BLOCKCHAIN TECHNOLOGY IN RETAIL INDUSTRY: INNOVATION IN SUPPLY CHAIN MANAGEMENT

Project report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

By

Venkatesh Gunda (1410110493)

Under supervision

of

Dr. Udit Satija
Mrs. Felisha Manos
Dell Technologies

DEPARTMENT OF ELECTRICAL ENGINEERING
SCHOOL OF ENGINEERING
SHIV NADAR UNIVERSITY

May, 2018

Candidate Declaration

I hereby declare that the thesis entitled “Blockchain Technology in Retail Industry: Innovation in Supply Chain Management” submitted for the B Tech Degree program. This thesis has written in my own words. I have adequately cited and referenced the original sources.

Venkatesh Gunda

1410110493

Date: 09 May, 2018

CERTIFICATE

It is certified that the work contained in the project report titled “Blockchain Technology in Retail Industry: Innovation in Supply Chain Management,” by “Venkatesh Gunda” has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

Dr. Udit Satija
Department of Electrical Engineering
School of Engineering
Shiv Nadar University
May, 2018

Felisha Manos
Felisha Manos
Senior Manager, Field Marketing
DCE - Dell Commerce Experiences
Dell Technologies Pvt. Ltd.
May, 2018

ACKNOWLEDGEMENT

I am indebted to my external Supervisor Felisha Manos & my mentors during internship KR, Praveen and Prerna Khatri of Dell, Bangalore for their help and advice during this project. Their generous support helped me throughout my work. Their Advice was most valuable to understand the obtained results and to determine next steps for the work presented in this thesis.

Special thanks to my internal Supervisor, Dr. Udit Satija. His enthusiasm motivated to learn more about this interesting project at Dell.

Further, I would like to thank all my fellow colleagues at Dell for their cooperation and suggestions for the accomplishment of the work.

ABSTRACT

Developing a Software for handling the production of goods in the end-to-end tracking of a Product Lifecycle in FMCG Retail companies from the point of raw material procurement to the point of purchase by the consumer which is the final aim. This system is built on the blockchain technology which provides transparency, efficiency and uncorrupted data to the companies and the customers if the companies choose to reveal the information of the product lifecycle. It is based on the Ethereum blockchain. Ethereum uses the concept of Proof-of-State for reduced computational power and increased speed compared to the traditional Proof-of-Work. In this project, the blockchain is run privately (the other alternative is public blockchain where information is open and available globally which has a higher level of security but also needs higher computing power) because most of the companies prefer using their own infrastructure for running the day-to-day operations. This project gives a detailed explain on the architecture of the application, the development and the backend technology etc.

Keywords: Blockchain, Supply Chain Management, Ethereum

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

ABBREVIATIONS

IPFS	Inter-Planetary File System
dApp	Distributed Application
SCM	Supply Chain Management
APS	Advanced Planning and Scheduling
MES	Manufacturing Execution System
FMCG	Fast Moving Consumer Goods

1. Introduction

FMCG (Fast Moving Consumer Goods) Industry is the 4th Largest Industry in India with around 50 Billion Dollars revenue in 2016. [1]

FMCG Industry has one of the most outreach in the Indian population [1]. To achieve this rapid growth, most of the companies introduce multiple tiers to improve the coverage area as distribution is extremely difficult to manage by a single entity in a huge country like ours.

Because of all the third parties involved, it became almost impossible for the companies to extract analytical data to boost growth. Therefore, reliable and untampered information is essential for improving logistical efficiency & market growth.

Very often, the third-party vendors involved in the Supply chain do not keep track of the orders due to lack of the infrastructure. The companies allow this to boost the sales in the rural markets where the sales aren't very high but because reaching out to this population is more important.

But, reliable tracking is now made possible by Blockchain because of the low infrastructure costs and extremely high resilience to tampering. The way Blockchain works allows it to be accessed with just a smartphone with Internet access. This makes it extremely convenient for both the company and the low-income third-party vendors to use the services. Also, if the data has been lost from one of the device in the network, it's extremely easy to recover that data.

Customers will be able to check untampered information on where the raw materials are procured, what quality certifications have they passed, where are the products manufactured etc. These make the companies more transparent encouraging them to maintain the quality standards.

Indian Market is highly retail based because of the low-income population who prefer buying products on need basis rather than weekly or periodically repeating shopping patterns [2]. This makes is harder for Supermarkets and other big malls to survive in small cities and towns. This introduces a lot of challenges because of the extremely high number of shops in each town and

each lacking sufficient infrastructure to maintain digital records of transactions being conducted in the shops.

Smartphone reach in India has recently touched bigger heights because of the recent improvements in technology making them very affordable [3]. Even Internet connectivity through smartphones (4G) has been rapidly increasing because of decline in Mobile data prices owing to higher adoption rates among Indian consumers.

All these developments make the idea of having Blockchain as the backbone of retail supply chain management even more insightful and revenue-generating than many other alternatives adopted by the companies.

2. Literature Review

2.1 Theoretical Concepts:

2.1.1 Merkle Tree:

Merkle Tree is a cryptographic hash tree [4]. Every leaf-node (i.e. nodes in the bottom most layer) is labelled with the hash of a data block. All the non-leaf nodes are labelled with the hashes of their child nodes. Hash trees reduce the computational complexity of a computer program when searching for an index.

Hash trees are used to verify if the data received from other nodes in the network are received without any alteration and damaged and prevent the nodes from sending fake blocks.

