By Gaddikoppula Srinija

Product Supply Chain Management

ABOUT THE PROJECT:

PROBLEM:

Inefficient supply chain management can lead to delays, errors, and higher costs for all participants involved. The traditional supply chain system relies on intermediaries and centralized databases that are vulnerable to hacking, data loss, and human errors. This can cause a lack of transparency and accountability, making it difficult to track the flow of goods and verify their authenticity. This leads to potential fraud, disputes, and legal issues.

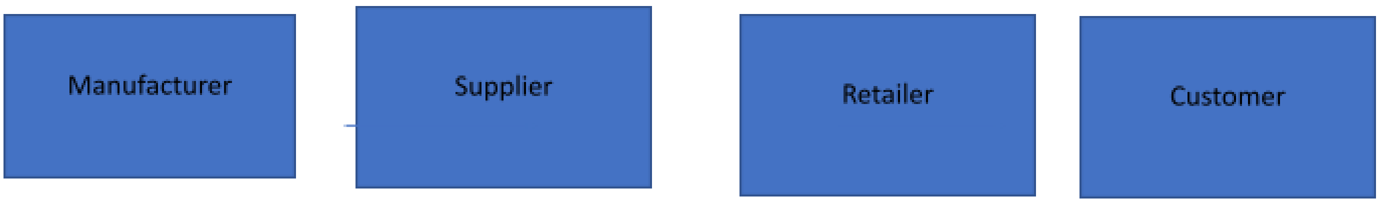
SOLUTION:

The proposed solution is to implement a distributed ledger technology (DLT) platform, which enables the creation of a shared, immutable, and decentralized database. This platform will be built using Hyperledger Fabric, a permissioned blockchain framework designed for enterprise use. The system will leverage smart contracts to automate and execute the business logic of the supply chain process. This will enable secure, transparent, and efficient tracking of products, from the point of origin to the end customer.

WHY FABRIC:

Hyperledger Fabric provides a flexible and modular architecture that allows for the customization of the consensus mechanism, privacy, and data sharing. The platform provides fine-grained access control, ensuring that only authorized participants can access the data. Fabric's modular design allows for the deployment of smart contracts in different programming languages, making it easy to integrate with existing enterprise systems. Additionally, Fabric provides scalability and performance, making it suitable for large-scale supply chain networks.

WORKFLOW:



Producer:

- The manufacturer creates a product with the details of the product, including the source of raw materials and the manufacturing process.
- The manufacturer produces the products and records the process on the distributed ledger.
- The products are transported to the supplier.
- The retailer receives the products and confirms the delivery, updating the record on the ledger.

Supplier:

- The supplier transfers the product and provides the necessary products to the customer.
- Records data on the blockchain at every stage of the supply chain, from the sourcing of products to the delivery of components to the customer

retailer:

- Approves the order contract from customer.

customer:

- creates the order.

RUNNING THE APPLICATION

PREREQUISITES:

npm install npm@latest -g

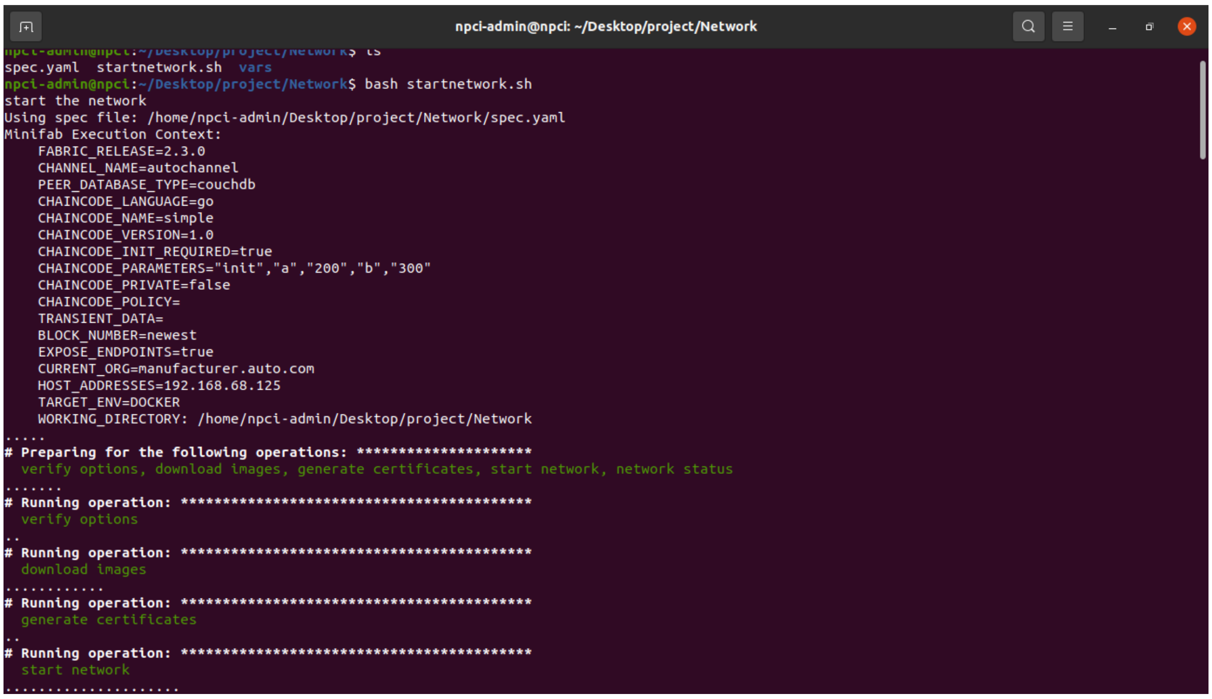
INSTALLATION:

Install NPM packages npm install

HOW TO RUN THE APPLICATION ACCORDING TO THE WORKFLOW:

