Blockchain Technology
Final Year B.Tech 2021-2022

A PROJECT REPORT ON

“BLOCKCHAIN IN FOOD SUPPLY CHAIN"

Group No: 27

Sr No.	Name	ID	Department
1	Shraddha Khedkar	191081904	IT
2	Bhagyshri Walde	191081907	IT
3	Manjushree Ghadge	191081908	IT
4	Tejas Jhamnani	181070067	Computer

Under the guidance of

Dr. Dhiren Patel

Dr. Mahesh Shirole

VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE, MATUNGA, MUMBAI -400019

INDEX

1. INTRODUCTION

1.1 Abstract

With the increase in the complexity of the industrial food supply chain, the traceability is becoming scarce. It has becoming a tedious and costly task to trace the processing of a packed product, as the data is stored in the relational databases it is quite easy to tamper the data stored in the database through an unauthorized access.To tackle all these issues and address the problem of traceability Blockchain is a suitable technique to store the data in the database. Blockchain technology offers important opportunities for supply chain management.

This aims to overview employment of blockchain technology in the field of supply chain. Although the technology has been widely associated with cryptocurrencies, non-financial applications such as supply chain, power and food industry are also promising. Blockchain can provide a permanent, shareable, auditable record of products through their supply chain, which improves product traceability, authenticity, and legality in a more cost-effective way.

By making the use of blockchain we can generate a ledger where each processing unit will make the transactions in the system and all the data related to the processing of product made at that unit is appended easily to the blockchain. By doing this, we are creating a tamper free ledger which is used to trace the processing details of the product in a really fast and cost effective manner.

We are here, creating a peer to peer network of different systems which will keep on making the transactions and the data will be stored in the ledger. A server will be there to monitor the transactions and to trace the product details if requested by the system administrator or by the end user. There will be a backup server also which will take the command of the requests once the main server becomes down and it also maintains a copy of the whole data. In this way it is feasible to trace the processing details of a product, once unique identity of the product is provided to the server.

1.2 Introduction and Motivation

Let’s take an example of Google spreadsheet or MS Excel (Windows). This spreadsheet is shared among different networks of computer, where everyone has copy of it. The spreadsheet contains information of the transactions committed by real people. Anyone can access that spreadsheet but no one can edit it. This is Blockchain. It works with Blocks, whereas spreadsheet works with “rows” and “columns”.

A block in a blockchain is a collection of data. The data is added to the block in blockchain, by connecting it with other blocks in chronological others creating a chain of blocks linked together. The first block in the Blockchain is called Genesis Block. Blockchain is a distributed ledger, which simply means that a ledger is spread across the network among all peers in the network, and each peer holds a copy of the complete ledger.

Some key objectives of Blockchain are which proves that blockchain is better than traditional systems of ledger information keeping:

1. **Peer-To-Peer:** No central authority to control or manipulate it. All participant talks to each otherdirectly.This allows for data exchange to be made directly with third-parties involvement.

2. **Distributed:** The ledger is spread across the whole network which makes tampering not so easy.

3. **Cryptographically Secured:** Cryptography is used for the security services to make the ledger tamper-proof.

4. **Add-Only:** Data can only be added in the blockchain with time-sequential order. This property implies that once data is added to the blockchain, it is almost impossible to change that data and can be considered practically immutable. We can say it has: “The right to be forgotten or right to erasure” defined.

5. **Consensus:** This is the most critical attribute of all. This gives blockchain the ability to update the ledger via consensus. This is what gives it the power of decentralization. No central authority is in control of updating the ledger. Instead, any update made to the blockchain is validated against strict criteria defined by the blockchain protocol and added to the blockchain only after a consensus has been reached among all participating peers/nodes on the network.

1.3Problem Description

The definition of supply chain is given as “the network of organizations that are involved, through upstream and downstream linkages, in the different processes and activities that produce value in the form of products and services in the hands of the ultimate customer”. Products cross at least one border in global supply chains. The global supply chains are generally very large scale formations that may consist of complex patterns of production processes, transactions and knowledge.

In a supply chain, ownership of products changes several times among participants until they are delivered to consumers. For low-added-value products such as agricultural commodities and certain types of mining commodities supply chains function as an aggregation method by which goods are provided by many small-scale producers to larger scale supply chain partners for further processing towards an end-product.

Existing supply chain models begin when two supply chain members, namely manufacturers and importers, send their products to the next stage of the supply chain. The next stage, also called as middle layer, includes the wholesaler, which processes the basic products received by the export, processor and supply chain. In the last step, there is a retailer and food service that sells products. The main problem with this model is that the data is encapsulated in elements of the supply chain and shared less. For example, it is not possible for the consumer to verify the source of the food to be purchased.

Moreover, sustaining operations across a complex chain of resources, activities and organizations can be hard for supply chain partners, especially when large number of smallholders are involved. Lack of visibility and incentives may cause difficulties for sustainability. Consumers cannot be sure about the reliability of data in current supply chain systems.

The existing model becomes even more burdensome in the global supply chain. A reliable system is difficult, even impossible, on a global scale without building trust. With the advent of blockchain as a disruptive technology for most processes related to our daily lives, the transition to the use of blockchain technology has begun to overcome all these challenges of supply chains. Participants and their roles in a typical blockchain integrated supply chain flow can be depicted as in Figure 1.

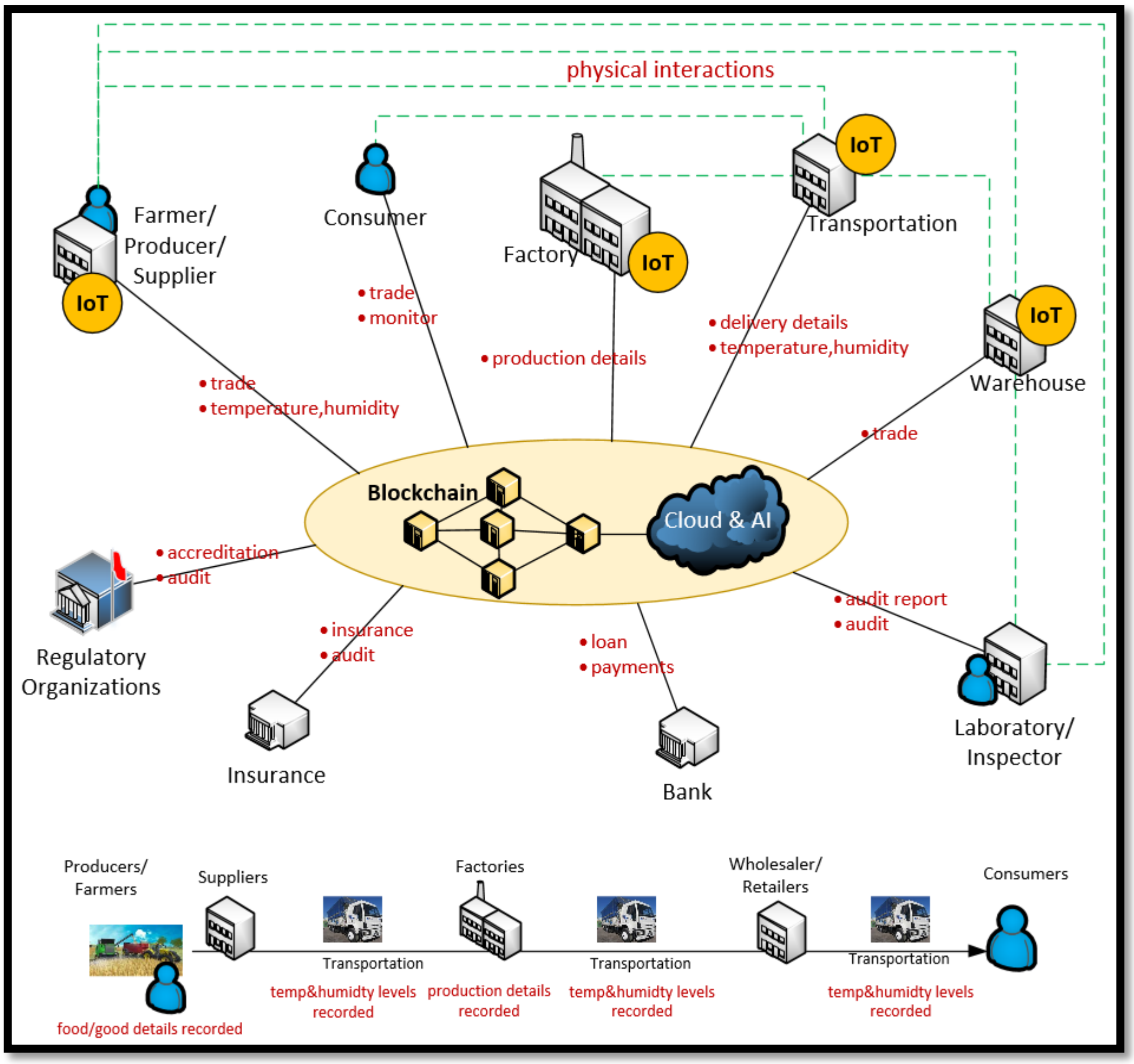


Fig. 1. Participants and their roles in a typical blockchain integrated supply chain flow.

1.4 Objective

The main objectives of the supply chain are listed as cost, quality, speed, dependability, risk reduction, sustainability, and flexibility. Manufacturing has been globalized, leads well defined supply chain management more crucial and valuable. In today's supply chain systems, it is difficult for customers to know exactly the value of a product due to lack of transparency. In addition, investigating supply chains mostly is not feasible in case of suspicion of illegal or unethical activities. Heavy paperwork, process costs, and slow processes are other main challenges of the supply chain.

In order to give an idea how blockchain might impact the needs of supply chain actors, we quoted Table. It presents, summary on how blockchain responds to the limitations that the actors of supply chains encounter today.

Employing blockchain in supply chain processes provides transparent, decentralized, secure, faster and low cost transactions. By eliminating unnecessary third parties and covering more daily life processes in digital systems minimizes paperwork.

Blockchain establishes trust among trading partners. Making more detailed data available in blockchain, improves supply chain monitoring ability and safety. This reduces insurance risks. Smart contracts and automated payments are game changer.

They add efficiency and remove bureaucracy especially in insurance, and traceability. They also allow escrowed payment by keeping money until terms of the deal are met and agreed, and then releasing automatically.Blockchain technology, in fact, provides missing infrastructure the cutting edge technologies need.

2. LITERATURE REVIEW

2.1Background

Original idea and building blocks of blockchain technology are coming from crypto money and date back to 1980s. Most recently, in 2008 December the article by an author nicknamed Satoshi Nakamoto titled as ‘’Bitcoin: A Peer-to-peer Electronic Cash System’’ popularized the blockchain technology. The blockchain concept consists of a combination of mathematics, crypto, computer and monetary science.

Blockchain technology, in fact, is a type of parallel and distributed computing architecture. It allows to eliminate central servers or trusted authority in digital interactions of partners. Thus it is classified as a disruptive technology which has potential to transform radically most of the processes in our daily life.Many sectors, like finance, medicine, manufacturing, and education, use blockchain applications to profit from the unique bundle of characteristics of this technology. Blockchain technology (BT) promises benefits in trustability, collaboration, organization, identification, credibility, and transparency.

The definition of supply chain is given as "the network of organizations that are involved, through upstream and downstream linkages, in the different processes and activities that produce value in the form of products and services in the hands of the ultimate customer".

In Supply Chain Everything is a Liquidity Problem. For Ex. Buyers-Sellers. Supply chains have become increasingly complex, making it difficult to ensure transparency throughout the whole supply chain. As a result, the increase of supply chain transparency is identified as the main objective of recent blockchain projects in supply chain management. The supply chain becomes a real-time global market.

Some of the benefits of blockchain in food supply chain management.

1. **Product traceability:** The immutable ledger of Blockchain keeps track of the fruits at every stop point of the fruit supply chain.

2. **Demand-supply balance:** Demand-supply balance Fruit supply chain in Blockchain coupled with artificial intelligence (AI) provides authentic insights on demand and supply of fruits. With timely insights, the participants of the fruit industry can maintain stocks as per the market requirements.

3. **Trust Booster:** The stakeholders of the fruit supply chain including the growers, suppliers, distributors, retailers and customers are not familiar with each other. The situation creates trust issues between the players of the fruit supply network. With Blockchain development in Fruit industry, the participants have no such problems as the superfluous technology does not allow fraudulence.

4. **Market Efficiency:** As said earlier, Blockchain implementation in the fruit supply chain keeps the quality of the fruits as such until it reaches the end customer. Henceforth, the overall efficiency of the fruit industry increases naturally.

5. **Peaceful Ecosystem:** With the elimination of fraudulence activities and increase in market efficiency, the fruit industry gradually moves on to a higher stratum. Fruit Blockchain Development paves way for the creation of a peaceful ecosystem where the stakeholders of the fruit supply network co-exist in perfect harmony.

As a summary, Blockchain technology can benefit in many ways in the supply chain as it does in many other application areas.

2.2Problems of Supply Chain and Opportunities with Blockchain

The main objectives of the supply chain are listed as cost, quality, speed, dependability, risk reduction, sustainability, and flexibility. Manufacturing has been globalized; leads well defined supply chain management more crucial and valuable. In today’s supply chain systems, it is difficult for customers to know exactly the value of a product due to lack of transparency. In addition, investigating supply chains mostly is not feasible in case of suspicion of illegal or unethical activities. Heavy paperwork, process costs, and slow processes are other main challenges of the supply chain.

A literature survey on the research focuses on blockchain for supply chain domain shows that the supply chain domain already benefits from blockchain technology because of its four main features (Figure 2).

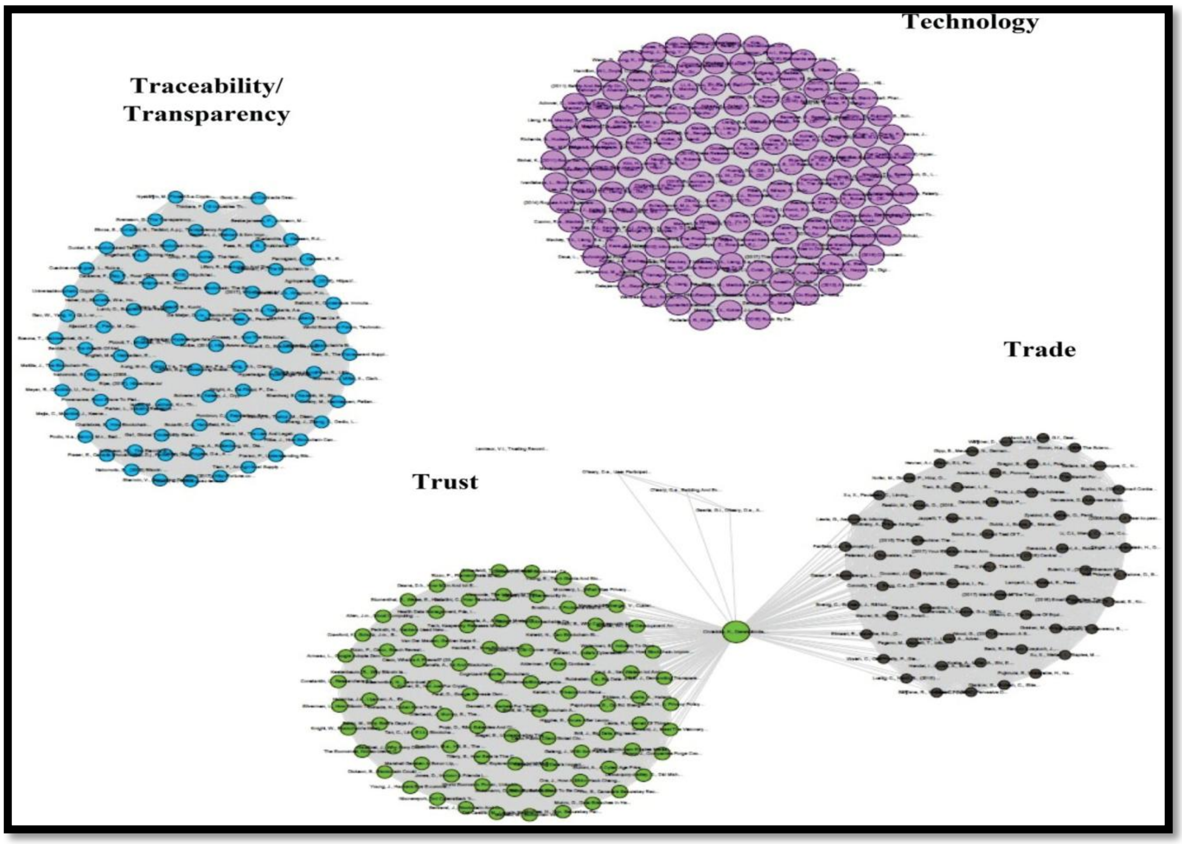
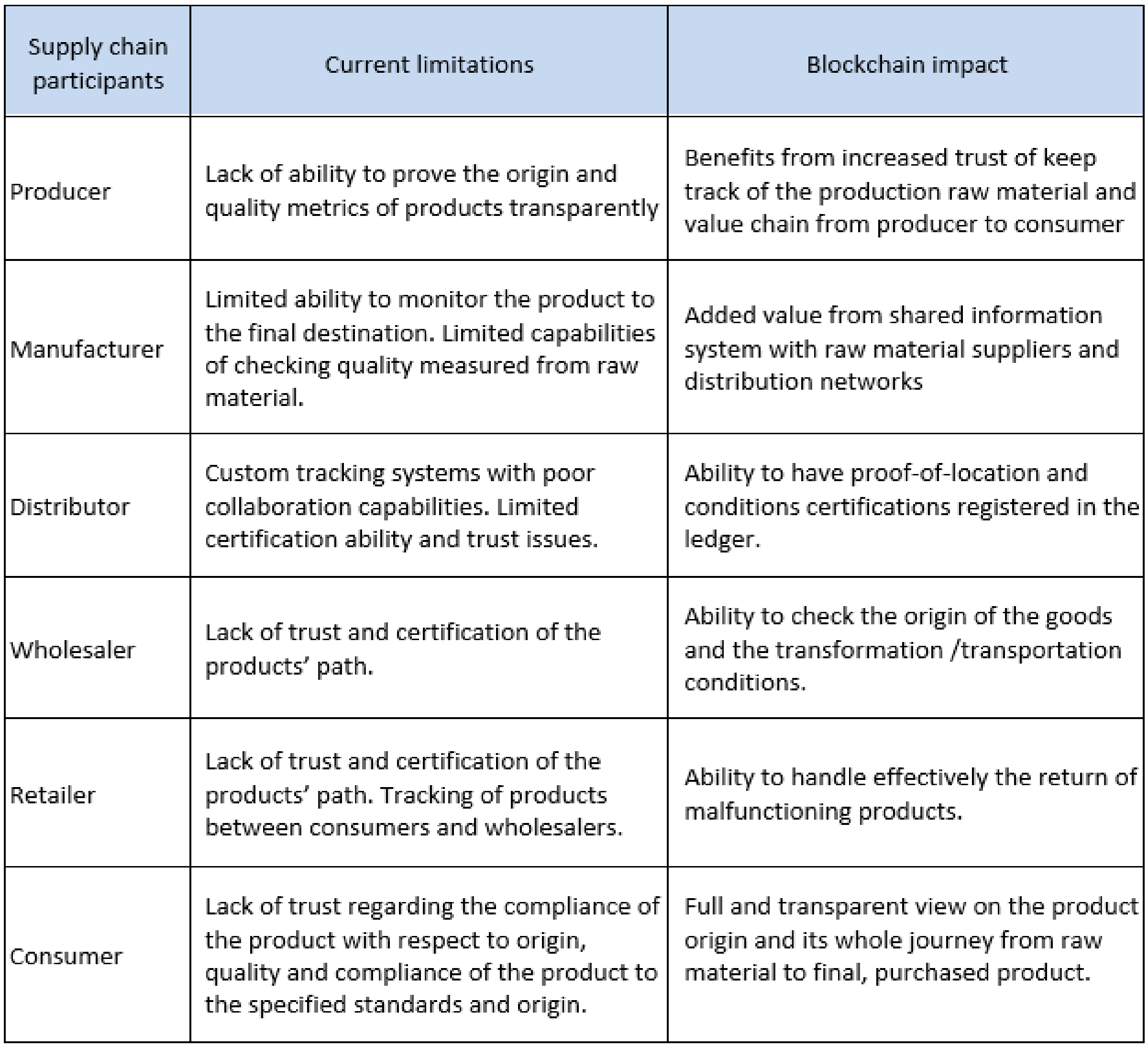