2.1.2 IPFS:

Inter-Planetary File System is a protocol and network that is designed with the intention of creating a method of sharing, storing, accessing files in a distributed file system. For every file, an address is generated and this address can be used to access the files. The IPFS Stack contains the layers shown in Figure 1. The software that can be used to access these services are also mentioned:

Figure 2 explains how Ethereum, Dapp and IPFS communicate with each other:

Figure 1: Visual Representation of the IPFS Stack & their Layers

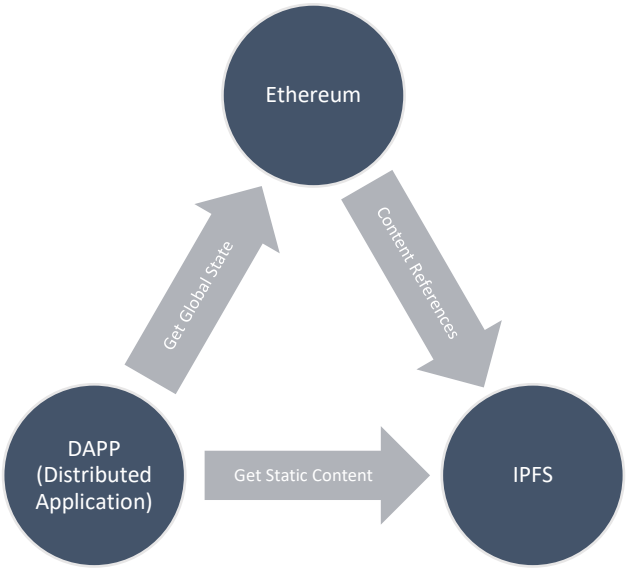

Figure 2: IPFS Stack in Ethereum Environment

2.2 Architecture of The Supply Chain Management:

In this paper [5], the authors have proposed a Supply Chain Management Architecture Model keeping in mind both the technical, business and operations standpoint. Therefore, for building our application, we're going to follow the architecture proposed in this paper and we're going to briefly discuss the inferences from the paper here.

Supply Chain Management mainly consists of these three pillars. They are:

- Supply Chain Business Processes
- Supply Chain Management Components
- Supply Chain Network Structure

Each of them is related as shown in the Figure 3

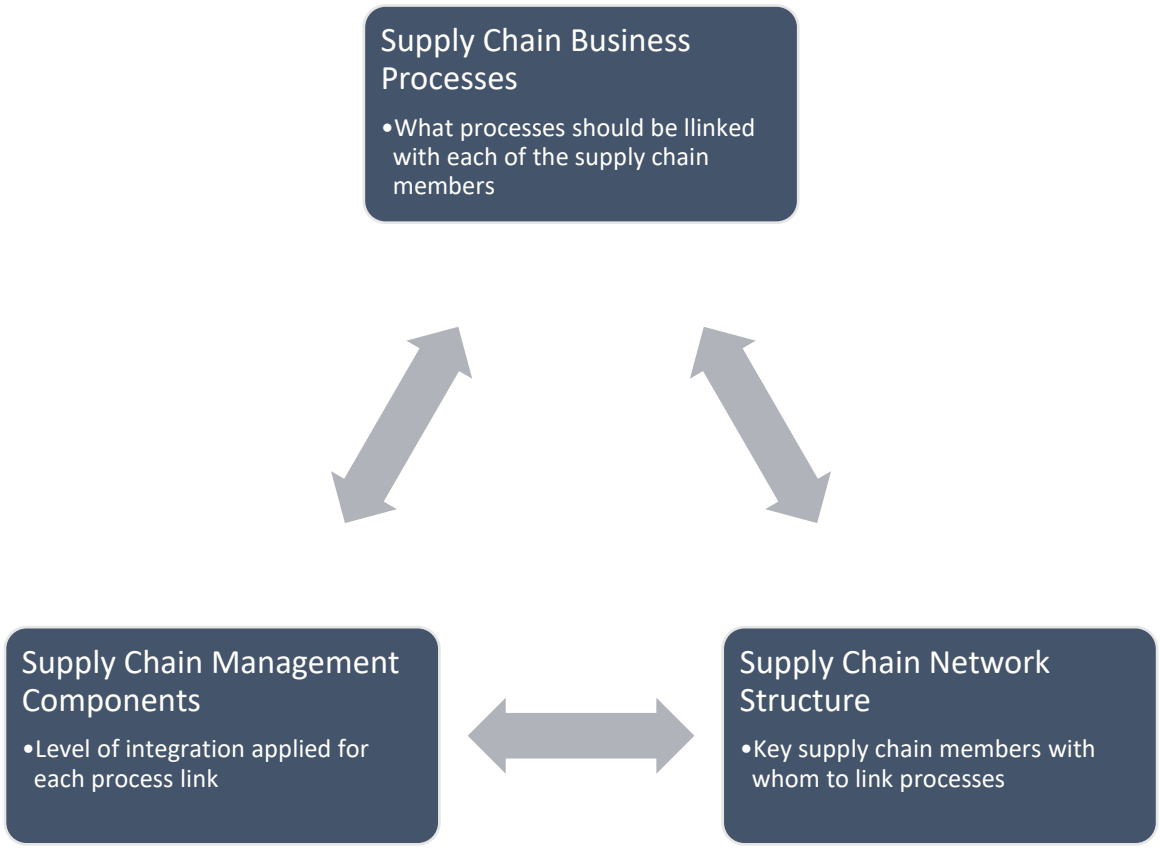

Figure 3: Three pillars of supply chain management

The Business processes decide what information should be shared with each of the member in the supply-chain. The Management Components decide what level of integration and management be applied for each process. The Network Structure decides who are the supply chain members that needs to be linked to each of the process.

Each Object in the Supply Chain Management Development cycle goes through the following three phases: Analysis, Design and Implementation. Initially the object model is very abstract and goes through the iterations of Analysis and design phases. Here, the emphasis is on solving the business problem in question (front-end problems) rather than the technicalities (back-end) of the problem. As the model keeps getting clearer, the concentration of the problem is slowly shifted towards the System Design by the systems developer who generates code and creates database access routines.

Unified Modelling Language is used for developing all the system/object models to maintain uniformity of data flow ad better understanding of the architecture throughout the development.

