AHMEDABAD UNIVERSITY

Global Education at Local Cost, Context and Ethos™

Summer Project Report

AGRIINPUT – A BLOCKCHAIN BASED SOLUTION

Submitted By

Raj Dhamsaniya | SEAS | July 15, 2018

Under the guidance of

Prof. Sanjay Chaudhary

INTRODUCTION

Bitcoin introduces us to distributed ledger technology known as blockchain. From that moment to present blockchain involved in many ways possible. Blockchain can be used in tracing of supply chain and can make products more reliable. It ensures that as a customer whatever we are paying it’s worth it or not. Agriculture inputs are no different from it and so, we can apply distributed ledger technology on the supply chain of it. It makes the input more reliable.

1.1 PROJECT SUMMARY

AgriInput is a blockchain based application for providing the traceability of Agriculture input products i.e. Fertilizer, Corps etc. It is developed for the use of customers in the agriculture domain. It is developed for the various organizations at a different level of the supply chain to optimize response time and minimize fake products in the market with its user friendly web experience. It also tackle with the problem of data manipulation. It also provides security features to restrict the viewership of content on blockchain network.

1.2 PURPOSE

Purpose of the development of software is to authenticate if the received product is real or fake. Another important aspect is to implement blockchain technology in a real-life solution. It helps the user by providing previous trace and detail of the product.

1.3 SCOPE

AgriInput is used for verification of the product by all the entities of the network. Besides that blockchain is used for such types of application where traceability will matter for the price of the product, we can use it. Because of decentralization single point of failure is not the problem in it anymore.

1.4 OBJECTIVES

To implement a blockchain based solution for authentication of agriculture input items.

1.5 TECHNOLOGY AND LITERATURE REVIEW

- Technology used:
    - Hyperledger Fabric, Hyperledger Composer, Node.js, CouchDB.

- Literature reviewed:
    - ‘Zero to Blockchain’ – IBM RedBook
    - Blockchain for Dummies – A book by IBM
    - The architecture of the Hyperledger Blockchain Fabric by Christian Cachin at IBM Research – Zurich, CH-8803 R¨uschlikon, Switzerland, July 2016.
    - Hyperledger composer documentation and tutorials.

2. Modelling the problem

This section identifies the modelling part of the problem containing a definition of transaction, assets and participants.

2.1 Participants

Participants are an entity which does a transaction and changes the state of assets.

Five types of participants are defined in the system:

1. **Manufacturer**
    - The manufacturer will create products and fulfil order requests.
    - The manufacturer does have the ability to create a product but, it can’t change its approved status.
    - The manufacturer is responsible for manufacturing the product and accepting order based on the product available.
    - While creating a product manufacturer defines the attributes of the product containing the required information for approval of the product.

2. **Agriculture Organization**
    - It is a government authority for authorizing the product to be able to sell in the market.
    - When asked for approval from Manufacturer, it will see the details of the products and provides the rating on it.
    - The only organization can able to change the approval of the product.

3. **Retailer**
   - The retailer is a participant which delivers products from manufacturer to customer.
   - For various scenario, there will be many middleman or brokers in the network which essentially doing the task of buying from the larger organization and selling to smaller will included in the network.

4. **Customer**
   - The customer is the end user in the network.
   - It will purchase products of the various manufacturer from a retailer.
   - It can also check of whereabouts of product and check if the product is real or fake from its batchId.
5. **Finance Company**
   - It is responsible for payments which are authorized by the retailer or customer about various orders.
   - It can view all the activity happened to an order with the timeline of the order.

2.2 Assets

Assets are tangible or intangible goods, services, or property in the network.

At this time Hyperledger does not support the traceability of state of change in the asset, so for the workaround that matter at every transaction, a parameter is edited containing the date of that transaction.

Four types of Assets are defined in this model.

