FINAL REPORT

MDS 6117 Blockchain

Blockchain-based Battery Traceability System

Group 1: Digital Transformation Team

CHEN Honghao, 119010014

LI Jiaqi, 220041009

LI Jingyu, 118010141

ZHANG Mingzhe, 118020526 (Team Leader)

Contents

Executive Summary

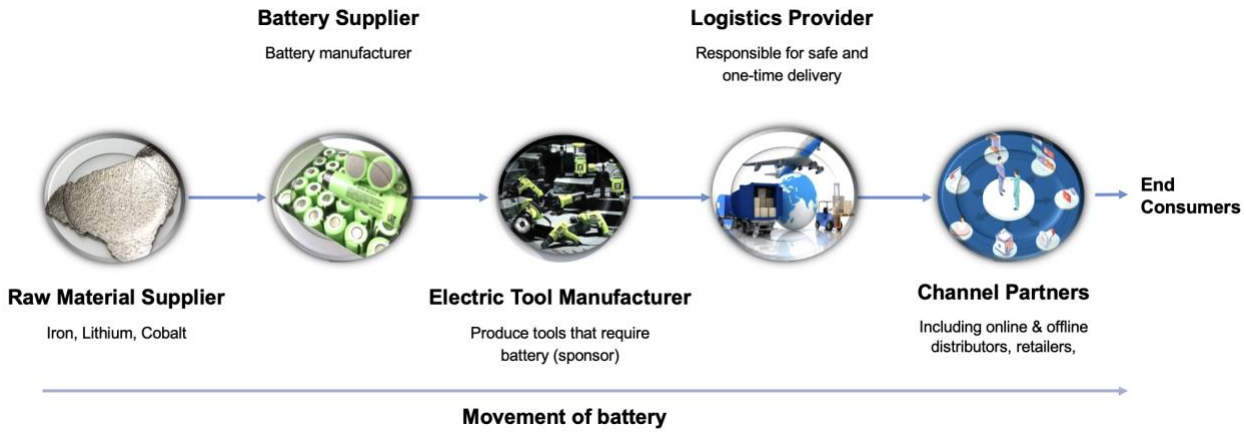
The battery industry is currently undergoing fierce competition enduring the bottleneck of battery supplies. The traceability and safety of battery supply have been industrial pain points for an extended period. Our project provides the sponsor with a blockchain-based traceability system to improve the supply chain visibility and responsiveness. The traceability system is based on a novel technology, blockchain, which drives decentralization and real-time data exchange among supply chain partners. Our blockchain solution is a good fit to solve the supply chain traceability problem based on the Blockchain Assessment Framework. Therefore, we develop the overall framework of the blockchain database system as the initial point of further project development.

1 Project Introduction

1.1 Sponsor Background

The sponsor of the project is an electronic tool company, with a focus on group purchase of battery-related raw materials and electronic instrument selling business. The battery supply chain of the company is demonstrated in **Exhibit 1**.

Exhibit 1. Battery & Electronic Tool Industry Supply Chain



1.2 Requirements Identification

Battery is expensive, complex, and requires specific manufacturing and logistics conditions. Meanwhile, there exists a high risk of the explosion of batteries, which contributes to huge cost and safety concerns to our sponsor and all supply chain partners. The current situation is, due to the potential risk of explosion, the company needs an efficient and reliable battery traceability system to monitor the production and logistics status of batteries throughout the supply chain.

The need of traceability is composed of three dimensions:

- Be able to monitor and control the battery production and selling process.
- Ensure the records are reliable and easily retrieved, and clear the allocation of responsibility.
- Share real-time logistics status to entities, enhancing supply chain visibility and trust.

1.3 Project Objective

Based on the needs analysis, our Digital Transformation Team proposes a blockchain-based information

system to trace the movement of batteries along the supply chain to achieve the main objective of **making the movement and conditions of batteries traceable and strengthening battery supply chain visibility**. Through the blockchain system, we expect to complete two value-added features that traditional databases cannot provide *at the operational level*:

- Enable the reliability and extractability of battery data.
- Enable real-time monitor of battery status.

At the strategical level:

- Build trust and partnership among supply chain partners.
- Improve visibility and responsiveness and the overall supply chain performance.

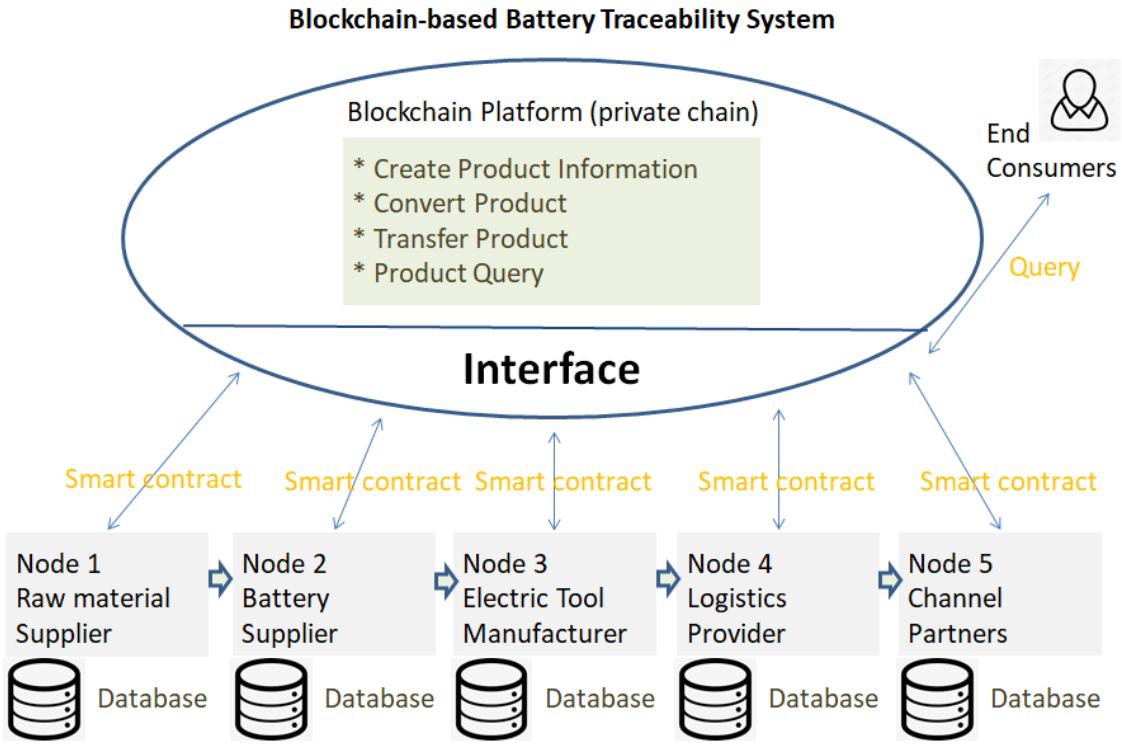
The logic of applying blockchain system is,

- First, blockchain is a decentralized system, and it is a reliable medium to share battery information to supply chain entities and boost trust.
- Also, blockchain has the properties of efficient and automated transaction processing, plus tamper-proofing data storage, which benefits the company by cutting labor costs and operations lead time.
- Moreover, blockchain is ultimately an information system to improve supply chain transparency/visibility. The visibility of battery status empowers the company to monitor and control the production and logistics process and equips the company with information needed to execute the battery recycling process.

1.4 System Overview

As shown in **Exhibit 2**, we build a blockchain-based system as a decentralized database shared among supply chain partners. Generally speaking, the ellipse represents the blockchain platform where different roles can perform the corresponding functions to influence the chain (e.g., create product information, query product, Convert / transfer product). The five nodes (participants) are listed below, namely the raw material supplier, battery supplier, electric tool manufacturer, logistics provider and channel partners. Each node represents a role during the battery processing procedure and possesses a distributed database containing all product information (assets). Individual nodes have interactions (transactions) with the blockchain platform interface through the smart contract mechanism. In addition, end customers can query product information through the platform. The detail of the use cases, data organization, data flow, and product transfer procedure will be shown in the following part.

Exhibit 2 Blockchain System Overview



2 Project description

2.1 Use case analysis

2.1.1 Industry Description

Electric power tools refer to tools that drive the working head through a transmission device. It has many different types, for example, electric drills, electric saws, electric wrenches, electric hammers, heat guns, etc. At present, the electric power tool industry concentration is low, and the market competition is fierce. To fight for market share, companies are committed to improving their capabilities in research, development, production, quality, marketing, business management, and other aspects. High product safety can contribute a lot to company's R&D efficiency, product quality, and corporate reputation. Electric power tools can be divided into AC power tools and lithium battery power tools in terms of power supply. The convenience of lithium battery power supply is far superior to that of AC power supply, so it has become the future development trend. However, the electrode materials and electrolytes of lithium batteries are flammable. A lithium battery of poor quality can easily cause combustion and explosion when being affected by heat. The anode and cathode materials, electrolyte and its additives, the

structure of the battery, and the manufacturing process had an important effect on the safety of the Li-ion battery. The environment in which the battery is transported and stored also affects the safety of the battery. Therefore, to ensure the safety of lithium batteries on the electric power tools, the companies must know and supervise the production process in their battery suppliers adequately. Besides, they also stipulate safety precautions for storage, transportation, and charging of electric power tools, such as temperature range, charging duration, etc.