In the system architecture, they have proposed three components. They are: System Portal, Object Repository and Database. These are the features that we are going to develop for our application. The functions of each of these blocks are given in Figure 4

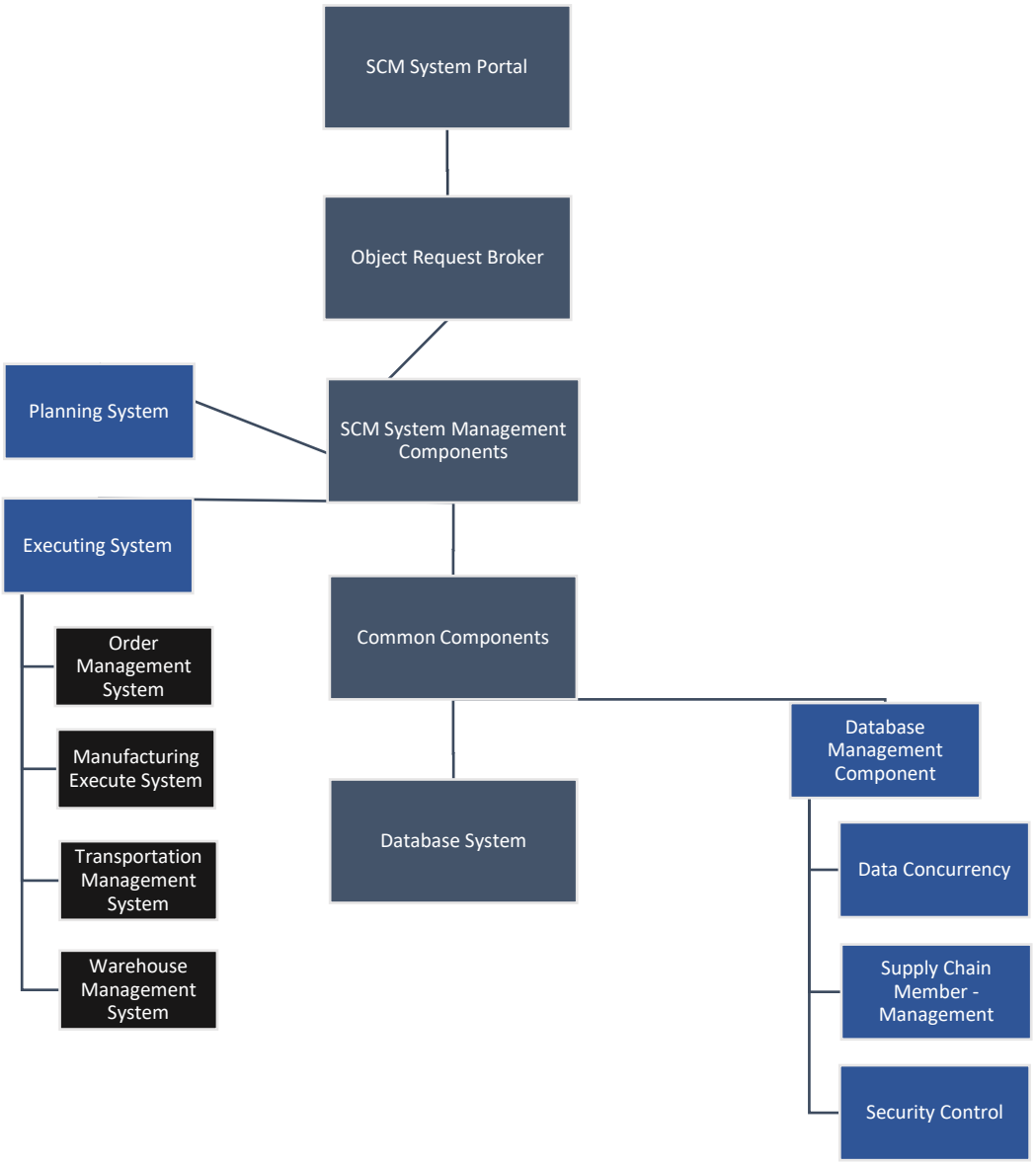

Figure 4: Application Architecture of Supply Chain Management

2.2.1 Supply Chain Management Systems Portal:

This portal acts as an interface between the users and the environment. It has two main functions: Taking inputs from the users and external sources and make sure the system is processing the information and data as intended. Providing outputs (data & information) when another system or a user request it.

In the Systems Portal design, the queries that are to be addressed are in Figure 5

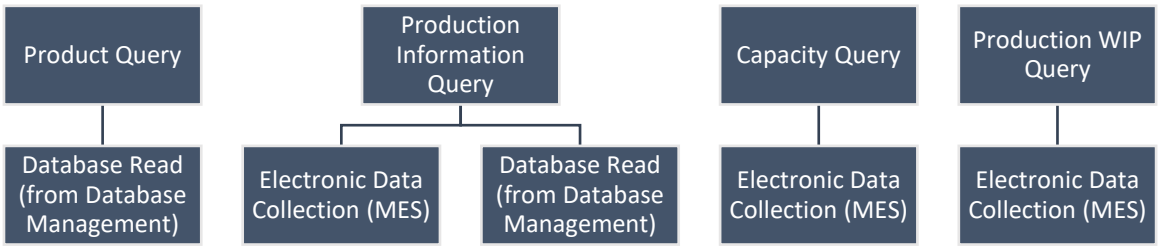

Figure 5: APIs/Applications for each query in SCM Systems Portal

2.2.1.1 Product Query:

This tool is used to get all the required details/specifications regarding the product. It can also be used to get the production information.

2.2.1.2 Production Information Query:

Users can get information regarding the supply, production plans and scheduling from this tool. They can make the production planning from their side accordingly making the business operations smoother.

2.2.1.3 Capacity Query:

This tool can be used to find out the capacity available on the production side at any point of time. Based on the results, the business parties will know if they can place the order with the company.

2.2.1.4 WIP Status Query:

This tool can be used to find out the status of the items in the factory based on the batch number. They can see the status of different stages of production inside the factory.

2.2.1.5 Advanced Planning and Scheduling (APS):

Using the queries, we can make Advanced Planning and Scheduling (APS) algorithms according to the query responses. In APS, there are two main components. They are: Capacity Planning and Production Planning.

Capacity Planning:

It is a constraint-based planning. Production planning is directly affected by capacity planning. APS makes calls for information both from internal and external sources to get the capacity planning information.