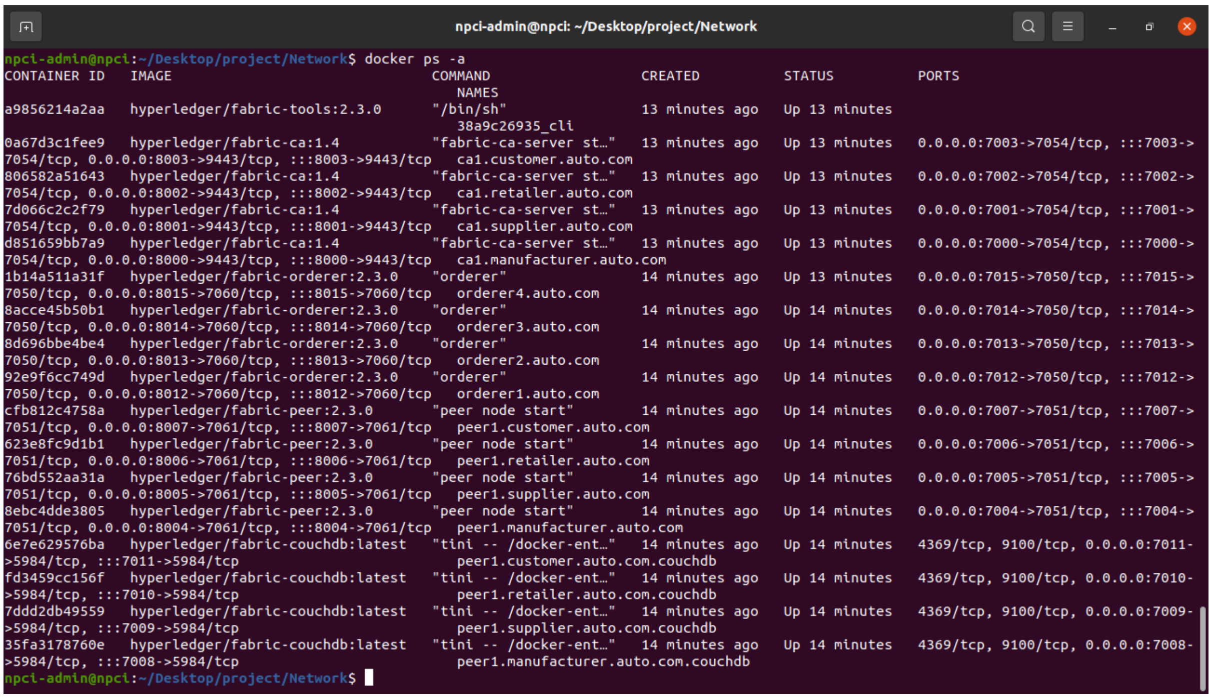
Firstly, we need to up the network, run the command docker ps -a to check whether any other network is up and running Then, run the network sh file in network terminal by using command

./startNetwork .sh



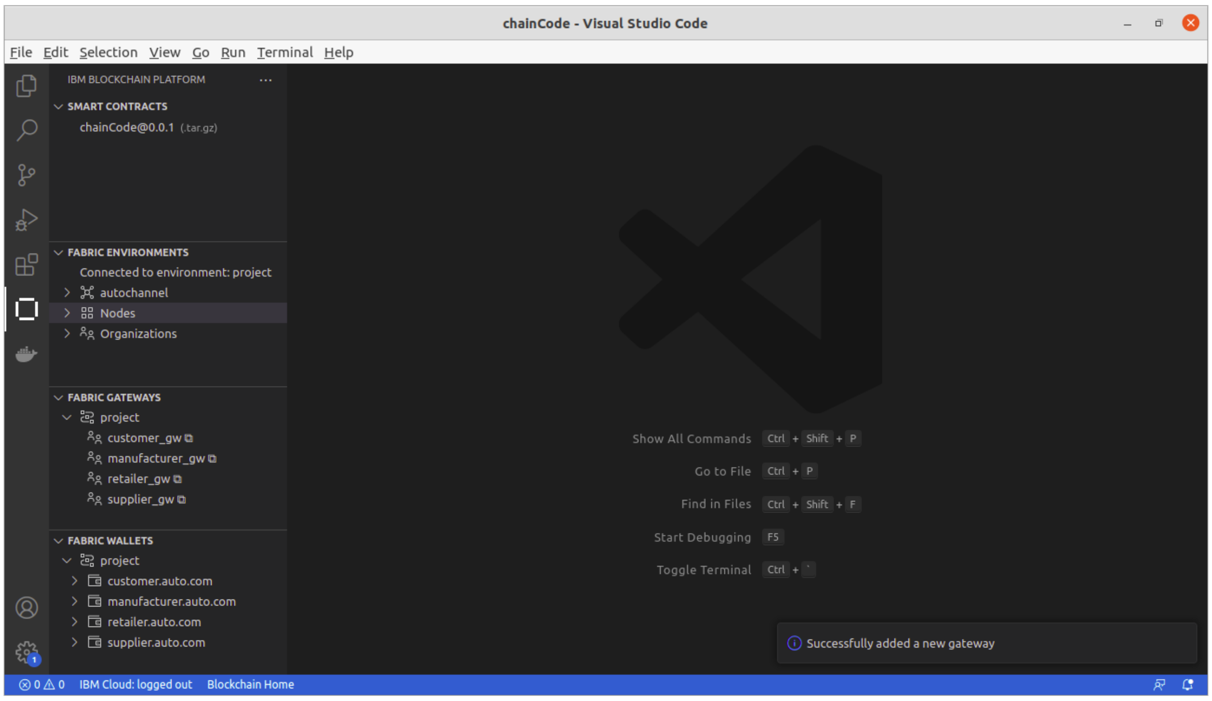
```
npci-admin@npci:~/Desktop/project/Network$ ls
spec.yaml  startnetwork.sh  vars
npci-admin@npci:~/Desktop/project/Network$ bash startnetwork.sh
start the network
Using spec file: /home/npci-admin/Desktop/project/Network/spec.yaml
Minifab Execution Context:
    FABRIC_RELEASE=2.3.0
    CHANNEL_NAME=autochannel
    PEER_DATABASE_TYPE=couchdb
    CHAINCODE_LANGUAGE=go
    CHAINCODE_NAME=simple
    CHAINCODE_VERSION=1.0
    CHAINCODE_INIT_REQUIRED=true
    CHAINCODE_PARAMETERS="init","a","200","b","300"
    CHAINCODE_PRIVATE=false
    CHAINCODE_POLICY=
    TRANSIENT_DATA=
    BLOCK_NUMBER=newest
    EXPOSE_ENDPOINTS=true
    CURRENT_ORG=manufacturer.auto.com
    HOST_ADDRESSES=192.168.68.125
    TARGET_ENV=DOCKER
    WORKING_DIRECTORY: /home/npci-admin/Desktop/project/Network
.....
# Preparing for the following operations: *********************
  verify options, download images, generate certificates, start network, network status
.......
# Running operation: ******************************************
  verify options
..
# Running operation: ******************************************
  download images
............
# Running operation: ******************************************
  generate certificates
..
# Running operation: ******************************************
  start network
.....................
```

After Running ./startNetwork.sh, by using docker ps -a command list out all the containers that are running.