Fig. 2. Research focuses in literature for deployment of blockchain for supply chain

In order to give an idea how blockchain might impact the needs of supply chain actors, we quoted Table.

Table 1. How blockchain can improve the existing limitations of supply chains (Litke, 2019)



Supply chain participants	Current limitations	Blockchain impact
Producer	Lack of ability to prove the origin and quality metrics of products transparently	Benefits from increased trust of keep track of the production raw material and value chain from producer to consumer
Manufacturer	Limited ability to monitor the product to the final destination. Limited capabilities of checking quality measured from raw material.	Added value from shared information system with raw material suppliers and distribution networks
Distributor	Custom tracking systems with poor collaboration capabilities. Limited certification ability and trust issues.	Ability to have proof-of-location and conditions certifications registered in the ledger.
Wholesaler	Lack of trust and certification of the products' path.	Ability to check the origin of the goods and the transformation /transportation conditions.
Retailer	Lack of trust and certification of the products' path. Tracking of products between consumers and wholesalers.	Ability to handle effectively the return of malfunctioning products.
Consumer	Lack of trust regarding the compliance of the product with respect to origin, quality and compliance of the product to the specified standards and origin.	Full and transparent view on the product origin and its whole journey from raw material to final, purchased product.

As the recent situation by coronavirus pandemic showed the importance of supply chain infrastructures to communicate with multiple ecosystems. As the existing supply chain network has been highly affected by Covid-19, thus interoperability and compatibility seem to be crucial for the global supply chain after this pandemic. For the purpose blockchain technology provides disconnected supply chain systems with low cost and maximum efficiency to interoperate securely.

Making more detailed data available in blockchain, improves supply chain monitoring ability and safety. This reduces insurance risks.As a summary, Blockchain technology can benefit in many ways in the supply chain as it does in many other application areas.

2.3 Proposed Solutions

The modern supply chain is facing incredible strain from increased demands from the digital ecosystem. Trade, shipping, and supply chains are making the news a lot lately. The modern supply chain is facing incredible strain from increased demands from the digital ecosystem.

The blockchain is not an entirely novel technology. Introduced alongside Bitcoin in 2009, the blockchain was initially intended to facilitate digital currency transactions using a decentralized ledger that boasted security and usability features that made a digital currency practically possible.

Of course, as Bitcoin and other digital currencies rose to prominence and ballooned in value, the blockchain gained ancillary notoriety as well. Since then, the blockchain has received near unanimous support and investment from industry leaders, entrepreneurs, and government officials.

Indeed, while the blockchain is frequently touted as the third iteration of the internet with disruptive potential for virtually every industry, perhaps none is more prepared to benefit from the blockchain than supply chains and the shipping sector.

In general, the blockchain will impact supply chain management in three profound ways:

1. Collaborative Technologies

The rise of the blockchain coincides with another technological revolution: the introduction of the Internet of Things (IoT). These technologies are making an imprint on everything from self-activating vacuum cleaners to autonomous vehicles, and they have an obvious and important role to play in supply chain management.

For example, cheap and accessible IoT devices can be secured to packages or shipping containers to provide real-time, actionable data to supply companies. This high-quality, reliable data can equip companies to more effectively manage and assess all aspects of the supply chain while also facilitating data-driven best practices for continual innovation and improvement.

2. Clear and secure records

The blockchain’s ledger was created to be an unchangeable, public record of transactions, and that feature applies to supply chain ecosystems as well. With multiple stakeholders all working towards a singular goal, the real-time data provided by IoT devices is accessible in the shared ledger.

Moreover, these public records provide a measure of accountability for an industry frequently cloaked in confusion and opacity. This transparency can improve the credibility of the supply chain while simultaneously improving its functionality and efficiency.

In short, when everyone is privy to all the necessary information, everyone wins, and the end product improves.

3. Convenient smart contracts

Supply chains are complex, always in-flux, and extremely reliant on speed, efficiency, and trust. The blockchain’s built-in smart contracts are a clear next step for enhancing companies’ ability to securely and automatically transfer payments and data to disparate members of the supply chain.

By reducing lag in the system, supply chain networks can operate more efficiently and expediently.In an exhaustive report on the blockchain’s role in supply chain management, Deloitte concludes, “Real-time tracking via smart contracts gives supply chain stakeholders the flexibility to make rapid decisions and update inventory levels on a continuous basis, thereby reducing working capital inactivity.”

In addition, companies can use smart contracts to remit immediate payments that correspond to predetermined conditions. In this way, couriers and other supply chain participants can automatically be compensated when their products reach a destination or pass a checkpoint.

3. METHODOLOGY

3.1 Basic Concept

As we know, a block in a blockchain is a collection of data. The data is added to the block in blockchain, by connecting it with other blocks in chronological orders creating a chain of blocks linked together. The first block in the Blockchain is called Genesis Block. Blockchain is a distributed ledger, which simply means that a ledger is spread across the network among all peers in the network, and each peer holds a copy of the complete ledger.

We have seen some basic key attributes of Blockchain which proves that blockchain is better than traditional systems of ledger information.

3.2 How does it work?

1. A node starts a transaction by first creating and then digitally signing it with its private key (created via cryptography). A transaction can represent various actions in a blockchain. Most commonly this is a data structure that represents transfer of value between users on the blockchain network. Transaction data structure usually consists of some logic of transfer of value, relevant rules, source and destination addresses, and other validation information.

2. A transaction is propagated (flooded) by using a flooding protocol, called Gossip protocol, to peers that validate the transaction based on preset criteria. Usually, more than one node is required to verify the transaction.

3. Once the transaction is validated, it is included in a block, which is then propagated onto the network. At this point, the transaction is considered confirmed.

4. The newly-created block now becomes part of the ledger, and the next block links itself cryptographically back to this block. This link is a hash pointer. At this stage, the transaction gets its second confirmation and the block gets its first confirmation.

5. Transactions are then reconfirmed every time a new block is created. Usually, six confirmations in a network are required to consider the transaction final.

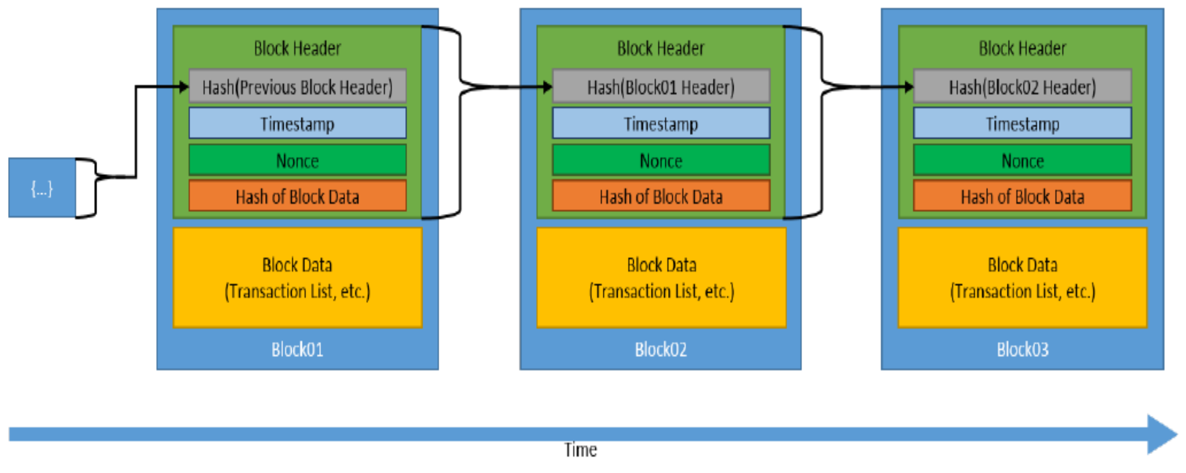
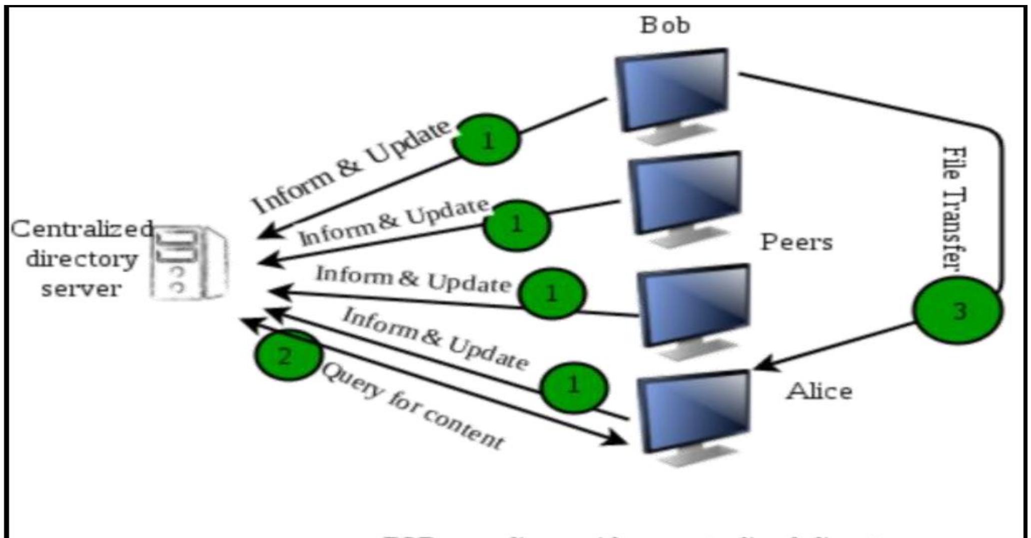


Figure 3: Generic Chain of Blocks

P2P (PEER TO PEER) File Sharing

In Computer Networking, P2P is a file sharing technology, allowing the users to access mainly multimedia files like videos, music, e-books, games etc. The individual users in this network are referred to as peers. The peers request for the files from other peers by establishing TCP or UDP connections.

How P2P Works?



P2P paradigm with a centralised directory

A peer-to-peer network allows computer hardware and software to communicate without the need for a server. Unlike client-server architecture, there is no central server for processing requests in a P2P architecture. The peers directly interact with one another without the requirement of a central server. Now, when one peer makes a request, it is possible thatmultiple peers have the copy of that requested object. Now the problem is how to get the

IP addresses of all those peers. This is decided by the underlying architecture supported by the P2P systems. By means of one of these methods, the client peer can get to know about all the peers which have the requested object/file and the file transfer takes place directly between these two peers.

Three Such Architectures Exist

1) Centralized Directory

2) Query Flooding

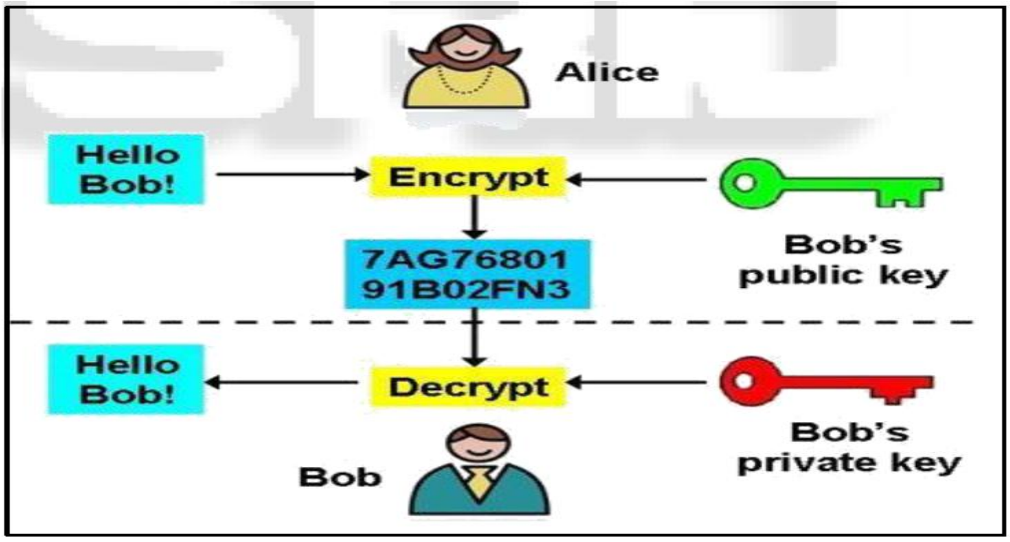
3) Exploiting Heterogeneity

3.3 Cryptography

Cryptography is the practice of developing protocols that prevent third parties from viewing private data. Modern cryptography combines the disciplines of math, computer science, physics, engineering, and more.

Some important terms are defined below:

1. **Encryption:** Encoding text into an unreadable format.
2. **Decryption:** Reserving encryption – converting a jumbled message into its original form.
3. **Cipher:** An algorithm for performing encryption or decryption, usually a well-defined set of steps that can be followed.



3.4 Public-key cryptography

It is a cryptographic system that uses a pair of keys – a public key and a private key.The public key may be widely distributed, but the private key is meant to be known only by its owner. Keys are always created in a pair every public key must have a corresponding private key.
Let’s say Alice wants to send message to Bob:
− Alice uses Bob’s public key to encrypt the message.
− Alice sends the encrypted message to Bob – if a third party intercepted it, all they would see is random numbers and letters.
− Bob uses his private key to decrypt and read the message.

The algorithm which we are going to use in the proposed model is the RSA algorithm. It is really helpful for the low scale system cryptography and is very efficient in terms of the memory and the processing resources. SHA algorithm was also an option but it comes with so many demerits and disadvantages which can affect the efficiency of the running system. Since we are dealing with the real time production industry even the minutes of delay can cause a great loss to the company so to be on a safer side we opted to use the RSA algorithm. In return we observed in the proposed system.