Production Planning:

After the capacity planning data has been retrieved, APS requests data regarding WIP status and information regarding raw materials, their availability etc., and make the production planning accordingly which is shown in Figure 6

2.2.1.6 Manufacturing Execution System (MES):

Manufacturing Execution System (MES) is used to control the manufacturing process and factory status, to be controlled by the capacity and production planning. It has the following five functions:

Dispatching:

This tool is used to know when the product will enter into the system and must wait in queue, dispatching tool uses the information from capacity and production planning and dispatch the products accordingly to minimize the wait timings.

Tracking:

When a product starts getting worked on in the station, tracking tool notes down the time it started and when the work on the product is finished at the end of the system.

Material Management:

This tool is used to manage the resources of manufacturing. It is used to track both the raw materials and the semi-finished goods.

Electronic Data Collection (EDC):

This tool is used to record the status of all the objects in the system. It could provide production planning to plan the production schedule

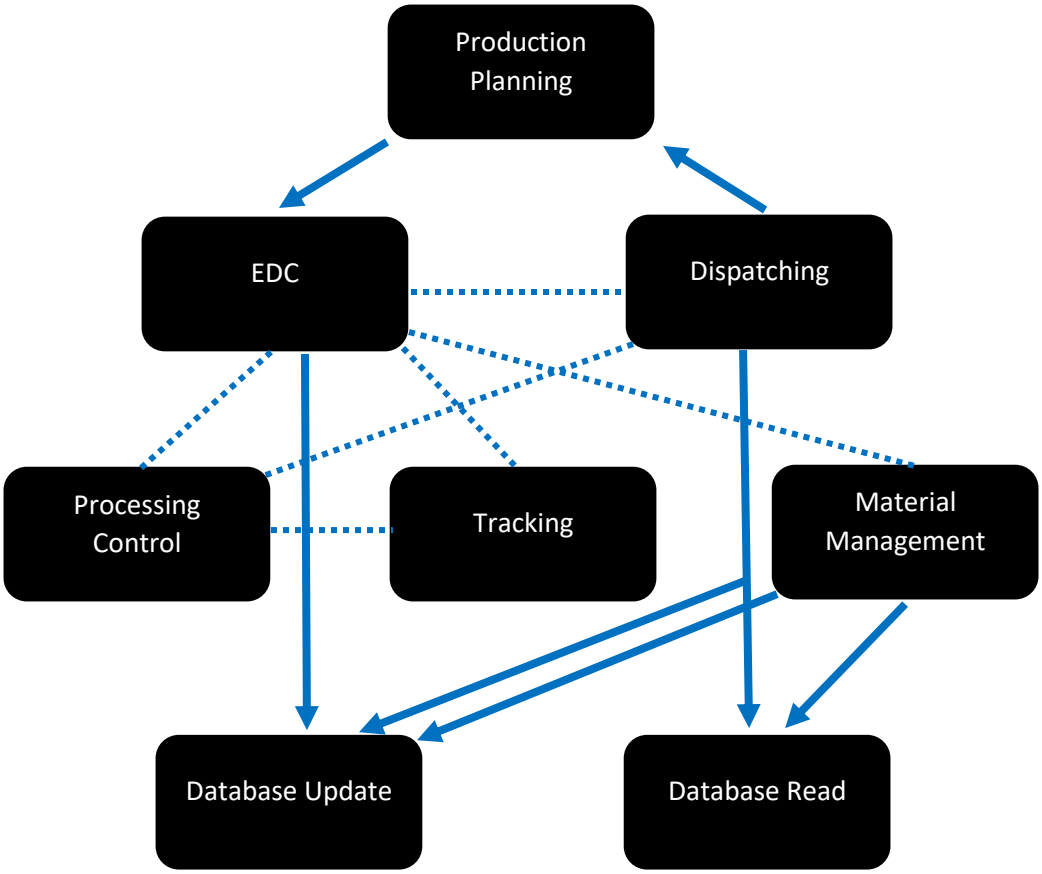

Figure 6: Applications/Programs and their dependencies for Production Planning

2.2.2 Supply Chain Management Object Repository:

This is the core of the Supply Chain Management System. All the processes, all the functions necessary for executing the operations, the logic required for validating the transactions, the required APIs are in this repository.

2.2.3 Database:

This is the storage unit for the data in the blockchain including the previous transactions, relevant files, documents, media files etc. It can store data in various data formats. It has the information regarding supply chain network member data, product ad production information of the entire system.

From the abstract supply chain management model and its components given in the paper, we're going to build tools satisfying the requirement of each block in the model. The model is given in Figure 7

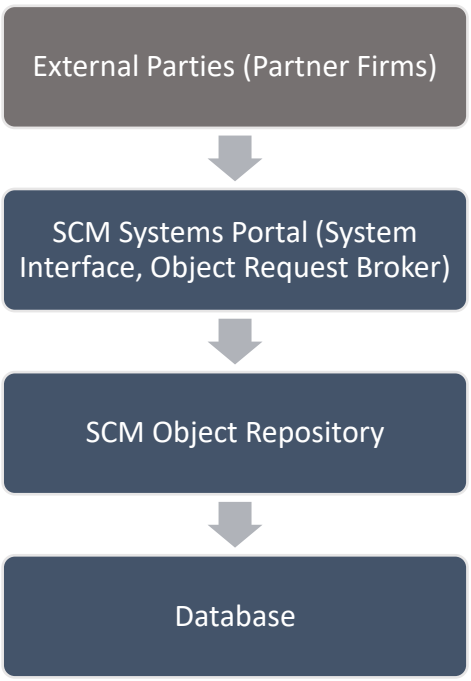

Figure 7: Database as the primary storage unit of all the transactions

2.3 Architecture of The Blockchain Environment:

The architecture/stack of the blockchain consists of the following five layers. The following architecture of Web 3.0 can be compared with the architecture of the traditional Web 2.0.

Web 3.0 is capable of "machine-machine" interactions and dynamic applications which allow for a web without human interference [6]. In Web 3.0, computers can interpret the commands given by humans and able to execute the commands accordingly.

2.3.1 Layer 0 (Consensus Layer):

In this layer, the format of the ledger is described [7]. It is open to everyone. There's a consensus function that can be used. Therefore, anyone can use this function to determine if the block/transaction they're trying to add agrees with the consensus ledger. This layer should allow new blocks to be added to the network.