2.1.2 Industry Pain Points and Challenges

Although the electric power tool companies understand the importance of safety of lithium battery on their products, they still have many obstacles to supervise the production process of the batteries. For lithium battery manufacturers, the core technology of lithium battery manufacturing is a corporate secret. Therefore, it is impossible to allow the demand side to send people to supervise production. They will only provide the demander with a third-party quality inspection report to prove that their products meet the requirements. This is also the mode that is widely adopted at present. However, there exists some pain points and challenges in this pattern:

- Lithium battery manufacturers may collude with third-party quality inspection agencies to issue false quality inspection reports. This may allow products that would otherwise not meet the standards to enter the market.
- Even if the quality inspection report itself is true, the performance it can detect cannot fully reflect the safety of lithium batteries. The manufacturer may pass the quality inspection through various means, but the product does not actually meet the standard (such as the melamine milk incident in the food field).
- For the manufacturers themselves, it is difficult to fully guarantee the quality of the raw materials and parts used in the production of batteries.
- The quality inspection of lithium batteries requires the manufacturer to pay a fee.
- The acquisition of the quality inspection report requires many steps, including the manufacturer's application, the manufacturer's mailing of samples, the quality inspection agency's quality inspection, and the mailing of the report after the inspection is completed, which often takes several weeks.
- As for the transportation, storage, and charging process of electric power tools, although electric power tool companies realize their importance to battery safety, it is quite challenging to control these related processes. The reason is that logistic providers conduct transportation, and channel partners run storage. Full supervision is not realistic. The current mode is that the

companies enclose detailed product specifications with the products. However, whether logistic providers and channel partners operate according to the product specification is not guaranteed.

2.1.3 Blockchain Assessment Framework

The below assessment table shows the feasibility of Blockchain-based Battery Traceability System in the battery supply chain industry.

Exhibit 3 Blockchain Assessment Framework

INDUSTRY FOR BLOCKCHAIN APPLICATION: Battery Traceability System			
Number	Factors	Assessment Framework	Does Blockchain Fit? Why?
1	Intermediary	l High fees for intermediary? l Latency due to processing through intermediary? l Does the intermediary exist due to lack of a trust?	Yes. Traditional quality supervision pattern highly relies on third-party inspection agencies due to lack of trust. Obtaining the report usually costs weeks. Every procedure in the supply chain can plug into the blockchain and record production and transportation details, which is a powerful supplement to inspection reports. After being mature, electric power tool companies can even purchase high-quality batteries without third-party's quality inspection.
2	Transparency	l Are multiple participants involved? l Does increase in transparency into the transaction help the participants?	Yes. Electric power tool companies, lithium battery manufacturers, raw material suppliers, logistic providers and channel partners are all involved. Higher transparency can help each downstream customer control their product quality by controlling the quality of product parts.

3	Golden Source	l Are multiple participants storing the same information in multiple locations? l Is data consistency an issue?	Yes. Common information like product id, location, timestamp needs to be stored by all participants in this supply chain. Blockchain can guarantee data consistency in all participants' servers. Product information has to be gathered from multiple sources for tracing.
4	Manual Processing	l Does the process involved manual operations? l Is the cost of reconciliation high?	Yes. Original lithium battery products and electric power tool transportation need to record many paper-intensive data like machine parameters, temperatures, which requires lots of manpower.
5	Trust	l Is there trust among participants? l Do multiple participants have the right to modify transactions? l Is there a risk of fraudulent transactions?	Yes. Lack of trust exists between battery suppliers and downstream electric power tool companies, as well as logistic providers, and channel partners. The risk of inferior batteries does exist because suppliers can get a passed quality inspection report for an inferior battery through any means. The risk of irregular storage and transportation for electric power tools also exists. Since information may be unknown to each other, there is a lack of trust and the possibility of fraudulent activities.
6	Documentation	l Is the documentation paper-based? l Is there a large number of documents/reports required to be generated (e.g. for	Yes. The product information, validations report, quality inspection report, logistic address, bills, the recordings of the process of quality inspection, etc. are all paper-

		regulatory purposed)?	based and require documentation.
7	Time Sensitivity	l Will the transactions benefit from being real-time or synchronous?	Yes. Every participant will benefit from real-time data visibility. It will help in providing enhanced customer experience and responsibility confirmation process, and reduce the exposure risk of quality problems if the recording process is in real-time.

2.1.4 Advantages and Disadvantages of Blockchain Application in The Battery Traceability System

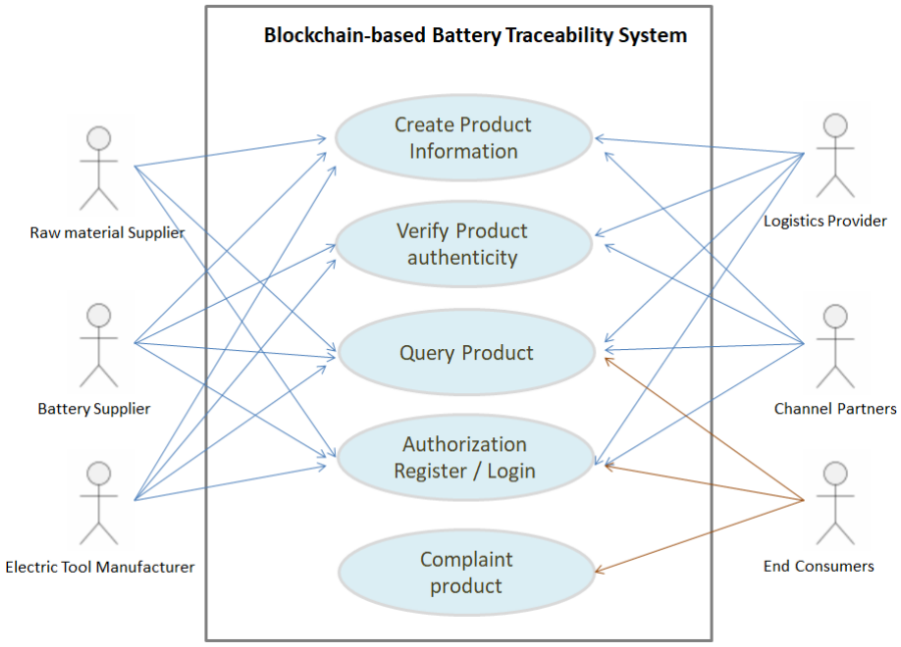
Based on the blockchain assessment framework, the advantages and disadvantages of the battery traceability system can be summarized. The battery traceability system based on blockchain has many advantages comparing with traditional quality inspection. First, the blockchain has the tamper-proof feature. When the production equipment is connected to the blockchain, the system can truly record all the processes of battery production, including raw material composition, production environment parameters, whether key operations to prevent explosions have been carried out, etc. It can be a powerful supplement to inspection report. After being mature, electric power tool companies can even purchase high-quality batteries without third-party's quality inspection. Blockchain can also record the process of battery transportation and storage. If irregular operation occurs, the electric power tool companies can remind the logistic providers and partners to make them correct the errors in operation. Second, the records in the blockchain are fully traceable. Once an accident occurs during the use of the product, the production, transportation and storage data can be quickly analyzed to investigate the cause and prosecute those responsible. Third, the generation of the records are nearly in real time, which saves a lot of time.

However, the disadvantages of the battery traceability system cannot be ignored. First, in this system, every participant's production process is visible, which may cause leaking of the secrets about the product design. Second, the data in the blockchain is highly duplicated. Every node in the system stores all the production data. As the quantity of the products increase, the nodes will store more and more data, which will be a burden for the storage hardware of the nodes. Third, the set-up of this traceability system may be difficult and expensive. Since blockchain application has not been popularized in supply chain, lots of things need to be done in the initial phase of the blockchain network construction of battery supply chain

2.1.5 Use case diagram

The following figure shows the use case diagram of our blockchain-based battery traceability system.

Exhibit 4 Use Case Diagram



There are mainly five operations:

1. Create product information: The participants in the supply chain can create information regarding the product. For example, raw material supplier can create copper information and upload a unique id and its purity; logistic provider can create logistic information about the product location, timestamp; Channel partners can upload transaction information. Since the data in a blockchain cannot be tampered, modifying product information is equivalent with creating new information. End customers cannot upload product information since they don't produce.
2. Transfer product: Every participant in the supply chain except the end consumers can transfer the product to downstream participant, while the receiver can verify whether the information from upstream is authentic so that it can proceed in the following steps. For example, raw material supplier can transfer their iron or lithium to next participant battery supplier and battery supplier can transfer battery to electric tool manufacturer for future production. The end customer belongs to lowest stream and cannot transfer the product. The core change is the data belonging attribute, where the owner changes from the original one to a new participant.
3. Convert product: The raw material supplier, batter supplier, and electric tool manufacturer have the ability to convert product into another form. For example, from raw material to battery, from battery

to electric tool. Converting a product does not change product's belonging but change its property. Only the left hand side participants can execute this use case.

4. Query product: Every participant can perform query on a product. Participants can use the information to arrange business strategy or identify responsibility. Customers mainly use this use case to trace product information. The function will return a complete list of the product information like its history owners, timestamp, attributes, quality, and so on.
5. Complaint product: This is the special function for end customers. If they find quality problems of a product, they can complaint through the system. Relevant bureau in charge / inspection agency / association of consumers will deal with the case and try to solve the problem. The function serves as the function work.

