Supply Chain Using Ethereum Blockchain

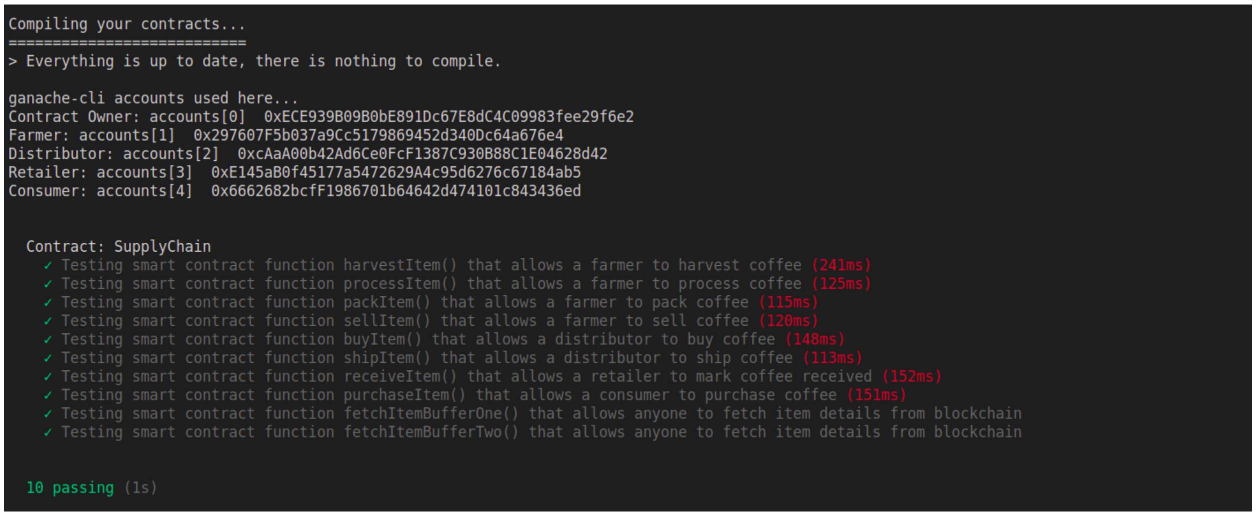
(Link for the code)

Installations :

1. Ganache
2. Npm recent version
3. Truffle recent version
4. Install little server (npm install lite-server --save-dev)

Project Setup :