2.3.2 Layer 1 (Economic/Mining Layer):

This layer is responsible for incentivizing the parties/nodes to constantly maintain the consensus in the network and accept/reject new blocks/transactions into the blockchain.

2.3.3 Layer 2 (Blockchain/Off-chain services):

This layer manages the content/transactional data inside the blocks. It is responsible for creating blocks according to the specified format filling all the required details like: Timestamp, Smart Contract, Name registry, API (Decentralised Oracle). It also manages the links from blocks to outside blockchain like: File Systems, Messaging etc. It validates if a specific link in the block is directed properly before adding that data to the block as the data cannot be changed once the block is created.

2.3.4 Layer 3 (Interop/Transaction Layer):

This layer is responsible for conducting the transactions. It exchanges the data between the blocks, transfers the value between accounts, prompts the layers above it that the transaction has either passed/failed. This layer is responsible for the actual transfer of value (for example, crypto-currency in bitcoin) in the blockchain.

2.3.5 Layer 4 (Browsers):

Not all browsers can run a blockchain application. There are some requirements for running the same [8]. The ones that can run the applications are called Dapp (Distributed Application) Browsers. These browsers have an in-built wallet to store the different cryptocurrencies available in the market and allow the users to make transactions. The browser acts like a wallet and a marketplace for using them.

2.3.6 Layer 5 (dApps):

Dapps (Distributed Applications) layer (like Applications layer in web 2.0 architecture) has the blockchain applications. These are the different services, websites that provide value to the consumers. They include the UI/UX, Application Architecture, JavaScript etc. of the application. Each time the application needs to communicate with the blockchain, it starts talking to the layers beneath it. It is shown in Figure 8

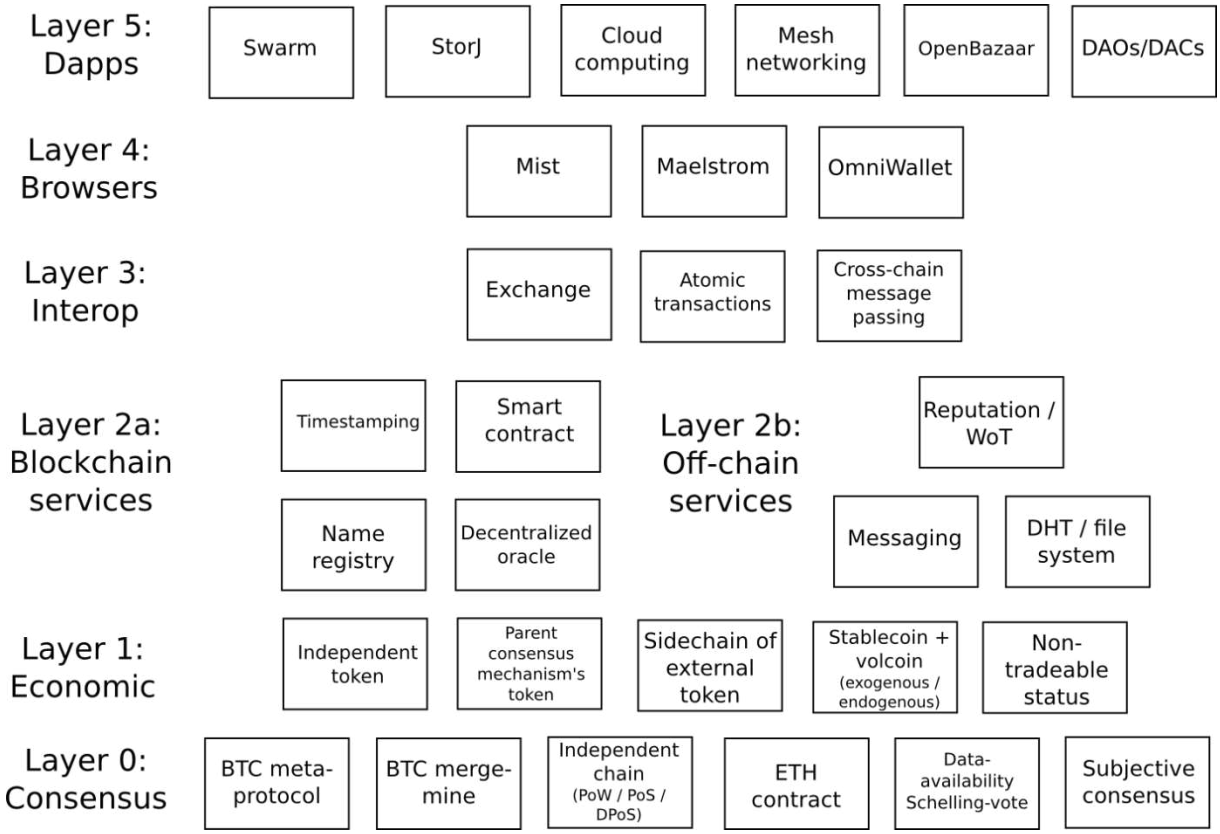

Figure 8: Layers in the Blockchain Stack (Also called as Web 3.0)

3. Blockchain Technology

A blockchain in its simplest meaning is a glorified and more complicated form of a traditional linked list. Each transaction is stored in the chain as a block [9]. The chain keeps increasing as the transactions are conducted and the blocks keeps getting added. Each block in the chain is connected to the previous by a cryptographic hash key. The hash key in the present block directs to the next block in the chain. All the blocks in the chain are encrypted. The basic components of a block are: A hash key of the previous block, a timestamp, and the transactional information.

So, a blockchain is a set of computers connected via a mesh network with no master/central server but instead connected to each other. These computers define for themselves how a state is achieved and develop some rules/ constraints to adhere to for attaining certain state and agree upon the state of each node/computer in the network accordingly.

