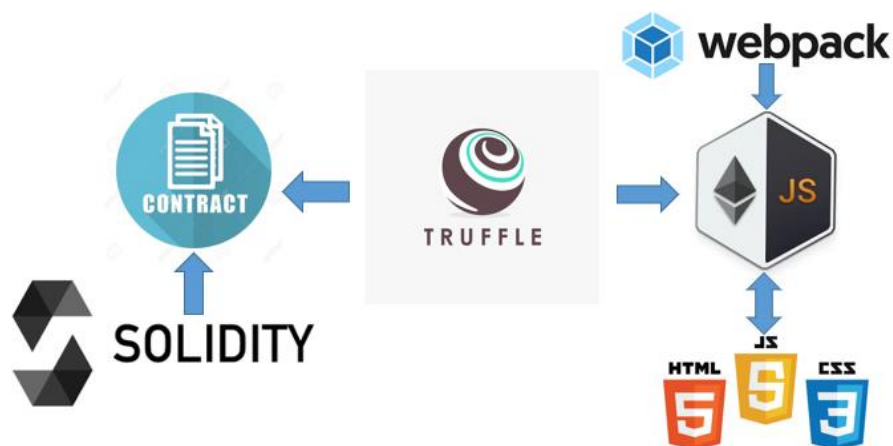


Figure 6-1 Architecture Overview

### 6.1.1.Process for Decentralized App Development

With the emergence of blockchain 2.0 related concepts and the popularity of decentralized applications, more standard procedures are gradually provided by truffle and this project is completed through following three major steps.

### 6.1.2.Design of Smart Contracts

The figure 6-1 shows that smart contracts are written using the programming language called solidity (<https://solidity.readthedocs.io/en/v0.4.23/>) which is static, contract-oriented and high-level compared with other languages. The syntax of solidity is similar to JavaScript and the code will be executed in Ethereum virtual machine.

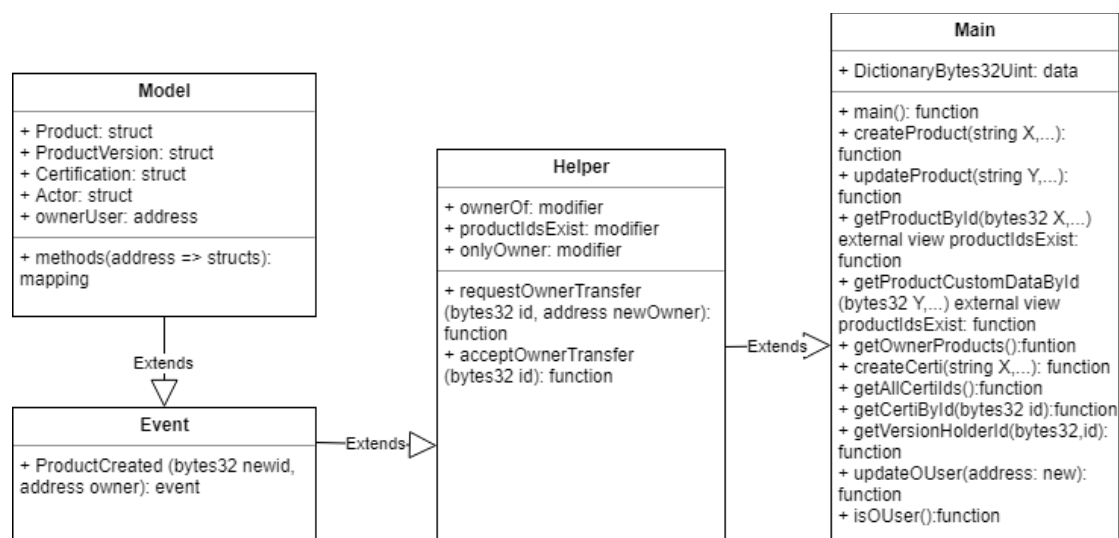


Figure 6-2 Smart Contract Design

The overall design of contract is displayed through above UML diagram. There are four significant contracts Model, Event, Helper and Main and each of them extends the functionalities of their previous contract. Model works as the database in traditional application and it defines different structs similar to entities and a special type called address for back-end storage and also different mappings between id to their structs. Event just contains one function to be fired by other smart contracts. Helper is designed with existence checking of productid and users and transfer administrator rights. While in Main several functions are defined such as createX or getX to be called by the methods at front-end side for sending information to smart contracts. Additionally, the

