Third-Party API's

Team Id	NM2023TMID04410
Project Name	Project-Drug Traceability

Third-Party API's in Drug Traceability:

In a drug traceability blockchain system, third-party APIs can play a crucial role in enhancing the functionality and data integration of the platform. Here are some ways in which third-party APIs can be used:

1. **Regulatory Compliance:**

   - Integrating APIs from regulatory bodies, such as the FDA or EMA, can help ensure that the system complies with the latest regulations and standards for drug traceability.

2. **Supply Chain Integration:**

   - APIs from supply chain partners, like pharmaceutical manufacturers, distributors, and retailers, can be used to track the movement of drugs throughout the supply chain, providing real-time data on product locations.

3. **Authentication and Verification:**

   - Third-party APIs can be utilized for identity verification and authentication, ensuring that only authorized entities have access to the blockchain network.

4. **Data Aggregation:**

   - APIs from data providers or aggregators can be used to enrich the blockchain with additional information, such as product batch details, expiration dates, and quality control data.

5. **IoT and Sensor Integration:**

   - APIs from Internet of Things (IoT) devices and sensors can feed real-time data into the blockchain, ensuring the quality and condition of drugs during transportation and storage.

6. **Serialization and Labeling:**

- APIs from labeling and serialization companies can be integrated to ensure that each drug package is uniquely identified and tracked on the blockchain.

7. **Data Analytics:**

- Third-party analytics APIs can be used to process and analyze the vast amount of data collected on the blockchain, providing valuable insights for stakeholders.

8. **Interoperability:**

- APIs can help in interoperability between different blockchain platforms or networks, facilitating data sharing and traceability across multiple organizations.
- Blockchain networks other than Ethereum work in their own unique way which leads to interoperability issues where the different blockchains are not able to communicate with each other.
- If a unified blockchain-based solution is used among healthcare centers, this problem can be avoided.
- However, if healthcare centers decide to use different blockchainbased solutions with different platforms, it will be very difficult to make them interoperable.

9. **Payment and Settlement:**

- APIs for financial transactions can be integrated for secure and transparent payment settlements within the supply chain.

10. **User Interface:**

- Third-party APIs can also be used to build user-friendly interfaces and mobile applications for end-users to access and interact with the blockchain system.

When implementing third-party APIs in a drug traceability blockchain, it's important to prioritize security, data integrity, and compliance with industry standards to ensure the trustworthiness and reliability of the system. Additionally, proper documentation and ongoing maintenance of these APIs are essential for the long-term success of the blockchain solution.

- The manufacturer will first deploy the smart contract in which details of the manufactured drug Lot will be defined, declared and an event will be triggered and announced to all participants in the supply chain. I

- n case new participants are added to the network, they will have access to the events since they are permanently stored on the ledger and therefore they can track and trace the history of any manufactured drug Lot.
- The manufacturer also has the option of uploading an image of the Lot to the IPFS so that it can be accessed by participating entities to visually inspect the drug Lot.