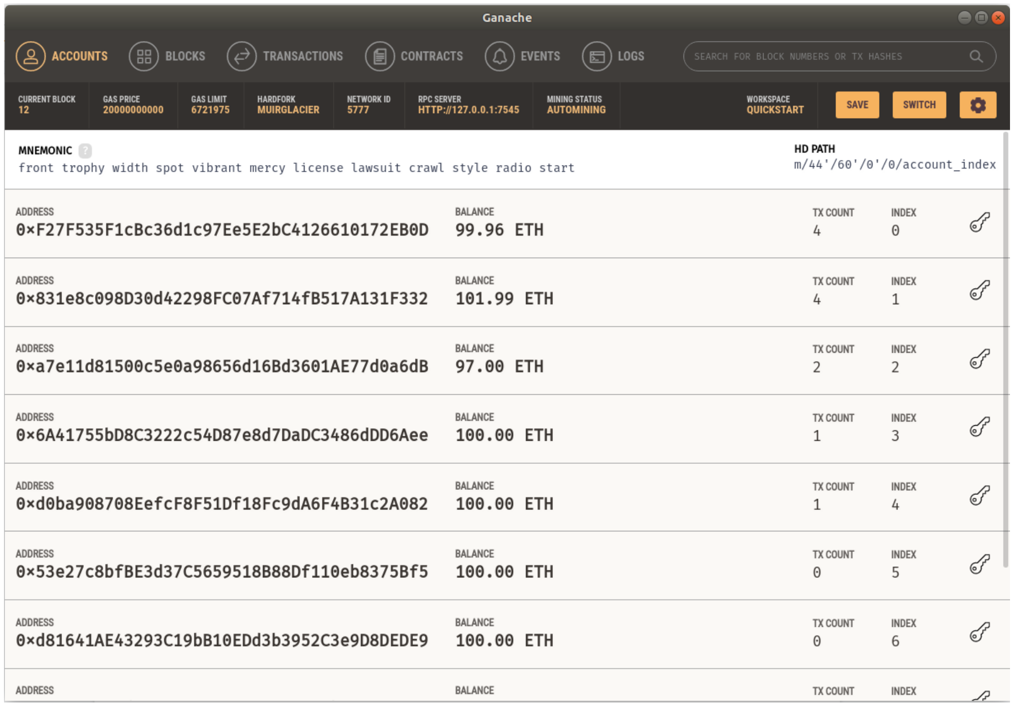
1. mkdir dapp
2. cd dapp
3. Work with Sudo or administrator mode
4. truffle init
5. Update truffle-config.js file to include host and port number on which ganache is running
6. Create a file 2_deploy_contracts.js in migrations folder and add the content in it.
7. Create a file SuppyChain.sol in contracts and add the solidity code in it.
8. Create a basic html view in index.html and write the api code to interact with smart contract in app.js
9. Instal the package and its dependencies using :
    a. npm install
10. Run the test
    a. From the base directory :
        i. ganache-cli -p 7545
    b. In a new command prompt from same base directory :
        i. truffle compile
        ii. truffle migrate
        iii. truffle test



```
Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.

ganache-cli accounts used here...
Contract Owner: accounts[0]  0xECE939B09B0bE891Dc67E8dC4C09983fee29f6e2
Farmer: accounts[1]  0x297607F5b037a9Cc5179869452d340Dc64a676e4
Distributor: accounts[2]  0xcAaA00b42Ad6Ce0FcF1387C930B88C1E04628d42
Retailer: accounts[3]  0xE145aB0f45177a5472629A4c95d6276c67184ab5
Consumer: accounts[4]  0x6662682bcfF1986701b64642d474101c843436ed


  Contract: SupplyChain
    ✓ Testing smart contract function harvestItem() that allows a farmer to harvest coffee (241ms)
    ✓ Testing smart contract function processItem() that allows a farmer to process coffee (125ms)
    ✓ Testing smart contract function packItem() that allows a farmer to pack coffee (115ms)
    ✓ Testing smart contract function sellItem() that allows a farmer to sell coffee (120ms)
    ✓ Testing smart contract function buyItem() that allows a distributor to buy coffee (148ms)
    ✓ Testing smart contract function shipItem() that allows a distributor to ship coffee (113ms)
    ✓ Testing smart contract function receiveItem() that allows a retailer to mark coffee received (152ms)
    ✓ Testing smart contract function purchaseItem() that allows a consumer to purchase coffee (151ms)
    ✓ Testing smart contract function fetchItemBufferOne() that allows anyone to fetch item details from blockchain
    ✓ Testing smart contract function fetchItemBufferTwo() that allows anyone to fetch item details from blockchain


  10 passing (1s)
```

Run the Network :

