PROBLEM STATEMENT

This project implements transportation of Gasoline between a Producer and a Distributor where an order is initiated by a Distributor and accepted by the producer if sufficient stock is available. During transportation, quality and quantity check is maintained with the help of sensors. If specifications are not met and the quality deteriorates, then payment amount decreases accordingly.

MODEL DESIGN AND WORKING

7.1 Model Layout

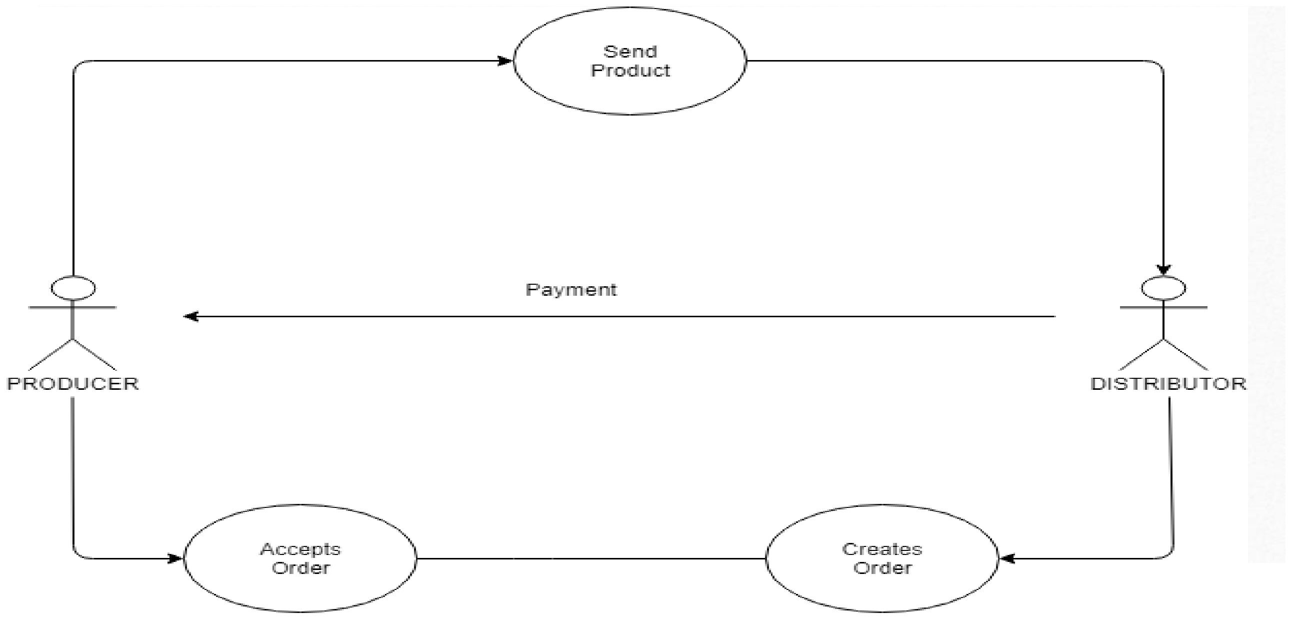


Figure 4: Model Layout

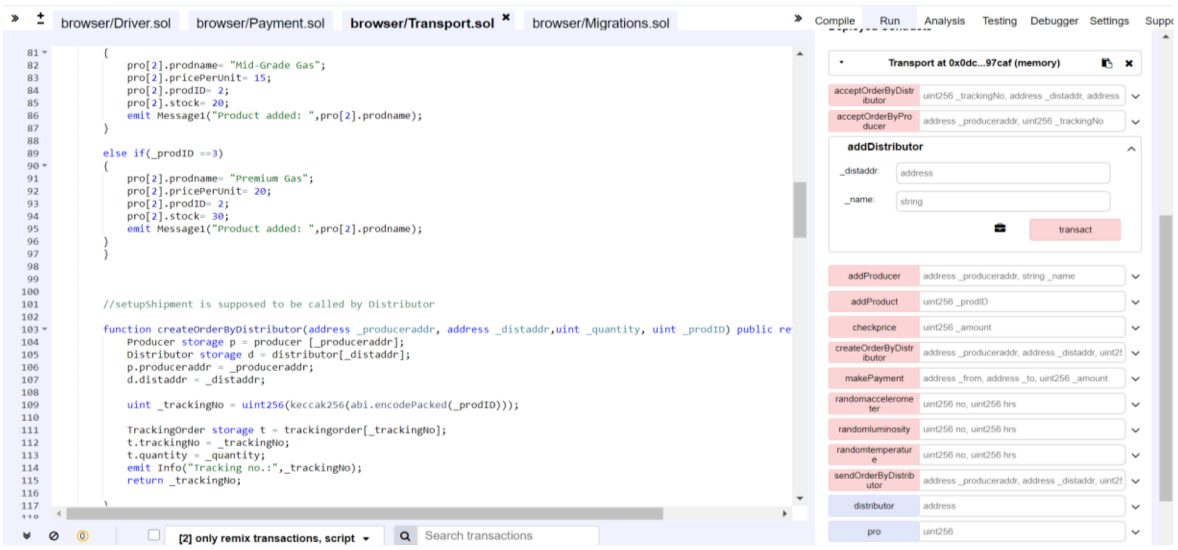
Entities encompass a Producer and a Distributor. Flow control for the model is as follows:

The Distributor initiates the order which consists of all the product details which is sent to the Producer. The Producer then checks his stock to see if he has sufficient quantity of the Gasoline. If yes, the order is accepted and sent to the Distributor with payment details. During transport, quality check is maintained. This process takes as input data from temperature, light and humidity sensor based on which product price decreases/increases according to the specified quality. Order is then accepted by the distributor and payment is sent to the producer’s account.

CONTRACTS USED IN PROJECT

Transport.sol

This contract takes care from placing an order by the Distributor to accepting that order by Producer and trasvering all the details to the driver contract.