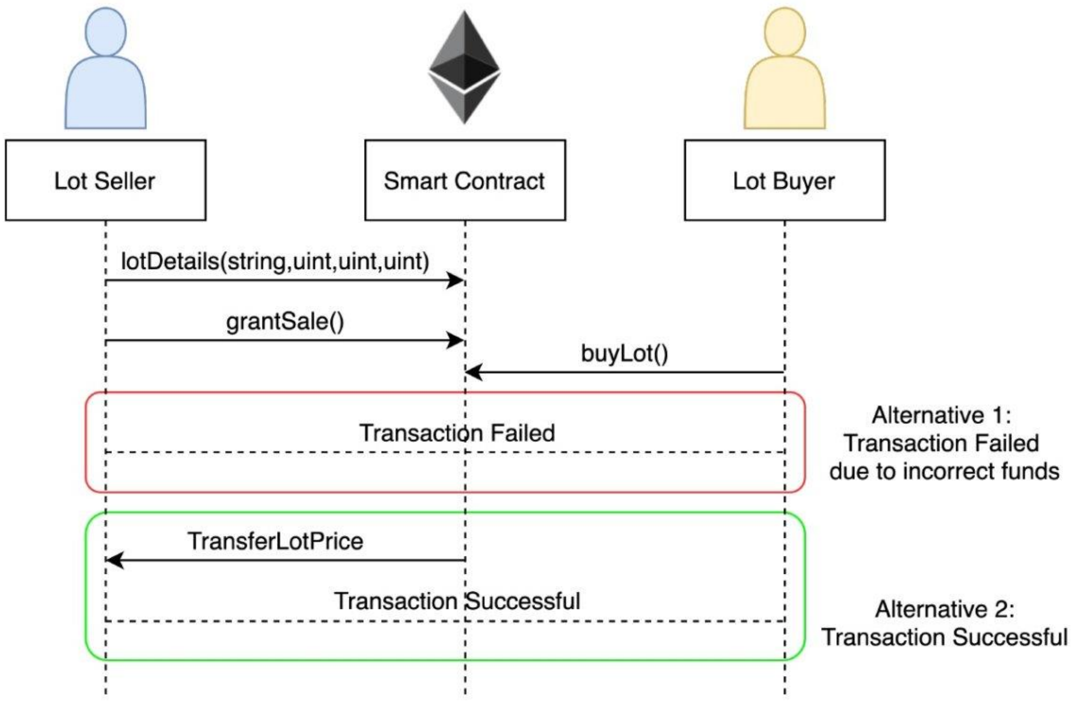

Lot Seller
Smart Contract
Lot Buyer
lotDetails(string,uint,uint,uint)
grantSale()
buyLot()
Transaction Failed
Alternative 1: Transaction Failed due to incorrect funds
TransferLotPrice
Transaction Successful
Alternative 2: Transaction Successful

FIG5. Function calls and events for two different scenarios for Lot sale

- Firstly, the Lot seller adds the Lot details by using the lotDetails functionoptionally uploading an image of that Lot to the IPFS as well as storing the hash along with other details of the Lot on chain. After that, the seller announces to all participants that the Lot is currently for sale via the grantSale function.
- An entity interested in buying the Lot executes the buyLot function which has two important requirements.
- Firstly, the executor of the function shouldn’t have the same Ethereum address as the owner of the Lot, and secondly, the buyer should have sufficient funds to buy the Lot, thereby leading to two possible alternatives.

- Specifically, if the specified funds by the buyer do not match the price of the Lot, the transaction will fail. However, if the funds are equal to the price of the Lot then the transaction will be considered successful, and the funds will be transferred to the buyer.