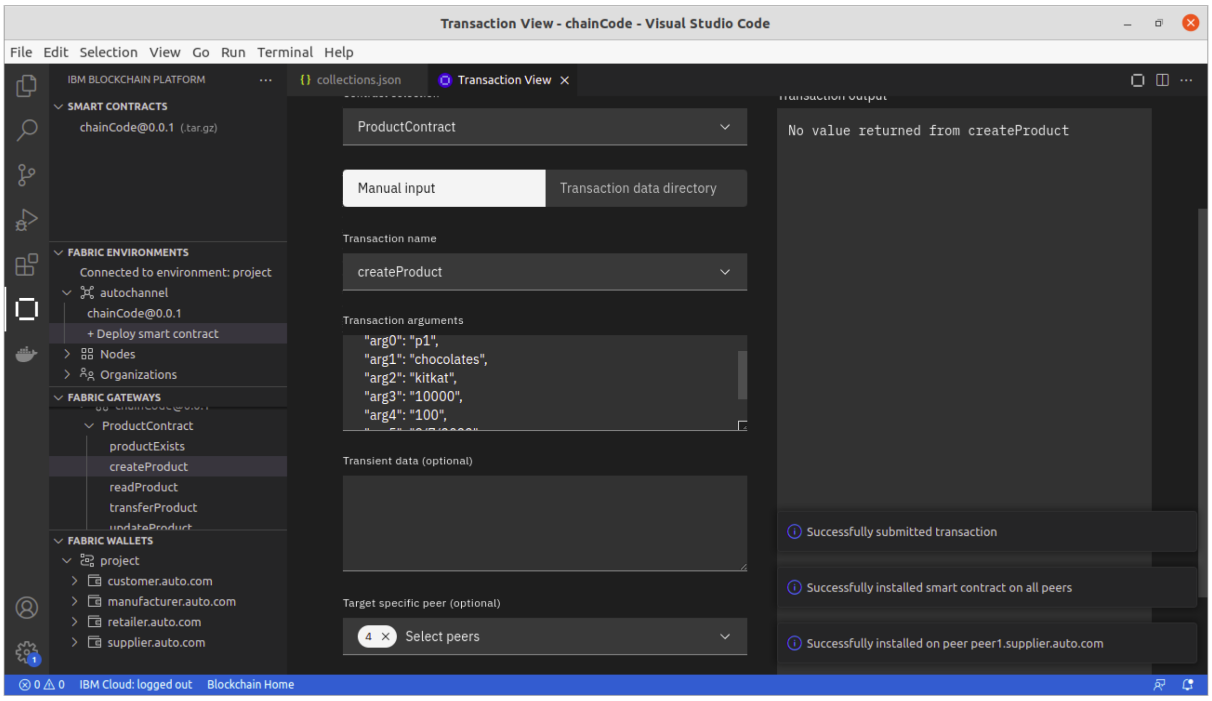
```
npci-admin@npci:~/Desktop/project/Network$ docker ps -a
CONTAINER ID   IMAGE                               COMMAND                  CREATED          STATUS          PORTS
                                                      NAMES
a9856214a2aa   hyperledger/fabric-tools:2.3.0      "/bin/sh"                13 minutes ago   Up 13 minutes
                                                      38a9c26935_cli
0a67d3c1fee9   hyperledger/fabric-ca:1.4           "fabric-ca-server st…"   13 minutes ago   Up 13 minutes   0.0.0.0:7003->7054/tcp, :::7003->
7054/tcp, 0.0.0.0:8003->9443/tcp, :::8003->9443/tcp   ca1.customer.auto.com
806582a51643   hyperledger/fabric-ca:1.4           "fabric-ca-server st…"   13 minutes ago   Up 13 minutes   0.0.0.0:7002->7054/tcp, :::7002->
7054/tcp, 0.0.0.0:8002->9443/tcp, :::8002->9443/tcp   ca1.retailer.auto.com
7d066c2c2f79   hyperledger/fabric-ca:1.4           "fabric-ca-server st…"   13 minutes ago   Up 13 minutes   0.0.0.0:7001->7054/tcp, :::7001->
7054/tcp, 0.0.0.0:8001->9443/tcp, :::8001->9443/tcp   ca1.supplier.auto.com
d851659bb7a9   hyperledger/fabric-ca:1.4           "fabric-ca-server st…"   13 minutes ago   Up 13 minutes   0.0.0.0:7000->7054/tcp, :::7000->
7054/tcp, 0.0.0.0:8000->9443/tcp, :::8000->9443/tcp   ca1.manufacturer.auto.com
1b14a511a31f   hyperledger/fabric-orderer:2.3.0    "orderer"                14 minutes ago   Up 13 minutes   0.0.0.0:7015->7050/tcp, :::7015->
7050/tcp, 0.0.0.0:8015->7060/tcp, :::8015->7060/tcp   orderer4.auto.com
8acce45b50b1   hyperledger/fabric-orderer:2.3.0    "orderer"                14 minutes ago   Up 14 minutes   0.0.0.0:7014->7050/tcp, :::7014->
7050/tcp, 0.0.0.0:8014->7060/tcp, :::8014->7060/tcp   orderer3.auto.com
8d696bbe4be4   hyperledger/fabric-orderer:2.3.0    "orderer"                14 minutes ago   Up 14 minutes   0.0.0.0:7013->7050/tcp, :::7013->
7050/tcp, 0.0.0.0:8013->7060/tcp, :::8013->7060/tcp   orderer2.auto.com
92e9f6cc749d   hyperledger/fabric-orderer:2.3.0    "orderer"                14 minutes ago   Up 14 minutes   0.0.0.0:7012->7050/tcp, :::7012->
7050/tcp, 0.0.0.0:8012->7060/tcp, :::8012->7060/tcp   orderer1.auto.com
cfb812c4758a   hyperledger/fabric-peer:2.3.0       "peer node start"        14 minutes ago   Up 14 minutes   0.0.0.0:7007->7051/tcp, :::7007->
7051/tcp, 0.0.0.0:8007->7061/tcp, :::8007->7061/tcp   peer1.customer.auto.com
623e8fc9d1b1   hyperledger/fabric-peer:2.3.0       "peer node start"        14 minutes ago   Up 14 minutes   0.0.0.0:7006->7051/tcp, :::7006->
7051/tcp, 0.0.0.0:8006->7061/tcp, :::8006->7061/tcp   peer1.retailer.auto.com
76bd552aa31a   hyperledger/fabric-peer:2.3.0       "peer node start"        14 minutes ago   Up 14 minutes   0.0.0.0:7005->7051/tcp, :::7005->
7051/tcp, 0.0.0.0:8005->7061/tcp, :::8005->7061/tcp   peer1.supplier.auto.com
8ebc4dde3805   hyperledger/fabric-peer:2.3.0       "peer node start"        14 minutes ago   Up 14 minutes   0.0.0.0:7004->7051/tcp, :::7004->
7051/tcp, 0.0.0.0:8004->7061/tcp, :::8004->7061/tcp   peer1.manufacturer.auto.com
6e7e629576ba   hyperledger/fabric-couchdb:latest   "tini -- /docker-ent…"   14 minutes ago   Up 14 minutes   4369/tcp, 9100/tcp, 0.0.0.0:7011-
>5984/tcp, :::7011->5984/tcp                          peer1.customer.auto.com.couchdb
fd3459cc156f   hyperledger/fabric-couchdb:latest   "tini -- /docker-ent…"   14 minutes ago   Up 14 minutes   4369/tcp, 9100/tcp, 0.0.0.0:7010-
>5984/tcp, :::7010->5984/tcp                          peer1.retailer.auto.com.couchdb
7ddd2db49559   hyperledger/fabric-couchdb:latest   "tini -- /docker-ent…"   14 minutes ago   Up 14 minutes   4369/tcp, 9100/tcp, 0.0.0.0:7009-
>5984/tcp, :::7009->5984/tcp                          peer1.supplier.auto.com.couchdb
35fa3178760e   hyperledger/fabric-couchdb:latest   "tini -- /docker-ent…"   14 minutes ago   Up 14 minutes   4369/tcp, 9100/tcp, 0.0.0.0:7008-
>5984/tcp, :::7008->5984/tcp                          peer1.manufacturer.auto.com.couchdb
npci-admin@npci:~/Desktop/project/Network$
```