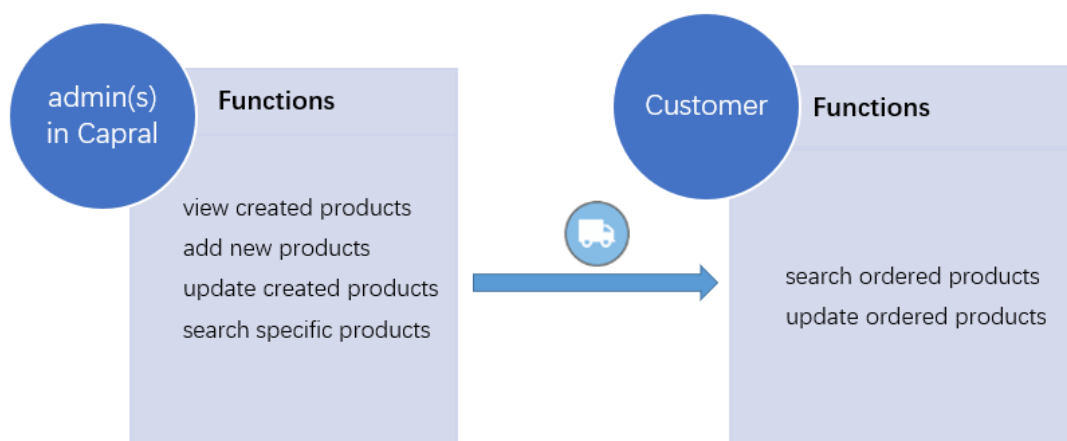
consensus and related concepts of blockchain are implemented in the solidity programs such as Dictionary and Migration

### 6.1.3.Compilation and Migration of Smart Contracts

Compiling contracts plays an important role for interpreting the code written by programmers into machine-level languages. After that, the build package including the json files of contracts are appeared. Then related files are migrated using the development network and unique transaction id, gas usage limits and block number are shown in the terminal. All previous steps are completed in our personal blockchain through executing ganache or truffle develop to test repeatedly whether the methods in contracts are valid.

### 6.1.4.Design of Front-end Side

Finally, it comes to UI design process and the front-end logic is based on functions in contracts and implemented in a plain style with fundamental operations for users. In the Figure 6-3, it shows two separate roles and here different scenarios are presented in next section. After configuration of local development server, clients can access the pages through browser to interact with Ethereum blockchain



*Figure 6-3 Basic Functions for Different Roles*

## 7. Implementation

At first, the users need to ensure the installation and configuration of Metamask (<https://metamask.io/>) which is an extension of different browsers such as Google Chrome since it is the most convenient way to interact with the Dapp in webpages.

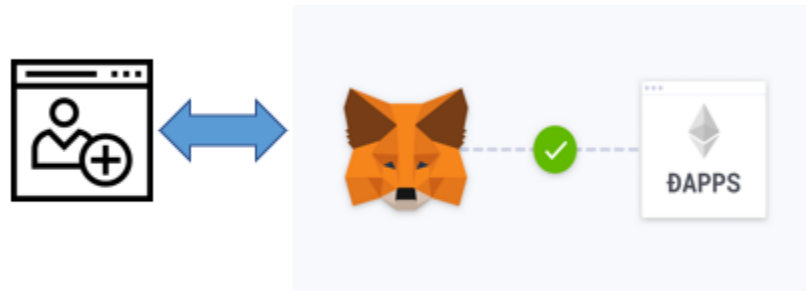


Figure 7-1 Interaction with the Dapp

After launching the Dapp, it is required to create the account using the mnemonic in truffle develop to import as the seed phrase and then the main page is displayed. The users could view the list of products they've created (suppose the mode of truffle develop is always running at the back-end).