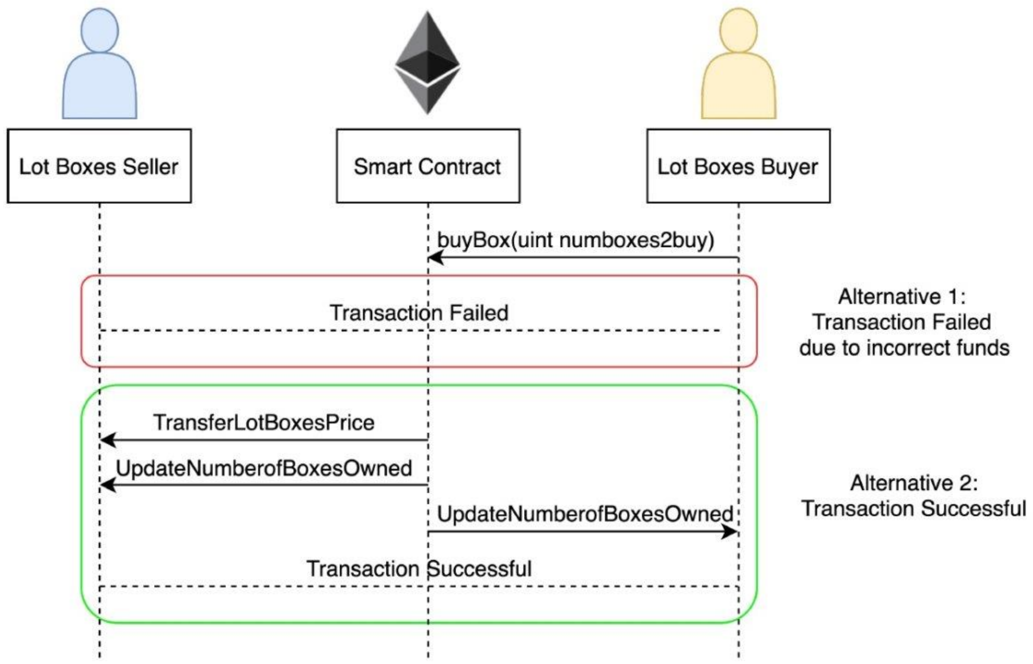


FIG. Function calls and events for two different scenarios for lot boxes sale

- In this case, the buyer will be specify a certain number of boxes from the Lot to be purchased. This scenario usually happens in a real pharmaceutical supply chain between the pharmacy and the patient.
- The buyer initiates the process by executing the buyBox function with the number of boxes needed and can result in two outcomes. If the transferred amount does not match the price of the boxes it will result in failure of the transaction.
- However, if the transferred amount is exactly equal to the price of the requested boxes, the price of the boxes will be transferred to the seller. Moreover, the number of boxes owned by the two entities will be updated according to the quantity purchased.

- Finally, the transaction will be finalized and declared successful to all participants. To further clarify the various functions of the smart contract, we present various algorithms used in our proposed solution.
- It should be noted that when referring to a buyer or a seller it only includes the authorized entities who are given the permission to execute the functions The following are the main functions with their corresponding algorithms.

Creating a Lot:

- Algorithm 1 explains the steps in creating a Lot. The inputs to the smart contract needed by the functions are shown with their descriptions.
- The function executes only if the address of the caller is the same as the address of the ownerID. If the caller is granted access, he/she will have the authority to update the fields in Algorithm 1.
- Once all fields have been updated the two events will update the status as shown in Algorithm 1.

Grant Lot Sale:

- The algorithm 2 describes Granting Lot Sale of the drug.
- This algorithm is responsible for sending an event alerting all participants that the Lot is currently available for sale, and can only be triggered if the caller is the ownerID holder.

Buying Lot:

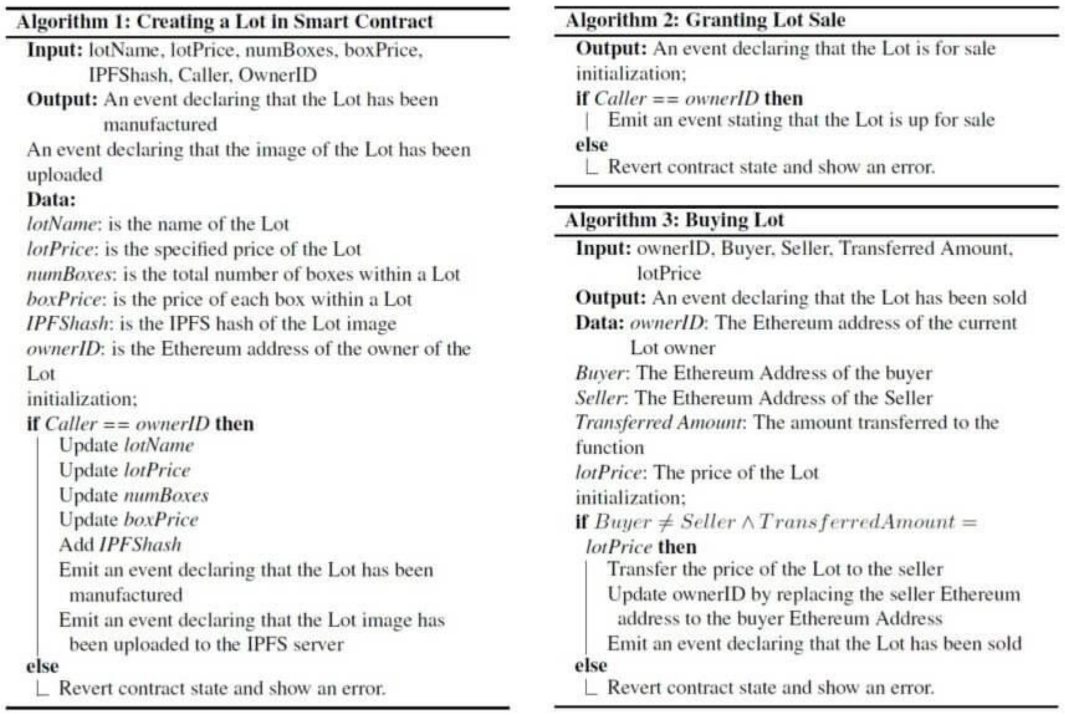
- Algorithm 3 describes the transactions between the buyer and the seller of the drug Lot. It requires the caller of the function (buyer) to not have the same address as seller (to ensures Lot owner doesn’t buy his own Lot) and requires the transferred amount to be exactly equal to the Lot price.
- Once both requirements are fulfilled, the sale amount will be transferred to the seller.
- Moreover, the ownerID will be updated. Finally, an event will be triggered to announce the sale of the Lot and update the new owner details. An important

thing to note here is that only trusted entities are allowed to use the smart contract.

- Therefore, when a Lot is announced sold, the buyer can rest assure that the seller is trusted and the Lot will be delivered.

Buying Lot Boxes:

- Algorithm 4 is similar to Algorithm 3, with subtle difference. The initiation of this algorithm is similar to Algorithm 3 but the buyer is required to specify the exact number of boxes within the Lot.
- The amount transferred by the buyer has to be exactly equal to the number of boxes the buyer wants to buy multiplied with the price of each box.
- The main difference here is that there is mapping for the addresses of the buyers with the number of boxes purchased, and the mapping gets updated every time this function gets executed successfully.



Algorithm 1: Creating a Lot in Smart Contract

```
Input: lotName, lotPrice, numBoxes, boxPrice,
       IPFShash, Caller, OwnerID
Output: An event declaring that the Lot has been
        manufactured
An event declaring that the image of the Lot has been
uploaded
Data:
lotName: is the name of the Lot
lotPrice: is the specified price of the Lot
numBoxes: is the total number of boxes within a Lot
boxPrice: is the price of each box within a Lot
IPFShash: is the IPFS hash of the Lot image
ownerID: is the Ethereum address of the owner of the
Lot
initialization;
if Caller == ownerID then
    Update lotName
    Update lotPrice
    Update numBoxes
    Update boxPrice
    Add IPFShash
    Emit an event declaring that the Lot has been
      manufactured
    Emit an event declaring that the Lot image has
      been uploaded to the IPFS server
else
    Revert contract state and show an error.
```

Algorithm 2: Granting Lot Sale

```
Output: An event declaring that the Lot is for sale
initialization;
if Caller == ownerID then
    Emit an event stating that the Lot is up for sale
else
    Revert contract state and show an error.
```

Algorithm 3: Buying Lot

```
Input: ownerID, Buyer, Seller, Transferred Amount,
       lotPrice
Output: An event declaring that the Lot has been sold
Data: ownerID: The Ethereum address of the current
      Lot owner
Buyer: The Ethereum Address of the buyer
Seller: The Ethereum Address of the Seller
Transferred Amount: The amount transferred to the
function
lotPrice: The price of the Lot
initialization;
if $Buyer \neq Seller \wedge TransferredAmount =$
  $lotPrice$ then
    Transfer the price of the Lot to the seller
    Update ownerID by replacing the seller Ethereum
      address to the buyer Ethereum Address
    Emit an event declaring that the Lot has been sold
else
    Revert contract state and show an error.
```