Then open the folder with visual studio code and Add the wallets, Environment, Gateways and Create the Project in smart contract and Deploy the chaincode in environment by using current version and connect to the manufacturer gateway



Successfully added a new gateway

createProduct() function create a product with required attributes Do the transaction with the functions in the ProductContract



```
"arg0": "p1",
"arg1": "chocolates",
"arg2": "kitkat",
"arg3": "10000",
"arg4": "100",
```

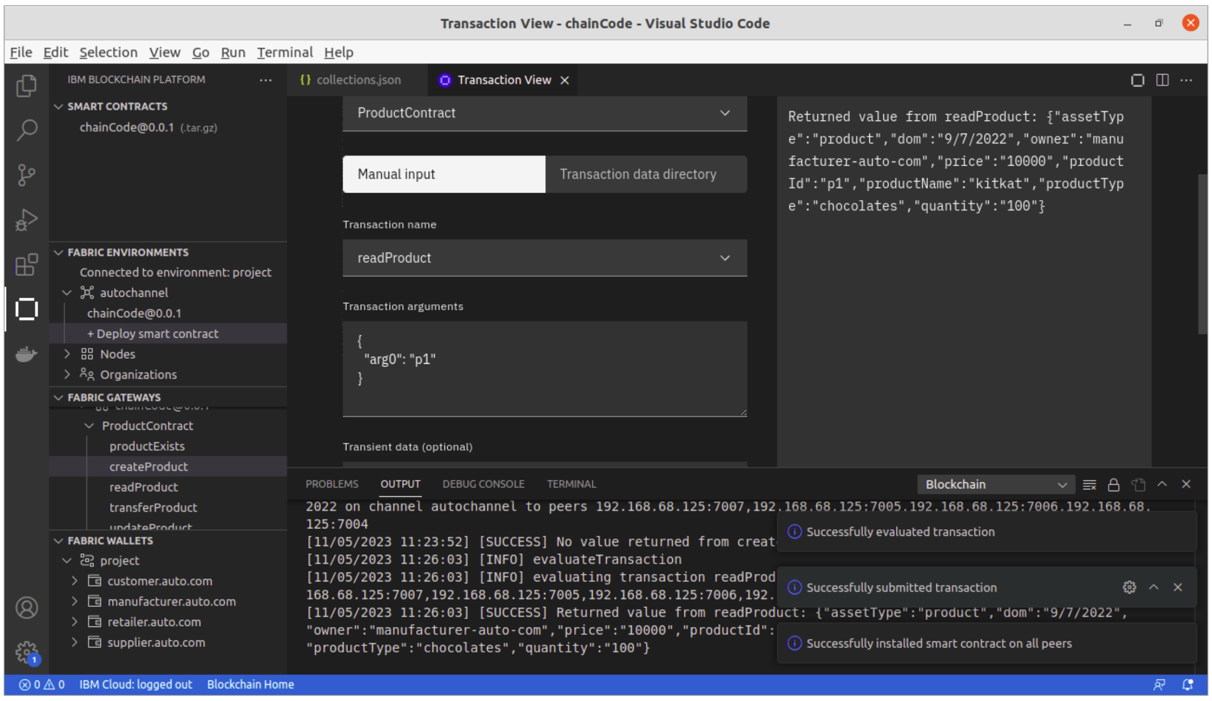
```
No value returned from createProduct
```

Successfully submitted transaction

Successfully installed smart contract on all peers

Successfully installed on peer peer1.supplier.auto.com

readProduct() function used to read a product by passing transient data along with the id in the readProduct() from the productContract



```
{
  "arg0": "p1"
}
```

```
Returned value from readProduct: {"assetTyp
e":"product","dom":"9/7/2022","owner":"manu
facturer-auto-com","price":"10000","product
Id":"p1","productName":"kitkat","productTyp
e":"chocolates","quantity":"100"}
```

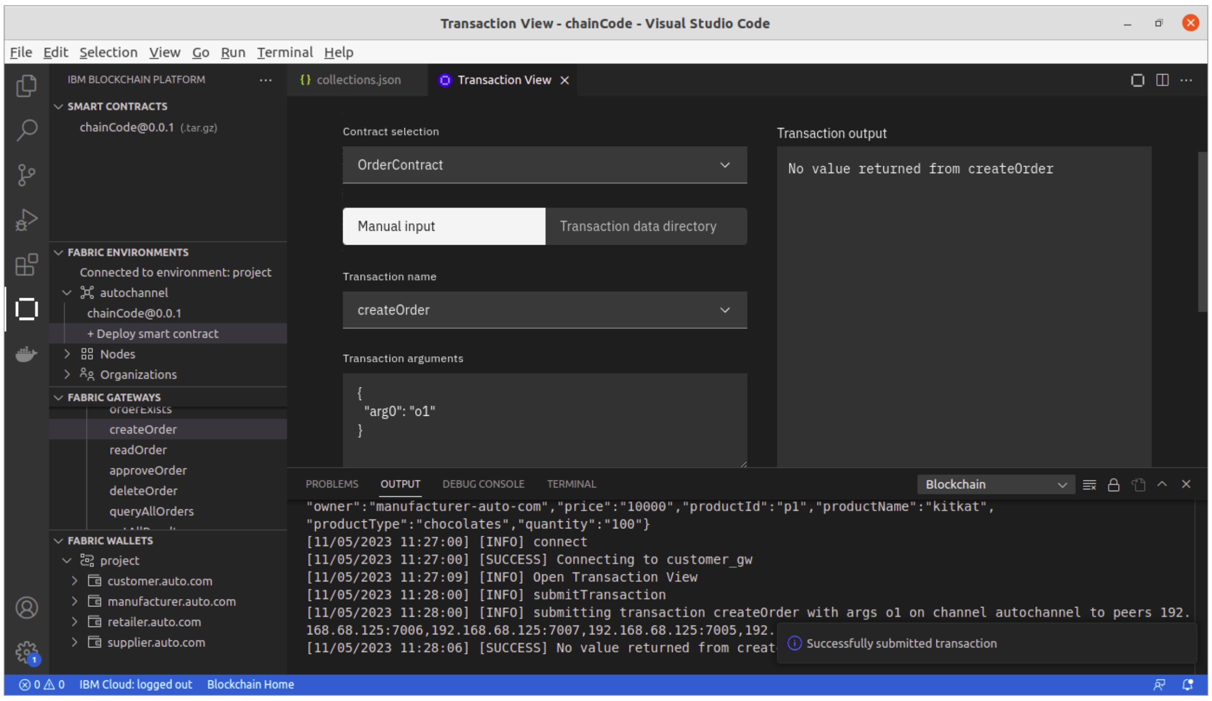
```
2022 on channel autochannel to peers 192.168.68.125:7007,192.168.68.125:7005.192.168.68.125:7006.192.168.68.
125:7004
[11/05/2023 11:23:52] [SUCCESS] No value returned from creat
[11/05/2023 11:26:03] [INFO] evaluateTransaction
[11/05/2023 11:26:03] [INFO] evaluating transaction readProd
168.68.125:7007,192.168.68.125:7005,192.168.68.125:7006,192.
[11/05/2023 11:26:03] [SUCCESS] Returned value from readProduct: {"assetType":"product","dom":"9/7/2022",
"owner":"manufacturer-auto-com","price":"10000","productId":
"productType":"chocolates","quantity":"100"}
```

