Process And Lineage Capture Tool

Team Members:

1.Swapnil Kole

2.Pooja Das

3.Pallavi Mishra

4.Atipha Mohanty

ABSTRACT

In today's fast-changing business environment, it's extremely important to be able to respond to client needs in the most effective and timely manner. If the customers wish to see the business online and have instant access to the products or services. Any business process is executed in steps with dependencies.In order to visualize the entire process lineage,a number of Technologies is used. These include Blockchain, Smart Contract, React.js., Web3.js ,Metamask and Ethereum .

1. INTRODUCTION

This document gives an overview of our application named Process and Lineage capture tool which we've developed keeping in mind the necessity to be able to respond to client's need to interact with business and products online and tell them about the business process.
Our area of work is to develop an application that visualises the entire process lineage,mandates unique identity proof of the users,secures the data in a decentralized database and secures the entire transaction process.We're expecting to achieve maximum efficiency and transparency in terms of security of user data through our business model,ensuring user satisfaction at the same time.

2. OBJECTIVE

In this Project our main objective is to develop a Dapplication to realize the entire process flow and identify its associated dependencies and bottlenecks in the process of Supply Chain. With this project we can detect and diagnose the process cycle to optimize the process which eventually leads to increase in productivity of the business.

This can be implemented using smart contracts which is platform-independent and can be designed using ETHEREUM,Co-Equal,Hyperledger-Fabric etc. Here, we've used Ethereum and all of this is integrated as a decentralized Application[DAPP]

3. LITERATURE REVIEW

The main technologies used in this project are:

A brief description of all these technologies follows:

- **Blockchain:** It is a distributed ledger-based technology that uses consensus-based decisions to come to a single point of truth. It involves three main technologies which are private key cryptography, peer-2-peer network, and Blockchain protocol. The data once entered becomes immutable.

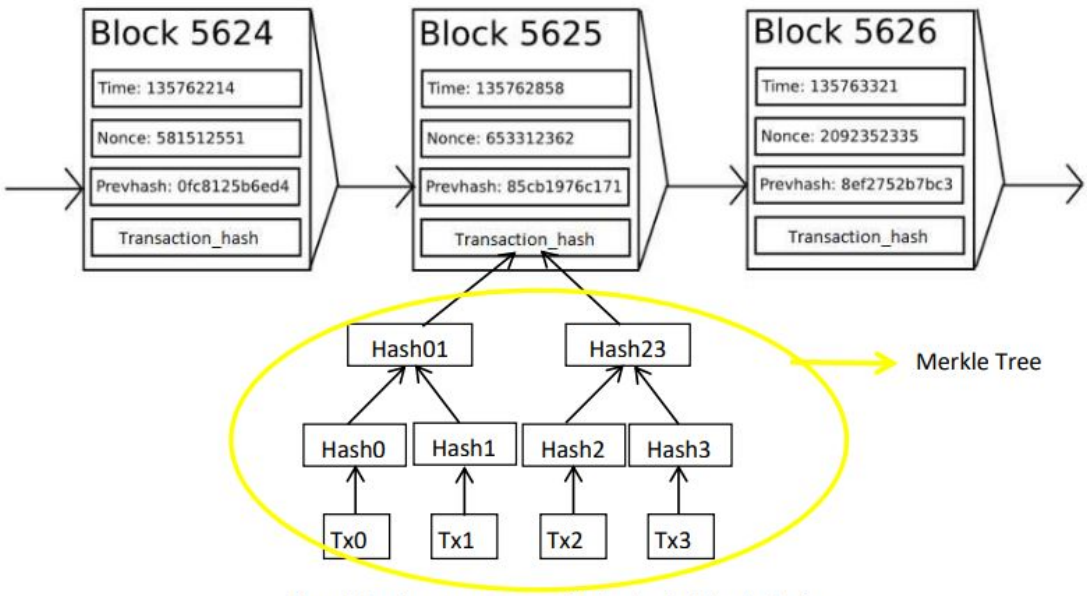


Fig. 1 Basic structure of blocks in Blockchain

Blocks in blockchain hold batches of valid transactions that are hashed and encoded into a Merkle tree. Each block includes the cryptographic hash of the prior block in the blockchain, linking the two. The linked blocks form a chain. This iterative process confirms the integrity of the previous block, all the way back to the original genesis block. Each block also has a timestamp and a nonce associated with it.

- **Ethereum:** Ethereum is an open software platform based on Blockchain technology that enables developers to build and deploy decentralized applications. Ethereum's coding language solidity helps write smart contracts. Its native currency is eth. It was founded by Vitalik Buterin.