2.2 Data Models

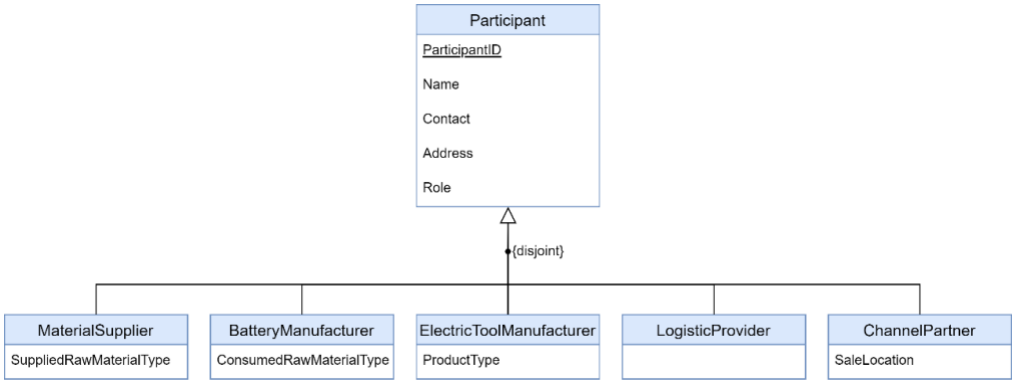
2.2.1 Entities

In this Blockchain-based Battery Traceability System, the entities can be divided into four categories, i.e., participant, battery raw material, battery, electric tool.

Participant

Participants are members of the business network. Each participant is associated with a specific role: material supplier, battery manufacturer, electric tool manufacturer, logistic provider, and channel partners. Different roles lead to different attributes stored in the database and different actions they can perform. Specialization is used in designing participant entities. The resulting design is shown in **Exhibit 5**.

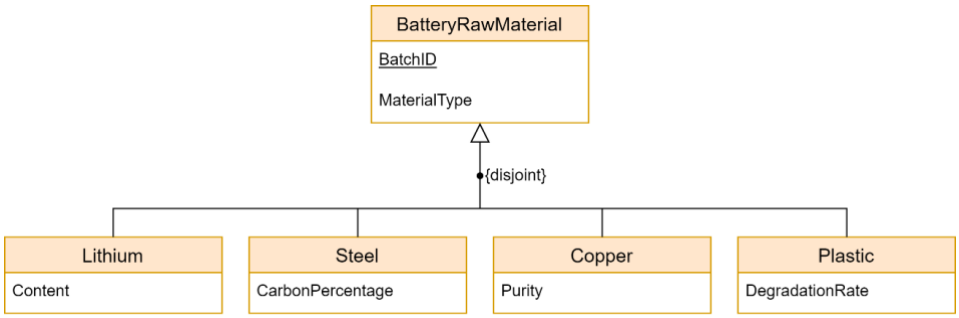
Exhibit 5 Composition of Participant Entity



Battery Raw Material

Battery raw materials are the materials used by battery manufacturers to produce batteries. There are several types of raw materials, including lithium, steel, copper, plastic, etc. Similar to participant entity, specialization is used in designing raw material entity, as shown below.

Exhibit 6 Composition of Battery Raw Material Entity



Battery

Batteries are produced by battery manufacturers from raw materials and are then used by electric tool manufacturers to produce electric tools. The producing process of batteries s kept track of by our system. The entity representation of battery will be shown in the complete Entity-Relationship Diagram later.

Electric Tool

Electric tools are produced by electric tool manufacturers using batteries from battery manufacturers. The entity representation of electric tool will be shown in the complete Entity-Relationship Diagram later.

2.2.2 Entity-Relationship Diagram

Linking the entities above with relations, the complete Entity-Relationship Diagram can be obtained as follow. Note that to simplify and increase readability of the diagram, only two of the raw materials presents in the figure. More raw materials can be added to the system following a similar manner. Detailed demonstrations of the entity-relationship diagram are shown in **Exhibit 7.** The relations linking different entities represent participants' actions (transactions) on assets (including battery raw materials, batteries, and electric tools). The relations can be categorized into three types:

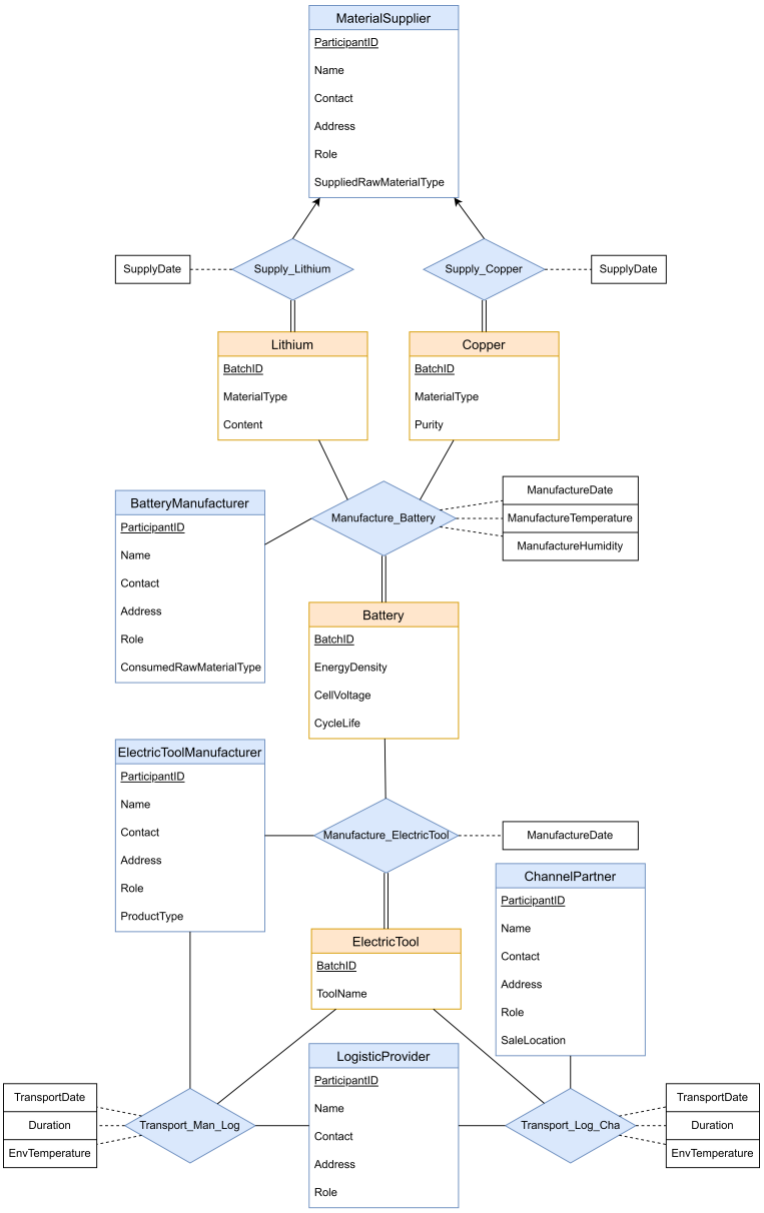
- Supplying Raw Materials

  New raw materials are introduced in the system by suppliers and are ready to be used to produce batteries.
- Manufacturing Battery / Electric Tool

  Manufacturers consume raw materials to produce batteries or consume batteries to produce electric tools.
- Transporting Electric Tool

  Electric tools are transported through the supply chain and finally delivered to consumers. manufacturers, logistic providers, and channel partners participate in the transportation process.

Exhibit 7 Entity-Relationship Diagram



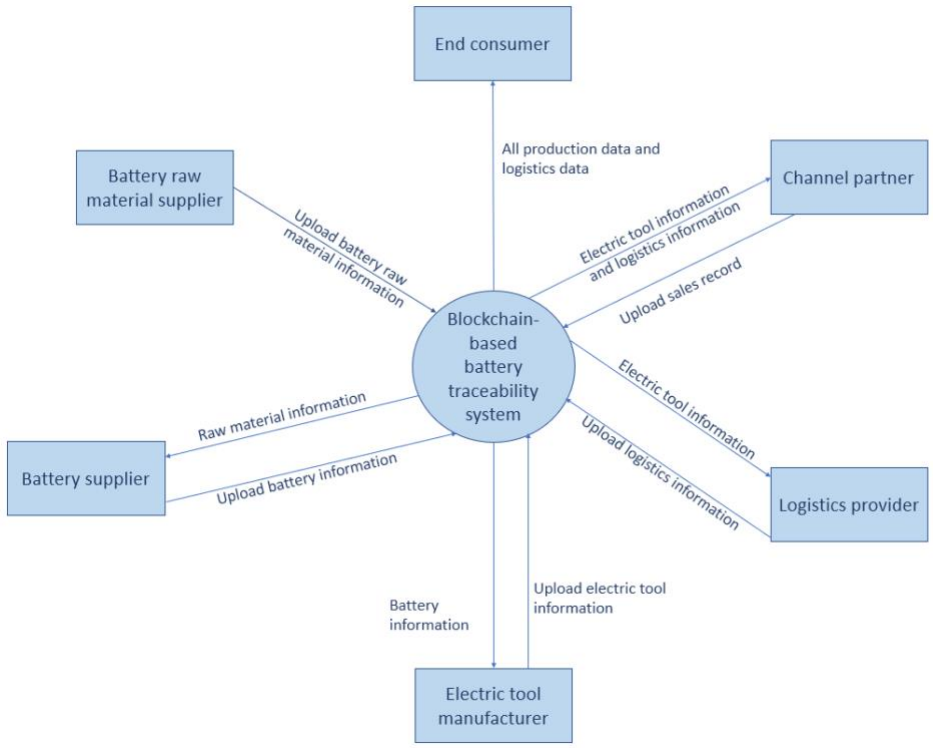
2.3 Dataflow diagram

The data flow diagram is shown in exhibit 2. We have seven kinds of participants, including six users and the system. Six users are battery raw material supplier, battery supplier, electric tool manufacturer, logistics provider, channel partner, end consumer.