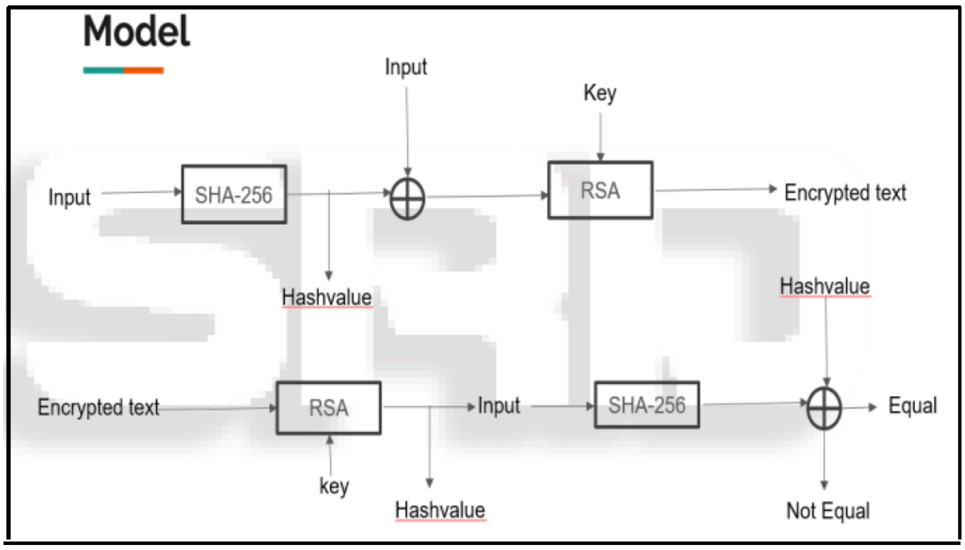
RSA algorithm (Rivest-Shamir-Adleman)

Public key cryptography, also known as asymmetric cryptography, uses two different but mathematically linked keys -- one public and one private. The public key can be shared with everyone, whereas the private key must be kept secret. In RSA cryptography, both the public and the private keys can encrypt a message; the opposite key from the one used to encrypt a message is used to decrypt it. This attribute is one reason why RSA has become the most widely used asymmetric algorithm. Encryption strength is directly tied to key size, and doubling key length can deliver an exponential increase in strength, although it does impair performance. RSA keys are typically 1024- or 2048-bits long, but experts believe that 1024-bit keys are no longer fully secure against all attacks. This is why the government and some industries are moving to a minimum key length of 2048-bits.

Problem of Traceability

The main issue of supply chain is lack of Traceability, or the ability to track the food product through all stages of the supply chain. Many consumers now want to know from where all products and their ingredients come from. Traceability problem is also faced within an organization. If an organization wants to get the complete processing detail of a product which was developed long ago then they have to invest a huge amount of money to get the detail. This causes loss to the organization. And even then also, it is not guaranteed that we could pinpoint the exact details of a product which was processed some time ago. The main reason behind this problem is the conventional data storage mechanisms in the production industry which uses human dependent registers and Relational Databases, sometimes it is observed that many industries are using flat files to maintain the product processing data. The data in the flat files becomes vulnerable and can be easily corrupted with human intervention.

Solution



The problem of traceability in the supply chain can be solved using Blockchain. We propose to develop an application through which the user can get all the details of the product, that is the whole cycle of the product from farm to his/her hand. The application can also be used by the organization in its operations like getting the history of a product which was manufactured long ago in very less time.

When we are using blockchain we can fully ensure that the data stored within our model is tamper proof, which means the data stored in such a manner that it cannot be altered with the efforts of someone who is not authorized to alter the data manually. The hashing techniques are used which are one way hash functions and are efficient from shielding the sensitive data from any malicious access. The document store model of NoSql is preferred over flat files which gives us ease of access in handling the data and making the transactions in order to log the data in the form of blocks in our blockchain network. This approach is also used to append new blocks.

3.5 Mathematical Model

1. Dependent Variables
   - Hash value generated
   - Encrypted text generated
   - Final status
2. Independent Variables
   - Block id
   - Block creation time
   - Block creation date

3. Assumptions
   - New chain of block will be created in every 10 min

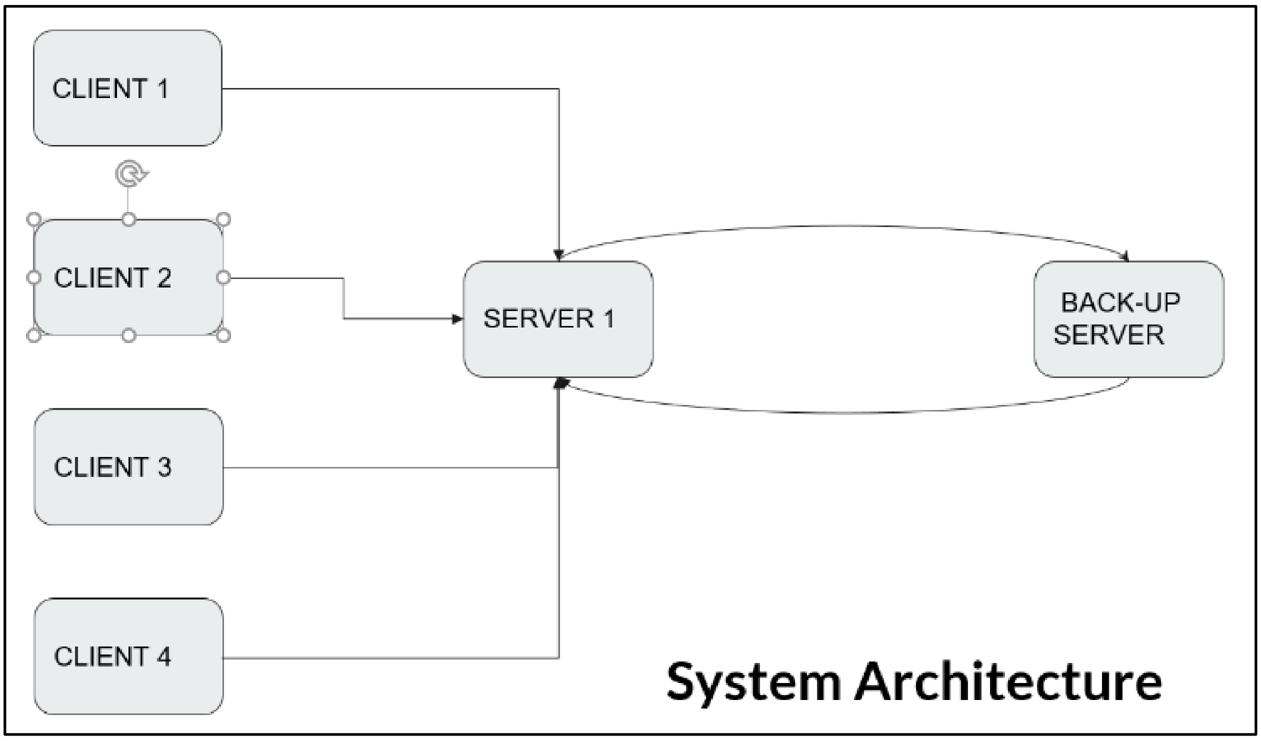
3.6 SHA-256

SHA-256 is a member of the SHA-2 cryptographic hash functions designed by the NSA. SHA stands for Secure Hash Algorithm. Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed "hash" a person can determine the data's integrity. A one-way hash can be generated from any piece of data, but the data cannot be generated from the hash.A sha256 is 256 bits long -as its name indicates. The larger the key the stronger hash value will be generated. The key can be any length. However, the recommended size is 64 bytes. If the key is more than 64 bytes long, it is hashed to derive a 64-byte key. If it is less than 64 bytes long, it is padded to 64 bytes.

By combining these two algorithms with our proposed model we developed a Web based application which is used to maintain the transactional data of different industrial processes running in an production industry and also the data related to the raw material and till the final product is processed is appended in the block chain and it can be accessed with proper authority just by few clicks which makes it highly traceable.

3.7 SHIP CHAIN: The Web Application

Our combined efforts on the research which we have presented above yielded an efficient and optimized web application to manage the transactional data of a product. The whole journey of the product from it being a raw material to a fully processed product and the data related to it is appended in the datastore which is based on the block chain.



To achieve this functionality we need to set up the client stations at every processing unit and we will append the data related to the current batch of the product to the server which will be centrally hosted in the organization. Server will have all the authority to publish the blocks in the blockchain. The client nodes will request for the ID of the current lot and when they get associated with the ID then they send all the product related data to the server, upon verifying the cryptography keys the server publishes the data received from the client nodes to the blockchain. Each block published to the ledger will have its unique id and data is stored in an encrypted format such that it cannot be tempered once it is published.

3.8 Tools

We have used following tools to create this application:

1. React.js
2. Node.js
3. MetaMask
4. Truffle

React.js

React.js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It's used for handling the view layer for web and mobile apps. React also allows us to create reusable UI components. React was first created by Jordan Walke, a software engineer working for Facebook. React first deployed on Facebook's newsfeed in 2011 and on Instagram.com in 2012.

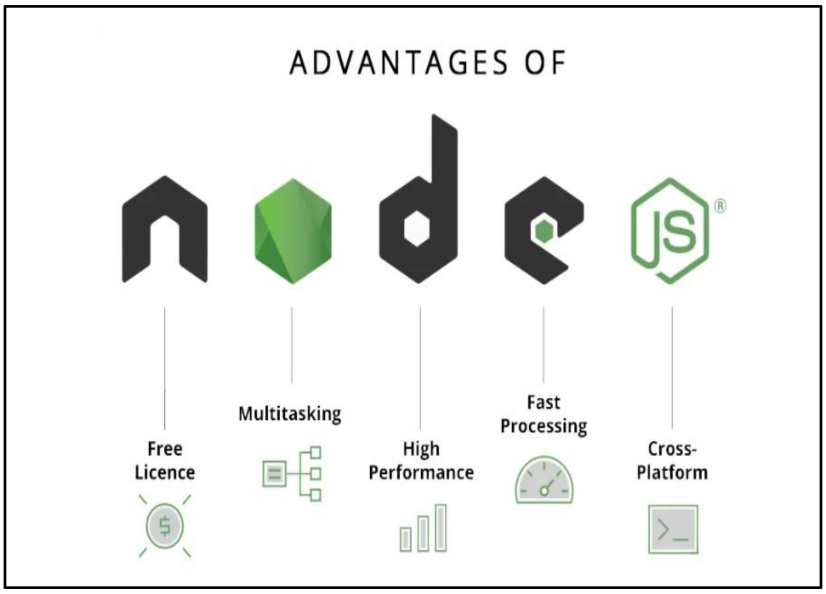
React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. It works only on user interfaces in the application. This corresponds to the view in the MVC template. It can be used with a combination of other JavaScript libraries or frameworks, such as Angular JS in MVC.

React.js properties includes the following

- React.js is declarative
- React.js is simple
- React.js is component based
- React.js supports server side
- React.js is extensive
- React.js is fast
- React.js is easy to learn

Node.js

Node.js is a runtime environment that allows software developers to launch both the frontend and backend of web apps using JavaScript. Although JS underpins all the processes for app assembly, as a backend development environment, Node.js, differs from the frontend environment. It has unique APIs that support HTTP requests, file systems and other server-side features for which frontend APIs provide limited support.



MetaMask

An Ethereum Wallet in your Browser

MetaMask is an extension for accessing Ethereum enabled distributed applications, or "Dapps" in your browser.

The extension injects the Ethereum web3 API into every website's javascript context, so that dapps can read from the blockchain.

MetaMask also lets the user create and manage their own identities (via private keys, local client wallet and hardware wallets like Trezor™), so when a Dapp wants to perform a transaction and write to the blockchain, the user gets a secure interface to review the transaction, before approving or rejecting it.

Because it adds functionality to the normal browser context, MetaMask requires the permission to read and write to any webpage. You can always "view the source" of MetaMask the way you do any Chrome extension, or view the source code on Github:

https://github.com/MetaMask/metamask-plugin

It enables access to Web 3.0, Dapps, NFTs, erc20, tokens, ICOs, erc271...and more.

Truffle

Truffle is a world-class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier.

Truffle is widely considered the most popular tool for blockchain application development with over 1.5 million lifetime downloads. Truffle supports developers across the full lifecycle of their projects, whether they are looking to build on Ethereum, Hyperledger, Quorum, or one of an ever-growing list of other supported platforms. Paired with Ganache, a personal blockchain, and Drizzle, a front-end dApp development kit, the full Truffle suite of tools promises to be an end-to-end dApp development platform.

- Built-in smart contract compilation, linking, deployment and binary management.
- Automated contract testing for rapid development.
- Scriptable, extensible deployment & migrations framework.
- Network management for deploying to any number of public & private networks.
- Package management with EthPM& NPM, using the ERC190 standard.
- Interactive console for direct contract communication.
- Configurable build pipeline with support for tight integration.
- External script runner that executes scripts within a Truffle environment.

3.9 Activity Diagram

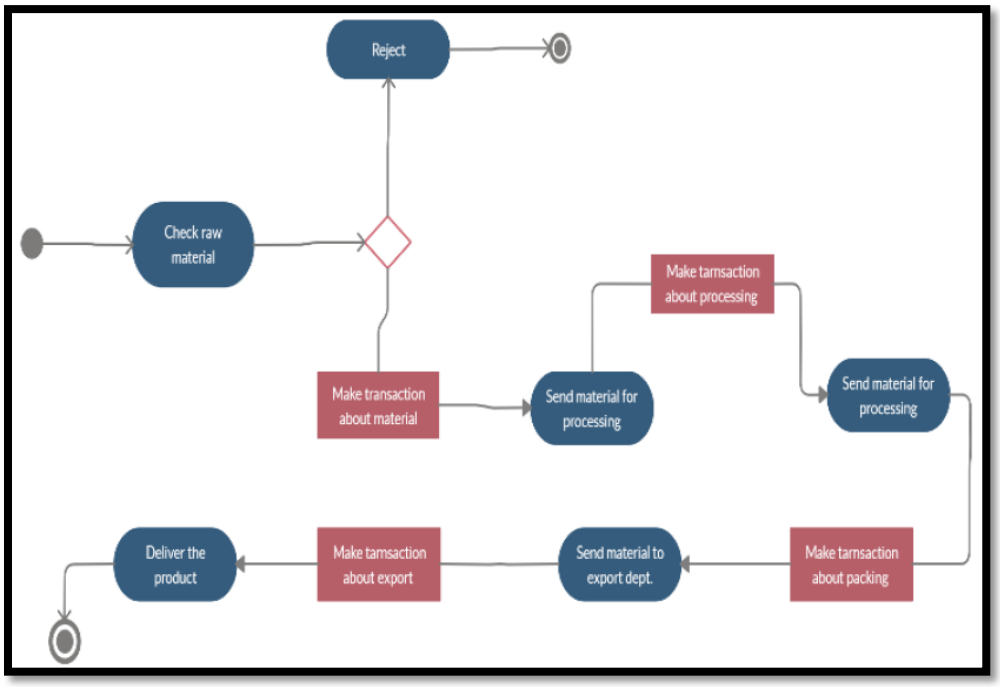

Reject
Check raw material
Make transaction about material
Send material for processing
Make tamsaction about processing
Send material for processing
Make tamsaction about packing
Send material to export dept.
Make tamsaction about export
Deliver the product

Fig 3: Activity Diagram of Blockchain in Supply Chain

4. CODING AND IMPLEMENTATIONS

4.1 Coding

Solidity code for migration.sol

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.4.22 <0.9.0;

contract Migrations {

  address public owner = msg.sender;

uint public last_completed_migration;

  modifier restricted() {

require(

msg.sender == owner,

      "This function is restricted to the contract's owner"

    );

    _;

  }

  function setCompleted(uint completed) public restricted {

last_completed_migration = completed;

  }

}
```

Deploy migrations

```
const Migrations = artifacts.require("Migrations");

module.exports = function (deployer) {
  deployer.deploy(Migrations);
};
```

- **Components**

First Stage (Raw Material Details)

```
import React, { useState } from 'react';
import { Form, Input, Button} from 'antd';
import moment from "moment";
import otpGenerator from 'otp-generator'
import { ToastContainer, toast } from 'react-toastify';

export default function FirstStage({accounts, contract}){

    const toastId = React.useRef(null);
    const [productName, setProductName] = useState(null);
    const [farmName, setFarmName] = useState(null);
    const [farmAddress, setFarmAddress] = useState(null);
    const [materialName, setMaterialName] = useState(null);
    const [quantity, setQuantity] = useState(0);
    const [price, setPrice] = useState(0);

    const infoToast = (msg) =>toastId.current = toast.info(msg);
    const successToast = (msg) =>toast.success(msg);
    const errorToast = (msg) =>toast.error(msg);
    const dismiss = () =>  toast.dismiss(toastId.current);

    const onFinish = async (values) => {
        infoToast("Processing Transaction!!")

        setTimeout(async() => {
            const date = moment().format("DD-MM-YYYY")
            console.log("date: ", date)
            const productId = otpGenerator.generate(6, { upperCase: false, specialChars: false });
            console.log("Product:", productId);
            console.log(contract.methods);
            const tx = await contract.methods.storeFirstStage(
                farmName,
                farmAddress,
                materialName,
                quantity,
                price,
                date,
                productId,
                productName
            ).send({ from: accounts[0] });
            console.log("transaction:", tx)

            if(tx.status){
                localStorage.setItem('productId', productId);
                dismiss();
                successToast(`Transaction successful. Product Id: ${productId}`);
            }else{
                errorToast("Transaction failed");
            }
        }, 1500);

    };

    const onFinishFailed = (errorInfo) => {
        console.log('Failed:', errorInfo);
    };

    return (
        <Form
            initialValues={{
            remember: true,
            }}
            layout="vertical"
            onFinish={onFinish}
            onFinishFailed={onFinishFailed}
            autoComplete="off"
            style={{marginLeft: '100px', marginRight: '100px', marginTop: '50px'}}
        >
            <Form.Item label="Product Name" name="productname" rules={[{required:
true,message: 'Please input product name'}]}>
                <Input type="text" placeholder="Enter product name"
onChange={(e)=>setProductName(e.target.value)}/>
            </Form.Item>

            <Form.Item label="Farm Name" name="farmname" rules={[{required: true,message:
'Please input farm name!'}]}>
                <Input type="text" placeholder="Enter farm name"
onChange={(e)=>setFarmName(e.target.value)} />
            </Form.Item>

            <Form.Item label="Farm Address" name="farmaddress" rules={[{required:
true,message: 'Please input farm address!'}]}>
                <Input  type="text" placeholder="Enter farm address"
 onChange={(e)=>setFarmAddress(e.target.value)}/>
            </Form.Item>