Successfully evaluated transaction

Successfully submitted transaction

Successfully installed smart contract on all peers

Then connect to customer gateway and by using createOrder() function create the order, and do the transaction by using remaining functions:

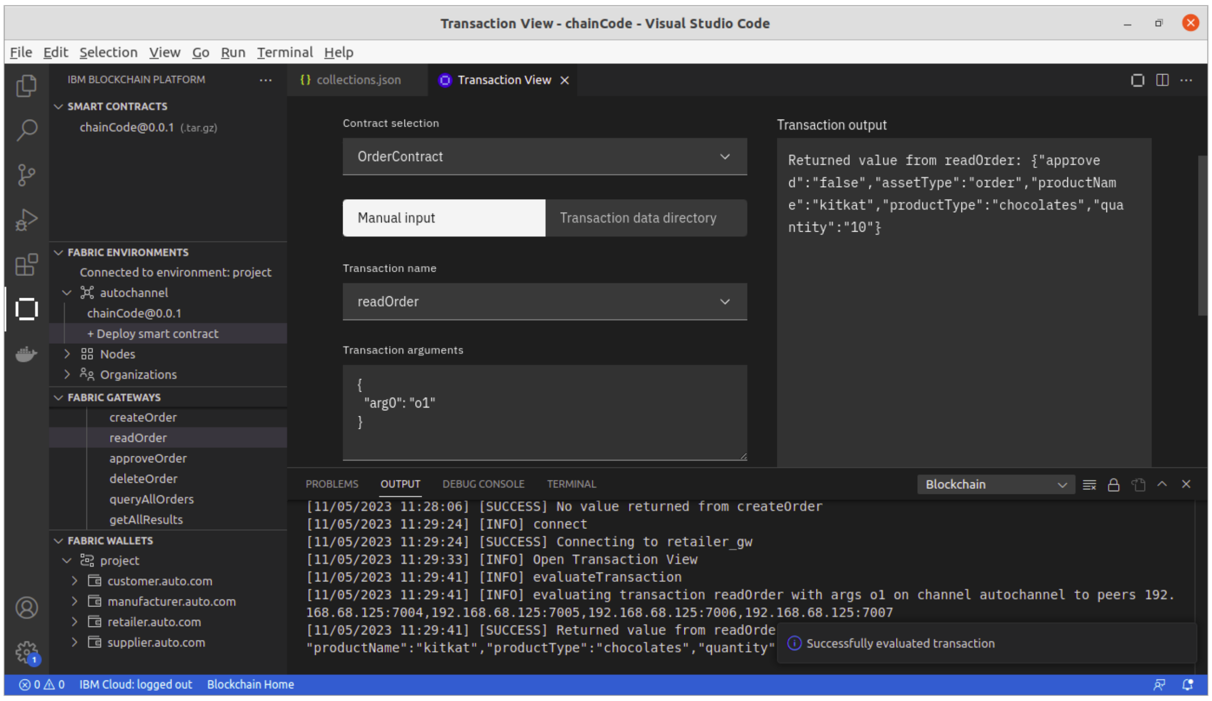


```
{
 "arg0": "o1"
}
```

```
No value returned from createOrder
```

```
"owner":"manufacturer-auto-com","price":"10000","productId":"p1","productName":"kitkat",
"productType":"chocolates","quantity":"100"}
[11/05/2023 11:27:00] [INFO] connect
[11/05/2023 11:27:00] [SUCCESS] Connecting to customer_gw
[11/05/2023 11:27:09] [INFO] Open Transaction View
[11/05/2023 11:28:00] [INFO] submitTransaction
[11/05/2023 11:28:00] [INFO] submitting transaction createOrder with args o1 on channel autochannel to peers 192.
168.68.125:7006,192.168.68.125:7007,192.168.68.125:7005,192.
[11/05/2023 11:28:06] [SUCCESS] No value returned from creat
```

Then by readOrder() function read a order by passing the id in the readOrder() from the OrderContract