All users except end consumer can upload sends information flow to the system to record the process of the production, transportation or marketing.

All users except the battery raw material supplier, which is in the most upper position in the industry chain, can query information from the system. Battery supplier and electric tool manufacturer can check whether their upstream products are up to standards or not. Logistic provider can also check electric tools' quality record. This behavior is quite useful in responsibility investigation if the products shipped to the destination has some quality problem. Due to the same motivation, channel partner is also authorized to check the electric tool information and logistics information. And end consumer can check all data generated in the production and transportation process of the electric tools.

Exhibit 8 Data Flow Diagram

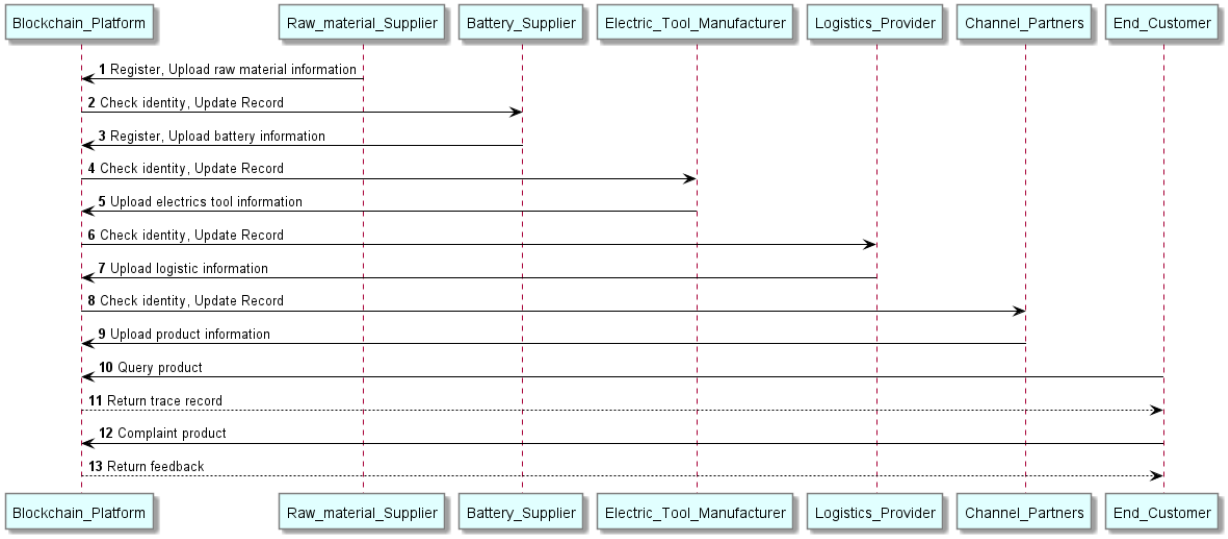


2.4 Sequence diagram

The **Exhibit 9** shows the sequence diagram of the system. Upstream participant raw material supplier registers to the system, and try to upload material information. The platform will verify the identity and broadcast the information to all other nodes using gossip mechanism. Once the next participant in the supply chain, the battery supplier receives the information, it can register and check the authenticity of the material, and then create new information in its level. By passing information to next and making use of previous one's, we transfer the product to the channel partners step by step. Different data from each part of the supply chain are gathered through this process (product id, location, material content, cost,

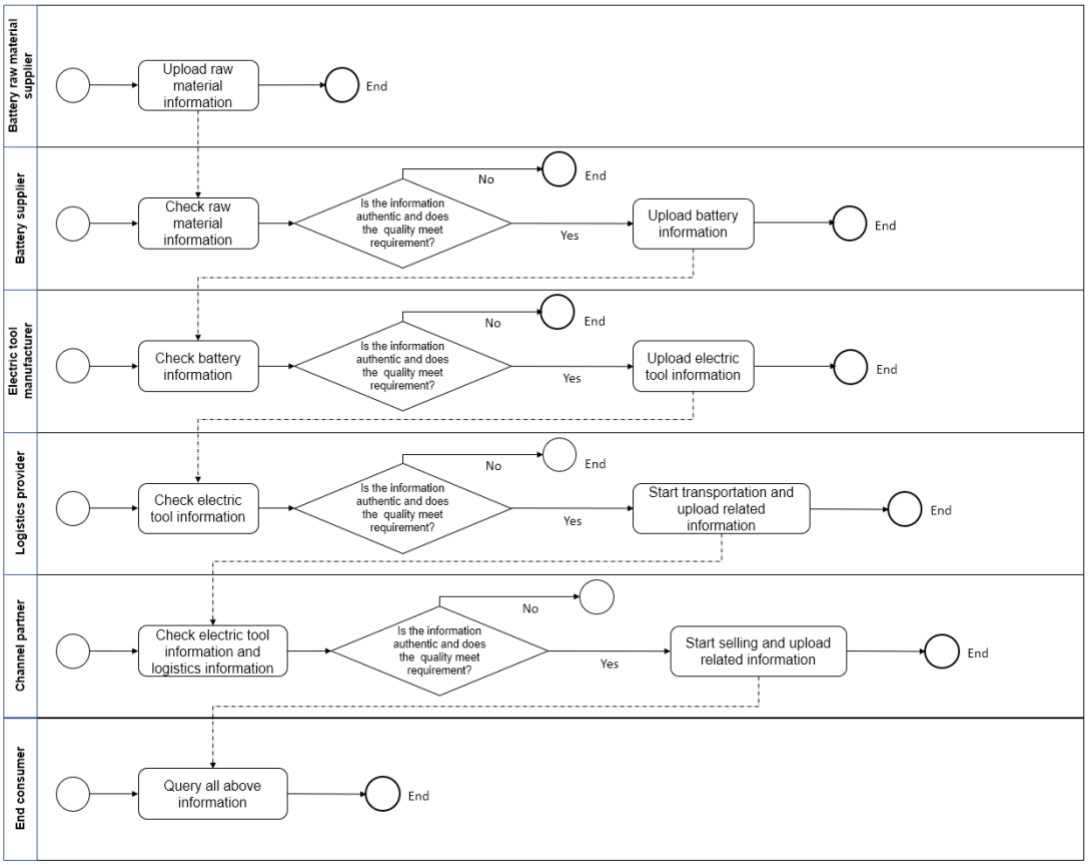
timestamp...). Finally, end customers can trace product with identification like product id or product QR code. If customers find quality problem, they can complaint about it.

Exhibit 9 Sequence Diagram



2.5 Information diagram

Exhibit 10 Information Flow Diagram



The **exhibit 10** shows the information flow diagram. The information flow starts from battery raw material supplier's uploading raw material information. After that, the battery supplier needs to check the raw material information and see whether the information is authentic, and the quality recorded meets the requirement. If the answer is no, the whole process will be aborted. If the answer is yes, the battery supplier will use the raw material to product battery and upload the battery information.

The next step is electric tool manufacturer's checking battery information. If the information is authentic and the quality recorded meets the requirement, the manufacturer will begin the production of electric tools, then upload related information.

After that, the logistics supplier will check the electric tool information. If there is no problem with the data, it will start transportation and upload logistics information.

The logistics information and electric tool information will be both checked by the channel partner. If the logistics storage conditions and quality of the electric tool meets the requirement, the channel partner will start selling the tools and upload related records.

The whole information flow is ended at the end consumer part. The end consumer purchases an electric tool and check all the information about that tool, including the production of batteries, the tool itself, the logistic and storage process.

3 Implementation of Blockchain Application

For the implementation part, our group first used the Hyperledger Composer to validate the logic of our smart contract. Then we use Golang to re-implement the smart contract. The source codes and complete test cases have been attached, and you can also find them in our Github repository: **https://github.com/Chen-Gary/Traceability-System.**

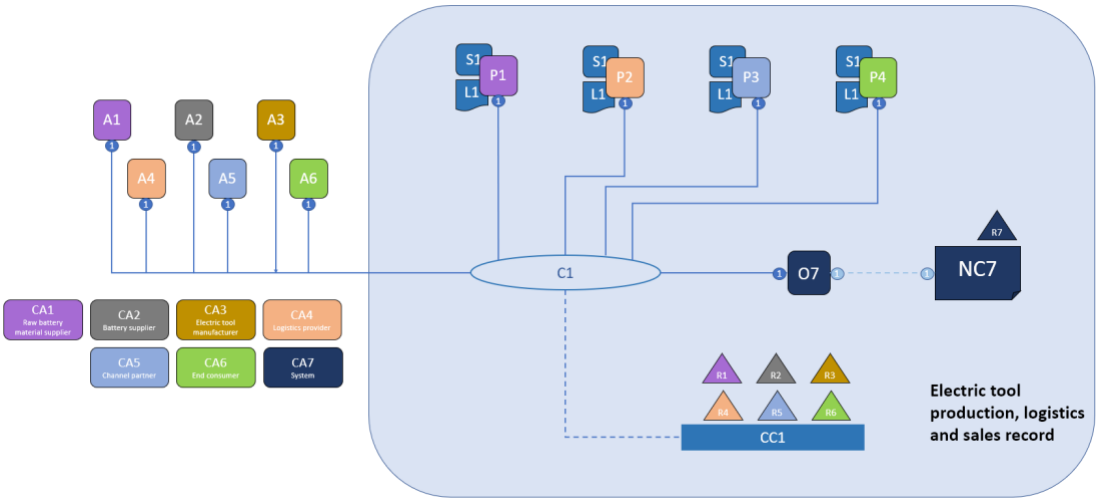
If you find it hard to deploy the environment, please login our server for fast test and running.