1. **Product**
   - Product contains the following properties.
     - productId – Identifier

    - productName – Name of the product
    - approved – Boolean approval from agriculture organization. Default false.
    - rating – For storing the rating of the product, provided by the organization.
    - mrp – Maximum retail price of the product.
    - batches – Array of strings, each containing details(Batch Id, Quantity, mfgDate) about a batch while invoking manufacture product.
    - archive – Array of a string containing batches which are sold by the manufacturer.
    - content – Array of a string containing details of content i.e. name, percentage.
    - status – status of the product.
    - requestApproval – the date when approval is requested by the manufacturer.
    - rejectProduct – the date when approval is rejected by agri organization.
    - approveProduct – the date when organization approve the product.
    - productCreated – the date when the product is created.
    - totalQuantity – a total available quantity that can be sold.
    - bestBefore – for defining expiry date of the product.
    - productType – if needed we can categorize product into a different type.
    - Relation with a manufacturer who creates that product and agriculture organization who approves that product.

2. **Order**
    - Order Contains Following Properties
        - orderId – Identifier
        - status – Status of order.
        - amount – the total amount of order
        - created – Date when the order is created by Retailer
        - cancelled – Date when Retailer cancels the order.
        - accepted – Date when Manufacturer accepts the order.
        - paymentRequested – the date when manufacturer asks for payment
        - approved – Date when Payment for order is approved.
        - paid – Date when the order amount is paid by the finance company.
        - Items – Array of a string containing details of order i.e. product details and its quantity in order with the amount of order.

    - Relationship with a manufacturer from the order is placed, a retailer who places the order and for finance related aspect, with a finance company.

3. Stockroom

- Stockroom Contains the following properties.
    - stockroomId – Identifier.
    - Stock – Array of string, each containing json data as given below.
        - ProductName – Product Name
        - Quantity – Total available quantity from all manufacturers.
        - Inventory – Manufacturer wise product details
            - Manufacturer – manufacturer Id
            - Quantity – Quantity of product from particular manu.
            - MRP – MRP of that product
            - Batches – Batches of the product from a manufacturer.
                - BatchId – batch Identifier
                - OrderId – details of the order containing that batch
                - MFGDate – MFG date of that batch
                - Quantity – Quantity in that batch
    - Relationship with Retailer who has this stockroom.

4. CustomerOrder

- CustomerProduct contains the following properties.
    - orderId – Identifier
    - status – Status of order.
    - amount – the total amount of order
    - created – Date when the order is created by the customer.
    - cancelled – Date when a customer cancels the order.
    - accepted – Date when Retailer accepts the order.
    - paymentRequested – the date when a retailer asks for payment
    - approved – Date when Payment for order is approved.
    - paid – Date when the order amount is paid by the finance company.

    - Items – Array of a string containing details of order i.e. same as details stored in the stockroom.
    - Relationship with the retailer from the order is placed, a customer who places the order and for finance related aspect, with a finance company.
- As of both the order and customerOrder are having same properties we can use inheritance in modelling language and reduce the complexity.

2.3 Transactions

The transaction is defined in order to interact with assets and change its states. Various transaction and it's invoke origin is defined in Fig. 1.