1. Start Ganache (Quickstart Ethereum)

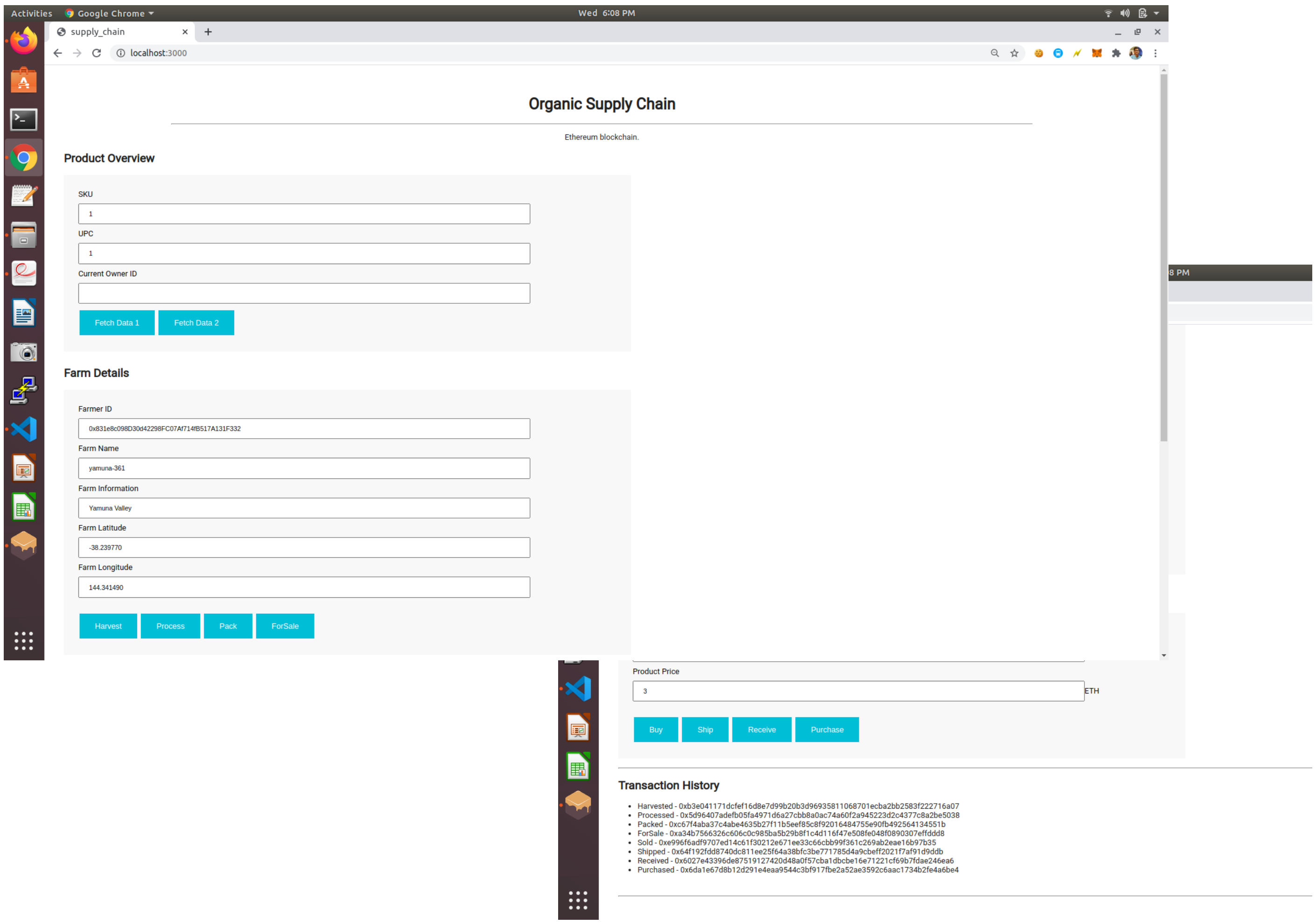


CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
12	20000000000	6721975	MUIRGLACIER	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART

MNEMONIC
```
front trophy width spot vibrant mercy license lawsuit crawl style radio start
```
HD PATH
```
m/44'/60'/0'/0/account_index
```

ADDRESS	BALANCE	TX COUNT	INDEX
0x F27F535F1cBc36d1c97Ee5E2bC4126610172EB0D	99.96 ETH	4	0
0x831e8c098D30d42298FC07Af714fB517A131F332	101.99 ETH	4	1
0xa7e11d81500c5e0a98656d16Bd3601AE77d0a6dB	97.00 ETH	2	2
0x6A41755bD8C3222c54D87e8d7DaDC3486dDD6Aee	100.00 ETH	1	3
0xd0ba908708EefcF8F51Df18Fc9dA6F4B31c2A082	100.00 ETH	1	4
0x53e27c8bfBE3d37C5659518B88Df110eb8375Bf5	100.00 ETH	0	5
0xd81641AE43293C19bB10EDd3b3952C3e9D8DEDE9	100.00 ETH	0	6