TRACEABILITY ANALYSIS OF THE PROPOSED SOLUTION:

- Every drug Lot is manufactured with a smart contract that is specifically designed for it and is responsible for triggering events and logging them on the ledger.
- A unique Ethereum address is generated for every drug Lot. However, copying Ethereum address of each drug is cumbersome, time consuming, and error prone process.
- Therefore, a QR code is used which can be easily scanned using smartphones. A QR code is a two-dimensional barcode that is readable by smartphones, and it can allow encoding over 4000 characters in a two dimensional barcode.
- Mapping an Ethereum address to a QR code can be done by using an Ethereum QR code generator in which the Ethereum address is passed and a unique QR code is generated which will exclusively map to that Ethereum address every time it gets scanned.
- Once the QR code gets attached to the drug Lot, it can be dispensed to patients.
- The first step is scanning the QR code that is attached to the drug by using a DApp which interacts with the Ethereum node (local or remote node) through web3j.
- To map the QR code to its corresponding Ethereum address, the DApp has to interact with the Ethereum node (Infura for example) through JSON-RPC.
- The Ethereum node has a replica of the ledger, and it is extremely important for the users because it makes the process smooth and easy by saving them the effort of having to set up their own Ethereum node which takes a lot of time.
- The gateway (Ethereum node) will map the Ethereum address of the drug Lot to the smart contract which will point to the events of the different functions of the smart contract that are stored in the ledger .

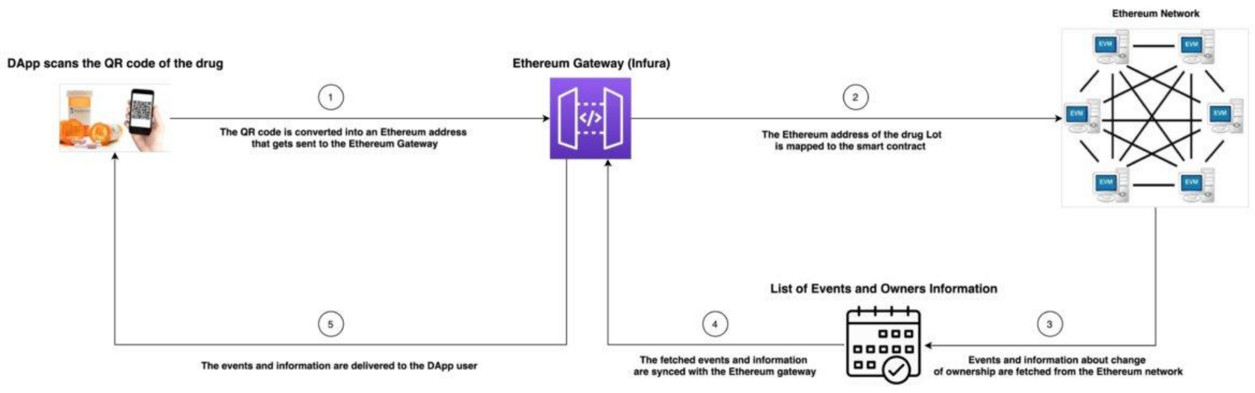

DApp scans the QR code of the drug
Ethereum Gateway (Infura)
Ethereum Network
EVM
1
The QR code is converted into an Ethereum address that gets sent to the Ethereum Gateway
2
The Ethereum address of the drug Lot is mapped to the smart contract
3
Events and information about change of ownership are fetched from the Ethereum network
List of Events and Owners Information
4
The fetched events and information are synced with the Ethereum gateway
5
The events and information are delivered to the DApp user