Login account: ssh alice@49.235.72.8

Password: alice6117

3.1 Blockchain Network Diagram

Exhibit 11 Blockchain Network Diagram



This is the blockchain network diagram. The network is managed according to the rules in network configuration NC7, which is controlled by the system, R7.

The other six organizations, from R1 to R6, are battery raw material supplier, battery supplier, electric tool manufacturer, logistics provider, channel partner, end consumer, respectively. The control the only channel C1 with the configuration CC1. Certificate authorities, from CA1 to CA7, issues the certificates for the six users and the system. A1, A2, A3, A4, A5, A6 are the applications of battery raw material supplier, battery supplier, electric tool manufacturer, logistics provider, channel partner, end consumer, respectively. These applications connect four peer nodes through the C1 channel. The peers share the same smart contract S1 and the same ledger L1.

3.2 Implementation with Hyperledger Composer

3.2.1 Implementation details

According to the design above, the key functions of the smart contract are listed in the following table, which contains six types of transactions. Note that customer querying should not be regarded as a transaction in general since there are no information updates on the blockchain. However, querying is implemented as a transaction, and the details will be explained later.

Exhibit 12 Key Functions of Smart Contracts

Participant	Asset	Action
Material Supplier	Raw Material	Create asset information
Battery Supplier	Raw Material, Battery	Convert raw materials to batteries
Electronic Tool Manufacturer	Battery, Electric Tool	Convert batteries to electric tools
Electronic Tool Manufacturer, Logistics Provider	Electric Tool	Transfer electric tools from electronic tool manufacturers to logistics providers
Logistics Provider, Channel Partner	Electric Tool	Transfer electric tools from logistics providers to channel partners
Customer	Electric Tool	Query electric tools information through the supply chain

Following the required structure of Hyperledger Composer, the smart contract codes are divided into four parts, including the model file, script file, access control, and query file.

Model File

Model files define the skeleton of the smart contract, including three subsections: asset definition, participant definition, and transaction prototypes.

- **Asset**

  Three types of assets are defined in this subsection, including raw material, battery, and electric tool, each of which is identified by a unique batch ID. Note that various types of raw materials will be involved in the system, such as Lithium, Copper, etc. Thus, there is an attribute, “material type”, in the raw material asset to distinguish between them.

- **Participant**

  Many participants with different roles will be present in the system. However, since all participants share similar attributes, such as institution name, contract, address, etc., only one general participant, called “Player”, is defined. We use an attribute, “role”, to distinguish different types of participants, including material supplier, battery supplier, electronic tool manufacturer, logistic provider, and channel partner.

- **Transaction**

  Transaction defines the actions the participants can perform on specific assets. In this smart contract, six transactions are defined according to the previous table.

Script File

Script files implement the transaction prototypes defined in the model file. The transaction handler functions implemented here will add/update necessary information on the blockchain for traceability records and later querying.

Because of the limitation of Composer Playground, customer querying is also implemented as a transaction, and the receipt containing all the detailed information will be logged to the browser console.

Query File

Query files define the rules of querying information from the blockchain. The queries are invoked by the customer querying transaction in the script file.

Access Control

Access control defines the rules about which participants can access what resources. In our project, we use the default access control file for simplicity.

3.2.2 Deployment, Testing, and Analysis

The codes and test cases can be found in the "composer" folder. Please refer to our demo video for complete guidance on deploying and testing the smart contract. The demo video can be found following this link. The general steps are also listed below:

1) Ensure the environment is correctly set up, and Hyperledger Composer Playground can launch normally.
2) The deployment follows the normal procedures of deploying smart contracts in Composer Playground. Copy "modelfile.cto", "logic.js", "queries.qry", and "permissions.acl" to Composer Playground and deploy changes.
3) Go to the "test_case" folder. According to the test case order, initialize participants, submit transactions, and execute queries. When executing the query, you should open the browser console because the returned message will be printed there.

You can find that all the test cases can be processed successfully and as expected.

3.3 Implementation with Go

3.3.1 Chaincode Overview

Chaincode is a program, written in Go that implements a prescribed interface. Chaincode runs in a secured Docker container isolated from the endorsing peer process. Chaincode initializes and manages the ledger state through transactions submitted by applications. A chaincode typically handles business logic agreed to by members of the network, so it similar to a "smart contract". Ledger state created by a

chaincode is scoped exclusively to that chaincode and can't be accessed directly by another chaincode. Given the appropriate permission, a chaincode may invoke another chaincode to access its state within the same network. In the following sections, we will present our battery traceability chaincode sample application and walk through the process of each method in the Chaincode Shim API.

3.3.2 Smart Contract Deployment

The deployment mainly follows the instructions and starting script in Lab2. After performing Lab1, the server has already installed the necessary packages to build the environment.

Notice that if you use our server (Alice) directly, you don't need to perform the following deployment steps. Instead, enter the following command and interact with the chaincode directly:

docker exec –it cli bash

Now you can experience, for example, query function use this command:

peer chaincode query -C mychannel -n trace -c '{"Args":["query","MDS0001"]}'

To initialize the project, first place the **trace.go** file in an appropriate chaincode folder. In our server, the address is listed as follows.

```
alice@VM-4-12-ubuntu:~/fabric-samples/chaincode/trace$ ls
trace.go
alice@VM-4-12-ubuntu:~/fabric-samples/chaincode/trace$ pwd
/home/alice/fabric-samples/chaincode/trace
```

Launch the first-network with the given script

```
# start the first network
./byfn.sh down
./byfn.sh up
```

Enter the docker command line environment with the following command:

```
docker exec -it cli bash
```

Install .go chaincode to 4 peer nodes, there are two organizations each with two peer nodes. Pay attention to the install address that should correspond to the code position.

```
chaincode > $ install&run.sh
# docker exec -it cli bash

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/
CORE_PEER_ADDRESS=peer0.org1.example.com:7051
CORE_PEER_LOCALMSPID="Org1MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizati
peer chaincode install -n trace -p github.com/chaincode/trace -v 1.6

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/
CORE_PEER_ADDRESS=peer1.org1.example.com:7051
CORE_PEER_LOCALMSPID="Org1MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizati
peer chaincode install -n trace -p github.com/chaincode/trace -v 1.6

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/
CORE_PEER_ADDRESS=peer0.org2.example.com:7051
CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizati
peer chaincode install -n trace -p github.com/chaincode/trace -v 1.6

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/
CORE_PEER_ADDRESS=peer1.org2.example.com:7051
CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizati
peer chaincode install -n trace -p github.com/chaincode/trace -v 1.6
```

List the current installed chaincode. For our project, we debugged the code for several times, so the version is updated to **trace 1.6**.

peer chaincode list –installed

```
Name: trace, Version: 1.6, Path: github.com/chaincode/trace, Escc: escc, Vscc: vscc
```

3.3.3 Code Structure

In this section, we will illustrate the Golang chaincode code structure.

At first, we import several libraries for necessary support. Then, we define the asset in the chaincode:

```
// Asset for tracibility system
type Asset struct{
    // batchId  e.g. ET0001, ET0345

    // creationTime: e.g. 2022-05-04 12:46:30 (GENERATE AUTOMATICALLY)

    // type {raw_material, battery, electric_tool}
    // name e.g. lithum, copper, CR123A
    // quality {bad, medium, good, perfect}
    // description e.g. lithium of high quality!!!
    // ownerID: e.g. materialSupplier_01
    // ownerContact: e.g. 110

    // HistoryOwner: e.g. materialSupplier_01+channelPartner_01+logisticsProvider_01
}
```

Different from the design in Hyperledger Composer, there is only one asset defined in the Golang chaincode version for management convenience and uniform purpose. The attributes, generation method, and sample values are shown in the following table.