```
        {
            pro[2].prodname= "Mid-Grade Gas";
            pro[2].pricePerUnit= 15;
            pro[2].prodID= 2;
            pro[2].stock= 20;
            emit Message1("Product added: ",pro[2].prodname);
        }

        else if(_prodID ==3)
        {
            pro[2].prodname= "Premium Gas";
            pro[2].pricePerUnit= 20;
            pro[2].prodID= 2;
            pro[2].stock= 30;
            emit Message1("Product added: ",pro[2].prodname);
        }
        }



        //setupShipment is supposed to be called by Distributor

        function createOrderByDistributor(address _produceraddr, address _distaddr,uint _quantity, uint _prodID) public re
            Producer storage p = producer [_produceraddr];
            Distributor storage d = distributor[_distaddr];
            p.produceraddr = _produceraddr;
            d.distaddr = _distaddr;

            uint _trackingNo = uint256(keccak256(abi.encodePacked(_prodID)));

            TrackingOrder storage t = trackingorder[_trackingNo];
            t.trackingNo = _trackingNo;
            t.quantity = _quantity;
            emit Info("Tracking no.:",_trackingNo);
            return _trackingNo;
```

Transport at 0x0dc...97caf (memory)

acceptOrderByDistributor	uint256 _trackingNo, address _distaddr, address
acceptOrderByProducer	address _produceraddr, uint256 _trackingNo