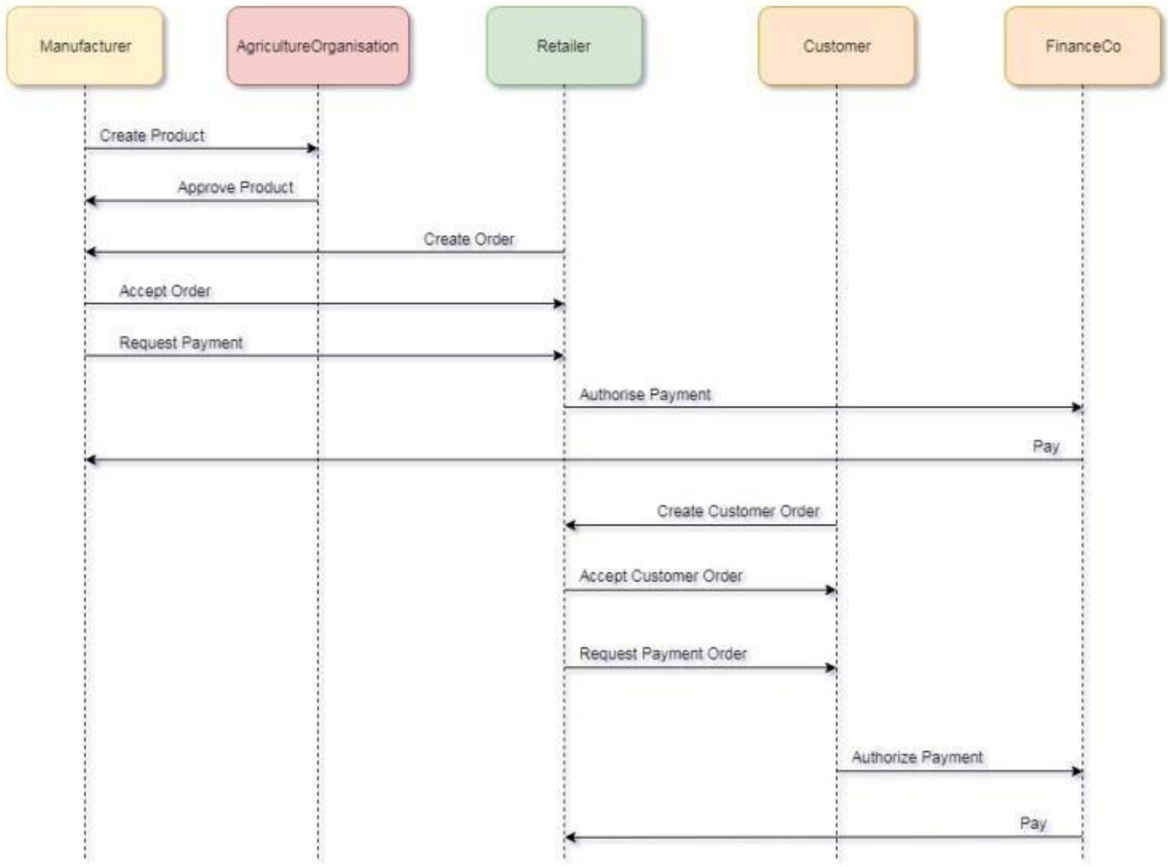

Figure 1: Participants and Transactions

Details of defines transactions are given below.

1. **CreateProduct:**

- Creates the asset Product with required parameters.
- Only manufacturer can create a product.

2. **RequestApproval***:*
   - Request to specific government Agriculture Organisation to approve the product to be used.
   - Only manufacturer can invoke this transaction.

3. **RejectProduct:**
   - Rejects the product because of any reason, but because of that no one can manufacture or sell that product on the network, or legally.
   - Only Agriculture Organisation can invoke this transaction.

4. **ApproveProduct:**
   - Approve a product to manufacture or sell with the rating of that product.
   - Only Agriculture Organisation can invoke this transaction.

5. **ManufactureProduct:**
   - If the product is approved then only transaction will happen.
   - This transaction will manufacture a batch of that product with provided quantity. It stores the information of batches as string as a parameter of Product asset.
   - Only Manufacturer can invoke this transaction.

6. **CreateOrder:**
   - The retailer will invoke this method to create order.
   - Only contains products that are approved.

7. **CancelOrder:**
   - It Cancels the created order if not accepted by manufacturer.
   - Only retailer can cancel the order.

8. **AcceptOrder:**
   - Manufacturer accepts the order by invoking this transaction.
   - This transaction will remove the batches that are produced and required to fulfil that order and move it to archives for future requirement. It also updates the value of the total quantity available for a product.
   - It is the same function to change or update the value of a stockroom of retailer, to add the details of the order to that.
   - If we want to implement the business logic of it then we have to add participants like Shipper and than, this function only updates the value of the product.