            <Form.Item label="Material Name" name="materialname" rules={[{required:
true,message: 'Please input material name!'}]}>
                <Input type="text" placeholder="Enter material name"
onChange={(e)=>setMaterialName(e.target.value)}/>
            </Form.Item>

            <Form.Item label="Quantity " name="quantity" rules={[{required: true,message:
'Please input quantity!'}]}>
                <Input type="number" placeholder="Enter quantity"
onChange={(e)=>setQuantity(e.target.value)}/>
```

```
          </Form.Item>

          <Form.Item label="Price" name="price" rules={[{required: true,message: 'Please
input price!'}]}>
              <Input type="number" placeholder="Enter price"
onChange={(e)=>setPrice(e.target.value)}/>
          </Form.Item>

          <Form.Item>
              <Button type="primary" htmlType="submit">
                  Submit
              </Button>
              <ToastContainer/>
          </Form.Item>
      </Form>
   );
}
```

Second Stage(Manufacturing Details)

```
import React, { useState } from 'react';
import { Form, Input, Button} from 'antd';
import moment from "moment";
import { ToastContainer, toast } from 'react-toastify';

export default function SecondStage({accounts, contract}){

   const toastId = React.useRef(null);
   const [machinNo, setMachineNo] = useState(null);
   const [temperature, setTemp] = useState(null);
   const [quantity, setQuantity] = useState(0);

   const infoToast = (msg) =>toastId.current = toast.info(msg);
   const successToast = (msg) =>toast.success(msg);
   const errorToast = (msg) =>toast.error(msg);
   const dismiss = () =>  toast.dismiss(toastId.current);

   const onFinish = async (values) => {
      infoToast("Processing Transaction!!")

      setTimeout(async() => {
         const date = moment().format("DD-MM-YYYY")
         console.log("date: ", date)
         const productId = localStorage.getItem('productId');

         const tx = await contract.methods.storeSecondStage(
            machinNo,
            temperature,
            quantity,
            date,
            productId
        ).send({ from: accounts[0] });

        console.log("transaction:", tx)

        if(tx.status){
            localStorage.setItem('productId', productId);
            dismiss();
            successToast(`Transaction successful. Product Id: ${productId}`);
        }else{
            errorToast("Transaction failed");
        }

    }, 1500);

};

const onFinishFailed = (errorInfo) => {
    console.log('Failed:', errorInfo);
};

return (
    <Form
        initialValues={{
        remember: true,
        }}
        layout="vertical"
        onFinish={onFinish}
        onFinishFailed={onFinishFailed}
        autoComplete="off"
        style={{marginLeft: '100px', marginRight: '100px', marginTop: '50px'}}
    >
        <Form.Item label="Machine Number" name="machineno" rules={[{required: 
true,message: 'Please input machine number'}]}>
            <Input type="number" placeholder="Enter machine number" 
onChange={(e)=>setMachineNo(e.target.value)}/>
        </Form.Item>

        <Form.Item label="Temperature" name="temperature" rules={[{required: 
true,message: 'Please input temperature!'}]}>
            <Input type="number" placeholder="Enter temperature" 
onChange={(e)=>setTemp(e.target.value)} />
        </Form.Item>

        <Form.Item label="Quantity " name="quantity" rules={[{required: true,message: 
'Please input quantity!'}]}>
            <Input type="number" placeholder="Enter quantity" 
onChange={(e)=>setQuantity(e.target.value)}/>
```

```
            </Form.Item>

            <Form.Item>
                <Button type="primary" htmlType="submit">
                    Submit
                </Button>
                <ToastContainer/>
            </Form.Item>
        </Form>
    );
}
```

Third Stage (Packing Details)

```

import React, { useState } from 'react';
import { Form, Input, Button} from 'antd';
import moment from "moment";
import { ToastContainer, toast } from 'react-toastify';

export default function ThirdStage({accounts, contract}){

    const toastId = React.useRef(null);
    const [totalPackets, setTotalPackets] = useState(null);
    const [boxNumber, setBoxNumber] = useState(null);

    const infoToast = (msg) =>toastId.current = toast.info(msg);
    const successToast = (msg) =>toast.success(msg);
    const errorToast = (msg) =>toast.error(msg);
    const dismiss = () =>  toast.dismiss(toastId.current);

    const onFinish = async () => {
        infoToast("Processing Transaction!!")

        setTimeout(async() => {
            const date = moment().format("DD-MM-YYYY")
            console.log("date: ", date)
            const productId = localStorage.getItem('productId');

            const tx = await contract.methods.storeThirdStage(
                totalPackets,
                boxNumber,
                date,
                productId
            ).send({ from: accounts[0] });
            console.log("transaction:", tx)

            if(tx.status){
                localStorage.setItem('productId', productId);
        dismiss();
        successToast(`Transaction successful. Product Id: ${productId}`);
      }else{
        errorToast("Transaction failed!! Check console");
        errorToast("Transaction failed");
      }
    }, 1500);

  };

  const onFinishFailed = (errorInfo) => {
    console.log('Failed:', errorInfo);
  };

  return (
    <Form
      initialValues={{
      remember: true,
      }}
      layout="vertical"
      onFinish={onFinish}
      onFinishFailed={onFinishFailed}
      autoComplete="off"
      style={{marginLeft: '100px', marginRight: '100px', marginTop: '50px'}}
    >
      <Form.Item label="Total Packet" name="totalpacket" rules={[{required: true,
message: 'Please input total packet!'}]}>
        <Input type="number" placeholder="Enter total packets in a box"
onChange={(e)=>setTotalPackets(e.target.value)}/>
      </Form.Item>

      <Form.Item label="Box number" name="boxno" rules={[{required: true, message:
'Please input box number!'}]}>
        <Input type="number" placeholder="Enter box number"
onChange={(e)=>setBoxNumber(e.target.value)} />
      </Form.Item>

      <Form.Item>
        <Button type="primary" htmlType="submit">
          Submit
        </Button>
        <ToastContainer/>
      </Form.Item>
    </Form>
  );
}
```

Fourth Stage (Delivery Details)

```
import React, { useState } from 'react';
import { Form, Input, Button} from 'antd';
import moment from "moment";
import { ToastContainer, toast } from 'react-toastify';

export default function FourthStage({accounts, contract}){

    const toastId = React.useRef(null);
    const [clientName, setClientName] = useState(null);
    const [clientAddress, setClientAddress] = useState(null);
    const [boxDelivered, setBoxDelivered] = useState(0);

    const infoToast = (msg) =>toastId.current = toast.info(msg);
    const successToast = (msg) =>toast.success(msg);
    const errorToast = (msg) =>toast.error(msg);
    const dismiss = () =>  toast.dismiss(toastId.current);

    const onFinish = async (values) => {
        infoToast("Processing Transaction!!")

        setTimeout(async() => {
            const date = moment().format("DD-MM-YYYY")
            console.log("date: ", date)
            const productId = localStorage.getItem('productId');

            const tx = await contract.methods.storeFourthStage(
                clientName,
                clientAddress,
                boxDelivered,
                date,
                productId
            ).send({ from: accounts[0] });
            console.log("transaction:", tx);

            if(tx.status){
                localStorage.setItem('productId', productId);
                dismiss();
                successToast(`Transaction successful. Product Id: ${productId}`);
            }else{
                errorToast("Transaction failed");
            }
        }, 1500);

    };

    const onFinishFailed = (errorInfo) => {
        console.log('Failed:', errorInfo);
    };
    return (
        <Form
            initialValues={{
            remember: true,
            }}
            layout="vertical"
            onFinish={onFinish}
            onFinishFailed={onFinishFailed}
            autoComplete="off"
            style={{marginLeft: '100px', marginRight: '100px', marginTop: '50px'}}
        >
            <Form.Item label="Client Name" name="clientname" rules={[{required: 
true,message: 'Please input client name'}]}>
                <Input type="text" placeholder="Enter client name" 
onChange={(e)=>setClientName(e.target.value)}/>
            </Form.Item>

            <Form.Item label="Client Address" name="clientaddress" rules={[{required: 
true,message: 'Please input client address!'}]}>
                <Input type="text" placeholder="Enter client address" 
onChange={(e)=>setClientAddress(e.target.value)} />
            </Form.Item>

            <Form.Item label="Total box delivered " name="totalbox" rules={[{required: 
true,message: 'Please input total box!'}]}>
                <Input  type="number" placeholder="Enter total number of boxes"
 onChange={(e)=>setBoxDelivered(e.target.value)}/>
            </Form.Item>

            <Form.Item>
                <Button type="primary" htmlType="submit">
                    Submit
                </Button>
                <ToastContainer/>
            </Form.Item>
        </Form>
    );
}
```

Get Details (Final Details)

```
import React, { useState } from 'react';
import { Form, Input, Button, Timeline} from 'antd';
import { ToastContainer, toast } from 'react-toastify';

export default function ProductDetail({accounts, contract}){

    const toastId = React.useRef(null);
    const [productId, setProductId] = useState(null);
    const [productInfo, setProductInfo] = useState([]);

    const notify = () =>toastId.current = toast.info("Getting data from chain!!");
    const dismiss = () =>  toast.dismiss(toastId.current);

    const onFinish = async (values) => {
        notify();
        setTimeout( async () => {
            const tx = await contract.methods.getProductDetails(
                productId
            ).call({ from: accounts[0] });

            console.log("Result:", tx)
            setProductInfo(tx)
            dismiss();
        }, 3000);

    };

    const onFinishFailed = (errorInfo) => {
        console.log('Failed:', errorInfo);
    };

    return (
        <div>
            <Form
                name="basic"
                initialValues={{ remember: true }}
                layout="vertical"
                onFinish={onFinish}
                onFinishFailed={onFinishFailed}
                autoComplete="off"
                style={{marginLeft: '100px', marginRight: '100px', marginTop: '50px'}}
            >
                <Form.Item label="Product Id" name="productId" rules={[{required: true,
message: 'Please input product id',}]}>
                    <Input type="text" placeholder="Enter product id"
onChange={(e)=>setProductId(e.target.value)}/>
                </Form.Item>
                <Form.Item>
                    <Button type="primary" htmlType="submit">
                        Submit
                    </Button>
                    <ToastContainer/>
                </Form.Item>
            </Form>
            <div>
                <h2>Product Detail Timeline</h2><br/><br/>
                <Timeline mode="left">

                    <Timeline.Item label="First Stage:"><h3><u>Raw material
details</u></h3></Timeline.Item>
                        { productInfo.length>0 &&productInfo["1"]["flag"] ?
                            <div>
                                <Timeline.Item>Farm Name:
{productInfo["1"]["farmName"]}</Timeline.Item>
                                <Timeline.Item>Farm Address:
{productInfo["1"]["farmAddress"]}</Timeline.Item>
                                <Timeline.Item>Material Name:
{productInfo["1"]["materialName"]}</Timeline.Item>
                                <Timeline.Item>Quantity:
{productInfo["1"]["quantity"]}</Timeline.Item>
                                <Timeline.Item>Price: {productInfo["1"]["price"]}</Timeline.Item>
                                <Timeline.Item>Date: {productInfo["1"]["date"]}</Timeline.Item>
                            </div> : null
                        }

                    <Timeline.Item label="Second Stage"><h3><u>Manufacturing
details</u></h3></Timeline.Item>
                        { productInfo.length>0 &&productInfo["2"]["flag"] ?
                            <div>
                                <Timeline.Item>Machine Number: { productInfo["2"]["machinNo"] }
</Timeline.Item>
                                <Timeline.Item>Temperature: { productInfo["2"]["temperature"]
}</Timeline.Item>
                                <Timeline.Item>Quantity: { productInfo["2"]["quantity"]
}</Timeline.Item>
                                <Timeline.Item>Date: { productInfo["2"]["date"] }</Timeline.Item>
                            </div> : null
                        }
                    <Timeline.Item label="Third Stage"><h3><u>Packing
details</u></h3></Timeline.Item>
                        { productInfo.length>0 &&productInfo["3"]["flag"] ?
                            <div>
                                <Timeline.Item>Total Packets: { productInfo["3"]["totalPackets"]
}</Timeline.Item>
                                <Timeline.Item>Box Number: { productInfo["3"]["boxNumber"]
}</Timeline.Item>
```

```
                                <Timeline.Item>Date: { productInfo["3"]["date"] }</Timeline.Item>
                            </div> : null
                        }
                    <Timeline.Item label="Fourth Stage"><h3><u>Delivery
details</u></h3></Timeline.Item>
                        { productInfo.length>0 &&productInfo["4"]["flag"] ?
                            <div>
                                <Timeline.Item>Client Name: { productInfo["4"]["clientName"]
}</Timeline.Item>
                                <Timeline.Item>Client Address: { productInfo["4"]["clientAddress"]
}</Timeline.Item>
                                <Timeline.Item>Total Box Delivered: {
productInfo["4"]["boxDelivered"] }</Timeline.Item>
                                <Timeline.Item>Date: { productInfo["4"]["date"] }</Timeline.Item>
                            </div> : null
                        }
                </Timeline>
            </div>
        </div>
    );
}
```

App.js

```

import React, { useEffect,useState } from "react";
import TrackProduct from "./contracts/TrackProduct.json";
import getWeb3 from "./getWeb3";
import "./App.css";
import 'antd/dist/antd.css';
import 'react-toastify/dist/ReactToastify.css';
import { Tabs } from 'antd';
import FirstStage from "./components/first_stage";
import SecondStage from "./components/second_stage";
import ThirdStage from "./components/third_stage";
import FourthStage from "./components/fourth_stage";
import ProductDetail from "./components/get_details";

const { TabPane } = Tabs;

function App(){

  const [contract, setContract] = useState()
  const [accounts, setAccounts] = useState()

  useEffect(()=>{
    async function connect(){
      try {
        const web3 = await getWeb3();
        const accounts = await web3.eth.getAccounts();
        console.log(accounts);
        const networkId = await web3.eth.net.getId();
        const deployedNetwork = await TrackProduct.networks[networkId];
        const instance = new web3.eth.Contract(
          TrackProduct.abi,
          deployedNetwork&&deployedNetwork.address,
        );
        instance.address = ["0x1B32a7F1826CD12E881109BeF306558DeA0A3C0e"]
        setAccounts(accounts);
        setContract(instance);
        console.log(instance);
      } catch (error) {
        alert(
          `Failed to load web3, accounts, or contract. Check console for details.`,
        );
        console.log(error);
      }
    }
    connect();
  },[])

  return (
    <div className="App">
      <Tabs defaultActiveKey="1" style={{marginLeft: '80px', marginRight: '80px'}}>
        <TabPane tab="Stage 1" key="1">
          <FirstStage accounts={accounts} contract={contract}/>
        </TabPane>
        <TabPane tab="Stage 2" key="2">
          <SecondStage accounts={accounts} contract={contract}/>
        </TabPane>
        <TabPane tab="Stage 3" key="3">
          <ThirdStage accounts={accounts} contract={contract}/>
        </TabPane>
        <TabPane tab="Stage 4" key="4">
          <FourthStage accounts={accounts} contract={contract}/>
        </TabPane>
        <TabPane tab="Get Detail" key="5">
          <ProductDetail accounts={accounts} contract={contract}/>
        </TabPane>
      </Tabs>
    </div>
  );

}

export default App;
```

Track product.sol

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.4.21 <0.7.0;
pragma experimental ABIEncoderV2;

contract TrackProduct {

   struct FirstStage {
      string farmName;
      string farmAddress;
      string materialName;
      uint256 quantity;
      uint256 price;
      string date;
      bool flag;
   }

   struct SecondStage {
      uint256 machinNo;
      uint256 temperature;
      uint256 quantity;
      string date;
      bool flag;
   }

   struct ThirdStage {
      uint256 totalPackets;
      uint256 boxNumber;
      string date;
      bool flag;
   }

   struct FourthStage {
      string clientName;
      string clientAddress;
      uint256 boxDelivered;
      string date;
      bool flag;
   }
   struct ProductDetail{
      string productName;
FirstStage detail1;
SecondStage detail2;
ThirdStage detail3;
FourthStage detail4;
   }

mapping(string =>ProductDetail) productDetail;

   function storeFirstStage(
      string memory farmName,
      string memory farmAddress,
      string memory materialName,
      uint256 quantity,
      uint256 price,
      string memory date,
      string memory productId,
      string memory productName
   ) public {

      require(productDetail[productId].detail1.flag == false, "Details already present!!");

FirstStage memory detail = FirstStage({
farmName: farmName,
farmAddress: farmAddress,
materialName: materialName,
        quantity: quantity,
        price: price,
        date: date,
        flag: true
      });

productDetail[productId].detail1 = detail;
productDetail[productId].productName = productName;
   }

   function storeSecondStage(
      uint256 machinNo,
      uint256 temperature,
      uint256 quantity,
      string memory date,
      string memory productId
```

```
    ) public {

        require(productDetail[productId].detail1.flag, "Insert stage1 first!!");
        require(productDetail[productId].detail2.flag == false, "Details already present!!");

SecondStage memory detail = SecondStage({
machinNo: machinNo,
          temperature: temperature,
          quantity: quantity,
          date: date,
          flag: true
        });

productDetail[productId].detail2 = detail;
    }

    function storeThirdStage(
        uint256 totalPackets,
        uint256 boxNumber,
        string memory date,
        string memory productId
    ) public {

        require(productDetail[productId].detail2.flag, "Insert stage2 first!!");
        require(productDetail[productId].detail3.flag == false, "Details already present!!");

ThirdStage memory detail = ThirdStage({
totalPackets: totalPackets,
boxNumber: boxNumber,
          date: date,
          flag: true
        });

productDetail[productId].detail3 = detail;
    }

    function storeFourthStage(
        string memory clientName,
        string memory clientAddress,
        uint256 boxDelivered,
        string memory date,
        string memory productId
    ) public {
        require(productDetail[productId].detail3.flag, "Insert stage3 first!!");
        require(productDetail[productId].detail4.flag == false, "Details already present!!");

FourthStage memory detail = FourthStage({
clientName: clientName,
clientAddress: clientAddress,
boxDelivered: boxDelivered,
          date: date,
          flag: true
        });

productDetail[productId].detail4 = detail;
    }

    function getProductDetails(string memory productId) public view returns (ProductDetail
memory detail){
        return productDetail[productId];
    }
}
```