- **Smart Contracts:** A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties. These transactions are trackable and irreversible.

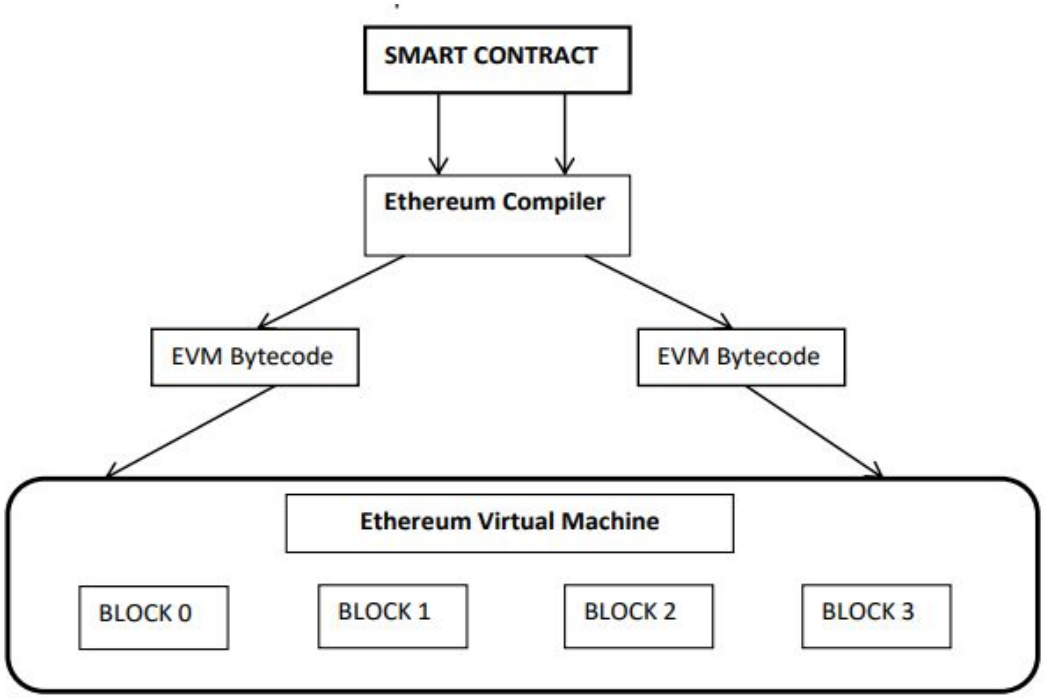


Fig 3: Execution of a smart contract

- **Ethereum Virtual Machine:** An Ethereum virtual machine provides a run anywhere obstruction layer for the smart contract. A smart contract written in HLL is translated into EVM bytecode and then deployed on EVM. Every node will host the smart contract codes on the EVM.

➢ **Dapplication:** A decentralized application is a computer application that runs on a distributed computing system. They have distributed ledger[DLT] based technology. It has a web-front and blockchain back-end and the smart-contract connecting both.