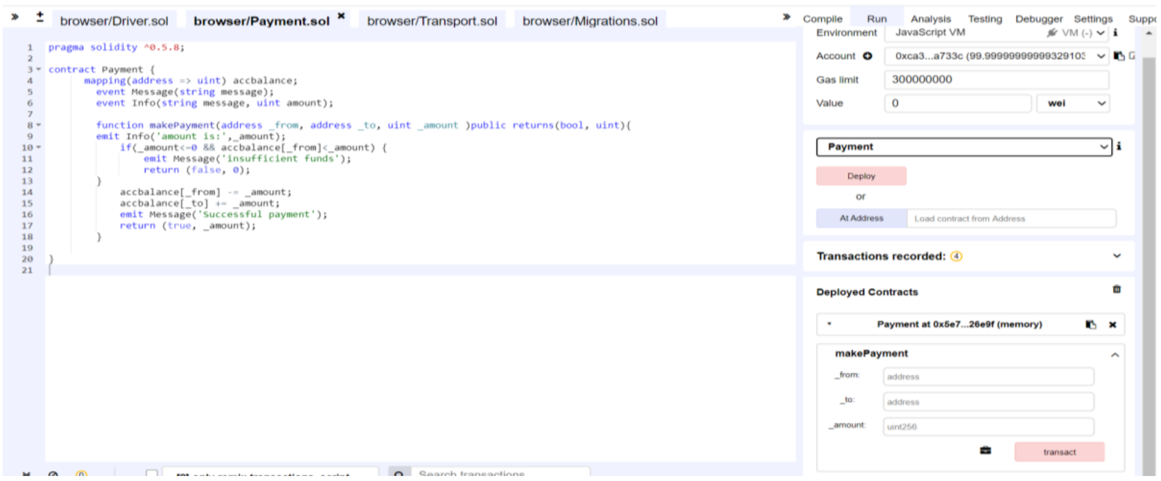
addDistributor
_distaddr: address
_name: string
transact

addProducer	address _produceraddr, string _name
addProduct	uint256 _prodID
checkprice	uint256 _amount
createOrderByDistributor	address _produceraddr, address _distaddr, uint2
makePayment	address _from, address _to, uint256 _amount
randomaccelerometer	uint256 no, uint256 hrs
randomluminosity	uint256 no, uint256 hrs
randomtemperature	uint256 no, uint256 hrs
sendOrderByDistributor	address _produceraddr, address _distaddr, uint2
distributor	address
pro	uint256

[2] only remix transactions, script

Payment.sol

This contract takes care of all the payments being done between the Distributor and the Producer, by taking the addresses and amount as it's input.



```
pragma solidity ^0.5.8;

contract Payment {
        mapping(address => uint) accbalance;
          event Message(string message);
          event Info(string message, uint amount);

          function makePayment(address _from, address _to, uint _amount )public returns(bool, uint){
          emit Info('amount is:',_amount);
              if(_amount<=0 && accbalance[_from]<_amount) {
                  emit Message('insufficient funds');
                  return (false, 0);
          }
              accbalance[_from] -= _amount;
              accbalance[_to] += _amount;
              emit Message('Successful payment');
              return (true, _amount);
          }

}
```

Environment: JavaScript VM
Account: 0xca3...a733c (99.9999999999932910
Gas limit: 300000000
Value: 0 wei

