May 3rd , 2021

Genesis Trail: A distributed application built on the Hyperledger Fabric

Dhruva Panyam

Rathi Kashi

Contents

1. Introduction

Knowing where a consumer's food comes from is the first step in making healthy food choices. With the world going increasingly 'organic', consumers rely on certifications to verify origins of a product and its quality. These certifications however, are misleading since not all ingredients in a product have to be organic for the product to be given an organic certification. Furthermore, organic traceability has been riddled with issues including certification fraud and lack of transparency of food information. In light of these issues, solutions involving blockchain technologies were thought of, since they provide a transparent and secure solution by way of their immutable ledger. Therefore, a blockchain solution would involve recording all raw materials, components, purchases, and production actions on a digital ledger. This would allow users to trace a product back to its origin without any uncertainty since the ledger cannot be tampered with. In fact, Walmart and IBM ran a proof of concept of this very idea and it met with success. In this project, we aim to implement such a network that allows for creating and querying transactions stored on a blockchain network using the Hyperledger Fabric.

2. Project Description

Genesis Trail is a distributed application built on the Hyperledger Fabric, which aims to harness the accountability and transparency of blockchain networks to ensure the validity of transactions (which in our case, refers to the buying and selling of food materials). Our application thus allows for users to create, validate and query transactions.

In line with the above, our application consists of two modes: The consumer mode and the trader mode.

1. The consumer mode allows consumers to query a product stored on the blockchain network through its product ID. On querying a product, the consumer can view the lifecycle of a product - from it's raw materials to the final product along with the organic certifications of the farmers involved in cultivating the raw materials. Further, they can also track each raw material through all the vendors and manufacturers who used it.

2. The trader mode allows farmers, vendors and manufacturers to create transactions on the ledger (indicating the purchase or production of a raw material or product). This mode also allows the above users to view all of their past transactions, validate a transaction and keep stock of the product and materials they have with them.

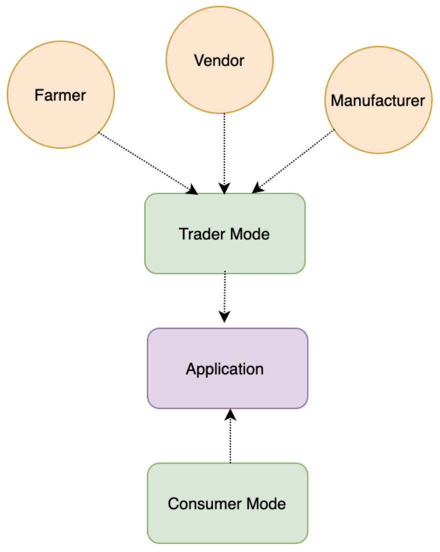


Figure 2.1: *User types*

3. Project Flow and Technical Details

The blockchain network is set up as follows: One orderer, certificate authority and peer. The chaincode is instantiated on the peer which the client application uses to access the ledger which is also stored on the peer. The client application has various user types - Farmer, Manufacturer, Vendor and Customer. The first three require a login to update transactions on the ledger.