Exhibit 12 Key Data Information in Golang

Attribute	Generation	Sample value
Batch ID	**Defined by Participant**	**MDS0001**
Creation Time	Generate automatically	2022-05-04 12:46:30
Type	Defined by Participant	{raw_material, battery, electric_tool}
Name	Defined by Participant	lithium, copper, lithium_battery, …
Quality	Defined by Participant	{bad, medium, good, perfect}
Description	Defined by Participant	lithium of high quality!!!
Owner ID	Defined by Participant	materialSupplier_01
Owner Contact	Defined by Participant	110
History Owner ID	Generate automatically	channelPartner_ljy <-- logisticsProvider_chh <-- electronicToolManufacturer_zmz <-- batterySupplier_ljq <-- materialSupplier_01

The asset is the main object operated by various transactions. There are multiple transactions as explained in the Hyperledger Composer implementation. The querying is also implemented as a transaction for better information display. Before implementing the transactions, we need to define two fundamental functions – *Init* and *Invoke*. Every chaincode program must implement the Chaincode interface whose methods are called in response to received transactions. In particular the **Init** method is called when a chaincode receives an instantiate or upgrade transaction so that the chaincode may perform any necessary initialization, including initialization of application state. The **Invoke** method is called in response to receiving an invoke transaction to process transaction proposals.

```go
func (t *Asset) Init(stub shim.ChaincodeStubInterface) peer.Response{
    //Get args from the instantiation proposal
    args := stub.GetStringArgs()
    attr_count := 7

    if len(args) != attr_count {
        return shim.Error("Incorrect arguments. Expecting a key and 6 values.")
    }

    value := time.Now().Format("2006-01-02 15:04:05")
    for i := 1; i < attr_count; i++ {
        value += "&" + args[i]
    }
    value += "&" + args[5]  // HistoryOwner, recorded automatically

    err := stub.PutState(args[0], []byte(value))
    if err != nil {
        return shim.Error(fmt.Sprintf("Failed to create asset with ID %s", args[0]))
    }
    return shim.Success(nil)
}
```

In the function, we'll retrieve the arguments to the Init call using the **ChaincodeStubInterface.GetStringArgs()** function and check for validity. We assign the value with the arguments passed and add the current timestamp as well as the history owner terms. Next, now that we have established that the call is valid, we'll store the initial state in the ledger. To do this, we will call ChaincodeStubInterface.PutState with the key and value passed in as the arguments. Assuming all went well, return a peer.Response object that indicates the initialization was a success.

```go
func (t *Asset) Invoke(stub shim.ChaincodeStubInterface) peer.Response{
    fn, args := stub.GetFunctionAndParameters()

    var result string
    var err error
    if fn == "create" {
        result, err = create(stub, args)
    } else if fn == "query" {
        result, err = query(stub, args)
    } else if fn == "transfer" {
        result, err = transfer(stub, args)
    } else if fn == "convert" {
        result, err = convert(stub, args)
    } else {
        errMsg := "Unsupported functions: " + fn
        return shim.Error(errMsg)
    }

    if err != nil {
        return shim.Error(err.Error())
    }

    return shim.Success([]byte(result))
}
```

In the Invoke function, as with the Init function above, we need to extract the arguments from the ChaincodeStubInterface. The Invoke function's arguments will be the name of the chaincode application function to invoke. In our case, our application will have four functions: create,

query, transfer, and convert. We first call ChaincodeStubInterface.GetFunctionAndParameters to extract the function name and the parameters to that chaincode application function.
Next, we'll validate the function name as being any one of the provided choices and invoke those chaincode application functions, returning an appropriate response via the shim.Success or shim.Error functions that will serialize the response into a message type.

The other interface in the chaincode "shim" APIs is the ChaincodeStubInterface which is used to access and modify the ledger, and to make invocations between chaincodes. This is what we called "transactions". Here we will introduce the four transactions implementations.

Create product information:

```go
func create(stub shim.ChaincodeStubInterface, args []string) (string, error) {
    attr_count := 7
    if len(args) != attr_count {
        return "", fmt.Errorf("Incorrect arguments. Expecting 7 parameters.")
    }

    value := time.Now().Format("2006-01-02 15:04:05")
    for i := 1; i < attr_count; i++ {
        value += "&" + args[i]
    }
    value += "&" + args[5]  // HistoryOwner, recorded automatically

    err := stub.PutState(args[0], []byte(value))
    if err != nil {
        return "", fmt.Errorf("Failed to create asset: %s", args[0])
    }

    return string(value), nil
}
```

The create function logic is exactly the same as Init function, that receives several arguments of an asset (usually the raw material), add the timestamp and history owners terms, and then store the state to the ledger. An example parameter list is:

["MDS0002","raw_material","copper","medium","copper of medium quality!!!","materialSupplier_02", "120"]

Transfer product from one participant to another:

```go
// batchID, original ownerID, new ownerID, new ownerContact
func transfer(stub shim.ChaincodeStubInterface, args []string) (string, error) {
    if len(args) != 4 {
        return "", fmt.Errorf("Incorrect arguments. Expecting 4 parameters.")
    }
    value, err := stub.GetState(args[0])
    if err != nil {
        return "", fmt.Errorf("Failed to query asset: %s with error: %s", args[0], err)
    }
    if value == nil {
        return "", fmt.Errorf("Asset not found: %s", args[0])
    }
    value_string := strings.Split(string(value), "&")
    if args[1] != value_string[5] {
        return "", fmt.Errorf("Failed to transfer asset: %s with error: %s", args[0], "Incorrect ownership")
    }
    new_value := value_string[0]
    for i := 1; i <= 4; i++ {
        new_value += "&" + value_string[i]
    }
    new_value += "&" + args[2]  // New ownership
    new_value += "&" + args[3]  // New ownerContact
    new_value += "&" + args[2] + " <-- " + value_string[7]  // New history owner
    put_err := stub.PutState(args[0], []byte(new_value))
    if put_err != nil {
        return "", fmt.Errorf("Failed to transfer asset: %s", args[0])
    }
    return string(new_value), nil
}
```

Transfer function mainly modifies the original product owner to a new participant. After receiving the product state from ledger, we split the value term and obtain different attributes. We then pick the relevant terms and reorganize the data with the new arguments passed in. In addition to the new ownership, we add the original owner to the history owner list now, recording the traceability information. An example parameter list is:

["MDS0001","materialSupplier_01","batterySupplier_ljq", "119"]

[BatchID, Original Owner ID, New Owner ID, Owner Contact]

Covert product type:

```go
func convert(stub shim.ChaincodeStubInterface, args []string) (string, error) {
    if len(args) != 5 {
        return "", fmt.Errorf("Incorrect arguments. Expecting 3 parameters.")
    }

    value, err := stub.GetState(args[0])
    if err != nil {
        return "", fmt.Errorf("Failed to query asset: %s with error: %s", args[0], err)
    }

    if value == nil {
        return "", fmt.Errorf("Asset not found: %s", args[0])
    }

    value_string := strings.Split(string(value), "&")
    if args[1] != value_string[1] {
        return "", fmt.Errorf("Failed to convert asset: %s with error: %s", args[0], "Wrong initial type")
    }

    new_value := value_string[0]
    new_value += "&" + args[2] + "&" + args[3] + "&" + value_string[3] + "&" + args[4]
    for i := 5; i <= 7; i++ {
        new_value += "&" + value_string[i]
    }

    put_err := stub.PutState(args[0], []byte(new_value))
    if put_err != nil {
        return "", fmt.Errorf("Failed to convert asset: %s", args[0])
    }
```

Convert product generally has similar implementation as the transfer function, we still obtain the product state from ledger, and then split the term, modify the product type from one to another (e.g. battery → electric tool), and then put the new state back. A sample parameter list is:

["MDS0001","battery","electric_tool", "lithum_electric_tool", "Now we have made up the lithum electrical tool"]

[BatchID, Original Type, New Type, New Name, New Description]

Query Product

```
// BatchID
func query(stub shim.ChaincodeStubInterface, args []string) (string, error) {
    if len(args) != 1 {
        return "", fmt.Errorf("Incorrect arguments. Expecting a key, which is the ID of the asset.")
    }

    value, err := stub.GetState(args[0])
    value_string := strings.Split(string(value), "&")

    receipt := "\n================== Receipt ==================\n"
    receipt += "Query Time: " + time.Now().Format("2006-01-02 15:04:05") + "\n"
    receipt += "\nYou are trying to query the information of Asset - " + args[0] + "\n";
    receipt += "The detailed information of this asset is as follows: \n\n";
    receipt += "ID: " + args[0] + "\n"
    receipt += "CreationTime: " + value_string[0] + "\n"
    receipt += "Type: " + value_string[1] + "\n"
    receipt += "Name: " + value_string[2] + "\n"
    receipt += "Quality: " + value_string[3] + "\n"
    receipt += "Description: " + value_string[4] + "\n"
    receipt += "Owner ID: " + value_string[5] + "\n"
    receipt += "Owner Contact: " + value_string[6] + "\n"
    receipt += "HistoryOwner: " + value_string[7] + "\n"
    receipt += "\n==================== End ====================\n\n"
```

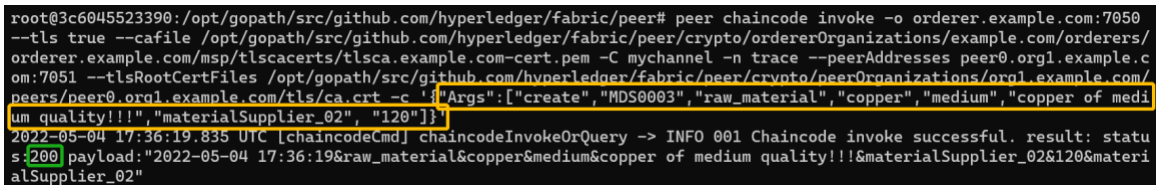
We display the complete product information and utilize our traceability fundamentals in the query function. We simply get the product state with its id and split the string byte to generate different product attributes. We store the info as well as their description in a formatted way for better visualization. The function will return the string(receipt) and print the variable out in the terminal. The parameter list is very simple, only the product id, since it's unique:

```
'{"Args":["query","MDS0001"]}'
```

3.3.4 Result & Analysis

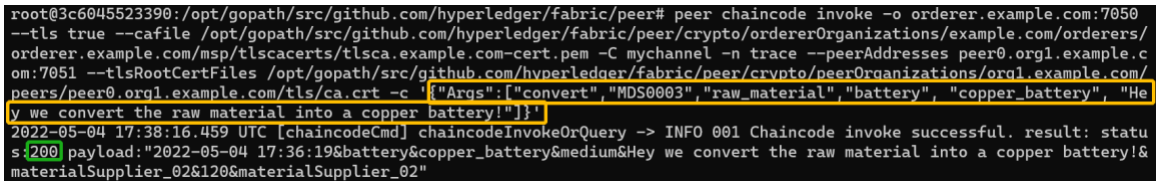
This part displays the execution results and relevant analysis. After the user enters the docker command mode, he can take a single transaction command from "install&run.sh" to execute each time.