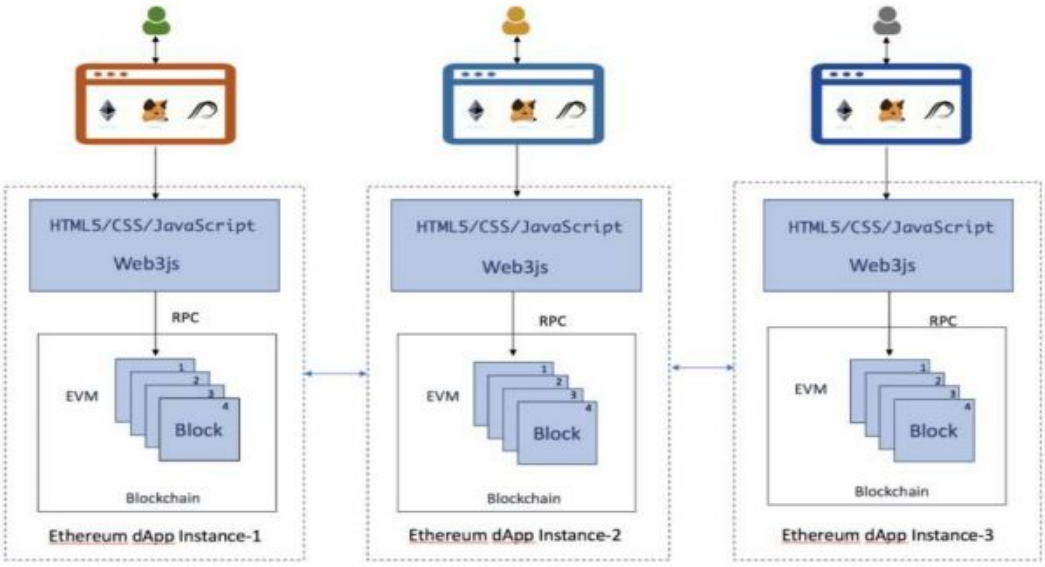


Fig 4: Ethereum Dapp instance

➢ **Solidity:** Solidity is a contract oriented language. It is designed to target the Ethereum Virtual Machine. It is statically typed language, supporting inheritance, libraries and complex user-defined types.

➢ **Hashing Algorithm:** It plays a crucial role in the blockchain process and also in the integrity of the transaction and confidentiality of data. It transforms and maps an arbitrary length of input data value to a unique fixed-length value. The algorithm should be one-way function and collision-free. Some majorly used hashing functions are SHA-256 and Keccak.

➢ **React-Js. :** We've used React.js to efficiently update and render the right components with data change thus making our code predictable and easier to debug. React can render

on the server using Node and power mobile apps using React Native. We've used Reactstrap instead of templates to pass rich data through our app.

- **Web3.Js :** Web3.js enables you to fulfill the responsibility of developing clients that interact with The Ethereum Blockchain. It is a collection of libraries that allow you to perform actions like send Ether from one account to another, read and write data from smart contracts, create smart contracts, and so much more!

4. WORK

- **Contract-Design:**This blockchain is based on a decentralized database with ethereum as its main technology. The smart contract will be coded as desired and will be deployed by the Ethereum Virtual Machine. This work will have 4 groups of users involved: government service providers, Manufacturers & Customers and general people who'll visit our Dapplication to get an idea about items,the GST on them, their respective MRPs, and offers.

- **Remix:** Remix Ethereum IDE is an open source web and desktop application. It fosters a fast development cycle and has a rich set of plugins with interactive GUIs. Remix is used for the entire journey of contract development as well as being a playground for learning Ethereum.
  Remix IDE is part of the Remix Project which is a platform for development tools that use a plugin architecture. It encompasses sub-projects including Remix Plugin Engine, Remix Libs, and of course Remix-IDE.We've made use of Remix Ethereum IDE that enables us to develop,deploy and administer smart contacts for Ethereum like **Block chain.**

- **Ganache:**Ethereum Ganache is a local in-memory blockchain designed for development and testing. It simulates the features of a real Ethereum network, including the availability of a number of accounts funded with test Ether. Here,we get a unique address for each person which can be accessed by that particular person only.

- **Metamask:**Metamask is a crypto-currency **wallet** available as a browser extension and as a mobile app.It generates passwords and keys on one's device, so only that person has access to his accounts and data. .It is the most secure way to connect to blockchain-based

applications.In our application, the customer or the manufacturer needs to connect to the wallet with the same address to do any kind of process or transaction so as to ensure security and uniqueness of identity.

❖ **FRONTEND**

➢ **Home:** Any Visitor/Customer/Manufacturer first takes a glance at the home page of the application where he gets the know about various new launches and products that the application deals with; be it purchasing or manufacturing. From home page he is redirected to all other pages through various

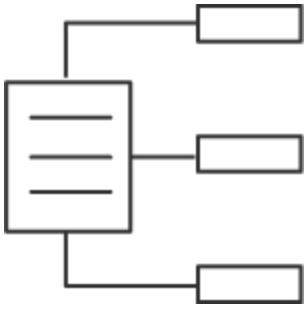
➢ **Registration/Update:** The new customer or the manufacturer needs to register himself first using the Registration Page.This is an important step as there are different functions available for the customer and manufacturer respectively in the website.Also, this creates an unique Id for each customer and the manufacturer in smart contract.Once they have registered they can login any time using login page as their Id is already stored.

➢ **Product:** If a customer logins the page then he can click on the particular product & add the no.of quantity which he wants to buy.The total amount will be generated automatically which has to be paid by the customer once the confirmation button is clicked.If the manufacturer logins the page then he gets an additional button known as Add item.In this the manufacturer can add his item to the to the product page.This button is not visible to the customer.

➢ **Shipment :** Once the customer confirms the product,it is directly shown to the customer in the shipment page along with the process and payment status.It has specific customer and manufacturer address.Order details will only be visible to that particular customer and manufacturer address.