2. Import private keys and add accounts in Metamask (For now add only two accounts listed on 2nd and 3rd number in ganache).
3. Verify that the accounts are showing the correct number of ethers associated with them.
4. From the basic directory Compile and Deploy the Smart Contract with command
   a. `truffle migrate`
5. You can check few ethers from account one of ganache got used up for deploying the contract and a file SupllyChain.json created in build/contract folder.
6. From another command prompt start the server with command
   a. `npm run dev`
7. Start the server on address ***http://localhost:3000/*** and connect the accounts of metmask with this site (mostly a pop-up will come up automatically to do so).
8. Your network is up and running !



Organic Supply Chain

Ethereum blockchain.

Product Overview

SKU: 1

UPC: 1

Current Owner ID:

Farm Details

Farmer ID: 0x831e8c098D30d42298FC07Af714fB517A131F332

Farm Name: yamuna-361

Farm Information: Yamuna Valley

Farm Latitude: -38.239770

Farm Longitude: 144.341490

Product Price: 3 ETH

Transaction History

- Harvested - 0xb3e041171dcfef16d8e7d99b20b3d96935811068701ecba2bb2583f222716a07
- Processed - 0x5d96407adefb05fa4971d6a27cbb8a0ac74a60f2a945223d2c4377c8a2be5038
- Packed - 0xc67f4aba37c4abe4635b27f11b5eef85c8f92016484755e90fb492564134551b
- ForSale - 0xa34b7566326c606c0c985ba5b29b8f1c4d116f47e508fe048f0890307effddd8
- Sold - 0xe996f6adf9707ed14c61f30212e671ee33c66cbb99f361c269ab2eae16b97b35
- Shipped - 0x64f192fdd8740dc811ee25f64a38bfc3be771785d4a9cbeff2021f7af91d9ddb
- Received - 0x6027e43396de87519127420d48a0f57cba1dbcbe16e71221cf69b7fdae246ea6
- Purchased - 0x6da1e67d8b12d291e4eaa9544c3bf917fbe2a52ae3592c6aac1734b2fe4a6be4

Performing Transactions :

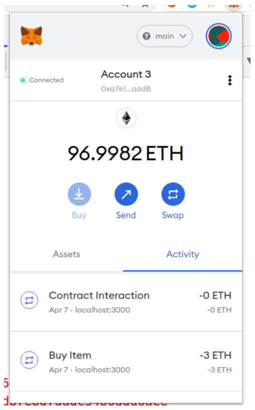
1. Press **F12,** to view the console as we are going to print logs and few information in it.
2. Fill up the form (already filled, if you want to change the information you can.) Update the owner ID and Farmer ID with the Second account's address listed in Ganache and switch to the same account in metamask.
3. In the console you can verify the filled up values that will get displayed, and verify the metmask account ID to be same as that of account we want.
4. Now you are ready with the farm, product and farmer (Account 2 of ganache) details. Now you can
   i. Harvest
   ii. Process
   iii. Pack
   iv. Sell Item (update the price before this, to set your own price for the product, by default it is set to 1).

5. You can see the transaction history below.
6. You can fetch the **fetchItemBufferOne and fetchItemBufferTwo,** containing every detail by hitting buttons **fetch data 1 and fetch data 2,** and see the results in console.