Create new product:



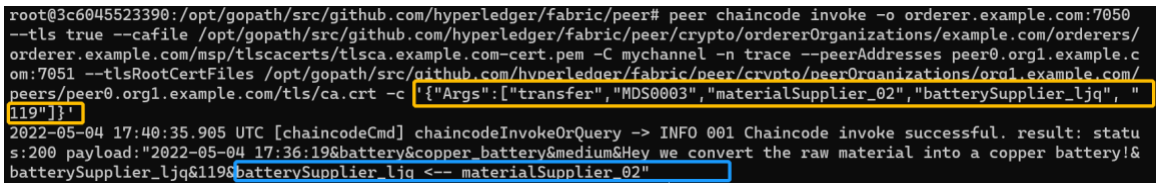
We use peer chaincode invoke command to invoke transaction. The ordered information, port information, channel name, application name, peer address, argument list should be specified, so the command seems long and complex. The command successfully creates a new product with id “MDS0003”, type “raw_material”, name “copper”, quality “medium”, description “copper of medium quality!!!”, owner id “materialSupplier02”, and owner contact “120”. The result status returns a 200, which means it’s a successful state storage. The payload stores the *value* to be put to the ledger. It’s in a plain string stream format, split by the **&** mark.

Convert:



Similar with the invoke method above, we try to convert product ”MDS0003” from raw material to battery and assign it a new name “copper_battery” and add a new description. Success!

Transfer:



With the command, we transfer the product “MDS0003” owner from materialSupplier_02 to battery_Supplier_ljq, with her contact “119”, we then add the original owner to the history owner list, as highlighted with Blue Square. Success!

Query:

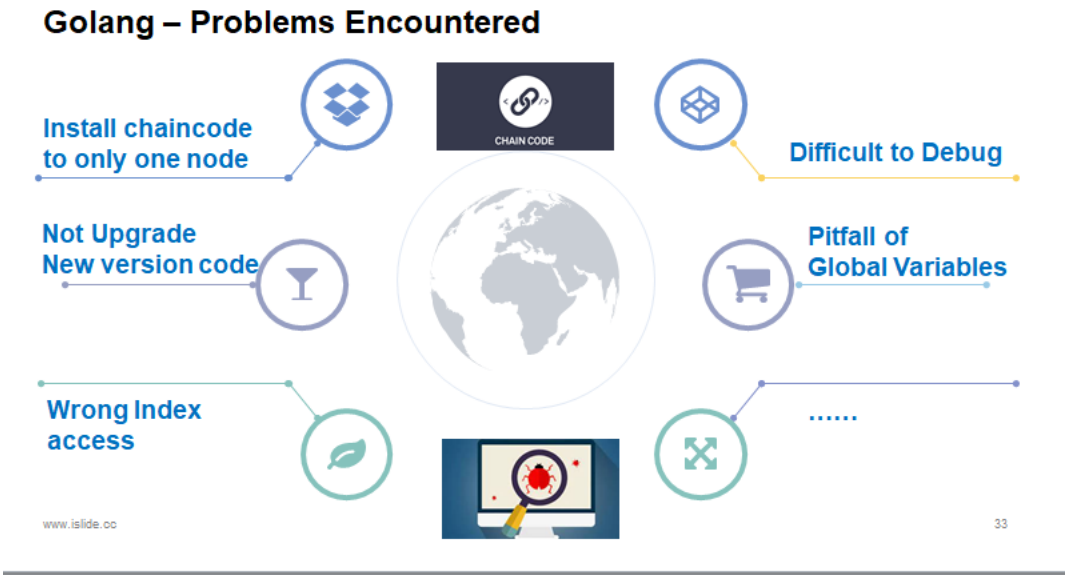
```
alice@VM-4-12-ubuntu:~$ docker exec -it cli bash
root@3c6045523390:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode query -C mychannel -n trace -c '{"A
rgs":["query","MDS0001"]}'

================== Receipt ==================
Query Time: 2022-05-04 17:34:09

You are trying to query the information of Asset - MDS0001
The detailed information of this asset is as follows:

ID: MDS0001
CreationTime: 2022-05-04 13:47:00
Type: electric_tool
Name: lithum_electric_tool
Quality: good
Description: Now we have made up the lithum electrical tool
Owner ID: channelPartner_ljy
Owner Contact: 118010141@link.cuhk.edu.cn
HistoryOwner: channelPartner_ljy <-- logisticsProvider_chh <-- electronicToolManufacturer_zmz <-- batterySupplier_ljq <-
- materialSupplier_01

=================== End ===================
```

We query the product id “MDS0001” we created before. A complete formatted list containing query time, greetings, ID, creation time, type, name quality, description, owner id, owner contact, history owner are printed out. The output is clear and one can trace the responsibility according to the history owner id.

3.3.5 Problem & Summary

The followings show some problems we have encountered during the coding.

Exhibit 12 Problems in Golang Implementation



It proves that we have successfully implemented Golang chaincode to realize our smart contract logic and generate satisfactory results.

4 Project Achievements & Drawbacks

4.1 Project Achievements

From operational to strategical, the achievement of our blockchain system has five levels.

- **Enable monitoring of battery.** Through the blockchain system, participants share the quality of position information of raw materials, batteries, and electric tools, which makes it easy and fast to find the root cause when accident happens.
- **Provide reliable and extractable battery information.** Through the strong Querying function, supply chain partners and consumers are now able to trace the sources of batteries.
- **Real-time transfer of order information.** The blockchain system enables real-time update of order information to notice all channel partners.
- **Build trust and partnership.** Strategic partnership has been a hot topic in the supply chain industry. By preventing the manual manipulation and collusion, the blockchain system boosts the trust among different players and makes the cooperation more reliable.
- **Improve supply chain visibility and responsiveness.** By employing the distributed ledger, battery information is now visible to all permissioned partners in a real-time manner.

Compared with traditional database, the blockchain system is more secured. With the technology of time stamp and hash, all transactions will be recorded in a transparent manner and cannot be modified. Also, in the traditional database, there exists data inconsistency issue since different companies store data in different formats. Blockchain well-solves the problem by unify the data format and data is consistent along the supply chain.

Compared with public blockchain, private blockchain enables authorization function which allows only permitted partners to access confidential data. The system can be fully controlled by the organization. Especially for the battery industry, some manufacturing information could be strategic secret which should better be kept among the supply chain partners.

4.2 Drawbacks of the Design and Implementation

Generally, we recognize two drawbacks during the implementation,

- **Insufficient asset definition.** We only define a single asset instead of classifying reasonably in Go.
- **Incomplete front-end user interface.** Due to the time limit, our team has not built a complete user interface. The simplicity and aesthetics of the front end still require improvements.

5 Future Work & Plans

5.1 Enrich Data Type

Our program is a prototype which includes only the most urgent data types (quality, position, time, etc.). In the future, we need to expand the data types, attributes to achieve more complex data contract logic, such as verifying transfers.

5.2 Improve Front-end User Experience

As mentioned in project drawbacks, front-end construction is an unfinished work in our project. Therefore, in the next step, we plan to employ advanced front-end tools, such as Angular framework (HTML, CSS, Javascript) to empower human computer interaction. Two essential dimensions of improving the front-end is Simplicity and beauty.

5.3 Enable Automated Data Update with RFID

To further reduce human works and improve real-time ability, we plan to combine industrial Radio Frequency Identification Devices (RFID) tags with the blockchain system. Transactions can be automatically established every time RFID is scanned.

6 Potential Impacts & Challenges

6.1 Potential Impacts

Our blockchain system is tailored to the Battery industry. Eventually, it is a digital solution of supply chain traceability for cross industries. For industry-wide companies, the blockchain database system have the following potentials.

Reduce data fraud

Blockchain system applies identity identification, time stamp, hash function to make each transaction identifiable and irreversible. All transactions are stored along the chain which empowers organizations to quickly inspect fault data manipulation.

Reduced long-term cost

The build-up cost of blockchain database is considerable. However, in the long-term, blockchain helps company to reduce the risk of data fraud which saves much time and money efforts.

Build trust

Blockchain requires partners along the supply chain to make an agreement on the data format and transfer rule, and also requires them to set up the same system. The intense corporation on information technology is potentially a breakpoint to build a long-term strategic partnership.

6.2 Potential Challenges

Besides the great potentials of blockchain to empower the battery industry, there still exist several challenges for organizations to implement the project.