Deploy Contracts

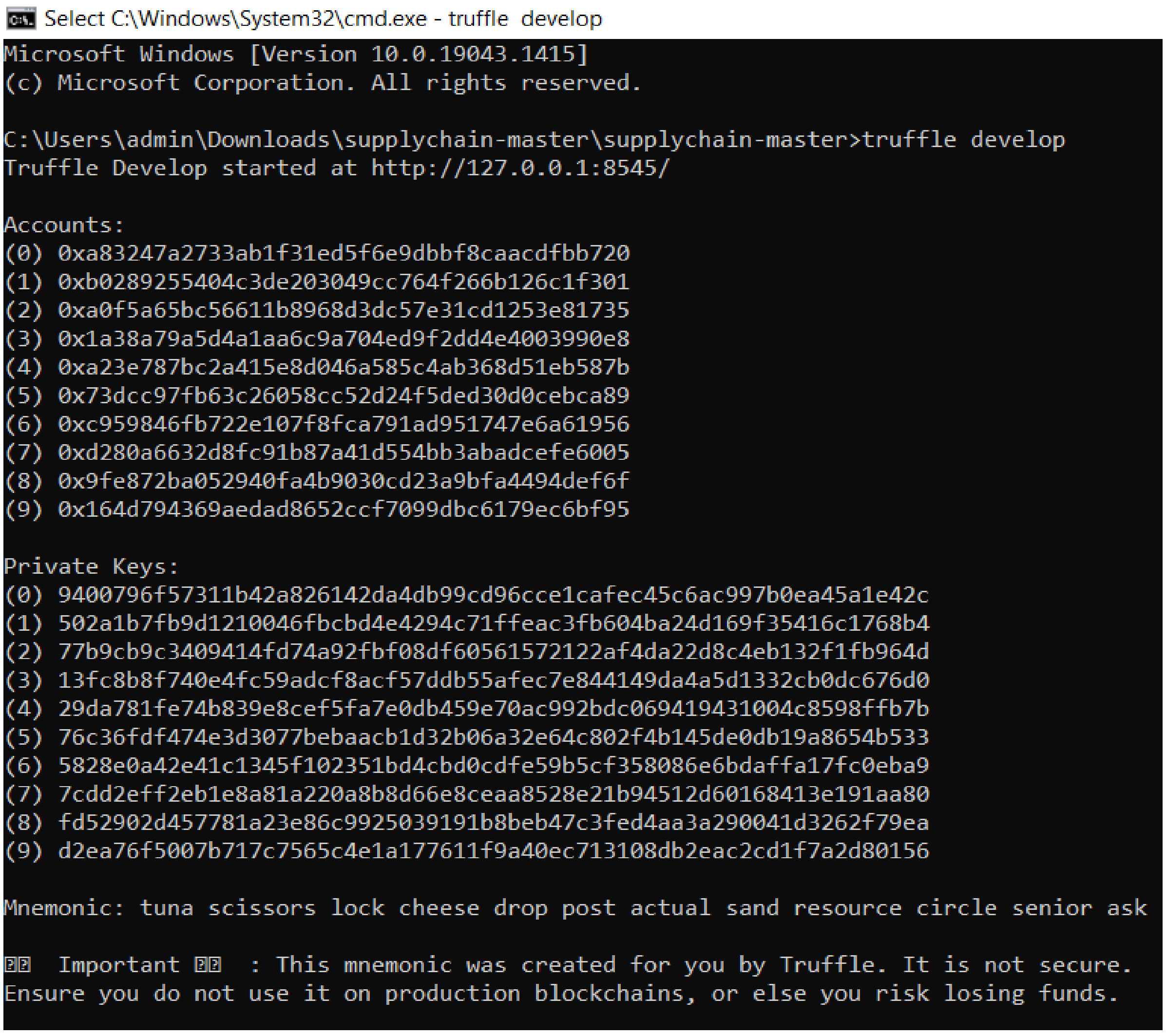
```
var TrackProduct = artifacts.require("./TrackProduct.sol");

module.exports = function(deployer) {
  deployer.deploy(TrackProduct);
};
```

4.2 Implementation and Output

Developed on Etherium smart contract using truffle react box.

A) To start Truffle Box

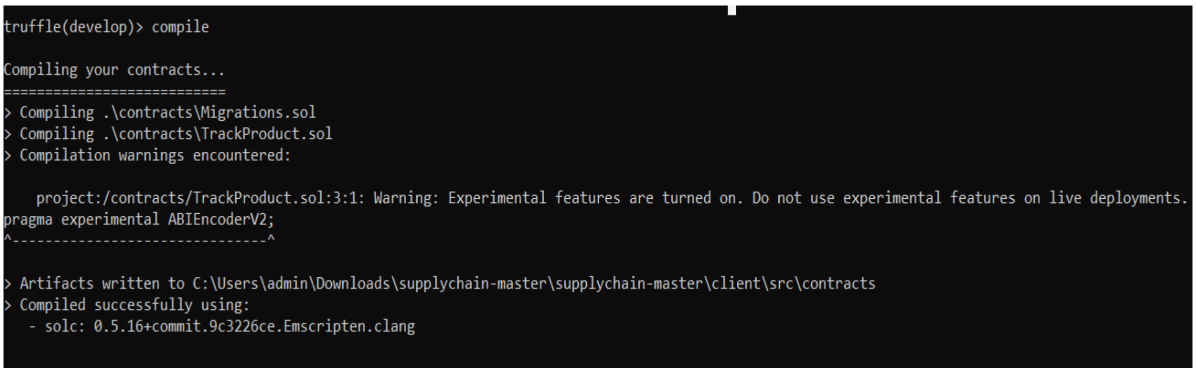
1. truffle develop



Select C:\Windows\System32\cmd.exe - truffle develop

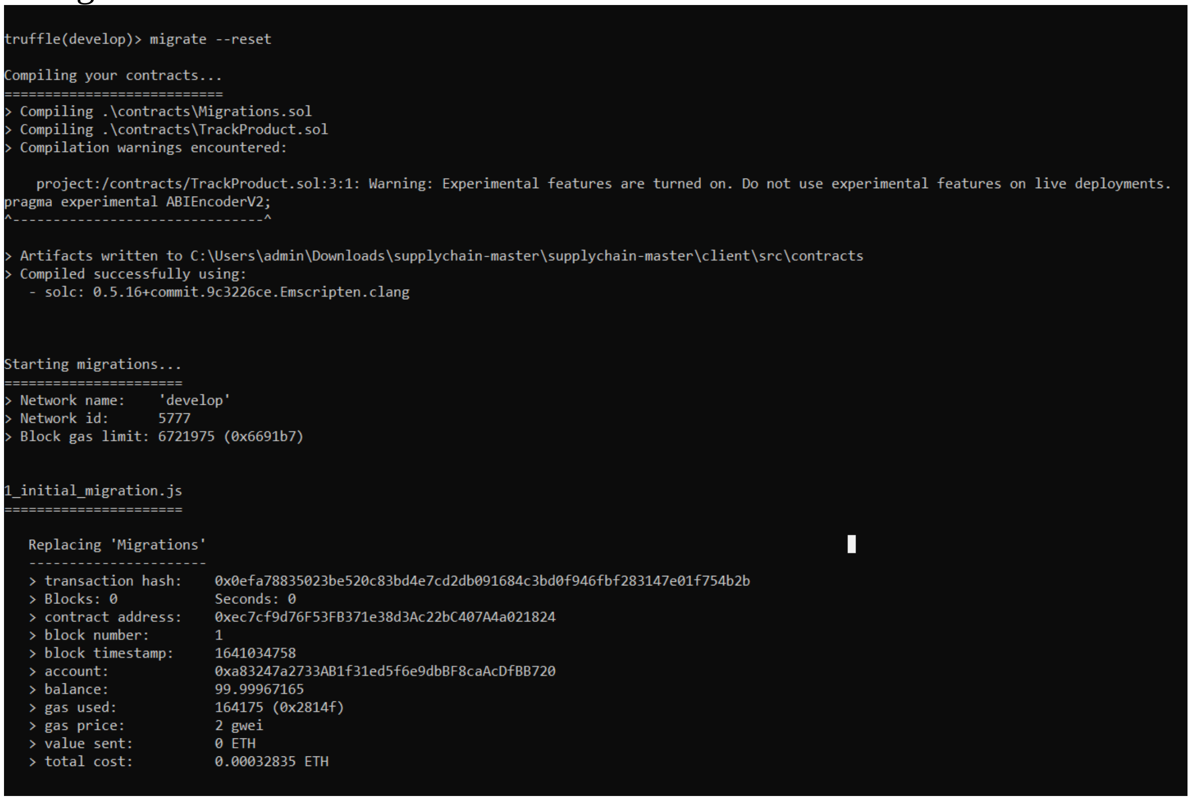
```
Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin\Downloads\supplychain-master\supplychain-master>truffle develop
Truffle Develop started at http://127.0.0.1:8545/

Accounts:
(0) 0xa83247a2733ab1f31ed5f6e9dbbf8caacdfbb720
(1) 0xb0289255404c3de203049cc764f266b126c1f301
(2) 0xa0f5a65bc56611b8968d3dc57e31cd1253e81735
(3) 0x1a38a79a5d4a1aa6c9a704ed9f2dd4e4003990e8
(4) 0xa23e787bc2a415e8d046a585c4ab368d51eb587b
(5) 0x73dcc97fb63c26058cc52d24f5ded30d0cebca89
(6) 0xc959846fb722e107f8fca791ad951747e6a61956
(7) 0xd280a6632d8fc91b87a41d554bb3abadcefe6005
(8) 0x9fe872ba052940fa4b9030cd23a9bfa4494def6f
(9) 0x164d794369aedad8652ccf7099dbc6179ec6bf95

Private Keys:
(0) 9400796f57311b42a826142da4db99cd96cce1cafec45c6ac997b0ea45a1e42c
(1) 502a1b7fb9d1210046fbcbd4e4294c71ffeac3fb604ba24d169f35416c1768b4
(2) 77b9cb9c3409414fd74a92fbf08df60561572122af4da22d8c4eb132f1fb964d
(3) 13fc8b8f740e4fc59adcf8acf57ddb55afec7e844149da4a5d1332cb0dc676d0
(4) 29da781fe74b839e8cef5fa7e0db459e70ac992bdc069419431004c8598ffb7b
(5) 76c36fdf474e3d3077bebaacb1d32b06a32e64c802f4b145de0db19a8654b533
(6) 5828e0a42e41c1345f102351bd4cbd0cdfe59b5cf358086e6bdaffa17fc0eba9
(7) 7cdd2eff2eb1e8a81a220a8b8d66e8ceaa8528e21b94512d60168413e191aa80
(8) fd52902d457781a23e86c9925039191b8beb47c3fed4aa3a290041d3262f79ea
(9) d2ea76f5007b717c7565c4e1a177611f9a40ec713108db2eac2cd1f7a2d80156

Mnemonic: tuna scissors lock cheese drop post actual sand resource circle senior ask

□□ Important □□ : This mnemonic was created for you by Truffle. It is not secure.
Ensure you do not use it on production blockchains, or else you risk losing funds.
```

2. compile

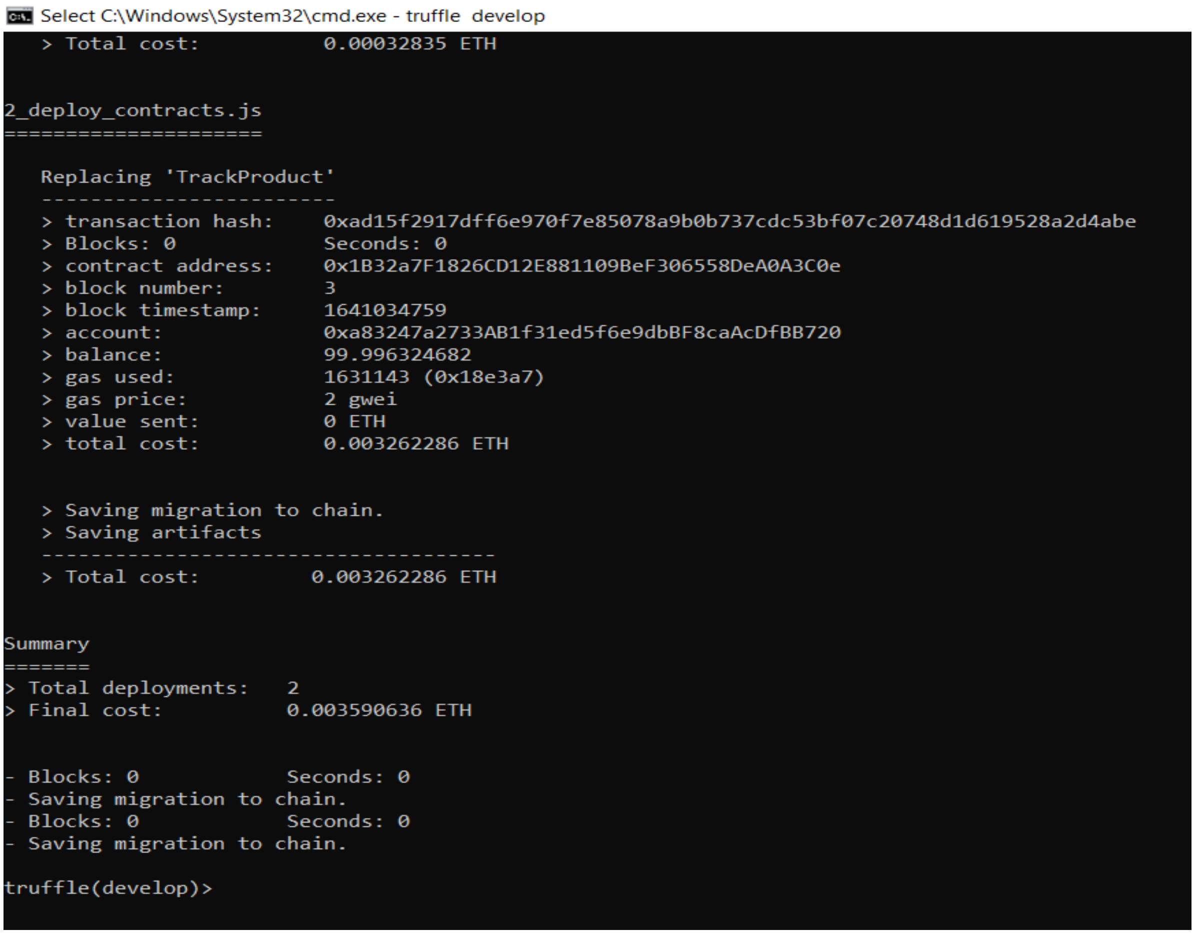


```
truffle(develop)> compile

Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\TrackProduct.sol
> Compilation warnings encountered:

    project:/contracts/TrackProduct.sol:3:1: Warning: Experimental features are turned on. Do not use experimental features on live deployments.
pragma experimental ABIEncoderV2;
^-------------------------------^

> Artifacts written to C:\Users\admin\Downloads\supplychain-master\supplychain-master\client\src\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

3. migrate --reset



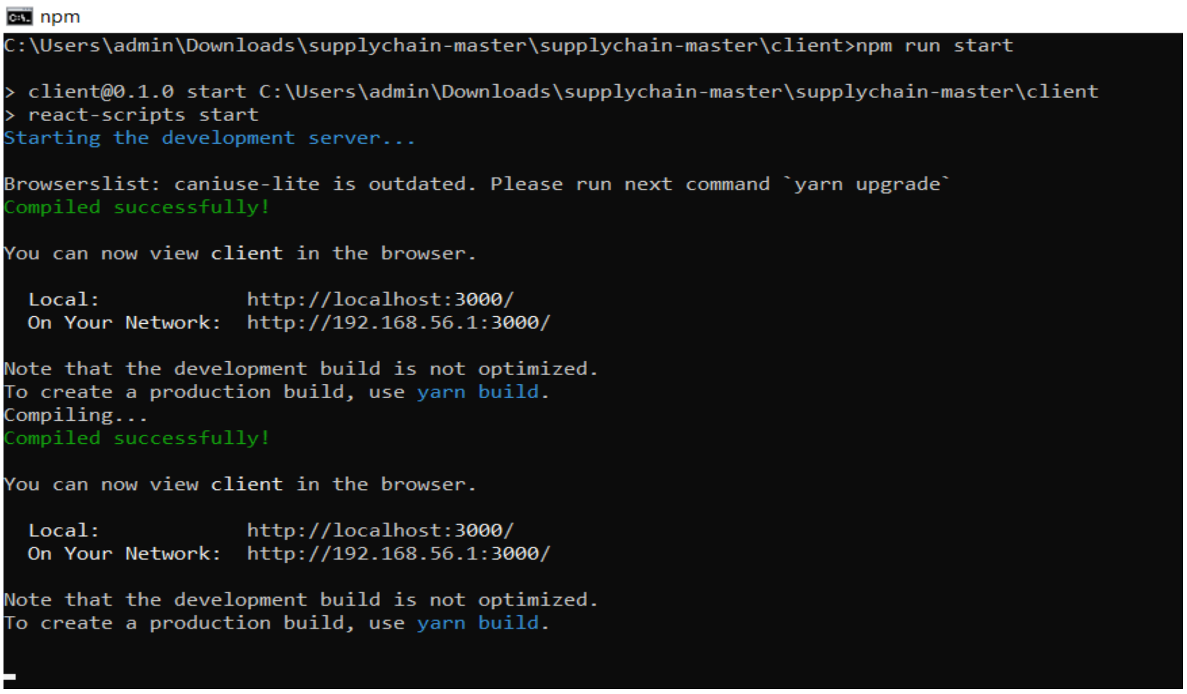
```
truffle(develop)> migrate --reset