One of the main security features of blockchain technology is the inability to change the data of the transactions that have already happened. If someone tries getting into the network and manipulates a block in the chain, the block that they have modified generates a new hash key because the hash key is generated from the data in the block. Therefore, the cryptographic hash of the block changes disconnecting it from all the blocks after it. Therefore, the chain is broken. Also, when the node where the data has been modified tries to validate the change that has been performed with the rest of the nodes in the network, a minimum of 50 percent of the network needs to agree with the change. In this case, if the person is trying to manipulate the chain, the node he's accessing needs to have computing power more than or equal to all the rest of the nodes combined in the network which is almost impossible in most of the cases. For example, if someone wants to manipulate a block in Bitcoin (Most famous blockchain in the world), the person needs to have all the computing power of the world's fastest 500 supercomputers.

This makes the blockchain technology one of the most secure technologies in the world right now.

Another feature of the blockchain is its Smart Contract method of validating transactions. Blockchain technology improves speed, efficiency and transparency in validating transactions. As discussed in [10], we can see that blockchains uses a single version of truth which every

node agrees upon before the transaction is validated. Although, the main issue comes with the anonymity required with some types of blockchains like Bitcoin where the transactions need to be anonymous to the public as it poses a threat to their security. This has been discussed briefly in [Research Paper: Introduction to Security and Privacy on the Blockchain] where the different scenarios are described in detail. Later, they have proposed the research that needs to be done to rectify the existing loopholes in the existing blockchain technology. You can see the status of each of the tools that serve each of the mentioned applications in Table 1.

	Web 2.0	**Web 3.0 (dApps)**	**Status**
Scalable Computation	Amazon EC2	Ethereum, Truebit	In Progress
File Storage	Amazon S3	IPFS/Filecoin, Storj	In Progress
External Data	3rd Party APIs	Oracle (Augur)	In Progress
Monetisation	Ads, Selling goods	Token Model	Ready
Payments	Credit Cards, Paypal	Ethereum, Bitcoin etc.,	Ready

Table 1: Status of development of different tools in Blockchain Stack

But, the most interesting and important feature of all that makes blockchain more relevant than any other technology is the cost of infrastructure. There's no single authority who can charge you for using the resources. It's all open source. Anybody can setup a node, join the network, download a copy of the blockchain and start making transactions. A Blockchain is a decentralised network and it looks like the network shown in the Fig 9

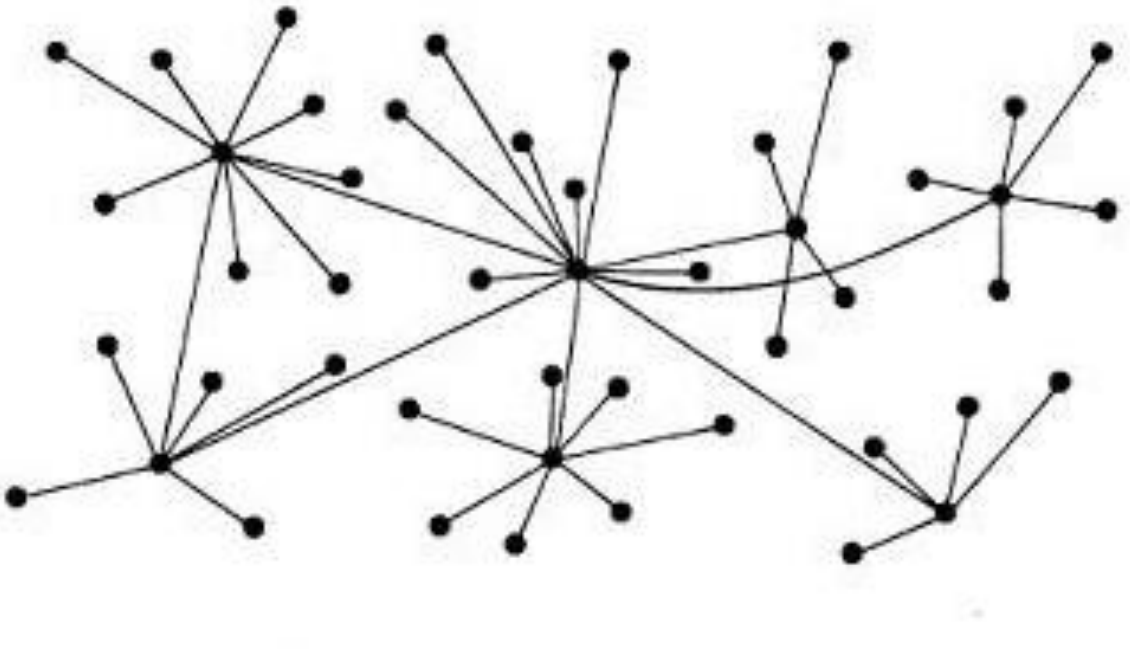

Figure 9: A visual representation of the decentralised network

4. Ethereum Implementation

The first full-fledged implementation of the Blockchain technology is Bitcoin (a cryptocurrency). Bitcoin is built only for transferring money. Its blocks, smart contracts and the concept of proof-of-work to validate the transactions are all hard-coded. The whole system is programmed to function as a wallet and nothing else. But Bitcoin (a cryptocurrency) is only a single application of how blockchain can be used. There are hundreds of use cases for blockchain applications. Therefore, there is a need for a programmable blockchain to modify the parameters according to the use case.

Ether is the cryptocurrency that is used for the operation of Ethereum. It also provides a public/private distributed ledger for transactions. Ether is used to pay for "GAS", which is the computational unit in transactions and state transitions.

Ethereum uses accounts and balances by a method called state transitions. They do not rely upon unspent transaction outputs (UTXOs). State tells the current balances in all the accounts and has more information. This state is not stored in blockchain. Instead it is stored in a Merkle tree. Ether accounts are pseudonymous, which means unlike traditional bank accounts that are linked to an individual, these accounts are linked to a specific address. Whoever has that address is the owner of the account. So, this address must be secured to protect the account.

Ethereum uses "Ethash" algorithm for calculating Proof-of-State opposed to Proof-of-Work for Bitcoin.

Ethereum smart contracts are written in Solidity, Serpent, LLL and Mutan. Smart Contracts are high-level abstractions like the Terms & Conditions of a Legal agreement. These are programmed and then deployed into the blockchain to validate the transactions.

Public Smart Contracts can be used as proof of functionality and ownership.