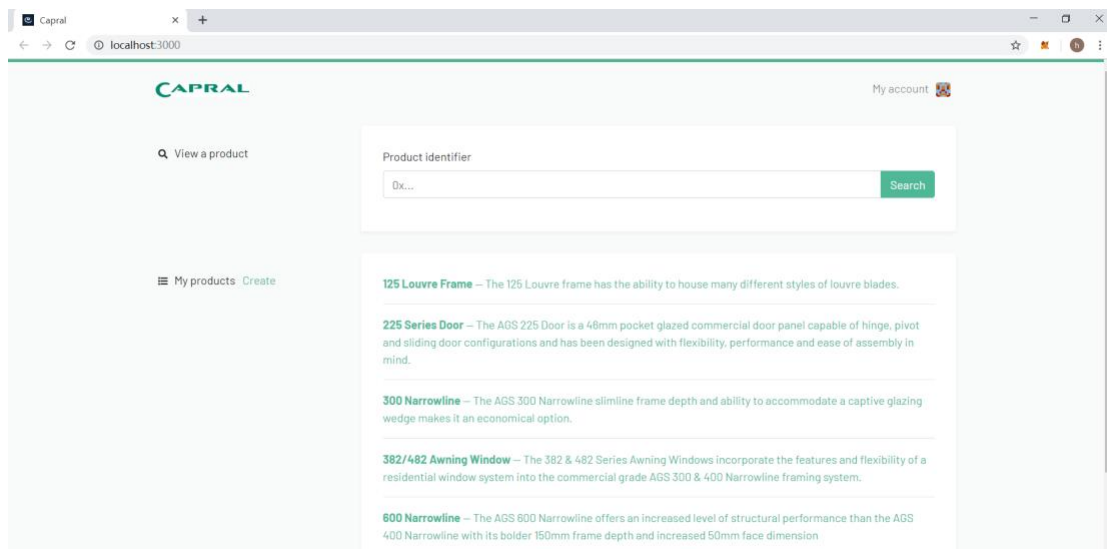


Figure 7-2 Main page

### 7.1. Add Products and Certifications

Firstly, we need to create certifications for the particular products and here we mainly select one product in Capral-Content-1 file as an example. After input the related details into the corresponding fields, the address of account is copied to the highlighted area in

Figure 7-3. The transaction of certification creation has to be confirmed on the blockchain through MetaMask when clicking <Create>.

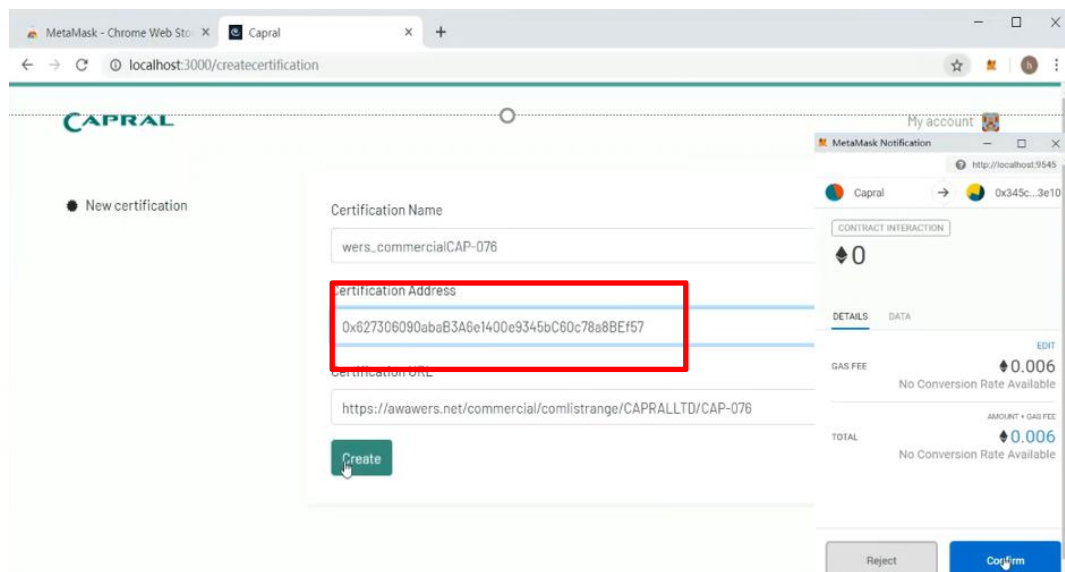


Figure 7-3 Create Certification(s)

The below screenshot shows the page for adding the specific products. The users could see the certifications they add in previous step and mark the certifications of this product for customer to check. Here, we suppose the first holder of this product is Capral and after input the information, users can also add their custom properties for the products and this transaction also requires confirming. For example, the cyclonic rating of this framing system is C4.