Compiling your contracts...
===========================
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\TrackProduct.sol
> Compilation warnings encountered:

    project:/contracts/TrackProduct.sol:3:1: Warning: Experimental features are turned on. Do not use experimental features on live deployments.
pragma experimental ABIEncoderV2;
^-------------------------------^

> Artifacts written to C:\Users\admin\Downloads\supplychain-master\supplychain-master\client\src\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



Starting migrations...
======================
> Network name:    'develop'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0x0efa78835023be520c83bd4e7cd2db091684c3bd0f946fbf283147e01f754b2b
   > Blocks: 0            Seconds: 0
   > contract address:    0xec7cf9d76F53FB371e38d3Ac22bC407A4a021824
   > block number:        1
   > block timestamp:     1641034758
   > account:             0xa83247a2733AB1f31ed5f6e9dbBF8caAcDfBB720
   > balance:             99.99967165
   > gas used:            164175 (0x2814f)
   > gas price:           2 gwei
   > value sent:          0 ETH
   > total cost:          0.00032835 ETH
```



```
Select C:\Windows\System32\cmd.exe - truffle  develop
   > Total cost:          0.00032835 ETH


2_deploy_contracts.js
=====================

   Replacing 'TrackProduct'
   ------------------------
   > transaction hash:    0xad15f2917dff6e970f7e85078a9b0b737cdc53bf07c20748d1d619528a2d4abe
   > Blocks: 0            Seconds: 0
   > contract address:    0x1B32a7F1826CD12E881109BeF306558DeA0A3C0e
   > block number:        3
   > block timestamp:     1641034759
   > account:             0xa83247a2733AB1f31ed5f6e9dbBF8caAcDfBB720
   > balance:             99.996324682
   > gas used:            1631143 (0x18e3a7)
   > gas price:           2 gwei
   > value sent:          0 ETH
   > total cost:          0.003262286 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:         0.003262286 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.003590636 ETH


- Blocks: 0            Seconds: 0
- Saving migration to chain.
- Blocks: 0            Seconds: 0
- Saving migration to chain.

truffle(develop)>
```

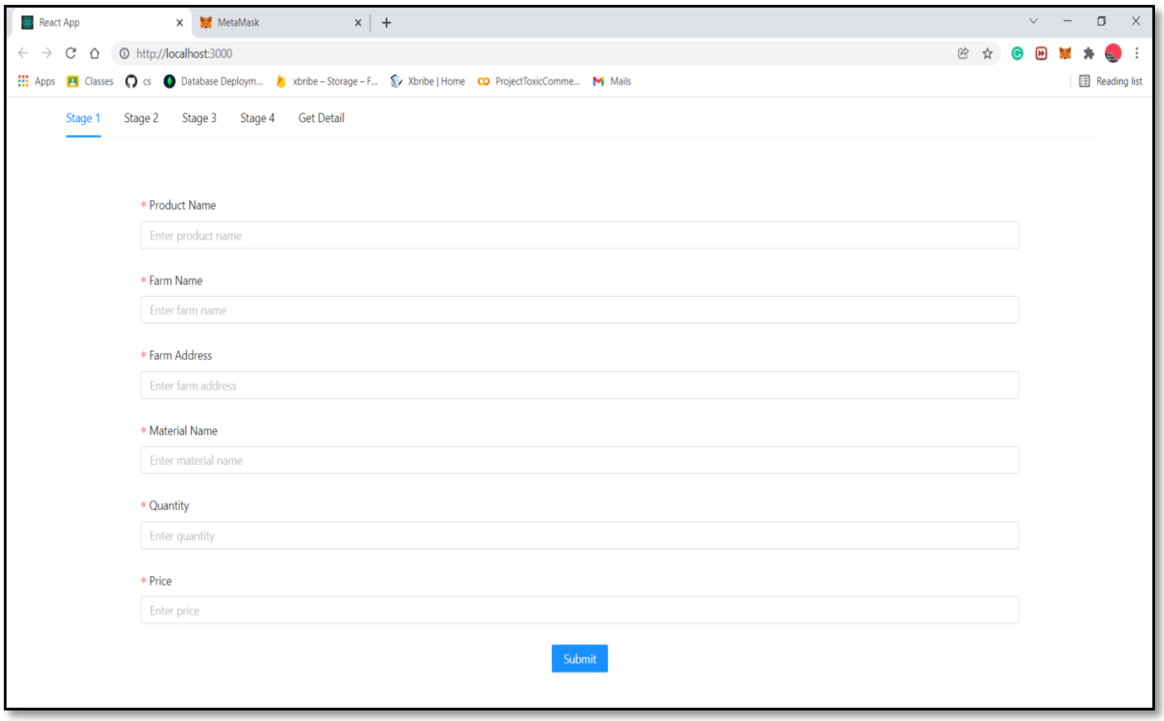
B) To start Localhost

1. Open another terminal and go inside client directory
2. npm install
3. npm run start => Navigate to => localhost:3000



```
npm
C:\Users\admin\Downloads\supplychain-master\supplychain-master\client>npm run start

> client@0.1.0 start C:\Users\admin\Downloads\supplychain-master\supplychain-master\client
> react-scripts start
Starting the development server...

Browserslist: caniuse-lite is outdated. Please run next command `yarn upgrade`
Compiled successfully!

You can now view client in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.56.1:3000/

Note that the development build is not optimized.
To create a production build, use yarn build.
Compiling...
Compiled successfully!

You can now view client in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.56.1:3000/

Note that the development build is not optimized.
To create a production build, use yarn build.

_
```

It will start react application on localhost:3000, on your browser



* Product Name
Enter product name

* Farm Name
Enter farm name

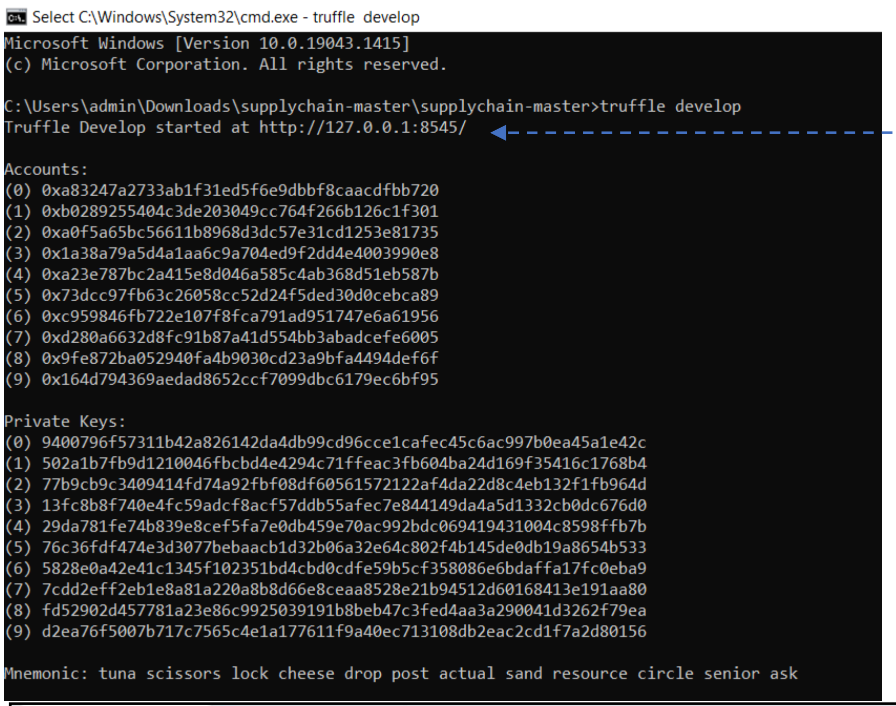
* Farm Address
Enter farm address

* Material Name
Enter material name

* Quantity
Enter quantity

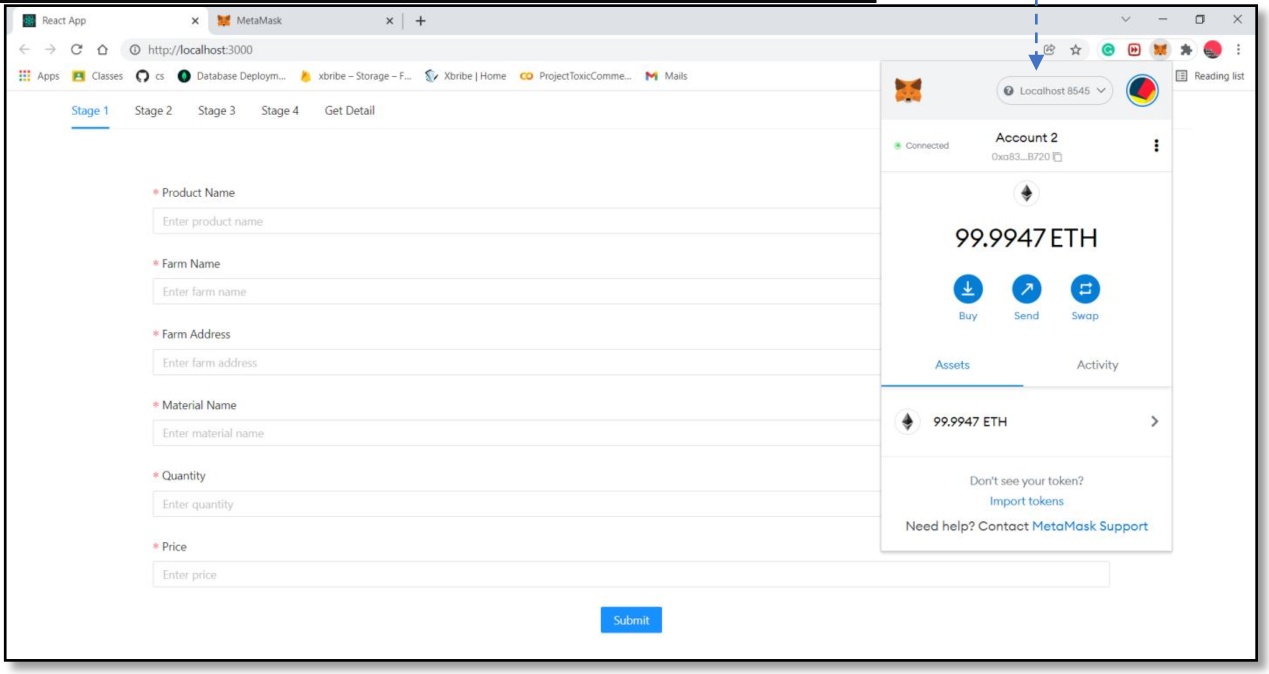
* Price
Enter price

C) To connect Metamask, make sure youron same port as on truffle and also your account must have at least 100ETH to make transactions.



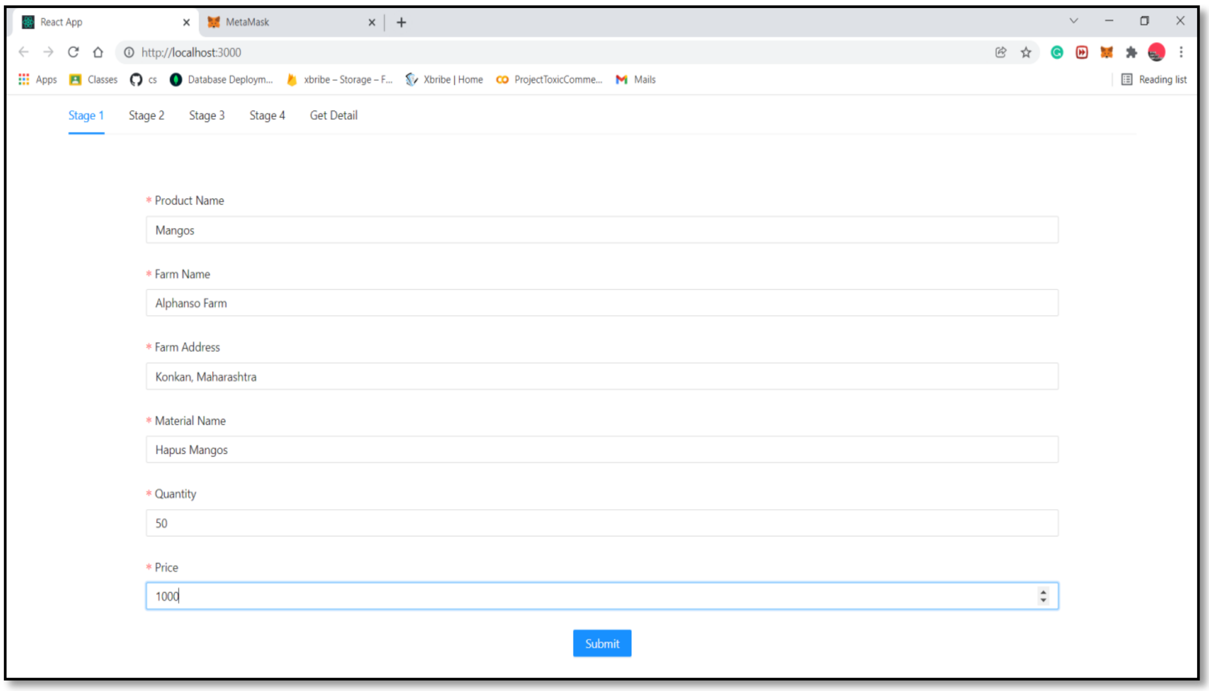
```
Select C:\Windows\System32\cmd.exe - truffle  develop
Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin\Downloads\supplychain-master\supplychain-master>truffle develop
Truffle Develop started at http://127.0.0.1:8545/

Accounts:
(0) 0xa83247a2733ab1f31ed5f6e9dbbf8caacdfbb720
(1) 0xb0289255404c3de203049cc764f266b126c1f301
(2) 0xa0f5a65bc56611b8968d3dc57e31cd1253e81735
(3) 0x1a38a79a5d4a1aa6c9a704ed9f2dd4e4003990e8
(4) 0xa23e787bc2a415e8d046a585c4ab368d51eb587b
(5) 0x73dcc97fb63c26058cc52d24f5ded30d0cebca89
(6) 0xc959846fb722e107f8fca791ad951747e6a61956
(7) 0xd280a6632d8fc91b87a41d554bb3abadcefe6005
(8) 0x9fe872ba052940fa4b9030cd23a9bfa4494def6f
(9) 0x164d794369aedad8652ccf7099dbc6179ec6bf95

Private Keys:
(0) 9400796f57311b42a826142da4db99cd96cce1cafec45c6ac997b0ea45a1e42c
(1) 502a1b7fb9d1210046fbcbd4e4294c71ffeac3fb604ba24d169f35416c1768b4
(2) 77b9cb9c3409414fd74a92fbf08df60561572122af4da22d8c4eb132f1fb964d
(3) 13fc8b8f740e4fc59adcf8acf57ddb55afec7e844149da4a5d1332cb0dc676d0
(4) 29da781fe74b839e8cef5fa7e0db459e70ac992bdc069419431004c8598ffb7b
(5) 76c36fdf474e3d3077bebaacb1d32b06a32e64c802f4b145de0db19a8654b533
(6) 5828e0a42e41c1345f102351bd4cbd0cdfe59b5cf358086e6bdaffa17fc0eba9
(7) 7cdd2eff2eb1e8a81a220a8b8d66e8ceaa8528e21b94512d60168413e191aa80
(8) fd52902d457781a23e86c9925039191b8beb47c3fed4aa3a290041d3262f79ea
(9) d2ea76f5007b717c7565c4e1a177611f9a40ec713108db2eac2cd1f7a2d80156