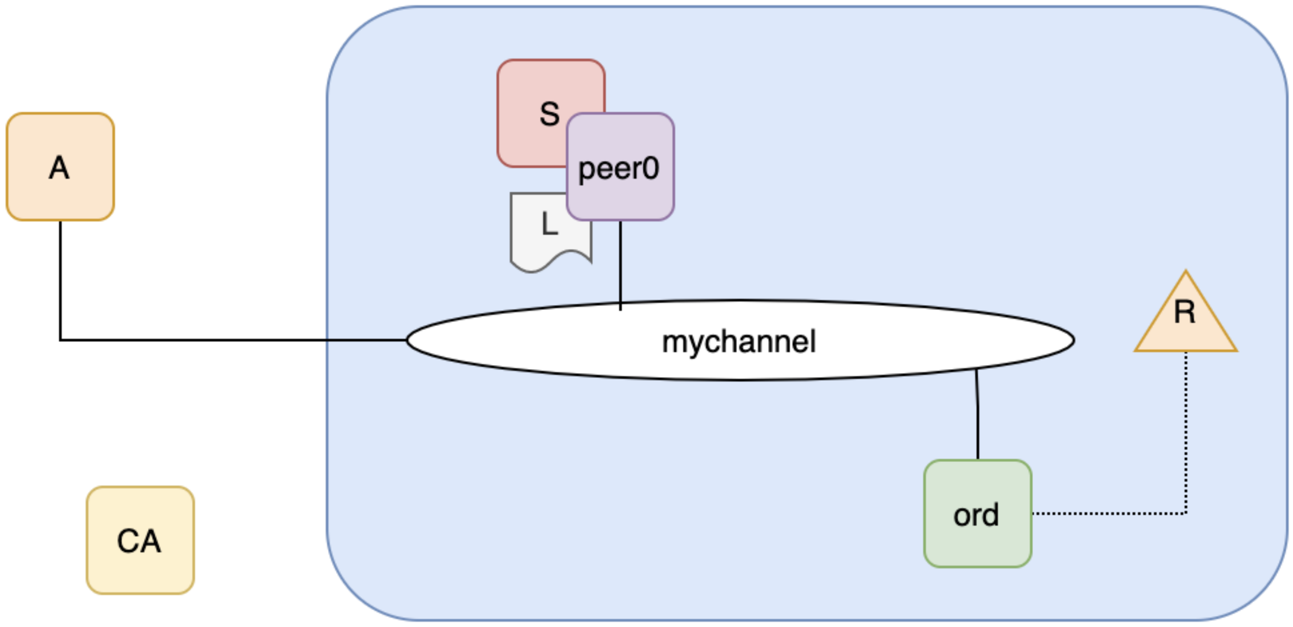
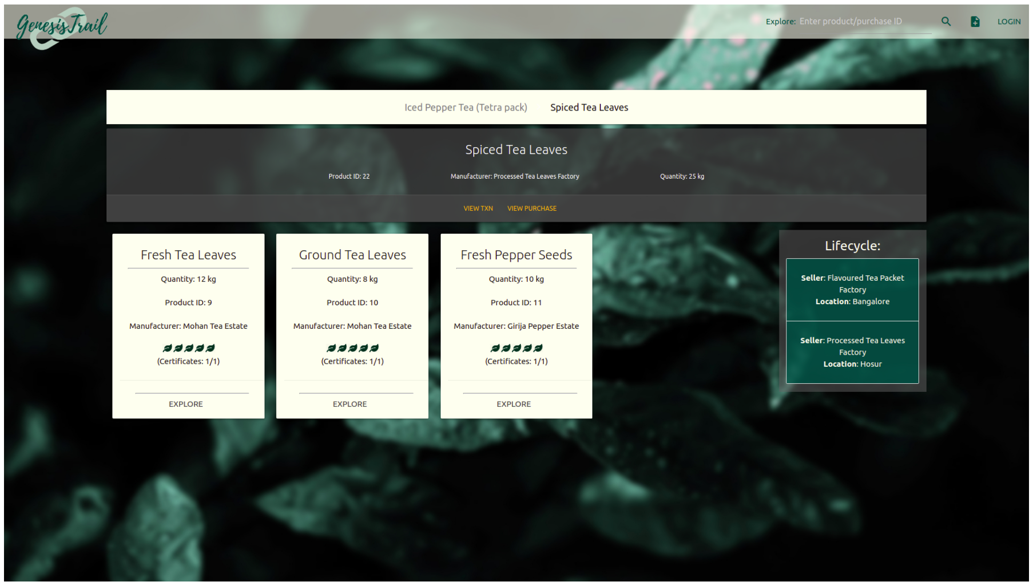


Figure 3.1: *The network structure is as follows: 1 organisation (R) with 1 orderer, 1 certificate authority, 1 channel with 1 peer. The chaincode/smart contract installed on the peer is S and the associated ledger is L. The application A (in organisation R) uses the chaincode S to access the ledger L via the peer.*

On loading the web application, a user can log in to their existing account, or register as a certain user type - farmer, vendor or manufacturer. Logging in allows the user to view their current inventory of products, their past transactions, and purchases that they must validate. They can also create new transactions or products based on their user role. Each of these transactions will be explained in further sections.

Any user can explore products without having to log in as a user on the network. Every product has a unique ID (*product ID*), and at each different stage of the product's location/possession, it has another unique ID (*purchase ID*). Users can explore a product by its product ID, or its purchase ID. Exploring a product will show the user the product's details, its ingredients, and its path from the manufacturer to the current owner. Each ingredient shows how many of its raw materials have an organic certificate linked. These ingredients can then further be explored to reveal their details, ingredients and path. This exploration allows a user to trace any ingredient of a product back to its raw form, and view the associated farmer's organic certification.



Spiced Tea Leaves
Product ID: 22
Manufacturer: Processed Tea Leaves Factory
Quantity: 25 kg

Fresh Tea Leaves
Quantity: 12 kg
Product ID: 9
Manufacturer: Mohan Tea Estate
(Certificates: 1/1)

Ground Tea Leaves
Quantity: 8 kg
Product ID: 10
Manufacturer: Mohan Tea Estate
(Certificates: 1/1)

Fresh Pepper Seeds
Quantity: 10 kg
Product ID: 11
Manufacturer: Girija Pepper Estate
(Certificates: 1/1)

Seller: Flavoured Tea Packet Factory
Location: Bangalore

Seller: Processed Tea Leaves Factory
Location: Hosur

Figure 3.2: *Explore page on the web application.*

3.1 Chainstate Assets / Transactions

The application stores 4 types of assets: **users**, **raw materials**, **purchases**, and **productions**. This section will explain each of these types briefly.