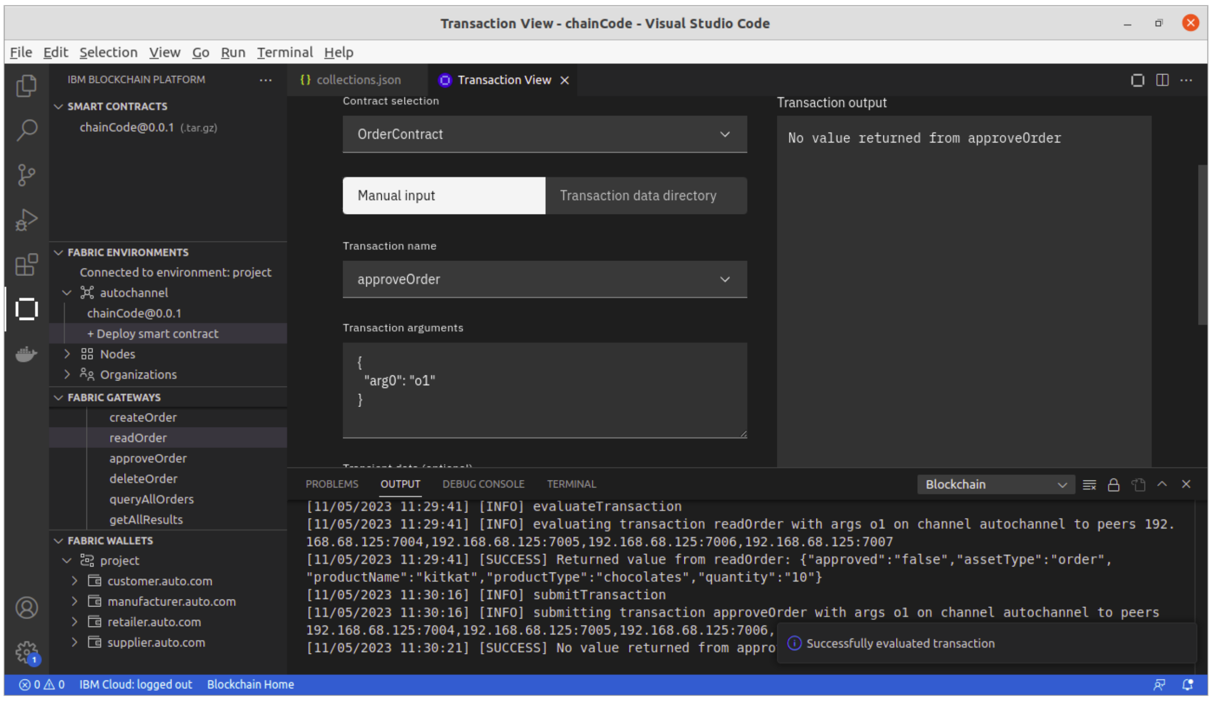


```
{
 "arg0": "o1"
}
```

```
Returned value from readOrder: {"approve
d":"false","assetType":"order","productNam
e":"kitkat","productType":"chocolates","qua
ntity":"10"}
```

```
[11/05/2023 11:28:06] [SUCCESS] No value returned from createOrder
[11/05/2023 11:29:24] [INFO] connect
[11/05/2023 11:29:24] [SUCCESS] Connecting to retailer_gw
[11/05/2023 11:29:33] [INFO] Open Transaction View
[11/05/2023 11:29:41] [INFO] evaluateTransaction
[11/05/2023 11:29:41] [INFO] evaluating transaction readOrder with args o1 on channel autochannel to peers 192.
168.68.125:7004,192.168.68.125:7005,192.168.68.125:7006,192.168.68.125:7007
[11/05/2023 11:29:41] [SUCCESS] Returned value from readOrde
"productName":"kitkat","productType":"chocolates","quantity"
```

Then connect to retailer gateway and by using approveOrder() function ,approve the order by passing the orderid in the readOrder() from the OrderContract.then,approved status will set to true which is initially in a false status.



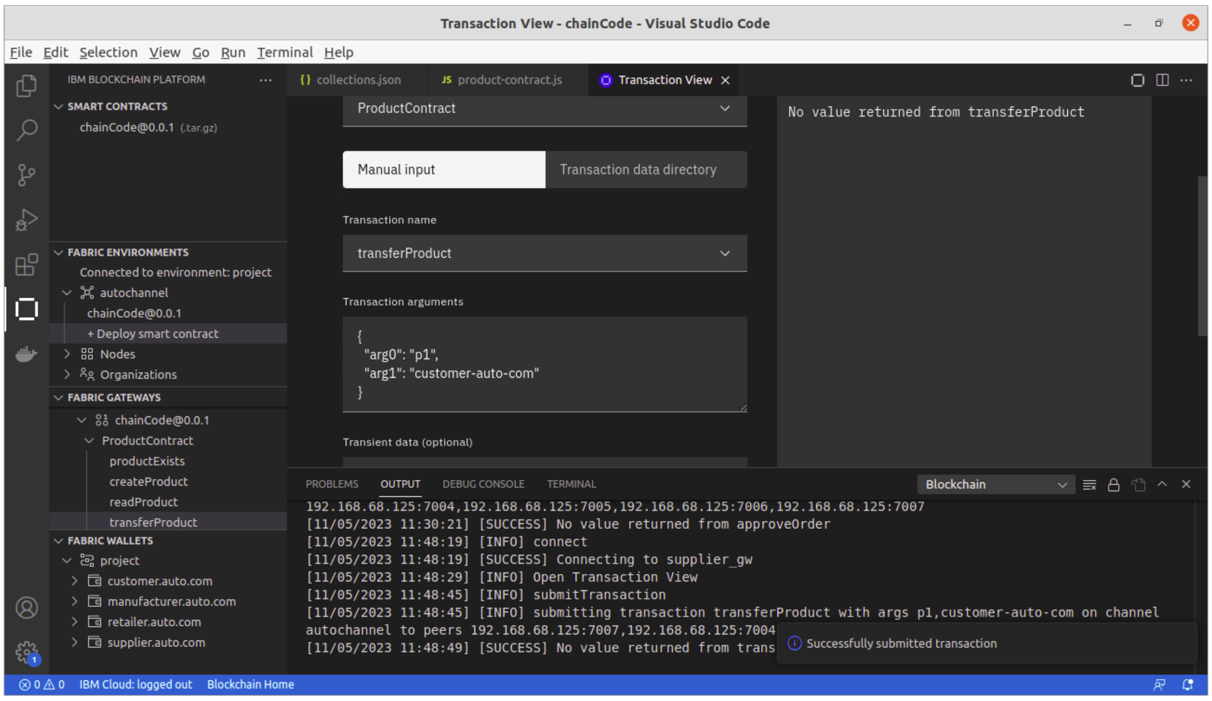
```
{
 "arg0": "o1"
}
```

```
No value returned from approveOrder
```

```
[11/05/2023 11:29:41] [INFO] evaluateTransaction
[11/05/2023 11:29:41] [INFO] evaluating transaction readOrder with args o1 on channel autochannel to peers 192.
168.68.125:7004,192.168.68.125:7005,192.168.68.125:7006,192.168.68.125:7007
[11/05/2023 11:29:41] [SUCCESS] Returned value from readOrder: {"approved":"false","assetType":"order",
"productName":"kitkat","productType":"chocolates","quantity":"10"}
[11/05/2023 11:30:16] [INFO] submitTransaction
[11/05/2023 11:30:16] [INFO] submitting transaction approveOrder with args o1 on channel autochannel to peers
192.168.68.125:7004,192.168.68.125:7005,192.168.68.125:7006,
[11/05/2023 11:30:21] [SUCCESS] No value returned from appro
```

Successfully evaluated transaction

Then connect to supplier gateway and by using transferProduct() function ,transfer the product in the transferProduct() from the productContract.