Mnemonic: tuna scissors lock cheese drop post actual sand resource circle senior ask
```

Same port no : 8545



D) To make transaction

1. First Stage (Raw Material Details)



* Product Name
Mangos

* Farm Name
Alphanso Farm

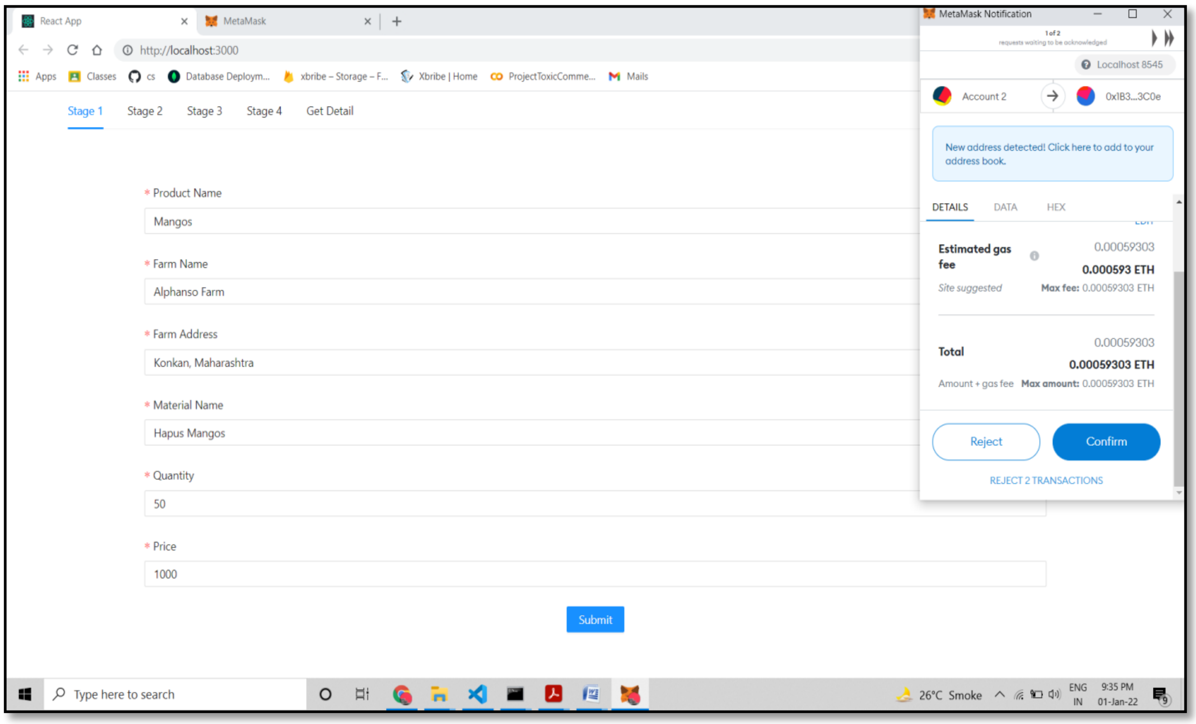
* Farm Address
Konkan, Maharashtra

* Material Name
Hapus Mangos

* Quantity
50

* Price
1000

Every time, Click on submit button, to confirm transaction.



* Product Name
Mangos

* Farm Name
Alphanso Farm

* Farm Address
Konkan, Maharashtra

* Material Name
Hapus Mangos

* Quantity
50

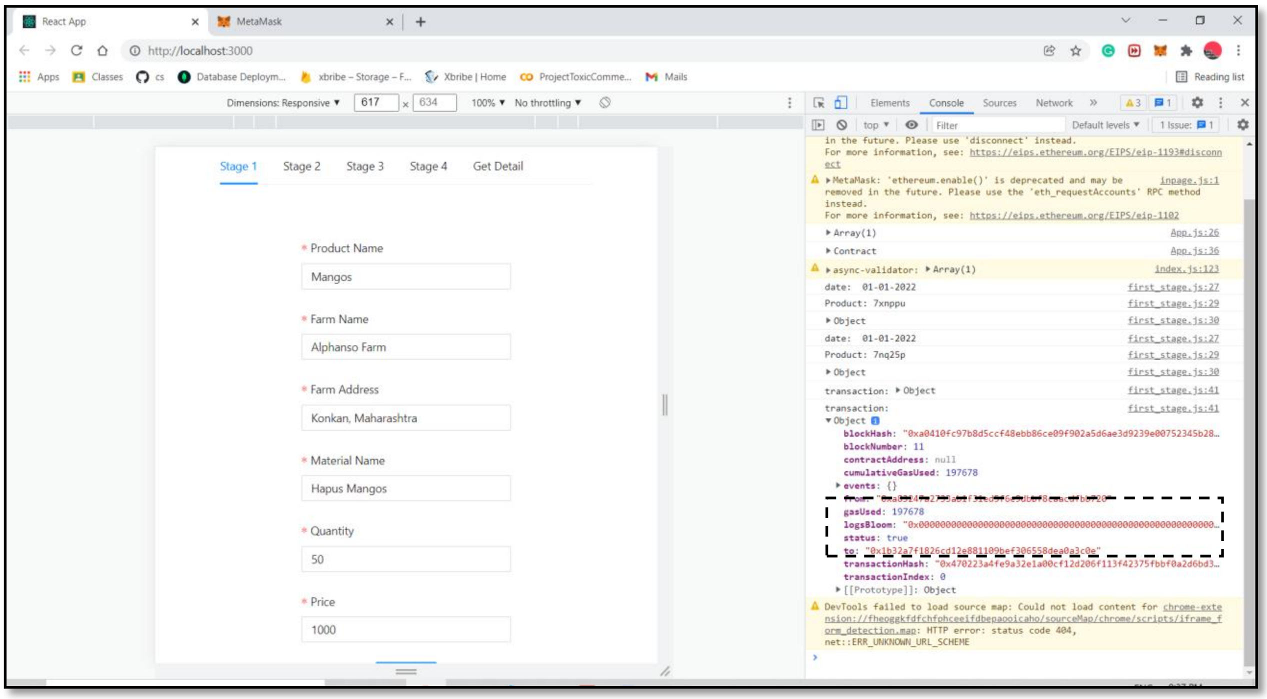
* Price
1000

1 of 2
requests waiting to be acknowledged
Localhost 8545
Account 2 → 0x1B3...3C0e
New address detected! Click here to add to your address book.

Estimated gas fee 0.00059303
0.000593 ETH
Site suggested Max fee: 0.00059303 ETH

Total 0.00059303
0.00059303 ETH
Amount + gas fee Max amount: 0.00059303 ETH

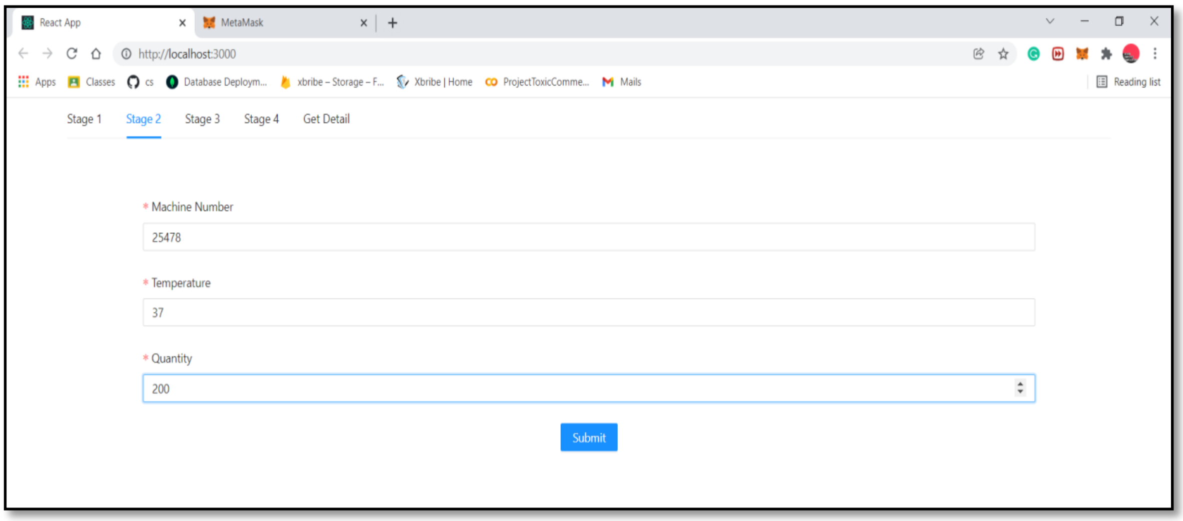
From console, you can check the status of transaction along with all details.



* Product Name	Mangos
* Farm Name	Alphanso Farm
* Farm Address	Konkan, Maharashtra
* Material Name	Hapus Mangos
* Quantity	50
* Price	1000