7. NOW BUYING AN ITEM :
    i. Switch to the account in metamask which is listed third in ganache. Verify the address after switching, it will flash in the console.

```
getMetaskID:                                                app.js:90
▶ ["0xa7e11d81500c5e0a98656d16bd3601ae77d0a6db"]
```

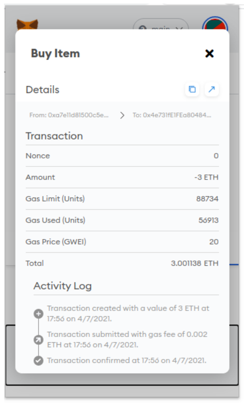
    ii. Update the amount in product price you want to pay for it (Should be greater than original product price).
    iii. Hit the buy item.
    iv. Verify that in **fetchItemBufferTwo,** the distributor ID will get updated.
    v. SHIP THE ITEM, with the same account. You can check in **fetchItemBufferOne,** the state of the item is getting updated with every transaction (it's enum variable).



96.9982 ETH

Contract Interaction -0 ETH
Apr 7 · localhost:3000 -0 ETH

Buy Item -3 ETH
Apr 7 · localhost:3000 -3 ETH



From: 0xa7e11d81500c5e... > To: 0x4e731fE1FEa80484...

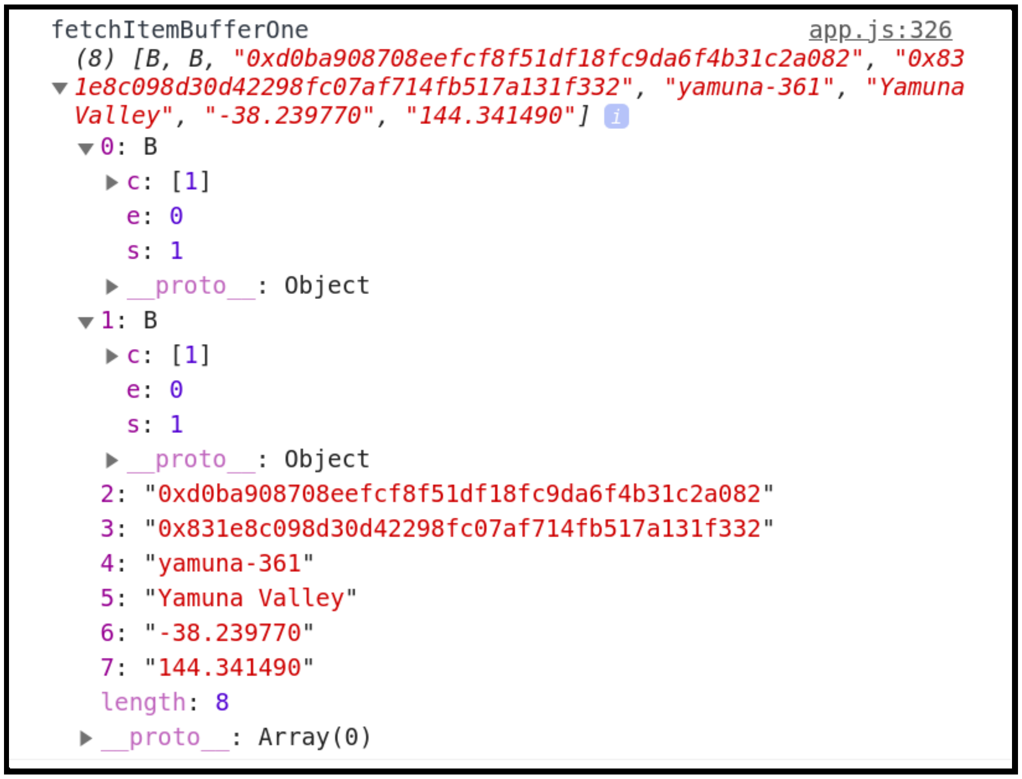
Transaction

Nonce	0
Amount	-3 ETH
Gas Limit (Units)	88734
Gas Used (Units)	56913
Gas Price (GWEI)	20
Total	3.001138 ETH