Scalability

Blockchain system is slower than the traditional transaction system since one transaction needs to be confirmed by multiple users. Meanwhile, if the transactions volume is high along the supply chain, blockchain transactions could cost seriously. The lower efficiency and higer cost make blockchain difficult to scale up.

Investment capital cost

It is expensive to engage supply chain partners to join the blockchain system in terms of time and money. There are blockchain development cost, training cost, etc. Most of the time, it requires a company to do digital transformation to get used to the blockchain system since it is quite different than the traditional database.

Difficult engagement and implementation

In a traditional supply chain, individual partners keep their own records and some of them treat the data as their competency. However, blockchain requires players to share information that is usually kept as their own assets.

7 Concluding Remarks

In our blockchain project, we have explored the great feasibility and potential of implementing blockchain technology to enhance the transparency in the battery supply chain. Furthermore, we designed the business network and information flow in the system, accordingly, developed a prototype of the traceability system based on Hyperledger Composer and Golang. Digital transformation has been a prevalent topic in the Industry 4.0 era. We firmly believe blockchain technology is a breakthrough point for supply chain to realize transparency and build trust among industrial partners.

8 Evaluation Letter from Sponsor

Hello team,

Thank you for your detailed analysis and design work. I don't have much experience in blockchain development. Here, I would like to talk about some personal views and suggestions from the perspective of blockchain technology and the application of traceability system for your reference.

In general, the design of the whole traceability system is relatively complete, considering all aspects of battery production and circulation, and making full use of the characteristics of blockchain technology, which can basically meet the requirements of traceability in different scenarios. In the summary part, the future improvement direction is also given, including enriching the traceability data and improving the usability of the system.

In addition to the optimization directions already mentioned in the report, some other aspects of the system can also be considered for future optimization, including:
1. The application scenario of the current traceability system design is slightly different from the actual application scenario. For example, in the actual industrial chain, there will be a **pack factory** (assembling a number of cells into battery packs) between the cell factory and the power tool factory.
2. Some business operations can be carried out in related business systems, such as quality inspection during production and inspection of goods before transportation, rather than in the traceability system. The traceability system should pay more attention to the collection and collation of traceability information. If the business process like quality inspection is put into the traceability system, the system process will be very complicated. For example, if the quality inspection is not qualified, the unqualified products need to be dealt with later.

All the best,
Kimi SHEN 沈震远

「Chinese Version」

各位同学好，

感谢你们组做了很详尽细致的分析与设计工作。在区块链开发方面我没有什么经验，下面我从区块链技术和溯源系统应用的角度，谈谈个人的一些小小看法和建议，供参考。

总的来说，整个溯源系统设计的比较完整，考虑了电池生产与流通的各个环节，也充分利用了区块链技术的特点，能够基本满足在不同场景下的溯源需求。在总结部分，也给出了未来改进的方向，包括在丰富溯源数据、提高系统易用性等方面。

除了在报告中已经提到的优化方向，系统的其他一些方面也可以考虑在今后优化，包括：

1、目前溯源系统设计的应用场景，和实际的应用场景，还略微有些差异。例如在实际的产业链中，在电芯厂和电动工具整机厂之间，还会有pack厂这一环节（将若干电芯组装成电池包）。

2、一些业务操作，可以放在相关业务系统中进行，例如生产过程中的质检和运输前的验货，而不需要放在溯源系统中。溯源系统应当更加侧重对于溯源信息的收集和整理。如果把类似质检的业务流程放入溯源系统，会让系统流程很复杂，例如如果质检不合格，那还需要对不合格品进行后续的处理。

祝一切顺利，

Kimi SHEN 沈震远

References

Backhaus R. (2021). Battery Raw Materials - Where from and Where to?. *ATZ worldwide*, *123*(9), 8–13. https://doi.org/10.1007/s38311-021-0715-5

Team Cellerite. (2020). *Battery Parameters.* https://www.cellerite.com/post/battery-parameters

Dinghao Liu. (2018). *hyperledgerDocs*.

https://hyperledgercn.github.io/hyperledgerDocs/chaincode_developers_zh/

Appendix

Appendix 1 Project Plan

Project Overview

The sponsor of our project is an electronic tool company, which is focusing on group purchase of battery-related raw materials and electronic instrument selling business. Due to the potential risk of explosion, the company needs an efficient and reliable battery traceability system to monitor the production and logistics status of batteries throughout the supply chain. Also, the visibility of the battery is vital in agile event management, responsibility assignment, and recycling. Our team proposes a blockchain-based information system to trace the movement of batteries along the supply chain for the listed main reasons:

- First, blockchain is a decentralized system, and it is a reliable medium to share battery information to supply chain entities and boost trust.
- Also, blockchain has the properties of efficient and automated transaction processing, plus tamper-proofing data storage, which benefits the company by cutting labor costs and operations lead time.
- Moreover, blockchain is ultimately an information system to improve supply chain transparency/visibility.

Team Composition & Responsibilities

The Project is initiated by the Chief Technology Officer (CTO) of the company, with four professional IT managers from the IT department and the Supply Chain manager. This crew is called **Digital Transformation Team**, with the primary goal of leading the digital revolution of the battery company by employing advanced blockchain technology.

Mingzhe ZHANG. **Project Coordinator (Team Leader)**. Mingzhe is responsible for organizing the project team and ensuring the overall blockchain project goes smoothly. He is also responsible for aligning the technology with business and supply chain strategy.

Jingyu LI. **Integration Leader**. Jingyu is responsible for the integration of the separate systems, including Login, Identity Management, testing.

Honghao CHEN. **Blockchain Tech Leader**. Honghao is responsible for developing blockchain security technology, including public/private key design, decryption, encryption, smart contract, etc. He is also in

charge of developing the distributed database, including defining data content (E-R diagram, index design, data import) and database function (Create, Delete, Record, Query).

Jiaqi LI. **Front-end Design Leader**. Jiaqi s responsible for the design of the front-end and Graphic User Interface.

Project Objectives

The main business objective at the **strategic level** is to **make the movement of battery traceable and therefore enhance efficiency, cut cost, and lift customer satisfaction.**

We expect the blockchain-based system to aaccomplish the following five sub-objectives,

7 Be able to monitor and control the battery production and selling process

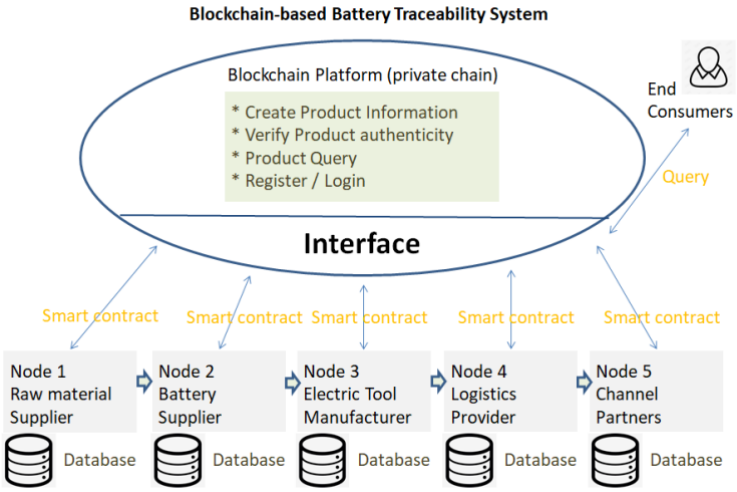
8 Ensure the records are reliable and easily retrieved, and clear the allocation of responsibility.

9 Share real-time logistics status to entities, enhancing supply chain visibility and trust.

10 Deploy smart contract system to trigger logistics event alerts and reinforce responsiveness.

11 Empower the recycling process of the wasted batteries and boost sustainability.

Blockchain System Components (first version)



Our Blockchain System will be implemented through Hyperledger Composer. It allows Participants to to send transactions to exchange assets easily. Above is the component diagram of our Blockchain-based Battery Traceability System. Transactions support business activities by allowing the interaction between the blockchain system and participants. The different parties can add product information to the chain for others to trace, and blockchain technology can ensure information security, not tampered with by anyone.

Also, our blockchain system would allow suppliers or manufacturers to create products in our blockchain through a smart contract. Downstream customer could query product origin to determine the part that possibly error would occur.

Project Timeline & Milestones

Our project will strictly follow the pre-set schedule, dynamically adjusting the plan according to the actual situation.

Task	Expected Time
Finish project plan	Feb 20
Finish database content design	Mar 1
Finish database function design	Mar 15
Finish smart contract design	Mar 31
Finish front-end GUI	Apr 12
System Integration	Apr 25
Finalize project report	May 5

Decision Protocol & Contingency Plan

To enhance the effectiveness and efficiency of team communication, we define the following protocols as communication methods.

12 Weekly meetings (Sat 600-630 pm): Team meet every week to synchronize the progress and share the achievements during the week

13 Shared documents: The team uses the shared documents system Shimo, as the progress monitor.

14 Task management tool. The team employs Trello to dynamically manage task progresses.

To mitigate predictable group behaviors, we set the contingency plan as a reference when inappropriate behaviors occur.

15 Disagreement and dispute. Both sides present their views in public for five minutes at weekly meetings, and team members collectively decide on a better option.

16 Communication breakdown. Anyone who falls out of the project will check the shared document.

Free riding. Theproject coordinator should arrange a small talk with the free-rider to inform the free-rider and then provide improvement suggestions. The job quality will be reviewed in every team meeting.