9. **RequestPayment:**

- After Accepting order, the manufacturer can invoke this transaction and request for payment of the order.

10. AuthorizePayment*:*
- After Accepting, order Retailer will authorise the payment for the order.

11. CreateCustomerOrder:
- Customer will invoke this method to create the Order from the retailer.
- It can create an order of a product from the different customer as long as the product is available in the stockroom of that order.

12. CancelCustomerOrder:
- The transaction for cancelling the order before it accepted.
- The only customer can invoke this transaction.

13. AcceptCustomerOrder:
- The retailer will invoke this transaction to accept the created order.
- This transaction will decrease the available product quantity and update the stockroom. It also adds the information of batch created date and quantity of that batch as a part of the order.
- Only retailer can invoke this transaction.

14. RequestCustomerPayment:
- The retailer can invoke this transaction only after order is accepted.
- It requests for payment to the customer.

15. AuthorizeRetailerPayment:
- Customer will authorise the payment for that order.

2.4 Assumptions

We have considered only one middle-man or broker as a retailer in a given scenario, while in real life there may be some levels. Also, we have assumed the end user to be the customer for which we are showing the trace of the product. For seeing trace it is necessary to provide retailer id.

3. Implementation

For implementation purpose followed the path is by ‘Zero to Blockchain’ – IBM Redbook course. We can divide the implementation into three parts.

1. Chain code
2. Node.js Server side Files
3. Node.js Client side Files

3.1 Chain Code

The first task of chain code is to define or conceptualise asset, participants and transactions. In hyperledger, we define that in the model folder. All the model files stay there. Every model file defines transactions requirements and not the code for the transaction. The actual code for the transaction will happen in files inside ‘lib’ folder. It will define ‘smart contract’ for all the transaction defined in the ‘model’ folder. In the modelling part, one has to decide access control in private blockchain network because of shared ledger technology. For that matter in hyperledger, two files are very important – 1. ‘permission.acl’, 2. ‘query.qry’. Permission file provides the information of which participant have permission to access which transaction and asset. If one has access to some the asset, but not all of it then we can use queries. Query files work like the first query provides the results from the current state and then the output of that query gets verified against the access control file. So, one can get only records for which one is authorised.

In the given scenario we are implementing the application with Node.js server and so it is also suggested to use mocha test framework to test the model. It tests the network asynchronously and from that, we can check if every transaction works fine before we develop server-side js files.

3.2 Node.JS Server side files

After checking the chain code using mocha, the next step in implementation is to create server-side files. This file stays on server fulfilling the queries coming from the client side files or browser. This file invoke the transaction from chain code. Router file provides the route to call a function based on the query coming from the client. In other words, it navigates the query. For querying or connecting the chain we will use given libraries from Node.js. We have created one file to handle all the queries from admin and the other one for handling a query from participants. Whenever a participant created, an Identity and a card will be given to it using which one can connect to the chain.

3.3 Node.JS Client side files

This files will be on the client’s web browser. It will maintain the design of the screen and also the events on the screen. For every Participant, there will be different js files. In this problem, we have not implemented the authorization process, but for better visualization purpose, whichever participant is selected it will log in as that participant.

- For more detailed implementation refer Appendix 1.

4. CONCLUSION

This paper shows a use case of blockchain technology. We can extend it by considering the ground rules of some real-life situation. This paper provides the information on a scenario implemented on the hyperledger composer. Adding product can be done using IoT and by this, we will have the precise and automated technology. One can extend the scenario by including the pre-process of manufacture product under the light and with going deep with it we will get a complete trace of product – from raw material to product to customer.

5. ACKNOWLEDGEMENT

This is to acknowledge the contribution of Dr Vikram Sorathia for guiding throughout the project on how to model and implement real-life scenarios on the blockchain, which leads to the current outcome of this project.