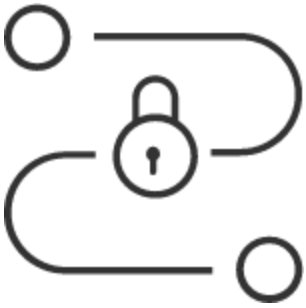
❖ **BACKEND**

➢ **Block chain:**Blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network. An **asset** can be tangible or intangible (intellectual property, patents, copyrights, branding). Virtually anything of value can be tracked and traded on a blockchain network, reducing risk and cutting costs for all involved.



Distributed ledger technology

All network participants have access to the distributed ledger and its immutable record of transactions. With this shared ledger, transactions are recorded only once, eliminating the duplication of effort that's typical of traditional business networks.



Records are immutable

No participant can change or tamper with a transaction after it's been recorded to the shared ledger. If a transaction record includes an error, a new transaction must be added to reverse the error, and both transactions are then visible.

Smart contracts

To speed transactions, a set of rules – called a smart contract – is running on the blockchain and executed automatically. A smart contract can define conditions which can automatically execute

& control actions that are mapped to real-world events and it works according to the terms of a contract or an agreement that we've written .

- **DEFI-App:** Stands for Decentralized Finance Application; DeFi draws inspiration from **blockchain**, the technology behind the digital **crypto-currency**, which allows several entities to hold a copy of a history of transactions, meaning it isn't controlled by a single, central source. That's important because centralized systems and human gatekeepers can limit the speed and sophistication of transactions while offering users less direct control over their money. DeFi is distinct because it expands the use of blockchain from simple value transfer to more complex financial use cases.
  We've made use of DEFi-App to run the financial system of our application on block chain using crypto-currencies and also gearing towards disrupting financial intermediaries at the same time.
- **Smart Contracts:** As we were very sure of what we want for our application without any intermediaries,we thought of implementing smart contracts in our application that automatically executes and transacts on the basis of the logic we've put in the smart contract. At the end of the day,our very goal is to ensure transparency,simplicity and efficiency, automatization without intermediaries in every transaction that is taking place in our system.

❖ **INTEGRATION**

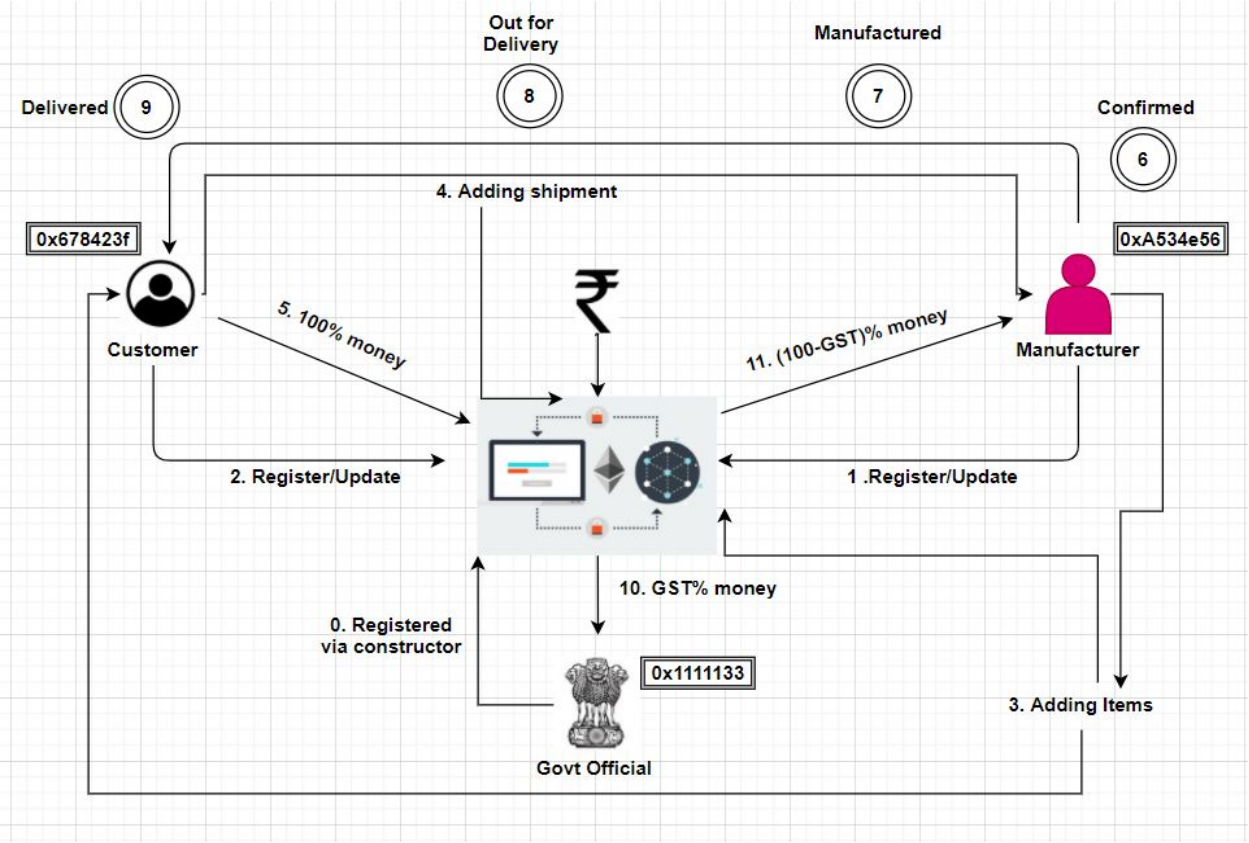
- **Web3.Js:** We've used Web3.Js library to integrate our React application with the Smart Contract to call all functions from Smart Contract and to retrieve data from the block chain. In short, we're performing read-write operations on the Blockchain.
  Also,the Web3.Js directly puts a notification with the metamask wallet enabling the user to easily confirm his transaction.