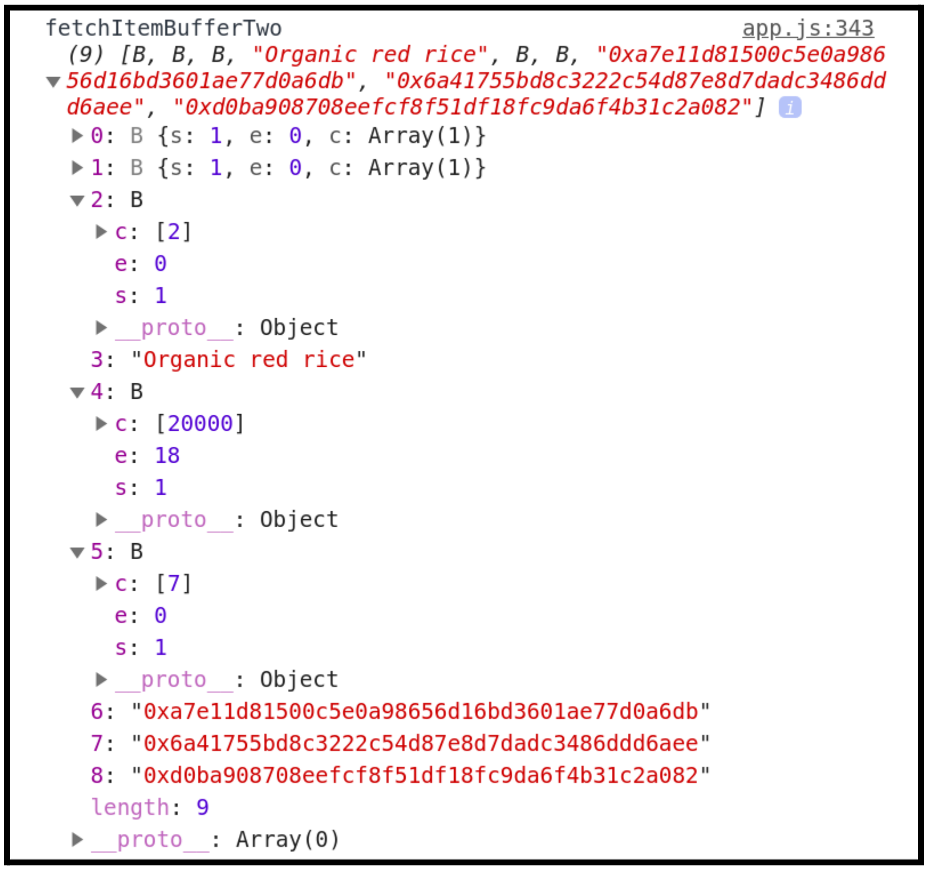
Activity Log

Transaction created with a value of 3 ETH at 17:56 on 4/7/2021.
Transaction submitted with gas fee of 0.002 ETH at 17:56 on 4/7/2021.
Transaction confirmed at 17:56 on 4/7/2021.

8. RECEIVE AN ITEM :
    i. Now add the fourth account listed in ganache in the metamask and connect it to localhost:3000.
    ii. Switch to it and verify the same in the console.
    iii. Press the receive button.
    iv. You can check in **fetchItemBufferTwo,** retailers address will get updated.