5. Software Requirements

5.1 Windows/Linux VM:

VMWare Workstation or any other equivalent software should be used for running a copy of Windows/Linux. I'm using Windows 10 64-bit on VMWare Workstation 14.

5.2 Java JDK: Java Applications Pre-requisite

Java JDK environment is necessary for developing and running Java applications. It has many tools including JRE (Java Runtime Environment), JavaC (compiler for Java) and other tools which are used for developing and running Java applications [11]. This package is ideal as it makes Java programs Operating System independent which makes the code/applications very portable.

5.3 NodeJS: JavaScript Runtime Environment

This package is necessary to execute JavaScript code on the server side which makes the webpages dynamic because there's no need to refresh the page to fetch the results from the server [12]. This is very useful for the applications being developed. It also includes "npm", a tool very useful for installing packages.

5.4 Go (Geth): Compiler Language

Go is developed by Google with integrating features from many programming languages [13]. Geth is the blockchain implementation using Go language. It is the one of the most used compilers of blockchain [14].

5.5 Python: Dependencies of Geth and Truffles

Python is another programming language that features dynamic type system and automatic memory management [15]. It is object-oriented, imperative, functional and procedural. It has dependencies with truffles framework.

5.6 Solidity: Programming Language

Solidity is a programming language used to write contracts [16]. Blockchain uses contracts to validate if the new blocks being added satisfy the terms of the contract. Solidity is used for implementing smart contracts on various blockchain platforms including Ethereum.

5.7 Truffle: Framework

Truffle is a development framework for Ethereum [17]. It has many features including Built-in Smart contract compilation, linking, deployment, automated contract testing, scriptable deployment and migrations, network management etc. It can be installed as an “npm” package using console.

5.8 TestRPC (Ganache)

TestRPC is used for testing and development of Ethereum projects [18]. It acts as a client and uses ethereumjs. This is now replaced with Ganache [19] which has all the features of testRPC and provides a better experience with the GUI.

5.9 Web3: Thin Client for connecting to the Ethereum Network

Web3.js is a thin-client that we use to run JavaScript and interact with a local or remote Ethereum node [20].

5.10 Abi-decoder: For decryption of the Blocks in the network

This is a library used for decrypting the blocks which contain the details about data parameters and events from Ethereum transactions [21].

5.11 Microsoft Visual Studio & Code: For developing the Application

We would be needing the Windows Forms Application package in the Microsoft Visual Studio Packages. The backend would be connected using JSON.

6. Setting-up the Development Environment

Running an Ethereum node requires some setup and system variables need to be modified. Therefore, running a new copy of Operating System on Virtual Machine is suggested.

For this project, I am running a copy of Windows 10 on VMWare Workstation 14. Allocate a minimum recommended storage of 100 GB as the Ethereum setup and other dependencies require space. Minimum amount of RAM to be allotted is around 4 GB. Anything less than that might result in crashing the node when running the blockchain.

The latest package of Java needs to be installed in the system. The Path needs to be added to the environment variables. For more information on how to setup Java, refer to [22]

Install NodeJS package on the system. It is required to install the rest of the packages. Make sure the folder you're installing it in has no read/write permission issues. All the installation instructions for NodeJS can be found on [23]

After installing both packages, open Windows Powershell/ Command Prompt as administrator.

Check if both Java and NodeJS are installed by running the command "Java" and "node". If Java is installed properly, you can expect a list of suggested commands that can be used after the word Java in the console. If you get an error message like "this command cannot be found", please re-install it and set the environment variables correctly. If NodeJS is installed properly, you enter the node as you execute the command "node". It starts with a ">". If not, the installation was not properly not done and you need to re-install it.

To exit the "node", please press "Ctrl + C" two times to get out of the node and into the normal mode.

Now you need to install the packages for Ethereum using "npm" package installer.

The first package to install is the "ethereumjs-testrpc", which creates ten sample Ethereum accounts that you can use to run the node privately and troubleshoot before deploying into the

actual network where you will be dealing with actual transactions. ("-g" for installing it globally to all users)

npm install -g ethereumjs-testrpc

If there are any dependencies asking for additional packages to be installed, please make sure you install them before proceeding to avoid unexpected behaviour during running the blockchain in the network.

The warnings I received were to install "webpack@4.0.0" during the installation of testrpc. You might also come across the error saying ethereumjs-testrpc has been deprecated. You have to use ganache-cli package instead of it. The reason for this error is that testrpc has combined in to the truffles framework. Which means no support will be given to testrpc if any bugs have been detected. Therefore, we would be installing ganache as well to avoid any errors later.

"Ganache" package installation is a three-step process. There are three different components of Ganache that needs to be installed to utilize the full capabilities provided by it. The first is Ganache-cli.

Ganache-CLI is the command line interface. It has all the commands required to use Ganache. It can be installed by the following command:

npm install -g ganache-cli

After this, please visit the page [24] for downloading ganache-gui application. This application allows you to see a lot of information regarding the blockchain without the need of running the commands for each detail like the blocks in the blockchain, all the transactions details, log of changes, all the accounts and their balances etc.

Then you need to install ganache core as some of the other applications need that as their dependency. The following command needs to be used for installing it:

npm install -g ganache-core

Now you must install Truffle framework in the system. Truffle is a multi-purpose tool used in blockchain development. The following command is used to install the truffle:

npm install -g truffle

Now, we must install Solidity compiler which compiles the solidity contracts that we have written for the blockchain. We can also use an online solidity compiler like [25], but a local compiler is more flexible.

npm install -g solc

You must install the web client we're going to use, which is "Web3". Please use the following command:

npm install -g web3

To decrypt the blockchain to read the blocks and access the information regarding the smart contracts, we need a decoder. This command installs "abi-decoder" which does the job:

npm install -g abi-decoder

After installing all the above components, the ethereum node is ready to run the blockchain.

To write the smart contracts using solidity, I used Visual Studio Code and installed the Solidity extensions. It is a very flexible application which has an in-built console window to compile the contracts and trigger the blockchain.

Once the contract is compiled without any errors, all the commands that we're going to use are to be performed in the "Ethereum node" and not in the normal command mode. To enter the node, please type "node" in the command prompt to enter it.