FIG. Application use case of the proposed blockchain-based solution

Algorithm 4: Buying Lot Boxes

```
Input: ownerID, Buyer, Seller, Transferred Amount,
       boxPrice, numBoxes, numBoxesToBuy,
       Transferred Amount, boxesPatient
Output: An event declaring that the Lot boxes have
        been sold
Data: ownerID: The Ethereum address of the current
      Lot owner
Buyer: The Ethereum Address of the buyer
Seller: The Ethereum Address of the Seller
Transferred Amount: The amount transferred to the
function
boxPrice: The price of the Lot box
numBoxes: The total number of boxes in the Lot
numBoxesToBuy: The number of boxes the buyer
wants to buy
boxesPatient: Maps the number of boxes bought to the
buyer address
initialization;
if $Buyer \neq Seller \wedge TransferredAmount =$
  $numBoxesToBuy * boxPrice$ then
    Transfer the price of the boxes to the seller
    Update ownerID by replacing the seller Ethereum
      address to the buyer Ethereum address
    Update numBoxes owned by the seller by
      decreasing the sold amount from it
    Update boxesPatient by assigning the purchased
      amount to the buyer address
else
    Revert contract state and show an error.
```

The service user will be able to verify the origin of the drug Lot by utilizing the event filtering feature which is based on the smart contract Ethereum address and the event name. Event filtering allows the service user to access the various events which are already stored on the immutable ledger of the Ethereum blockchain from which the service user can confirm if the drug is authentic or not. First, the service user can use the lotSold event to enter the pharmacy Ethereum address to verify the drug Lot was sold to the pharmacy legally. After that, the lotSale event can be used to fetch information about the drug Lot such as its name, number of boxes, and the price which allows the service user to verify that the pricing of the drug Lot is correct. Next, the imageuploaded event can be used to view the image of the manufactured Lot and boxes which shows if the product the service user receives matches the authentic one. Then, the lotManufactured event can be used to check if the Ethereum address of the manufacturer matches the original one. Finally, the newOwner event is used to view the Ethereum address of the original owner of the smart contract to confirm its authenticity.

The Ethereum network presented in Figure 7 illustrates how the information is distributed among the participating nodes. Each node in the network will have a replica of the ledger that is immutable which will ensure that any information that is fetched from the ledger is authentic and there is no way it has been manipulated with. The requested events and information about change of ownership will be fetched from the Ethereum network and they will be synced with the Ethereum gateway (Infura), and once the syncing is done they will be transferred to the DApp and displayed to the user [46].

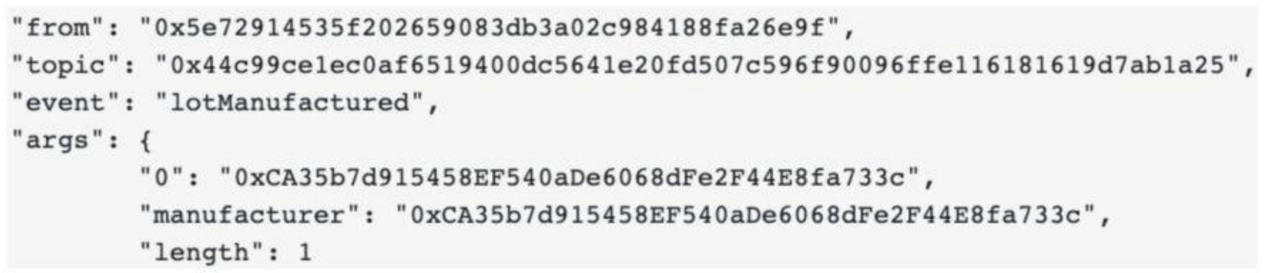
This application use case demonstrates the effectiveness of our proposed solution with respect to effective track and trace of drugs within a pharmaceutical supply chain. It achieves this by automating processes without requiring manual input from the user and utilizing different features of the Ethereum blockchain such as web3j, JSON-RPC, and Infura.