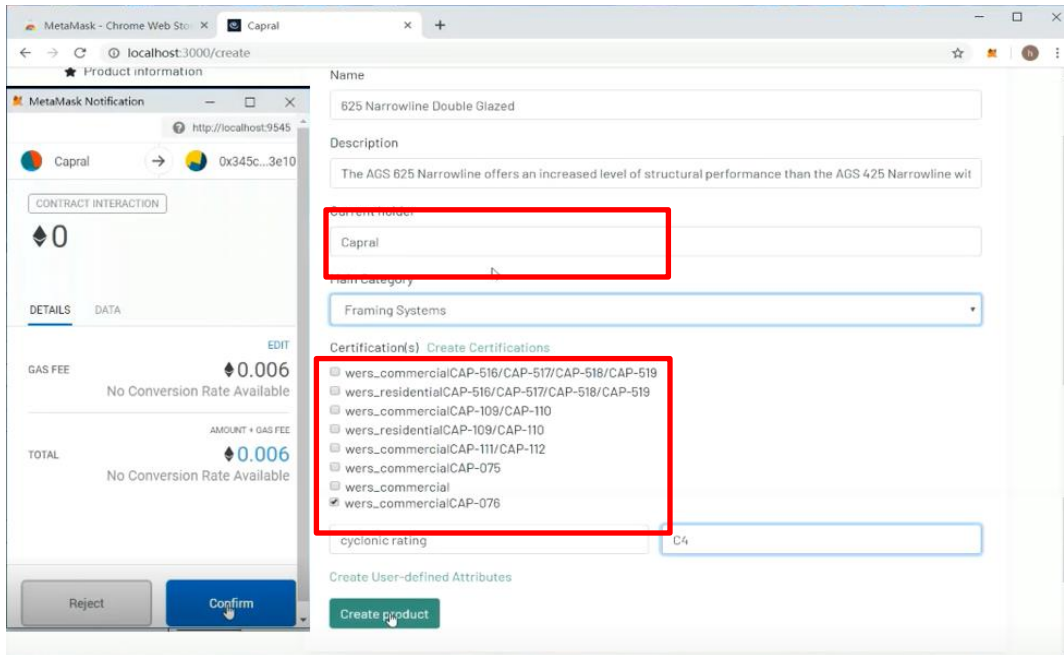


Figure 7-4 Create Product(s)

## 7.2. View created products

The new product we have added just now is displayed at the end of the list of products shown in Figure 7-5. After clicking the link of the products, users can view related details and here we try to bind unique id and QR code together in Figure 7-6.

