Project Name - Electronics Supply Chain

1. Use Case Overview

Current Scenario

The electronics supply chain involves three organizations:

1. **Manufacturer**: Creates electronic items using the `electronics` chaincode.
2. **Dealer**: Manages the sale and distribution of finished electronic items.
3. **Supplier**: Provides raw materials to the manufacturer.

Key Functionalities

- **Shared Asset**: Electronic items created by the manufacturer are accessible to all three organizations.
- **Private Data Collections (PDC)**:
  - The supplier creates raw material assets.
  - Only the manufacturer can access raw material data.
  - The dealer is restricted from accessing raw material data.

Pros

- **Transparency**: Shared data ensures that all organizations can access relevant information about electronic items.
- **Data Privacy**: Private Data Collections (PDC) allow sensitive information (e.g., raw material data) to be shared selectively.
- **Auditability**: Blockchain provides a tamper-proof ledger, ensuring traceability for compliance and reporting.

Cons

- **Complex Setup**: Establishing and maintaining a Hyperledger Fabric (HLF) network can be resource-intensive.
- **Scalability Concerns**: As the number of organizations and transactions grow, managing resources and performance might require significant scaling efforts.
- **Access Control Complexity**: Properly configuring access control (e.g., PDC policies) requires careful planning and execution.

2. Why Fabric?

Advantages of Introducing Hyperledger Fabric

1. **Permissioned Network**: Fabric provides a secure, permissioned environment, ensuring that only authorized entities participate.

2. **Modular Architecture**: Features like PDC and chaincode allow fine-grained control over data sharing and business logic.
3. **Scalability**: Supports multiple organizations with different roles and access levels.
4. **Interoperability**: Fabric's chaincode can integrate with existing enterprise systems, enabling seamless workflows.
5. **Data Privacy**: PDC ensures confidential data is accessible only to authorized participants.
6. **Consensus Flexibility**: Allows custom consensus mechanisms tailored to specific use cases.

3. Chaincode Functions

Functions in Electronics Contract

1. **CreateElectronicItem**
   Description: Creates a new electronic item and stores it in the world state. Emits a CreateElectronicItem event upon successful creation. Only allowed for users under ManufacturerMSP.
2. **ReadElectronicItem**
   Description: Retrieves an electronic item from the world state based on its itemID.
3. **DeleteElectronicItem**
   Description: Deletes an electronic item with the given itemID from the world state. Only allowed for users under ManufacturerMSP.
4. **GetAllElectronicItems**
   Description: Fetches all electronic items stored in the world state, filtered by the electronics asset type. The results are sorted by the color field in descending order.
5. **GetElectronicItemsByRange**
   Description: Fetches electronic items within the specified range of keys (startKey and endKey) from the world state.
6. **GetElectronicItemHistory**
   Description: Retrieves the transaction history of a specific electronic item, including all updates and deletions, with their respective timestamps and transaction IDs.
7. **GetElectronicsItemsWithPagination**
   Description: Fetches electronic items from the world state with support for pagination, returning a defined number of results (pageSize) and a bookmark for subsequent queries.

Functions in RawMaterial Contract

1. **CreateRawMaterial**
   Description: Creates an instance of a raw material and stores it in the private data collection. Emits an event upon successful creation. Only allowed for users under SupplierMSP.
2. **ReadRawMaterial**
   Description: Retrieves an instance of a raw material from the private data collection based on its materialID.
3. **DeleteRawMaterial**
   Description: Deletes a raw material with the given materialID from the private data collection. Only allowed for users under SupplierMSP.
4. **GetAllRawMaterials**
   Description: Fetches all raw materials stored in the private data collection, filtered by the rawmaterial asset type.

5. **GetRawMaterialsByRange**
   Description: Fetches raw materials within the specified range of keys (startKey and endKey) from the private data collection.

4. Resources and Technologies Used

1. **Programming Languages**
   - Go (Golang): Used to write the smart contract logic.
2. **Hyperledger Fabric**
   - Fabric Chaincode: The contract implements chaincode for managing private data in the blockchain network.
   - Fabric Private Data Collections: Used to store sensitive data securely.
3. **Libraries and APIs**
   - github.com/hyperledger/fabric-contract-api-go/contractapi: For developing chaincode using the contract API.
   - github.com/hyperledger/fabric-chaincode-go/shim: For accessing stub functions like private data handling and query results.
4. **Data Formats**
   - JSON: Used for structuring raw material data and for transient data serialization/deserialization.
5. **Security and Identity**
   - Transient Data: For securely passing private details during raw material creation.
   - Client Identity (GetMSPID): Ensures role-based access control (e.g., restricting SupplierMSP for certain functions).
6. **Blockchain Query Mechanisms**
   - Key-Value Storage: For storing raw material data in a private collection.
   - Query Functions: Using GetPrivateDataQueryResult and GetPrivateDataByRange for data retrieval.
7. **Development Tools**
   - Chaincode Dev Environment: A configured Hyperledger Fabric network for deploying and testing contracts.
   - JSON Marshaling/Unmarshaling: For encoding/decoding Go structs to/from JSON.
8. **Error Handling**
   - Standard Go Error Mechanisms: For returning informative messages when operations fail.