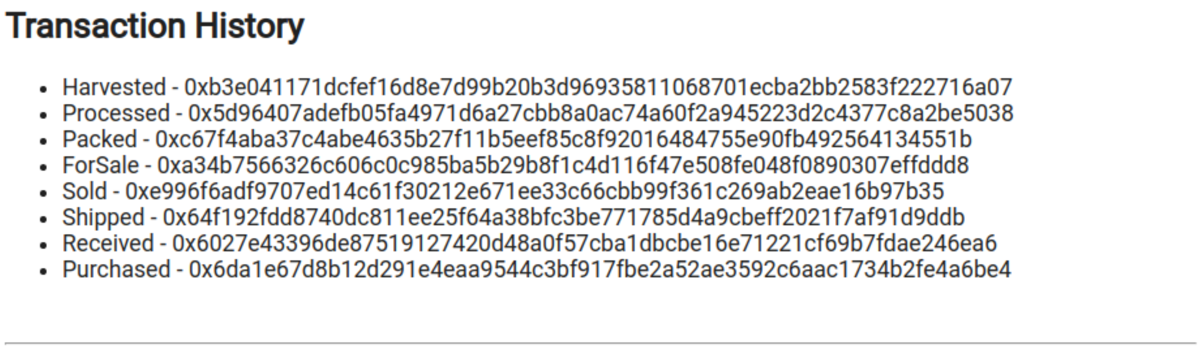


```
fetchItemBufferOne                                              app.js:326
 (8) [B, B, "0xd0ba908708eefcf8f51df18fc9da6f4b31c2a082", "0x83
▼1e8c098d30d42298fc07af714fb517a131f332", "yamuna-361", "Yamuna
 Valley", "-38.239770", "144.341490"] i
  ▼0: B
    ▶c: [1]
     e: 0
     s: 1
    ▶__proto__: Object
  ▼1: B
    ▶c: [1]
     e: 0
     s: 1
    ▶__proto__: Object
   2: "0xd0ba908708eefcf8f51df18fc9da6f4b31c2a082"
   3: "0x831e8c098d30d42298fc07af714fb517a131f332"
   4: "yamuna-361"
   5: "Yamuna Valley"
   6: "-38.239770"
   7: "144.341490"
   length: 8
  ▶__proto__: Array(0)
```



```
fetchItemBufferTwo                                      app.js:343
 (9) [B, B, B, "Organic red rice", B, B, "0xa7e11d81500c5e0a986
▼56d16bd3601ae77d0a6db", "0x6a41755bd8c3222c54d87e8d7dadc3486dd
 d6aee", "0xd0ba908708eefcf8f51df18fc9da6f4b31c2a082"] i
  ▶0: B {s: 1, e: 0, c: Array(1)}
  ▶1: B {s: 1, e: 0, c: Array(1)}
  ▼2: B
    ▶c: [2]
     e: 0
     s: 1
    ▶__proto__: Object
   3: "Organic red rice"
  ▼4: B
    ▶c: [20000]
     e: 18
     s: 1
    ▶__proto__: Object
  ▼5: B
    ▶c: [7]
     e: 0
     s: 1
    ▶__proto__: Object
   6: "0xa7e11d81500c5e0a98656d16bd3601ae77d0a6db"
   7: "0x6a41755bd8c3222c54d87e8d7dadc3486ddd6aee"
   8: "0xd0ba908708eefcf8f51df18fc9da6f4b31c2a082"
   length: 9
  ▶__proto__: Array(0)
```

9. RECEIVE AN ITEM (with consumer) :
    i. Add the fourth account listed in ganache in the metamask and connect it to localhost:3000.
    ii. Switch to it and verify the same in the console.
    iii. Press the purchase button.
    iv. You can check in **fetchItemBufferTwo,** the consumer address will get updated.
10. Complete Transaction history for upc = 1 will be printed at the bottom.



Transaction History

- Harvested - 0xb3e041171dcfef16d8e7d99b20b3d96935811068701ecba2bb2583f222716a07
- Processed - 0x5d96407adefb05fa4971d6a27cbb8a0ac74a60f2a945223d2c4377c8a2be5038
- Packed - 0xc67f4aba37c4abe4635b27f11b5eef85c8f92016484755e90fb492564134551b
- ForSale - 0xa34b7566326c606c0c985ba5b29b8f1c4d116f47e508fe048f0890307effddd8
- Sold - 0xe996f6adf9707ed14c61f30212e671ee33c66cbb99f361c269ab2eae16b97b35
- Shipped - 0x64f192fdd8740dc811ee25f64a38bfc3be771785d4a9cbeff2021f7af91d9ddb
- Received - 0x6027e43396de87519127420d48a0f57cba1dbcbe16e71221cf69b7fdae246ea6
- Purchased - 0x6da1e67d8b12d291e4eaa9544c3bf917fbe2a52ae3592c6aac1734b2fe4a6be4