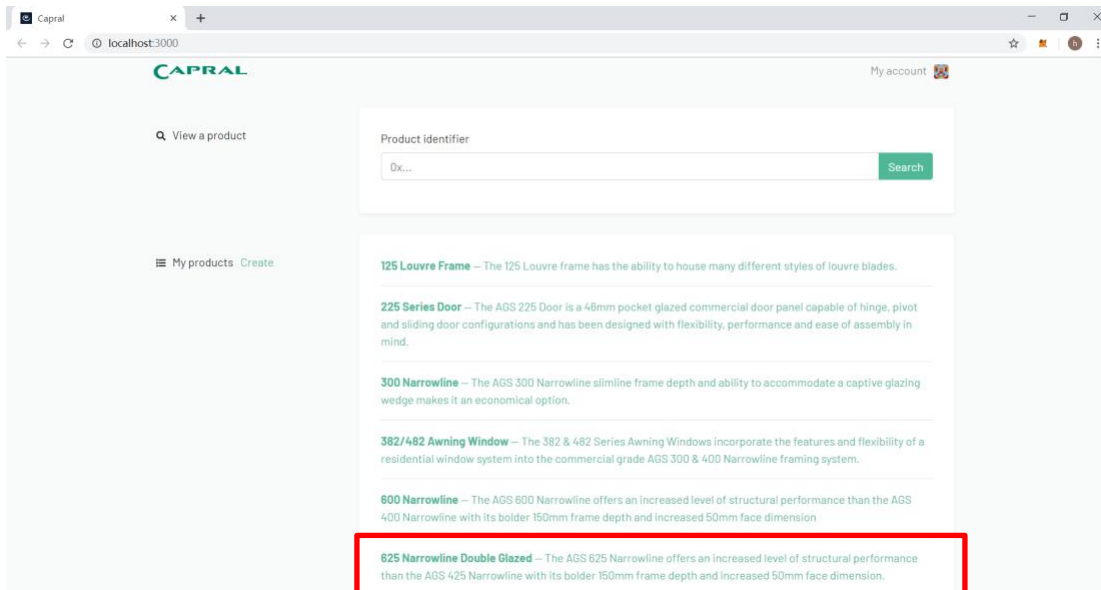


Figure 7-5 Main Page after Creation

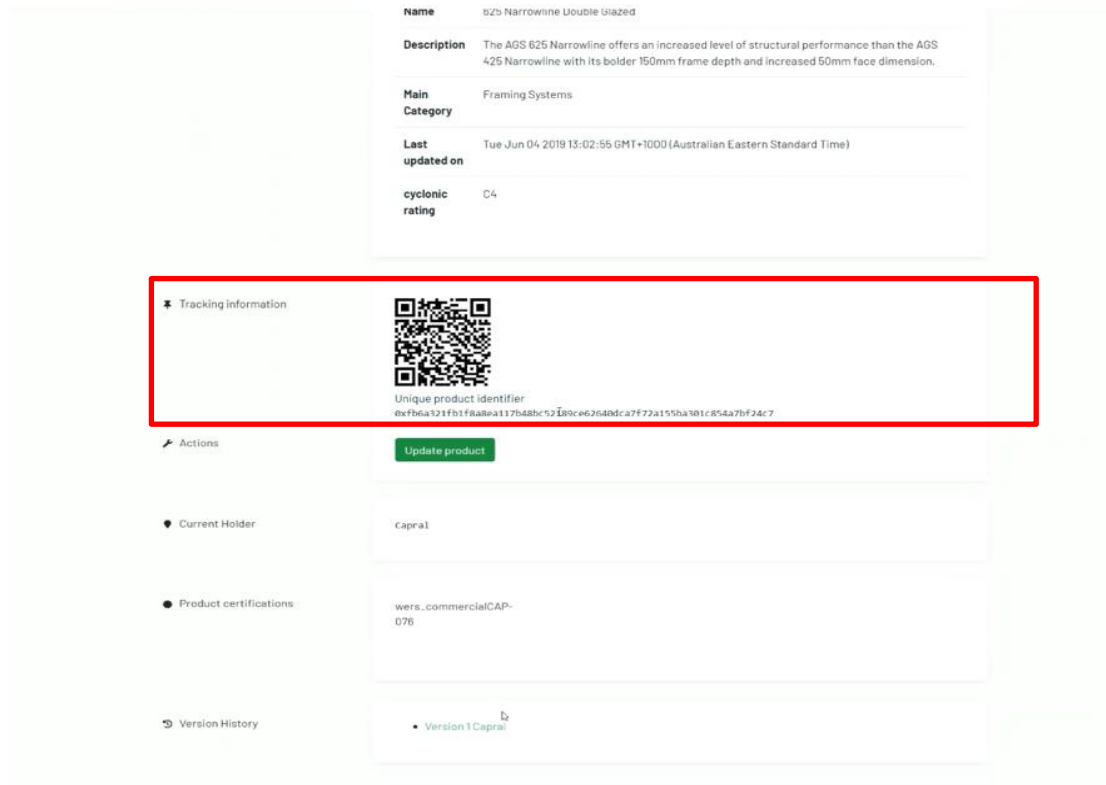


Figure 7-6 Tracking Info of the product

### 7.3. Update information about the product

In this part, we consider a scenario of updating the current holder of the product and adding more user-defined attribute for this specific product. Here, we suppose the communication between users in Capral and distributor is trustworthy. This operation is also required to confirm on the blockchain.