5. Workflow Diagram

Rough Workflow

- **Manufacturer Workflow:**
  1. Manufacturer creates an electronic item using the electronics chaincode.
  2. Asset is shared with all three organizations.

    3. Updates and transactions are recorded on the ledger.
- **Supplier Workflow:**
    1. Supplier creates a raw material asset using the PDC feature.
    2. Asset is accessible only to the manufacturer.
    3. Transactions involving raw materials are visible only to authorized parties.
- **Dealer Workflow:**
    1. Dealer can view electronic item details but cannot access raw material data.
    2. Dealer records transactions related to sales or distribution.

6. Shortcomings and Future Enhancements

Shortcomings

1. **Scalability Issues**: The current setup may face performance bottlenecks as the number of assets and transactions increase.
2. **PDC Overhead**: Managing private data collections requires additional storage and administrative effort.
3. **Limited Access**: The dealer's restricted access to raw material data could pose challenges if expanded business logic needs partial access.

Further Enhancements

1. **Addition of Order Asset**
    - Introduce an Order asset as private data shared between the distributor and manufacturer.
    - This can include order details like order ID, items ordered, quantity, and delivery timelines.
2. **Enhance Raw Material Details**
    - Add fields for certification details (e.g., ISO certification, safety compliance, quality assurance).
    - Certification information will increase the trust and credibility of the raw material.
3. **Certification Process**
    - Add a certification mechanism by including new participants in the network, such as:
        - **Certification Authorities**: To verify and certify raw materials or final products.
        - **Auditors**: To validate the certification claims.
4. **Expand Network Participants**
    - Add **Vendors/Retailers** to the blockchain network for better supply chain visibility and traceability.
    - Allow retailers to access product history and certification details to assure quality.
5. **Inter-Participant Collaboration**
    - Establish secure data-sharing mechanisms between participants to streamline collaboration.
    - Examples: Manufacturer-supplier agreements, distributor-retailer contracts.
6. **Improve Query Capabilities**
    - Extend chaincode to support multi-criteria search queries for raw materials and orders.

    - Enable filtering by certification status, date ranges, or supplier details.

7. Prerequisites

Need the following softwares/tools before

Installing the Dependencies

Note: If any of the following dependencies are available on your laptop, then no need to install it.

Update Packages

In case of a fresh Ubuntu 22 installation, use the following commands to update the packages before installing other dependencies.

```
sudo apt update

sudo apt upgrade
```

Visual Studio Code

Download and install the latest version of VS code from here: https://code.visualstudio.com/download

To install, execute the following command from the same folder where VS Code is being downloaded.

Note: Replace file_name with the actual name of the file you've downloaded.

```
sudo dpkg -i file_name
```

eg: sudo dpkg -i code_1.95.2-1730981514_amd64.deb

cURL

Install curl using the command

```
sudo apt install curl -y

curl -V
```

NVM

Install NVM (Node Version Manager), open a terminal and execute the following command.

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.0/install.sh | bash
```

Close the current terminal and open a new one.

In the new terminal execute this command to verify nvm has been installed

```
nvm -v
```

NodeJS (Ver 22.x)

Execute the following command to install NodeJs

```
nvm install 22
```

Check the version of nodeJS installed

```
node -v
```

Check the version of npm installed

```
npm -v
```

Docker

Step 1: Download the script

```
curl -fsSL https://get.docker.com -o install-docker.sh
```

Step 2: Run the script either as root, or using sudo to perform the installation.

```
sudo sh install-docker.sh
```

Step 2: To manage Docker as a non-root user

```
sudo chmod 777 /var/run/docker.sock
```

```
sudo usermod -aG docker $USER
```

To verify the installtion enter the following commands

```
docker compose version
```

```
docker -v
```

Execute the following command to check whether we can execute docker commands without sudo

```
docker ps -a
```

JQ

Install JQ using the following command

```
sudo apt install jq -y
```

To verify the installtion enter the following command

```
jq --version
```

Build Essential

Install Build Essential uisng the commnad

```
sudo apt install build-essential -y
```

To verify the installtion enter the following command

```
dpkg -l | grep build-essential
```

Go

Step 1: Download Go

```
curl -OL  https://go.dev/dl/go1.22.0.linux-amd64.tar.gz
```

Step 2: Extract

```
sudo rm -rf /usr/local/go && sudo tar -C /usr/local -xzf go1.22.0.linux-amd64.tar.gz
```

Step 3: Add /usr/local/go/bin to the PATH environment variable. Open the /etc/environment file

```
sudo gedit /etc/environment
```

Step 4: Append the following to the end of PATH variable and save

```
:/usr/local/go/bin
```

```
source $HOME/.profile
```

To verify the installtion enter the following command

```
go version
```

Note: If go version is not listed, then restart the system and execute the command again.

Git

Install git using the command

```
sudo apt install git -y
```

To verify the installtion enter the following command

```
git --version
```