TESTING AND VALIDATION:

- In order to asses the smart contracts developed via Ethereum, Remix IDE in-browser developing and testing environment was used to test and validate different functions.
- The scenarios involved three different participants and their corresponding Ethereum Addresses as presented in Table 3.
- We further present the transactions and logs of the smart contract’s functions below

TABLE 3. The Ethereum address of each participant in the testing scenario

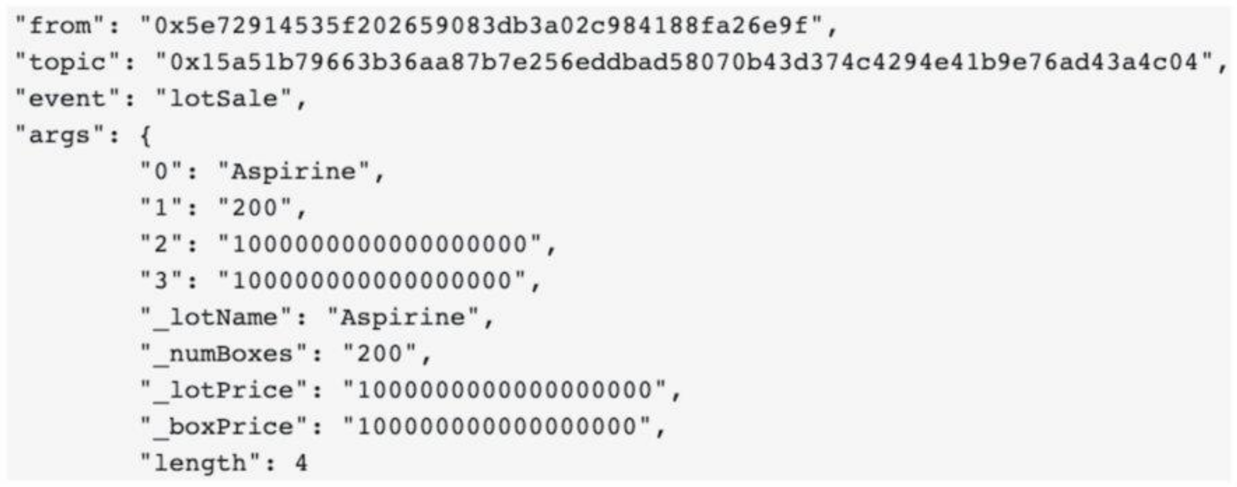
	Ethereum Address
Participant1	0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c
Participant2	0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C
Participant3	0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2Db



```
"from": "0x5e72914535f202659083db3a02c984188fa26e9f",
"topic": "0x44c99ce1ec0af6519400dc5641e20fd507c596f90096ffe116181619d7ab1a25",
"event": "lotManufactured",
"args": {
        "0": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c",
        "manufacturer": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c",
        "length": 1
```