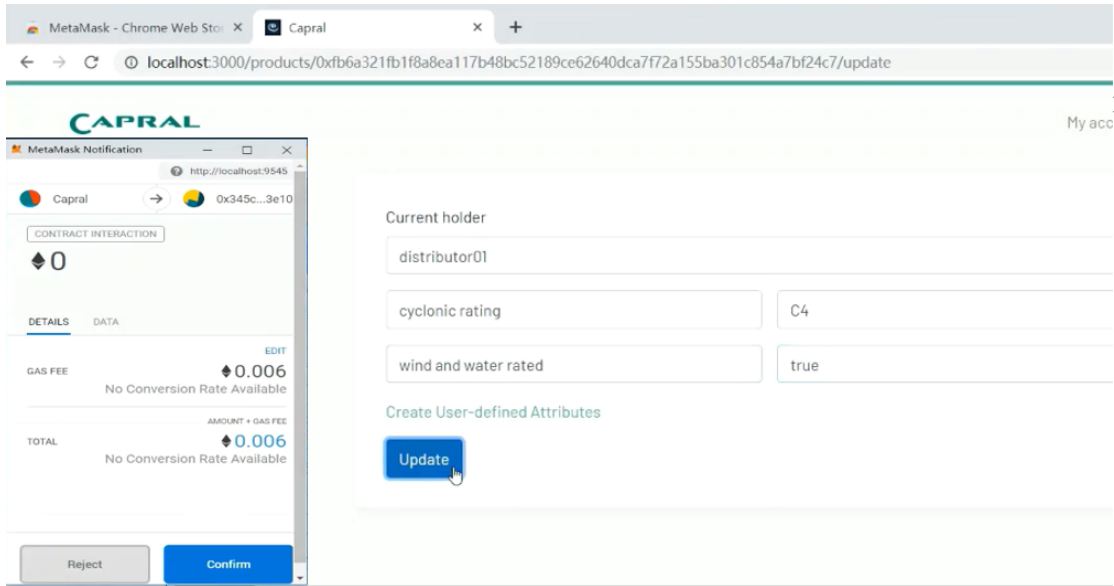


Figure 7-7 Update the Product

## 7.4. Search and View

For this section, new account called customer are created in MetaMask and we mainly simulate the searching process for customer entering the unique id of the product. In the future, when the products are delivered to customers, they are allowed to scan the QR code printed on products which directs to the page of the product definition.

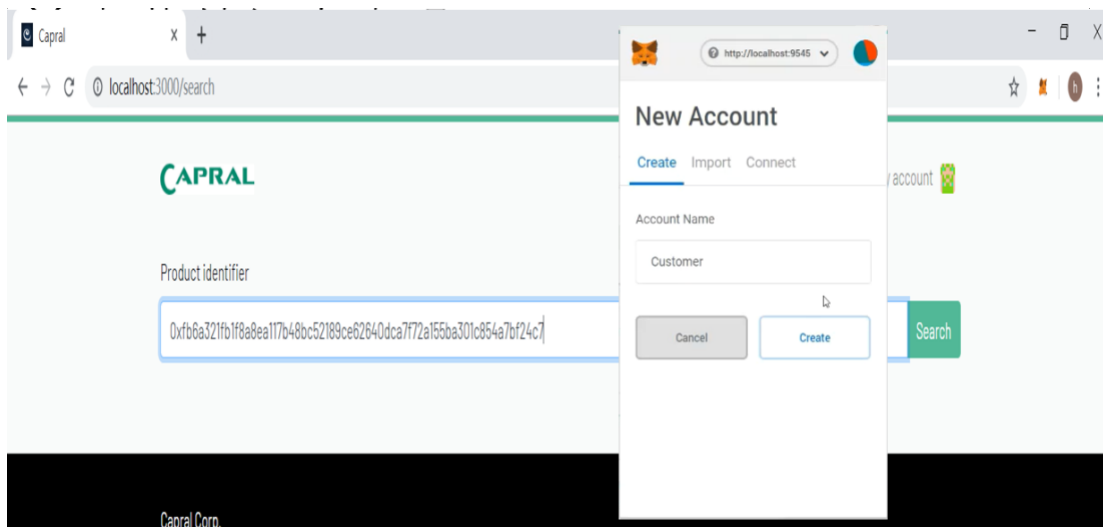



Figure 7-8 Simplified Search Page for Customer

After clicking <Search>, the related details about the product they ordered are shown in Figure 7-8. The customer can view the new attributes Capral created in section 7.3, mark this item as 'Received' (updating the current holder) and comment the quality of

this product. Also, the version history presents the holders in each state and it verifies the provenance of this particular product.

CAPRAL

My account 

● Product definition

Name

625 Narrowline Double Glazed

Description

The AGS 625 Narrowline offers an increased level of structural performance than the AGS 425 Narrowline with its bolder 150mm frame depth and increased 50mm face dimension.