```
in the future. Please use 'disconnect' instead.
For more information, see: https://eips.ethereum.org/EIPS/eip-1193#disconn
ect
▸MetaMask: 'ethereum.enable()' is deprecated and may be        inpage.js:1
removed in the future. Please use the 'eth_requestAccounts' RPC method
instead.
For more information, see: https://eips.ethereum.org/EIPS/eip-1102
▸Array(1)                                                      App.js:26
▸Contract                                                      App.js:36
▸async-validator: ▸Array(1)                                    index.js:123
date:  01-01-2022                                              first_stage.js:27
Product: 7xnppu                                                first_stage.js:29
▸Object                                                        first_stage.js:30
date:  01-01-2022                                              first_stage.js:27
Product: 7nq25p                                                first_stage.js:29
▸Object                                                        first_stage.js:30
transaction: ▸Object                                           first_stage.js:41
transaction:                                                   first_stage.js:41
▾Object
   blockHash: "0xa0410fc97b8d5ccf48ebb86ce09f902a5d6ae3d9239e00752345b28…
   blockNumber: 11
   contractAddress: null
   cumulativeGasUsed: 197678
  ▸events: {}
   from: "0xa83247a2753ab1f31ed5f6e9dbbf8caacdfbb720"
   gasUsed: 197678
   logsBloom: "0x0000000000000000000000000000000000000000000000000000000…
   status: true
   to: "0x1b32a7f1826cd12e881109bef306558dea0a3c0e"
   transactionHash: "0x470223a4fe9a32e1a00cf12d206f113f42375fbbf0a2d6bd3…
   transactionIndex: 0
  ▸[[Prototype]]: Object
DevTools failed to load source map: Could not load content for chrome-exte
nsion://fheoggkfdfchfphceeifdbepaooicaho/sourceMap/chrome/scripts/iframe_f
orm_detection.map: HTTP error: status code 404,
net::ERR_UNKNOWN_URL_SCHEME
>
```

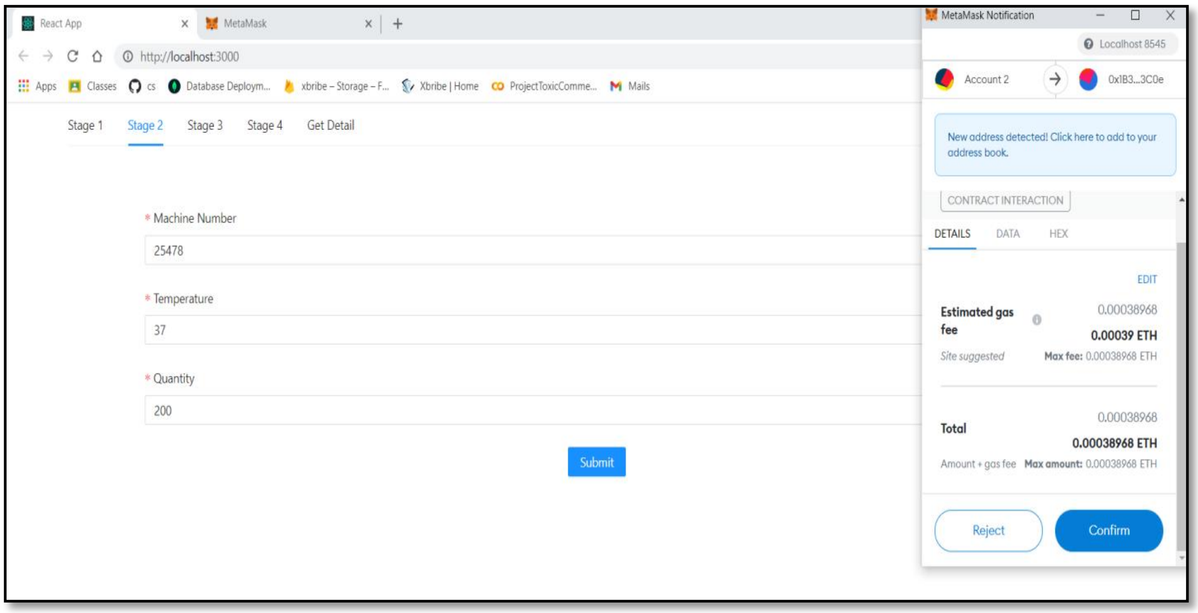
2. Second Stage (Manufacturing Details)


Stage 1 Stage 2 Stage 3 Stage 4 Get Detail

* Machine Number: 25478
* Temperature: 37
* Quantity: 200

Submit

Click on Submit button to confirm transaction.


Stage 1 Stage 2 Stage 3 Stage 4 Get Detail

* Machine Number: 25478
* Temperature: 37
* Quantity: 200

Submit

MetaMask Notification

Localhost 8545

Account 2 → 0x1B3...3C0e

New address detected! Click here to add to your address book.

CONTRACT INTERACTION

DETAILS DATA HEX

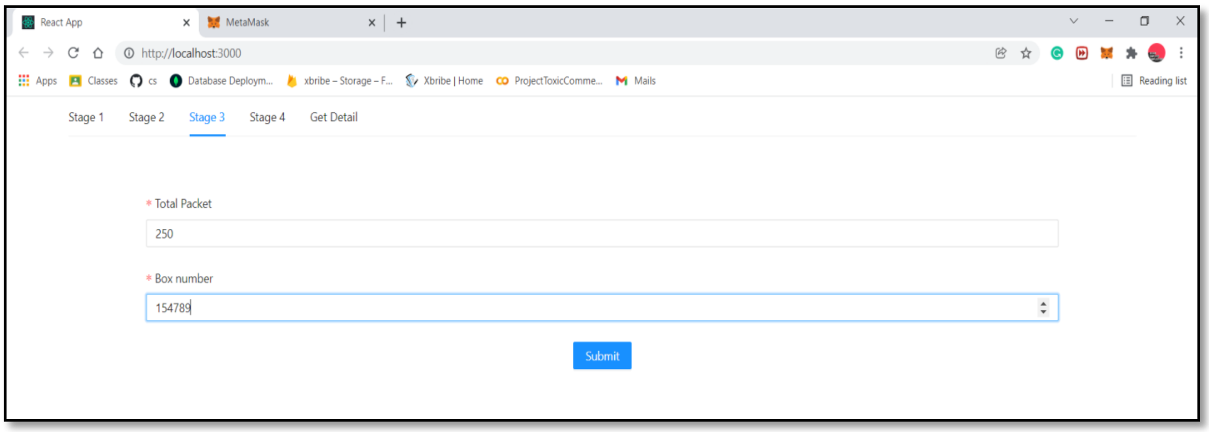
EDIT

Estimated gas fee 0.00038968
0.00039 ETH
Site suggested Max fee: 0.00038968 ETH

Total 0.00038968
0.00038968 ETH
Amount + gas fee Max amount: 0.00038968 ETH

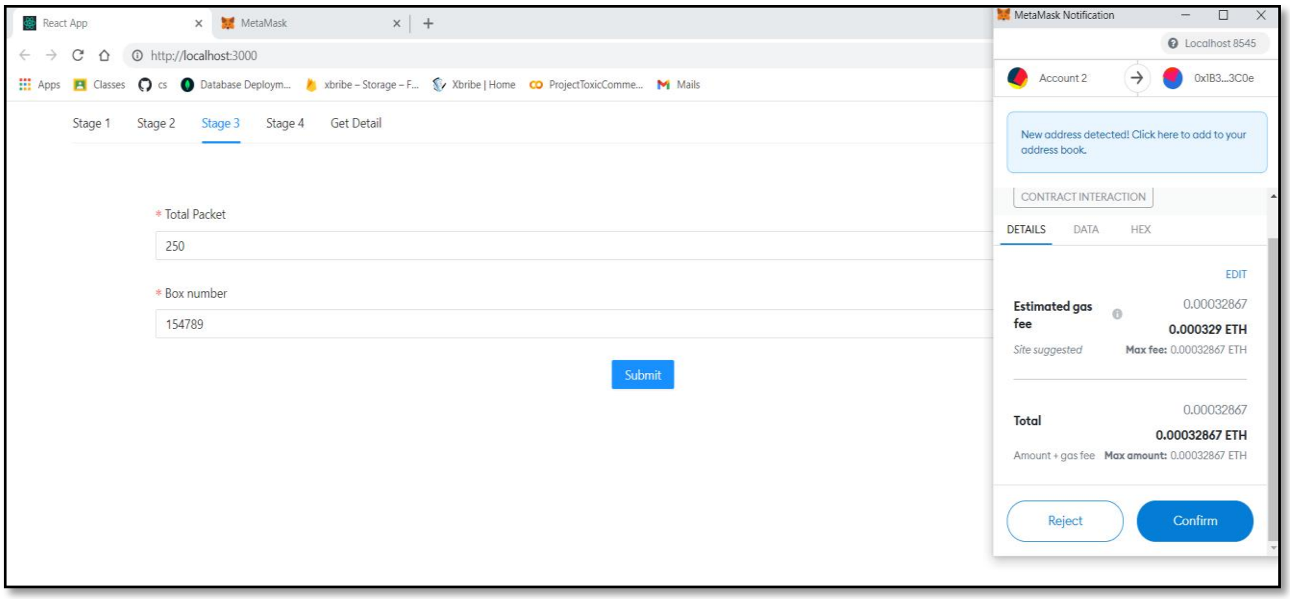
Reject Confirm

3. Third Stage (Packing Details)



* Total Packet
250
* Box number
154789

Click on Submit button to confirm transaction.



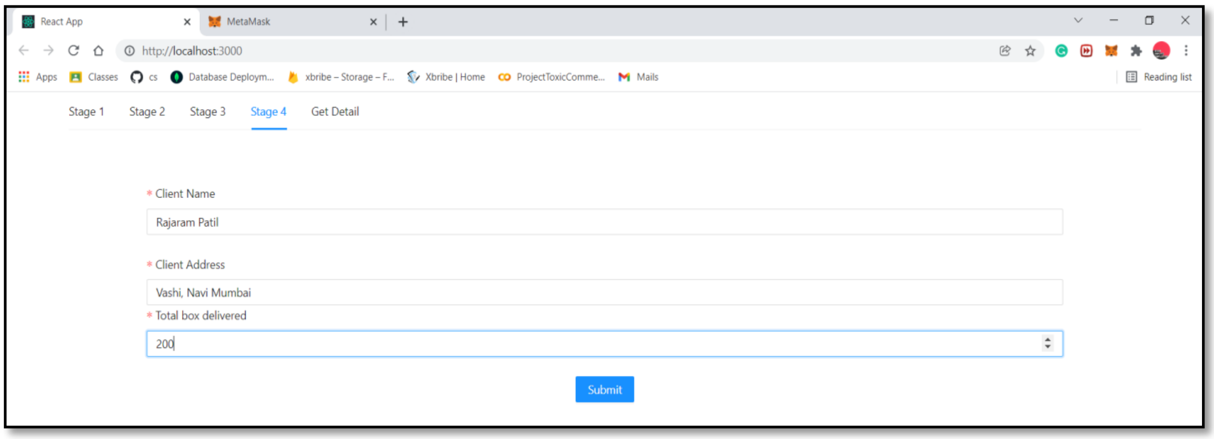
* Total Packet
250
* Box number
154789

New address detected! Click here to add to your address book.

Estimated gas fee 0.00032867 0.000329 ETH
Site suggested Max fee: 0.00032867 ETH

Total 0.00032867 0.00032867 ETH
Amount + gas fee Max amount: 0.00032867 ETH

4. Fourth Stage (Delivery Details)

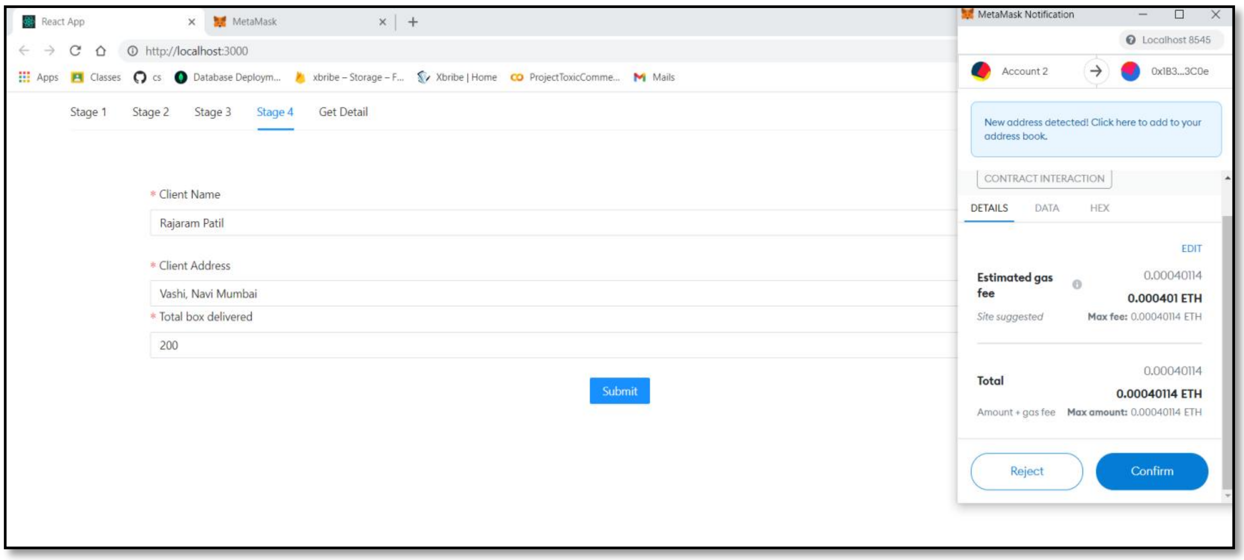


* Client Name
Rajaram Patil

* Client Address
Vashi, Navi Mumbai

* Total box delivered
200

Click on Submit button to confirm transaction.



* Client Name
Rajaram Patil

* Client Address
Vashi, Navi Mumbai

* Total box delivered
200

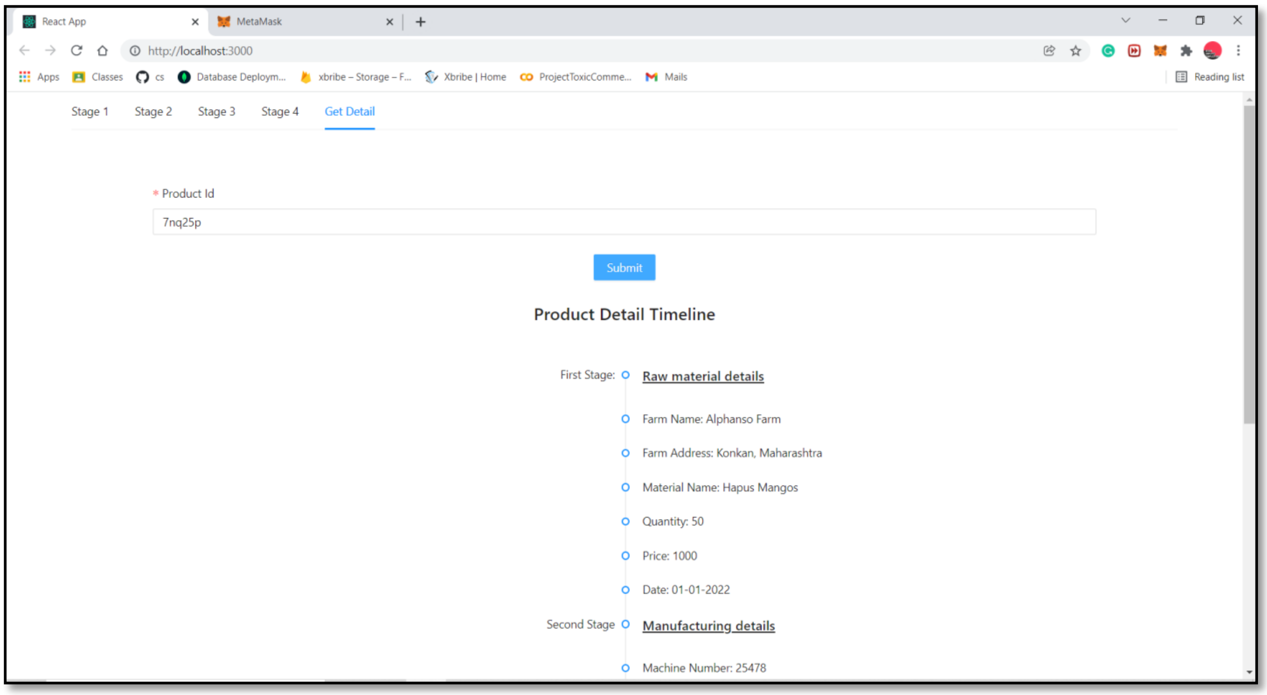
New address detected! Click here to add to your address book.

Estimated gas fee 0.00040114 0.000401 ETH
Site suggested Max fee: 0.00040114 ETH

Total 0.00040114 0.00040114 ETH
Amount + gas fee Max amount: 0.00040114 ETH

5. Get Details on supply chain

Enter Product id



* Product Id

7nq25p

Product Detail Timeline

First Stage: Raw material details

Farm Name: Alphanso Farm

Farm Address: Konkan, Maharashtra

Material Name: Hapus Mangos

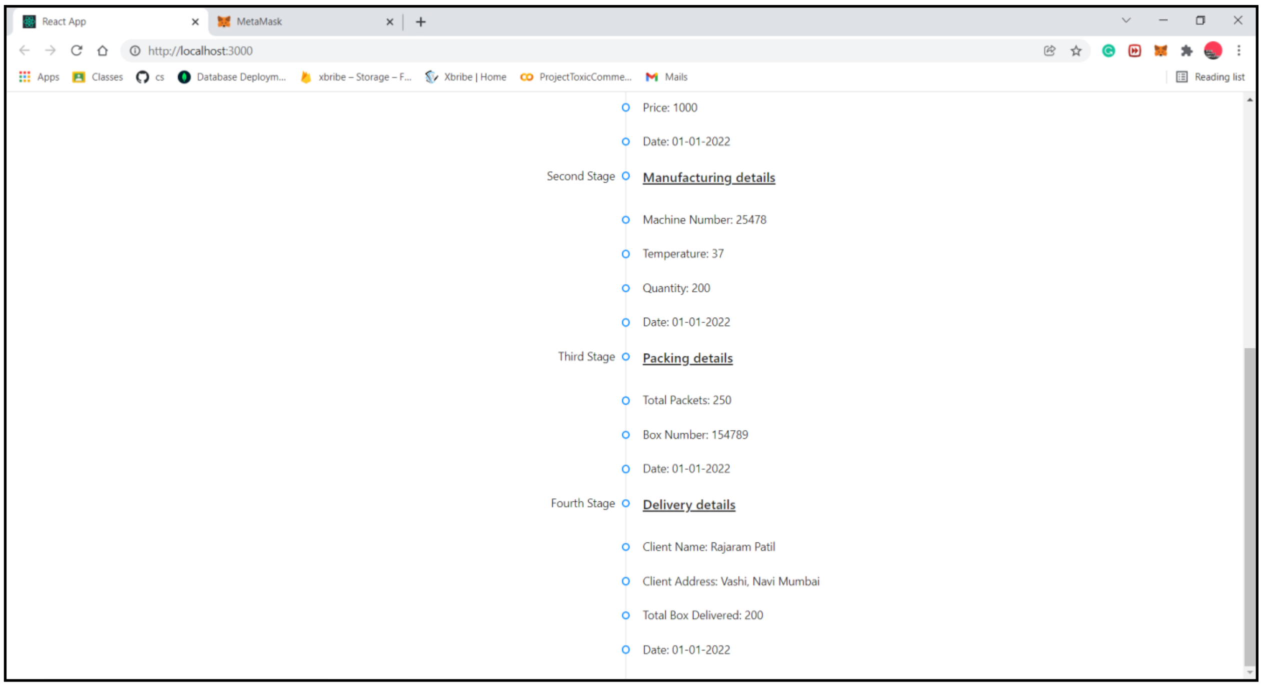
Quantity: 50

Price: 1000

Date: 01-01-2022

Second Stage Manufacturing details

Machine Number: 25478



Price: 1000

Date: 01-01-2022

Second Stage Manufacturing details

Machine Number: 25478

Temperature: 37

Quantity: 200

Date: 01-01-2022

Third Stage Packing details

Total Packets: 250

Box Number: 154789

Date: 01-01-2022

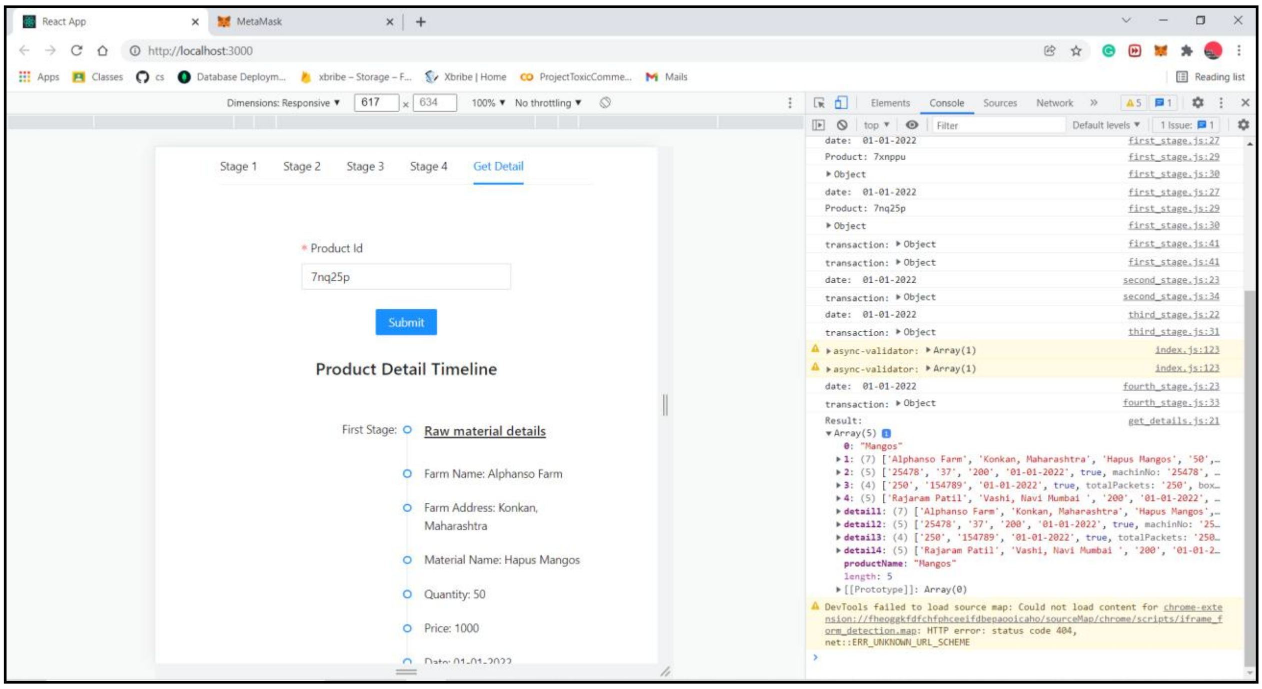
Fourth Stage Delivery details

Client Name: Rajaram Patil

Client Address: Vashi, Navi Mumbai

Total Box Delivered: 200

Date: 01-01-2022



* Product Id

7nq25p

Product Detail Timeline

First Stage: Raw material details

Farm Name: Alphanso Farm

Farm Address: Konkan, Maharashtra

Material Name: Hapus Mangos

Quantity: 50

Price: 1000

Date: 01-01-2022

```
date:  01-01-2022                                                   first_stage.js:27
Product: 7xnppu                                                     first_stage.js:29
▸ Object                                                            first_stage.js:30
date:  01-01-2022                                                   first_stage.js:27
Product: 7nq25p                                                     first_stage.js:29
▸ Object                                                            first_stage.js:30
transaction: ▸ Object                                               first_stage.js:41
transaction: ▸ Object                                               first_stage.js:41
date:  01-01-2022                                                  second_stage.js:23
transaction: ▸ Object                                              second_stage.js:34
date:  01-01-2022                                                   third_stage.js:22
transaction: ▸ Object                                               third_stage.js:31
▸ async-validator: ▸ Array(1)                                          index.js:123
▸ async-validator: ▸ Array(1)                                          index.js:123
date:  01-01-2022                                                  fourth_stage.js:23
transaction: ▸ Object                                              fourth_stage.js:33
Result:                                                              get_details.js:21
▾ Array(5)
    0: "Mangos"
  ▸ 1: (7) ['Alphanso Farm', 'Konkan, Maharashtra', 'Hapus Mangos', '50',…
  ▸ 2: (5) ['25478', '37', '200', '01-01-2022', true, machinNo: '25478', …
  ▸ 3: (4) ['250', '154789', '01-01-2022', true, totalPackets: '250', box…
  ▸ 4: (5) ['Rajaram Patil', 'Vashi, Navi Mumbai ', '200', '01-01-2022', …
  ▸ detail1: (7) ['Alphanso Farm', 'Konkan, Maharashtra', 'Hapus Mangos',…
  ▸ detail2: (5) ['25478', '37', '200', '01-01-2022', true, machinNo: '25…
  ▸ detail3: (4) ['250', '154789', '01-01-2022', true, totalPackets: '250…
  ▸ detail4: (5) ['Rajaram Patil', 'Vashi, Navi Mumbai ', '200', '01-01-2…
    productName: "Mangos"
    length: 5
  ▸ [[Prototype]]: Array(0)
DevTools failed to load source map: Could not load content for chrome-exte
nsion://fheoggkfdfchfphceeifdbepaooicaho/sourceMap/chrome/scripts/iframe_f
orm_detection.map: HTTP error: status code 404,
net::ERR_UNKNOWN_URL_SCHEME
```

5. FUTURE SCOPE AND CONCLUSION

5.1 Future Scope

This system can be employed by the production industries in order to trace the processing details of a product which is already processed and packaged. Transparency can be achieved and hence increase in the profits of the firm.In order to trace the details, a unique number is attached with every product and when that unique number is entered in the system then the system looks for the valuesassociated with that key and fetches the information to the user.Statistical analysis can also be done with the help of the proposed model and the efficiency of the different processing units can also be determined. Sources can be identified on the basis of quality, the sources from which good quality products are packaged can be marked as genuine class of sources.

5.2 Conclusion

On the concluding notes, we proposed a web based cryptographic solution to solve the problem of traceability in the food supply chain. The proposed web application can be used to log the transactional data from a food processing industry and it is stored over a blockchain ledger in a tamper proof manner. The algorithms used to solve the problem are RSA and SHA256, there were no significant changes made to the original approach of the problems. The web application can be set up by different clients and a single server node, the server node has all the authority to publish blocks on verifying the authenticity of the data received from the client nodes

REFERENCES

1. Blockchain Security and Relevant Attacks, Joanna Moubarak, Eric Filiol, 2018.
2. A Concept Framework for Agri-Food Supply Chain Integration in China, Juan Xu, DeBin Zhang.
3. Blockchain Application in Food Supply Information Security, Daniel Tse, Haoran Mu, 2018.
4. Challenges in the Management of Food Products Supply chain, Aleksandra PABIAN.
5. Blockchain-based Traceability in Agri-Food Supply Chain Management, Muhammad Salek Ali, 2017.
6. Blockchain In supply chain (https://hackernoon.com/how-is-blockchain-disruptingthe-supply-chain-industry-f3a1c599daef)
7. https://blockgeeks.com/guides/blockchain-and-supplychain/
8. Supply chain problems(https://www.dataversity.net/blockchainsolution-underlying-issues-supply-chain-management/)