3.1.1 Users

When a user registers onto the network, an asset is created on the chainstate, which contains their relevant data. This asset ((key, value) pair) is stored at some key, which is the user's permanent user ID. The data at this key contains the user's metadata, like full name, location, organic certification (if applicable). It also stores the current inventory of the user. The inventory is a dictionary for all the products that the user currently owns. It keeps track of how much of each product the user has left. Finally, the asset contains lists that maintain the user's transaction history, and their pending transactions.

Key	Value
UserID	Metadata, History, Pending, Storage

Asset for a User Transaction

Metadata	Full name, Location, Organic certification, Password
History	Type (Raw, Purchase (Sold), Production), TxnID
Pending	Type (Purchase (Bought)), TxnID
Storage	TxnID: Quantity, Unit Type

Description of each Value

3.1.2 Raw Materials

A farmer who has registered onto the network is allowed to create a raw material. On providing the material's name, the quantity produced, and having logged in, a raw material asset is created on the chainstate. This asset ((key, value) pair) is stored at some key, which identifies the raw material as its product ID.

Key	Value
TxnID	Type (Raw), Metadata, Certification, Unit Type, Quantity

Asset for a Raw Material Transaction

Type	Raw, Purchase, Production (In this case, Raw)
Metadata	UserID, Product name, Producer (User Full name)
Certification	Organic Certification of Producer
Unit Type	Kg/L/etc.
Quantity	Amount of Raw material produced

Description of each Value

3.1.3 Purchases

Any user has the ability to sell any product from their inventory. This sale is recorded as an asset, which contains the following fields: product data, product ID, quantity sold, and a purchase link. This purchase link links the sale to the asset where the seller bought this particular product. For example, if vendor V_2 buys product with ID P from vendor V_1, then this sale's purchase link will contain the key of the asset where V_1 buys P. To give you a better understanding, consider the following tables.

Key	Value
TxnID	Type (Purchase), Metadata, ProductID, PurchaseID, Validated, Quantity

Asset for a Purchase Transaction

Type	Raw, Purchase, Production (In this case, Purchase)
Metadata	BuyerID, SellerID, Product name
ProductID	TxnID of the raw material/product being sold
PurchaseID	TxnID of the previous purchase of this product
Validated	True or False (True, when the buyer has approved the transaction)
Quantity	Amount of product sold

Description of each Value

3.1.4 Productions

A manufacturer is allowed to create productions, which represent produced food items. These assets contain the product data and ID, its manufacturer, a list of ingredients, and the quantity produced. Each ingredient (sub-product) contains the sub-product's product ID, and last purchase ID, and quantity used.

Key	Value
TxnID	Type (Production), Metadata, Sub Products, Unit Type, Quantity

Asset for a Production Transaction

Type	Raw, Purchase, Production (In this case, Production)
Metadata	UserID, Product name, Producer (User Full name)
Sub Products	Array of sub products used, each element consists of: ProductID, PurchaseID, Quantity
Unit Type	Kg/L/etc.
Quantity	Amount of product produced

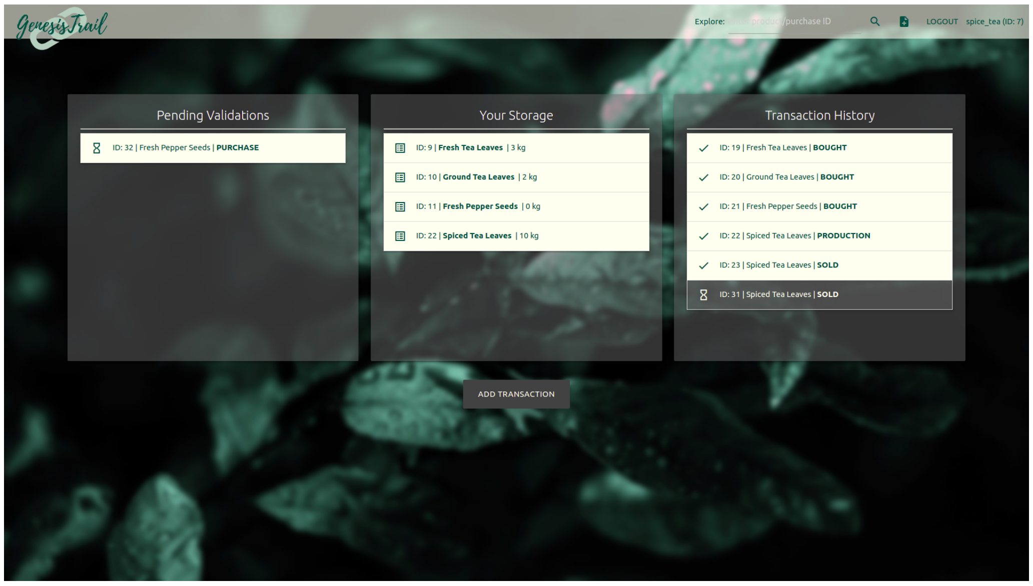
Description of each Value

3.2 Asset Updates

The assets that are created as per the previous section can undergo certain changes:

1. Consider a user data asset. This asset has fields that maintain a list of past assets. This list is updated every time the user creates a new asset.