Main Category

Framing Systems

Last updated on

Tue Jun 04 2019 13:20:40 GMT+1000 (Australian Eastern Standard Time)

cyclonic rating

C4


wind and water rated

true

Comment

good quality

✦ Tracking information



Unique product identifier

0xf6a321fb1f8a8ea117b48bc52189ce62640dca7f72a155ba301c854a7bf24c7

✎ Actions

Update product

📍 Current Holder

Customer

● Product certifications

wers\_commercialCAP-076

📜 Version History

• Version 3 Customer

• Version 2 distributor01

• Version 1 Capral

Figure 7-9 Details of the Searching Product

- 19 -

## 8.Evaluation and Future Work—Improvement

### 8.1.Critical Evaluation

Since there are few real cases applying the decentralized concept to develop an application, it lacks a formal standard to evaluate this project. However, we provide the pros and cons of this project based on comparison with traditional application and characteristics of blockchain.

The first advantage is that the Dapp supports the synchronization of information. The other factories of Capral around the world are able to share the product details if they register on the blockchain. Furthermore, each operation which represents transaction on the chain is more secured due to the encryption mechanism of blockchain.

Additionally, the cost of maintenance and development is not relatively high compared with the normal application, because the resource might be released at the free time when using a lite server in truffle

However, one of the disadvantages of the Dapp is that the efficiency of performance might drop, and the client is required to gain the basic knowledge of blockchain. Also, it is quite difficult to integrate with mobile application due to lack of complete ecosystem thus almost all Dapp can be accessed through web pages in their own PC.

### 8.2.Improvement

- User Experience

According to the current system, the first improvement we could make in the future is designing a more sophisticated access control and different roles such as retailers and suppliers in supply chain are added. Furthermore, in order to make our customers to get the actual locations (latitude and longitude) of products, we might insert a map for them to view.

- Backend limitation

Due to the current restriction of the local development environment, we could design a more complicated system deploying on a cloud server. For example, it is possible to

store the unique id of products and registered users in centralized database located at the cloud.

- Security enhancement

After designing a more complex access control, the security of current system needs to improve through the built-in library of OpenZeppelin (<https://openzeppelin.org/>) because it makes blockchain 2.0 more robust.

## 9. Conclusion

As a cutting-edge technology, blockchain brings both chances and challenges to different fields. In this project, we use blockchain-based technology to construct a web app. Users can use this app to track, search and verify the Capral product. Ethereum is a useful blockchain platform for developers and smart contracts provides the reliability. However, the project still has a lot to improve in the future. In conclusion, blockchain technology is used in more and more fields nowadays. It's meaningful to learn about it for everyone.

# Reference

- Brito, J., & Castillo, A. (2013). Bitcoin: A primer for policymakers. Mercatus Centre at George Mason University.
- Hajimirza, A. (2019). Azure-Samples/blockchain. [online] Available at: <https://github.com/Azure-Samples/blockchain/tree/master/blockchain-workbench/application-and-smart-contract-samples> [Accessed 11 Jun. 2019].
- Economist, T. (2015). The Great Chain of Being Sure about Things. The Economist. Accedido desde <http://www.economist.com>.
- Alon, S. (2019). Top 5 Working Products in Blockchain (Updated — October 2018). [online] Available at: <https://medium.com/theblock1/top-5-working-products-in-blockchain-updated-october-2018-9b18548bf8fc> [Accessed 11 Jun. 2019].
- Munro, A. (2019). Bitcoin vs Ethereum: A side-by-side comparison | finder.com.au. [online] finder.com.au. Available at: <https://www.finder.com.au/bitcoin-vs-ethereum> [Accessed 11 Jun. 2019].
- MetaMask. (2019). Retrieved from <https://metamask.io/>
- OpenZeppelin. (2019). Retrieved from <https://openzeppelin.org/>
- Raval, S. (2016). Decentralized applications: harnessing Bitcoin's blockchain technology. " O'Reilly Media, Inc.".
- Solidity — Solidity 0.4.23 documentation. (2019). Retrieved from <https://solidity.readthedocs.io/en/v0.4.23/>
- Truffle Suite | Boxes | react-box-web3-todo. (2019). Retrieved from <https://www.trufflesuite.com/boxes/react-box-web3-todo>
- Truffle Suite | Sweet Tools for Smart Contracts. (2019). Retrieved from <https://www.trufflesuite.com/>
- Vaillancourt, M. (2019). maximevaillancourt/trace. Retrieved from <https://github.com/maximevaillancourt/trace>

# Appendix

Demonstration Video: <https://www.youtube.com/watch?v=-zSWo0IQH0g>

Coding of the application: <https://github.com/yubingy/bcproject>