5.PROCESS-FLOW

In this project initially the government service provider will deploy the smart contract on the Ethereum Virtual Machine. Then all the Manufacturers and Customers will request for Registering themselves. Once registered, the Customer'/Manufacturer's public key is mapped with his aadhar no. and this aadhar no. will be mapped with his data. Based on the public key the users can view their activity on the application anytime and can be notified if in case of a process failure. By getting

access through their unique addresses through metamask,they can easily get information about the exact timings of process failure/success. Once a customer pays securely through his account the payment is stored in the smart-contract and the manufacturer gets notified about the same and hence proceeds with the manufacturing and shipment stages one-by one which can be viewed by both Manufacturer and Customer. We've made provision for the Govt Officials to inspect the payment process flow at any desired time through Smart Contract which makes our application unique to true sense.Also, the general people will be provided with a Dapplication that will have a web3 connection directly to the smart contract.

Here, we attach the brief **Process Flow.**



❖ Also,we'd like to attach the **Smart Contract Code** for your reference below:

```
pragma solidity ^0.6.0;
pragma experimental ABIEncoderV2;
contract ProLine{
 address payable public govt;
  constructor(address payable _govt)public{
     govt = _govt;
  }
  enum State { Added,Pending,Confirmed,Manufactured,Outfordel,Delivered,Cancelled }
  enum Status {Notpaid, inSc, received}
  uint public manufacturercount = 0;
  uint public customercount = 0;
  uint public itemcount = 0;
  uint public shipmentcount = 0;

  event processchange(uint indexed ship_id,State indexed shstate,uint times);
  event processpay(uint indexed ship_id,Status indexed pay,uint times);

  struct Manufacturer{
     string manuname;
     uint manuid;
     address payable manuadd;
     uint manupincode;
  }

  mapping(address  => Manufacturer) public Manufacturers;
  mapping(uint => address ) public Manu_ids;

  struct Customer{
     string custname;
     uint custid;
     address custadd;
     uint custpincode;
  }
 mapping(address => Customer) public Customers;
  mapping(uint => address ) public cust_ids;
 struct Item {
     uint itemid;
     uint itemtype;
     address manadr;
     string description;
     uint price;
     uint gst;
  }

  struct Shipment{
     uint shid;
     uint itemcat;
     uint qty;
```

```
        State shipstate;
        uint totalamt;
        Status payment;
        address custadr;
        address manadr;
    }

    mapping(uint => Item) public Items;
mapping(uint => Shipment) public Shipments;
   function addManufacturer(string memory _name,uint _pincode)public {
        manufacturercount++;
        Manufacturers[msg.sender].manuname = _name;
        Manufacturers[msg.sender].manuid = manufacturercount;
        Manufacturers[msg.sender].manuadd = msg.sender;
        Manufacturers[msg.sender].manupincode = _pincode;
        Manu_ids[manufacturercount] = msg.sender;

    }
    function updateManufacturer(string memory _name,uint _pincode)public {

        Manufacturers[msg.sender].manuname = _name;
        Manufacturers[msg.sender].manuadd = msg.sender;
        Manufacturers[msg.sender].manupincode = _pincode;
    }

    function addCustomer(string memory _custname,uint _pincode)public{
        customercount++;
        Customers[msg.sender].custname = _custname;
        Customers[msg.sender].custid = customercount;
        Customers[msg.sender].custadd = msg.sender;
        Customers[msg.sender].custpincode = _pincode;
        cust_ids[customercount] = msg.sender;
    }
    function modifyCustomer(string memory _custname,uint _pincode)public{
        Customers[msg.sender].custname = _custname;