Payment
Deploy
or
At Address: Load contract from Address

Transactions recorded: 4

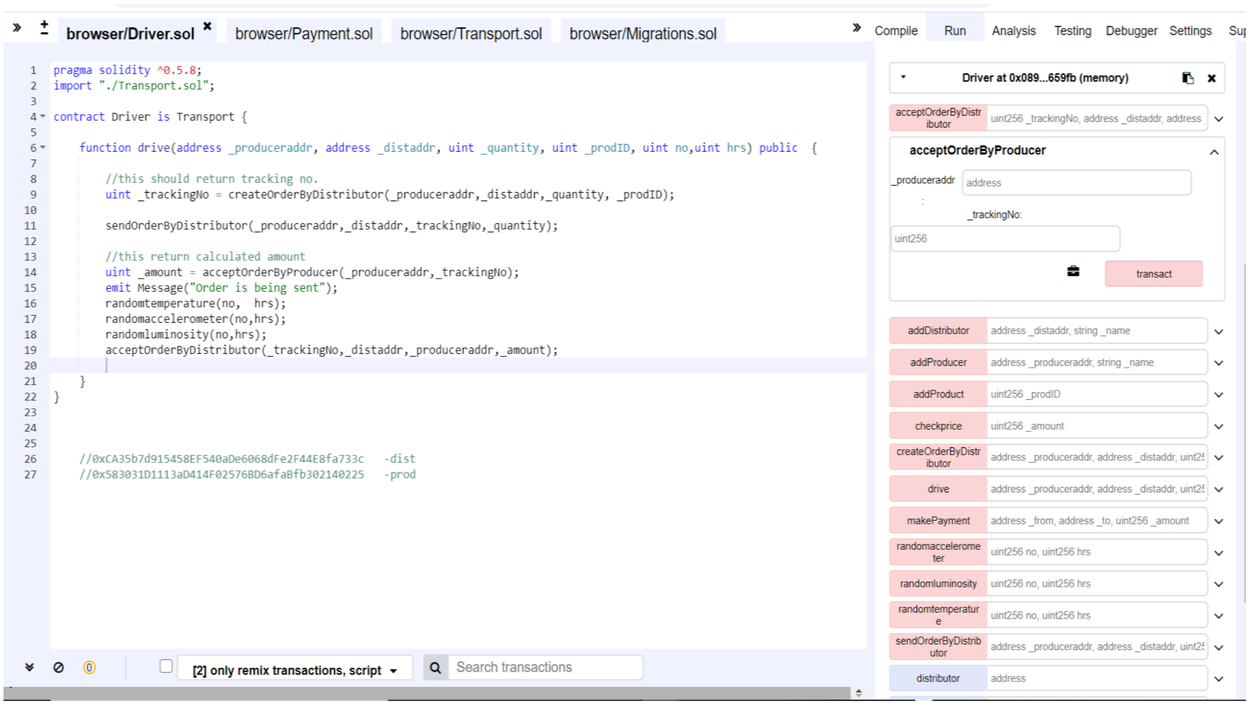
Deployed Contracts

Payment at 0x5e7...26e9f (memory)

makePayment
_from: address
_to: address
_amount: uint256
transact

Driver.sol

This contract generates a unique tracking number for every order being placed by the Distributor and that unique tracking number is used for accepting the order by the Producer and maintain a record for the transportation.



```
pragma solidity ^0.5.8;
import "./Transport.sol";

contract Driver is Transport {

    function drive(address _produceraddr, address _distaddr, uint _quantity, uint _prodID, uint no,uint hrs) public  {

        //this should return tracking no.
        uint _trackingNo = createOrderByDistributor(_produceraddr,_distaddr,_quantity, _prodID);

        sendOrderByDistributor(_produceraddr,_distaddr,_trackingNo,_quantity);

        //this return calculated amount
        uint _amount = acceptOrderByProducer(_produceraddr,_trackingNo);
        emit Message("Order is being sent");
        randomtemperature(no,  hrs);
        randomaccelerometer(no,hrs);
        randomluminosity(no,hrs);
        acceptOrderByDistributor(_trackingNo,_distaddr,_produceraddr,_amount);

    }
}



    //0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c   -dist
    //0x583031D1113aD414F02576BD6afaBfb302140225   -prod
```

Driver at 0x089...659fb (memory)

acceptOrderByDistributor	uint256 _trackingNo, address _distaddr, address

acceptOrderByProducer

_produceraddr: address

_trackingNo: uint256

transact

addDistributor	address _distaddr, string _name
addProducer	address _produceraddr, string _name
addProduct	uint256 _prodID
checkprice	uint256 _amount
createOrderByDistributor	address _produceraddr, address _distaddr, uint2
drive	address _produceraddr, address _distaddr, uint2
makePayment	address _from, address _to, uint256 _amount
randomaccelerometer	uint256 no, uint256 hrs
randomluminosity	uint256 no, uint256 hrs
randomtemperature	uint256 no, uint256 hrs
sendOrderByDistributor	address _produceraddr, address _distaddr, uint2
distributor	address

CONCLUSION

In this project we addressed the potential of Blockchain. We delved into the benefits of Blockchain compared to the traditional centralized system that is currently being used.

Overall, we confer that the use of Blockchain in transportation of Gasoline is a potential use case. We have designed a basic working model of a Blockchain-based Application for transportation of Gasoline.