2. When a sale asset is created, it is considered valid only if the buyer marks the asset as valid. So, when a sale asset is made, it gets added to the buyer's list of pending purchases. Once this sale is validated by the buyer, it moves from their pending list to the history list. The asset also updates its 'validated' field to True.



Pending Validations

- ID: 32 | Fresh Pepper Seeds | **PURCHASE**

Your Storage

- ID: 9 | **Fresh Tea Leaves** | 3 kg
- ID: 10 | **Ground Tea Leaves** | 2 kg
- ID: 11 | **Fresh Pepper Seeds** | 0 kg
- ID: 22 | **Spiced Tea Leaves** | 10 kg

Transaction History

- ✓ ID: 19 | Fresh Tea Leaves | **BOUGHT**
- ✓ ID: 20 | Ground Tea Leaves | **BOUGHT**
- ✓ ID: 21 | Fresh Pepper Seeds | **BOUGHT**
- ✓ ID: 22 | Spiced Tea Leaves | **PRODUCTION**
- ✓ ID: 23 | Spiced Tea Leaves | **SOLD**
- ID: 31 | Spiced Tea Leaves | **SOLD**

Figure 3.3: *The dashboard of the web application, with access to validate transactions ad view your storage and transaction history.*

3.3 Product Explore (Pseudocode)

The following describes the function to query a product's history:

- To query the history for a particular product ID, we must find the path of purchases made, to lead us from the manufacturer all the way to the farmer, depending on which subproduct's path we choose each time. The response will be a dictionary indexed by product IDs that were used to make this product.

- Each entry in this dictionary will correspond to a product, its lifecycle (all purchases leading to the manufacturer) and subproducts used to make it. These subproducts will then yield more objects in the dictionary.

```
func getHistory(id):
    product = getChainstate(id)
    lifecycle = expandLinkedList(product's parent
                                 product)
    subproducts = product.sub_products
    addToDictionary({product_name, lifecycle,
                    subproducts})
    getHistory(subID) for each subID in subproducts
```

All functions in the chaincode perform checks on the data received to ensure that the requested service is valid. For instance, a user cannot sell a product that they do not possess, a user cannot validate a sale they are not involved in, etc.

3.4 Login

When a user registers onto the network, they must provide a username and password, among other fields. These fields are stored as attributes in the user's wallet, with an encrypted certificate containing the user's data. To log in to the web application, the login details are checked against the specified certificate for access to the application.

3.5 Web Application

This section describes the web application's views, along with the relevant data that is fetched.

The landing page leads to either the login page or the explore page. If a user explores a product, a query is submitted to the chaincode, and the received data is formatted onto the webpage. After a user logs in, they are redirected to their dashboard, where they see their asset history, current inventory, and pending validations. To acquire these, the chaincode receives the corresponding query requests for the user's data. To add an asset (raw material, sale or production), there is a separate webpage which shows the user their product inventory, and allows them to select one or more of these products to sell or combine to make a production.

The app was made using the web framework Flask to manage the various templates, and maintain user data variables and request data from the chaincode.

4. Future Work

While this project demonstrates a fully functioning demo of the application's core features and design, to make it usable in a widespread manner, there are several things to add:

1. *Farmer Authorization*: To verify a farmer's organic certification number, we require special users who are authorized to verify them. This would make the registration process more robust, since farmers need a digital signature from the verifying authority, which makes their account verifiable by anyone.

2. *Manufactured Goods*: Currently, manufacturers can create productions using the products that they possess. However, the manufacturing process is not being overseen by an authority. To ensure that manufacturers' productions are authentic, we require authorities similar to those verifying farmers. These authorities would verify the contents of each production as well as the metadata being issued.

3. *ID tracking*: In order to scale the application, each product must be identifiable to any consumer easily. Further, when a sale is made, a new ID must be created to log that sale. The use of special QR codes can be used on product purchases and productions. These QR codes, on scanning, can redirect a user to the explore page of the product in question and view the details instantly.

4. *Payment Security*: Finally, purchases can either by made through the application, or using normal currency. In either case, the security of a payment must be maintained. For instance, in cases where the product is sent to the buyer, but the payment is not, or vice versa, there must be an arbitrator to oversee the transaction. Only upon successful payment can the sale asset be created.