        Customers[msg.sender].custadd = msg.sender;
        Customers[msg.sender].custpincode = _pincode;
    }
    function createItems(uint _itemtype,string memory _description,uint _price,uint _gst)public{
        itemcount++;
        Items[itemcount].itemid = itemcount;
        Items[itemcount].itemtype = _itemtype;
        Items[itemcount].description = _description;
        Items[itemcount].price = _price;
        Items[itemcount].gst = _gst;
        Items[itemcount].manadr = msg.sender;

    }
```

```
  function createShipment(uint _itemid,uint _qty,State _shipstate,uint _totalamt,Status
_payment,address payable _manadr)public{
      shipmentcount++;
      Shipments[shipmentcount].shid = shipmentcount;
      Shipments[shipmentcount].itemcat =  _itemid;
      Shipments[shipmentcount].qty =  _qty;
      Shipments[shipmentcount].shipstate = _shipstate;
      Shipments[shipmentcount].payment=  _payment;
      Shipments[shipmentcount].totalamt =  _totalamt;
      Shipments[shipmentcount].custadr =  msg.sender;
      Shipments[shipmentcount].manadr =  _manadr;
      emit processchange(shipmentcount,_shipstate,now);
      emit processpay(shipmentcount,_payment,now);

    }
  function updateShstate(uint _shipid,State _shipstate)public{
    require(Shipments[_shipid].shipstate != State.Cancelled,"Cancelled");
      Shipments[_shipid].shipstate = _shipstate;
      emit processchange(_shipid,_shipstate,now);
  }

  function updateShstatus(uint _shipid,Status _payment)public{
      Shipments[shipmentcount].payment=  _payment;
      emit processpay(_shipid,_payment,now);
  }
   function payitem(uint totamt,uint _shipid)public payable {
      require(msg.value == totamt,"less money");
      Shipments[_shipid].payment = Status.inSc;
      emit processpay(_shipid,Status.inSc,now);

    }
   function withdrawmoney(uint _shipid)public payable{
      Shipment memory y = Shipments[_shipid];
      Item memory x = Items[y.itemcat];
      uint totalpay = x.price*(y.qty);
      if(y.shipstate == State.Cancelled){
          address payable cus = payable(y.custadr);
          cus.transfer(totalpay);
      }
      else{
      require(y.shipstate == State.Delivered);
      require(y.payment == Status.inSc);
      uint govtmoney = (totalpay*x.gst)/100;
      uint manumoney = totalpay - govtmoney;
      govt.transfer(govtmoney);
      address payable mann = payable(y.manadr);
      mann.transfer(manumoney);
```

```
        Shipments[_shipid].payment = Status.received;
        emit processpay(_shipid,Status.received,now);

      }
    }
}
    function getbal(address addr)public view returns(uint){
      return (addr).balance;
    }
}
```

6. CONCLUSION

Data immutability and privacy is achieved by this system and now all the data is stored in a decentralized database forever,so no one hacks data from a decentralized database thus ensuring security and integrity. The Consumer/Manufacturer who wants to access his data has to input an unique address to prove identity and proceed as per his needs. The tracker directly accesses the data of the items and shipments and shows it to the concerned parties. Also, we'd like to highlight upon the fact that the uniqueness of our Dapplication is because of this system in which the GST is sent to the government without any intermediaries.

7.FURTHER RESEARCH

We are looking forward to the application of evidence-based behavioral insights and predictive analytics in our DAPP. To not settle for traditional, easily gameable metrics, we're thinking of digging deep into the world of process mining and to eliminate system complexity with the **Execution Management System**.

8.REFERENCES

I. Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," www.bitcoin.org
II. Gavin Wood, "Solidity: readthedocs," https://solidity.readthedocs.io/
III.Buterin, V, "Ethereum White Paper," https://github.com/ethereum/wiki/wiki/White-Paper

THANK YOU