ProductContract

Manual input | Transaction data directory

Transaction name: transferProduct

Transaction arguments

```
{
  "arg0": "p1",
  "arg1": "customer-auto-com"
}
```

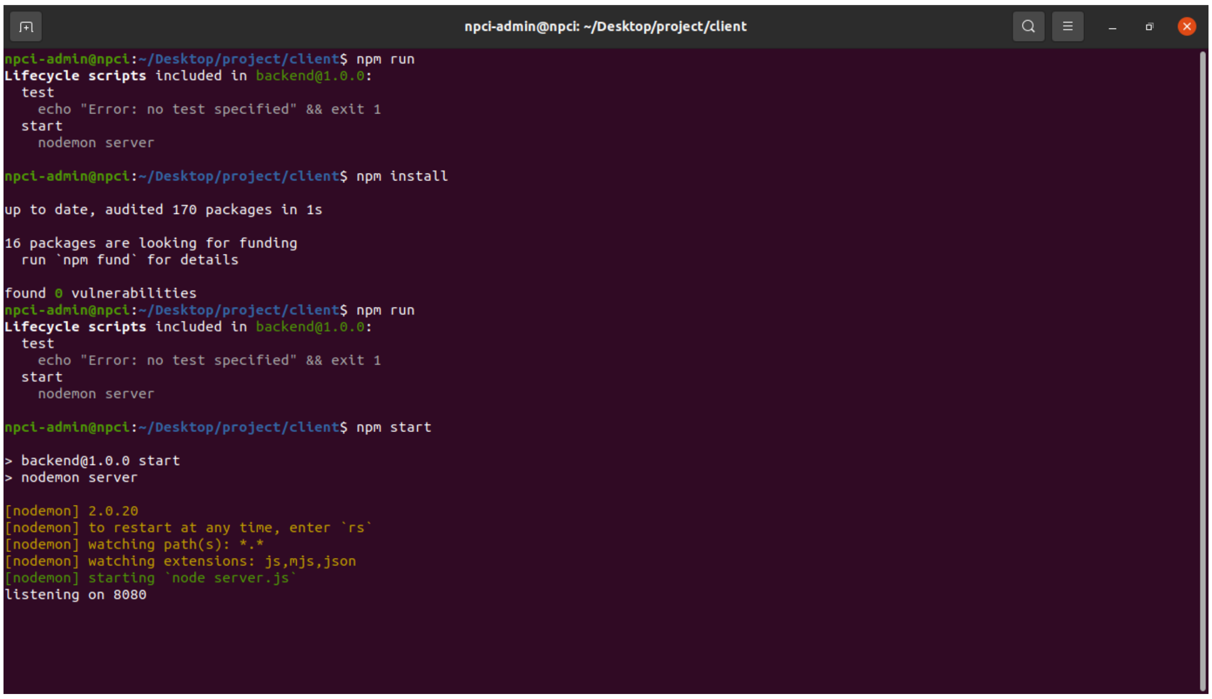
Transient data (optional)

```
No value returned from transferProduct
```

```
192.168.68.125:7004,192.168.68.125:7005,192.168.68.125:7006,192.168.68.125:7007
[11/05/2023 11:30:21] [SUCCESS] No value returned from approveOrder
[11/05/2023 11:48:19] [INFO] connect
[11/05/2023 11:48:19] [SUCCESS] Connecting to supplier_gw
[11/05/2023 11:48:29] [INFO] Open Transaction View
[11/05/2023 11:48:45] [INFO] submitTransaction
[11/05/2023 11:48:45] [INFO] submitting transaction transferProduct with args p1,customer-auto-com on channel
autochannel to peers 192.168.68.125:7007,192.168.68.125:7004
[11/05/2023 11:48:49] [SUCCESS] No value returned from trans
```

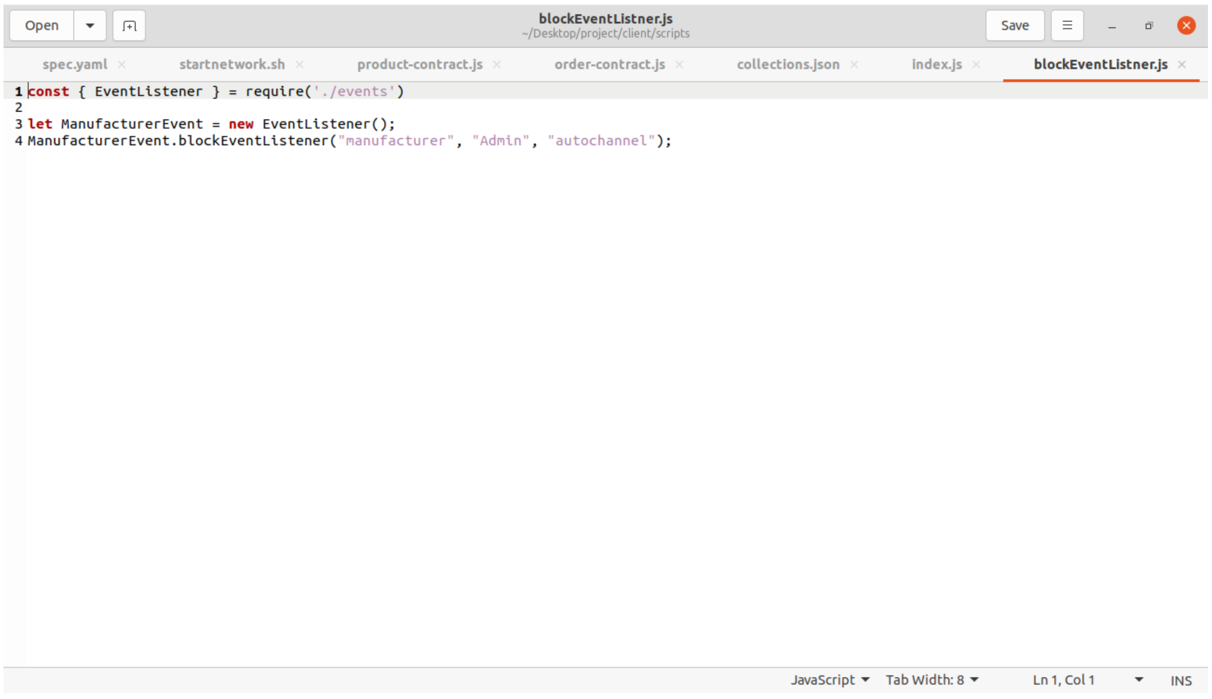
Successfully submitted transaction

Then, open terminal in client folder, test the client folder by executing node in terminal



```
npci-admin@npci:~/Desktop/project/client$ npm run
Lifecycle scripts included in backend@1.0.0:
  test
    echo "Error: no test specified" && exit 1
  start
    nodemon server

npci-admin@npci:~/Desktop/project/client$ npm install

up to date, audited 170 packages in 1s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npci-admin@npci:~/Desktop/project/client$ npm run
Lifecycle scripts included in backend@1.0.0:
  test
    echo "Error: no test specified" && exit 1
  start
    nodemon server

npci-admin@npci:~/Desktop/project/client$ npm start

> backend@1.0.0 start
> nodemon server

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
listening on 8080
```