FIGURE 8. Successful execution of lotDetails Function

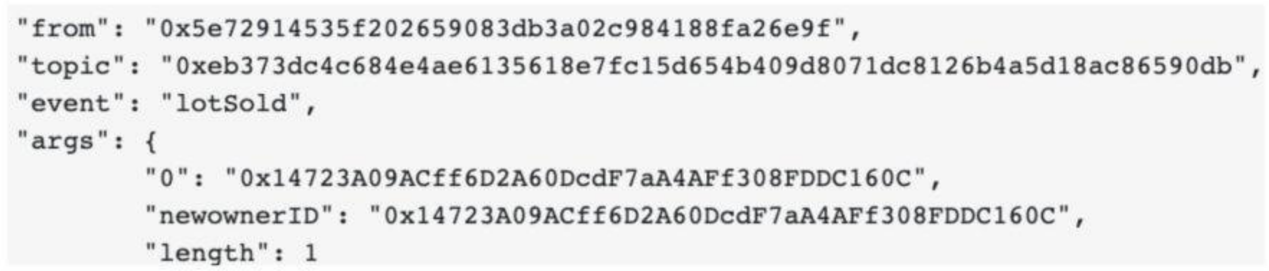
- **lotDetails:** In this function, it was tested whether the current owner of the smart contract is able to add the details of a newly manufactured Lot such as the Lot name, Lot price, number of boxes within the Lot, and the price of each box.
- A successful execution of the function and its corresponding logs and events are displayed in Figure 8.



```
"from": "0x5e72914535f202659083db3a02c984188fa26e9f",
"topic": "0x15a51b79663b36aa87b7e256eddbad58070b43d374c4294e41b9e76ad43a4c04",
"event": "lotSale",
"args": {
        "0": "Aspirine",
        "1": "200",
        "2": "1000000000000000000",
        "3": "100000000000000000",
        "_lotName": "Aspirine",
        "_numBoxes": "200",
        "_lotPrice": "1000000000000000000",
        "_boxPrice": "100000000000000000",
        "length": 4
```

FIGURE 9. Successful execution of grantSale Function

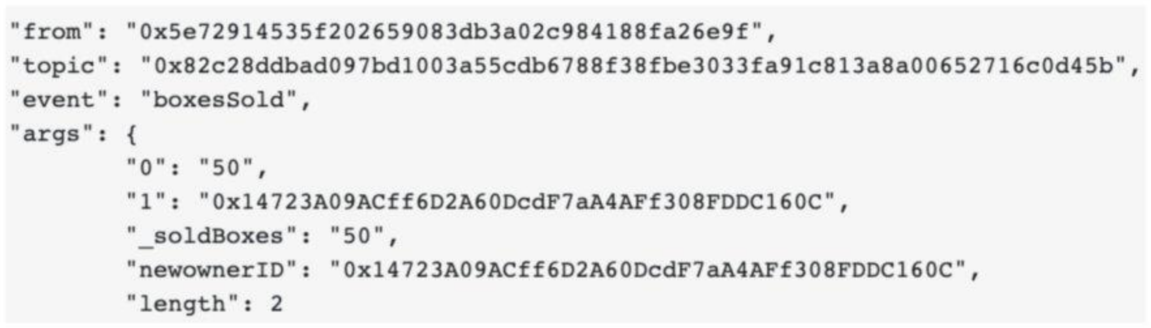
- **grantSale:** The grantSale function has a simple task yet it's very important, it basically notifies all the entities that the manufactured Lot is currently for sale. A successful execution of the function is given in Figure 9.



```
"from": "0x5e72914535f202659083db3a02c984188fa26e9f",
"topic": "0xeb373dc4c684e4ae6135618e7fc15d654b409d8071dc8126b4a5d18ac86590db",
"event": "lotSold",
"args": {
        "0": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
        "newownerID": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
        "length": 1
```

FIGURE 10. Successful execution of buyLot Function

- **buyLot:** In this function, Participant2 (Table 3 shows the corresponding Ethereum address) is used to buy the Lot from Participant1. Participant1 has specified the correct amount of ether (which is 1Ether as shown in Figure 8) to transfer and the successful execution of the function is shown in Figure 10.



```
"from": "0x5e72914535f202659083db3a02c984188fa26e9f",
"topic": "0x82c28ddbad097bd1003a55cdb6788f38fbe3033fa91c813a8a00652716c0d45b",
"event": "boxesSold",
"args": {
        "0": "50",
        "1": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
        "_soldBoxes": "50",
        "newownerID": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
        "length": 2
```

FIGURE 11. Successful execution of buyBox Function

- **buyBox:** This function deals with transactions related to purchase of specific number of boxes from the Lot (usually happens between a patient and the pharmacy). Figure 11 shows a successful execution of this function where Participant3 purchases 50 boxes from Participant2. The price of the boxes has been selected arbitrarily and they may not be logical but the purpose here is to confirm that the execution of the functions properly.