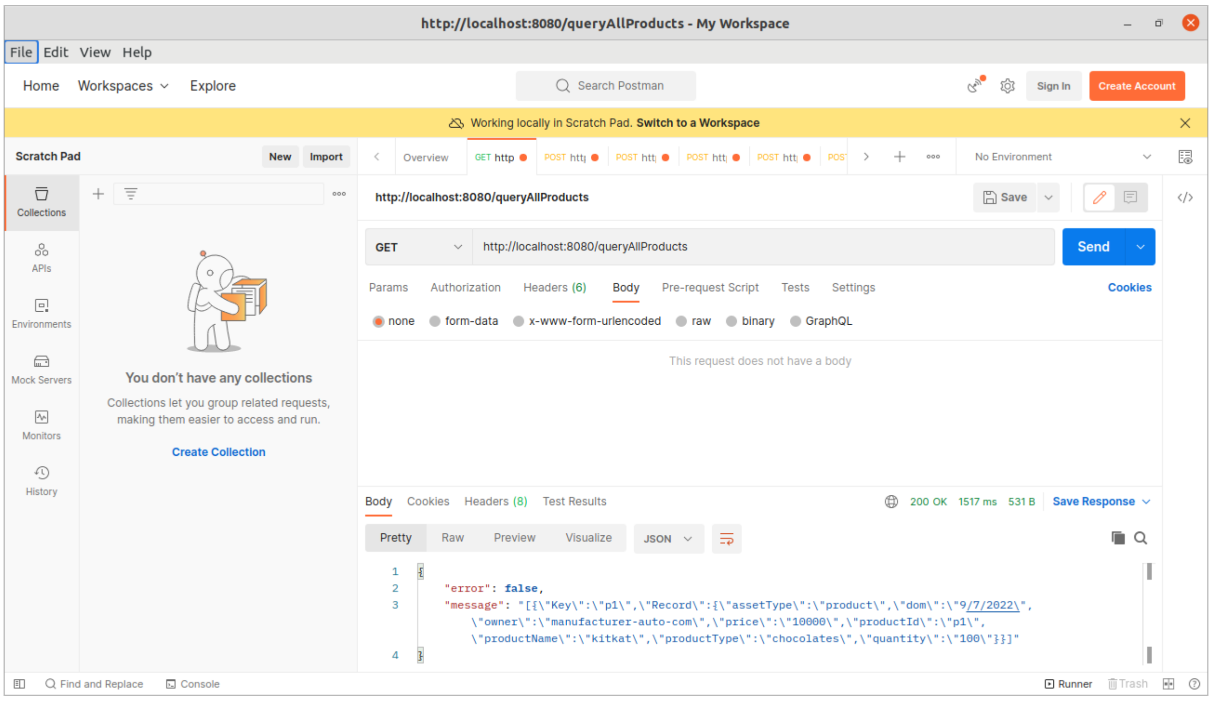
Then, open the terminal in event folder, test the event folder by executing command node



```javascript
const { EventListener } = require('./events')

let ManufacturerEvent = new EventListener();
ManufacturerEvent.blockEventListener("manufacturer", "Admin", "autochannel");
```

Then Final Project Output from backend interface:

GET()



http://localhost:8080/queryAllProducts - My Workspace

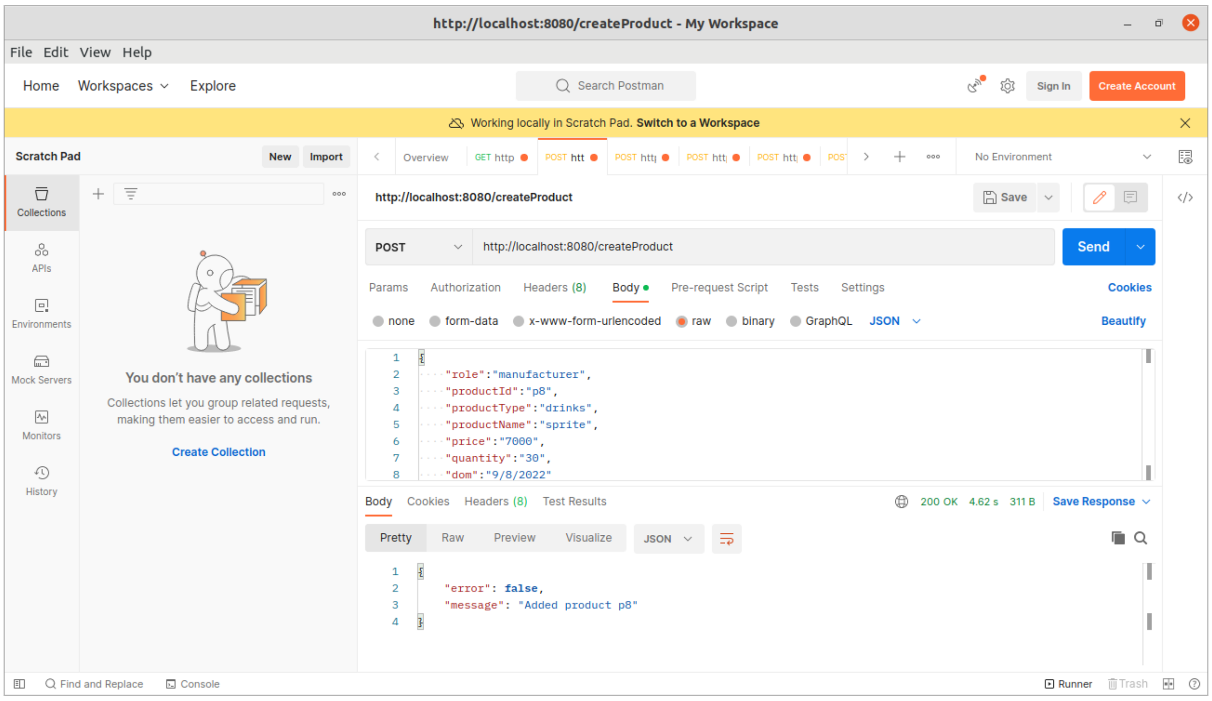
GET http://localhost:8080/queryAllProducts

This request does not have a body

200 OK 1517 ms 531 B

```
{
    "error": false,
    "message": "[{\"Key\":\"p1\",\"Record\":{\"assetType\":\"product\",\"dom\":\"9/7/2022\",
        \"owner\":\"manufacturer-auto-com\",\"price\":\"10000\",\"productId\":\"p1\",
        \"productName\":\"kitkat\",\"productType\":\"chocolates\",\"quantity\":\"100\"}}]"
}
```

POST()



http://localhost:8080/createProduct - My Workspace

POST http://localhost:8080/createProduct

```
{
    "role":"manufacturer",
    "productId":"p8",
    "productType":"drinks",
    "productName":"sprite",
    "price":"7000",
    "quantity":"30",
    "dom":"9/8/2022"
```

200 OK 4.62 s 311 B

```
{
    "error": false,
    "message": "Added product p8"
}
```

CHAINCODE FUNCTIONS:

createProduct - creates the product

readProduct - read the data of particular id

transferProduct - transfer the specific product to particular customer.

queryAllProducts - shows all the created products

deleteProduct - deletes the particular product with given id

getProductHistory - gets history of particular product

getProductsByRange - gets the details of products between the given ids createOrder - creates the order readOrder- read the order of mobile

createOrder – create the orders

readOrder – read the orders

approveOrder-approves the order

queryAllOrder - shows all the created orders

deleteOrder - deletes the particular id order which is given

RESOURCES USED:

Used internet browser for information regarding project.