7. Testing the Environment

For testing the integrity of the environment, a voting application is designed. If this runs successfully, the environment is setup properly. Please follow these steps to execute the program:

(These commands should be typed in the same order in the command prompt)

C:Users/Venk…./$ node

> Web3 = require('Web3')

> Web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:7545"));

(The localhost address varies with computer. Please check in ganache-gui in the settings section to find out the localhost address ganache is running)

> web3.eth.accounts

(This command returns all the account details running in the blockchain)

(The following commands are to compile the smart contract)

> code = fs.readFileSync('vote.sol').toString()

> solc = require('solc')

> compiledCode = solc.compile(code)

(If these steps are run without any errors, the smart contract that you have written is correct)

(The following steps are to deploy the contract in the blockchain)

> abiDefinition = JSON.parse(compiledCode.contracts[':Vote'].interface)

> VotingContract = web3.eth.contract(abiDefinition)

> bytecode = compiledCode.contracts[':Vote'].byteCode

> deployedContract = VotingContract.new(['CandA','CandB','CandC'],{data: bytecode, from: web3.eth.accounts[0], gas: 20})

> deployedContract.address

> contractInstance = VotingContract.at(deployedContract.address)

8. Software Development Process

The members involved in this project is only me. Therefore, I have used the principles of Agile for One which is the modified version of Agile for Teams. In this process, I made a list of features/concepts that I'm going to develop/learn for every week. Some of them are incremental features (additions to the previous versions) and some of them are iterative (mostly new features). These are the two-week sprints that I have divided the work into:

Sprint-1 (Week 1&2):

Done the Literature Review. Formulated the Problem Statement. Read articles, research papers etc. about the topic to deepen the understanding and finding the best suitable use cases.

Sprint-2 (Week 3&4):

Evaluating the options available for setting up the environment. Learning how to setup the environment. Setting it up. Making sure the environment is intact and all the necessary tools are present for development.

Sprint-3 (Week 5&6):

Developed the application architecture. Took required training in the Solidity programming language for writing the Smart contracts. Developing smart contracts architecture to be implemented at each stage.

Sprint-4 (Week 7&8):

Developed an application (voting application). Deployed the blockchain in a private node. Developed the front end of the Supply Chain Application. API development for the backend queries of the Systems Portal.

9. Problem Statement & Development

Blockchain Application for FMCG Retail Space.

The main aim of this application is to streamline the supply chain by increasing the transparency & traceability.

This application consists of multiple dashboards, each with a different functionality serving different sets of users, all depending on the same backend network that runs on blockchain.

As seen in the literature review [**5**], we're going to build the tools according to the architecture proposed by the authors.

The architecture of the application is divided into parts based on the requirements for each of the tool being developed. They are as follows:

9.1 Supply Chain Management Systems Portal:

This tool is mainly used to get the information from the blockchain regarding the production and products.

9.1.1 Product Query:

This portal needs to have the following options. A search bar to type the product name that they want to know the information. Alternative is having a drop-down menu with all the different categories of products. After a product is selected, the product information should appear on the right. For product catalogue, another drop-down menu must be added. Once a product category is selected, the catalogue appears on the right. It is shown in Figure 10

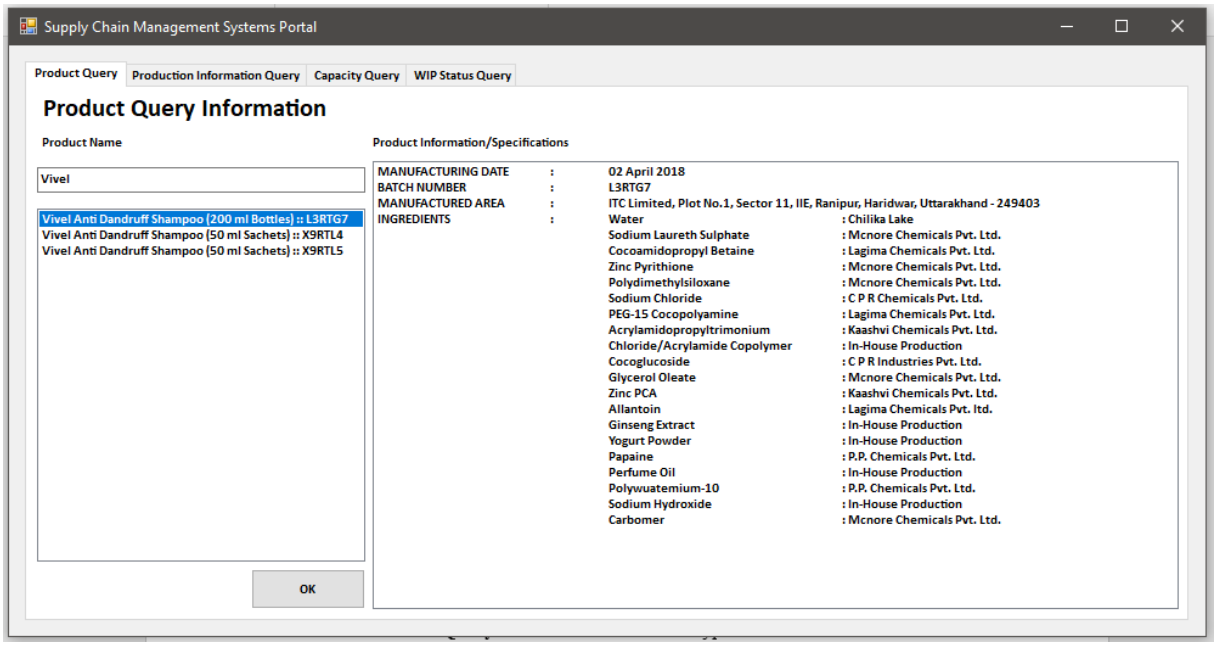

Figure 10: Product Query in SCM Systems Portal

9.1.2 Production Information Query:

A search bar where the batch number of the production needs to be typed. Once typed, all the details like what's the status, where is it right now, estimated time of completion of each stage etc. will be shown. It is shown in Figure 11

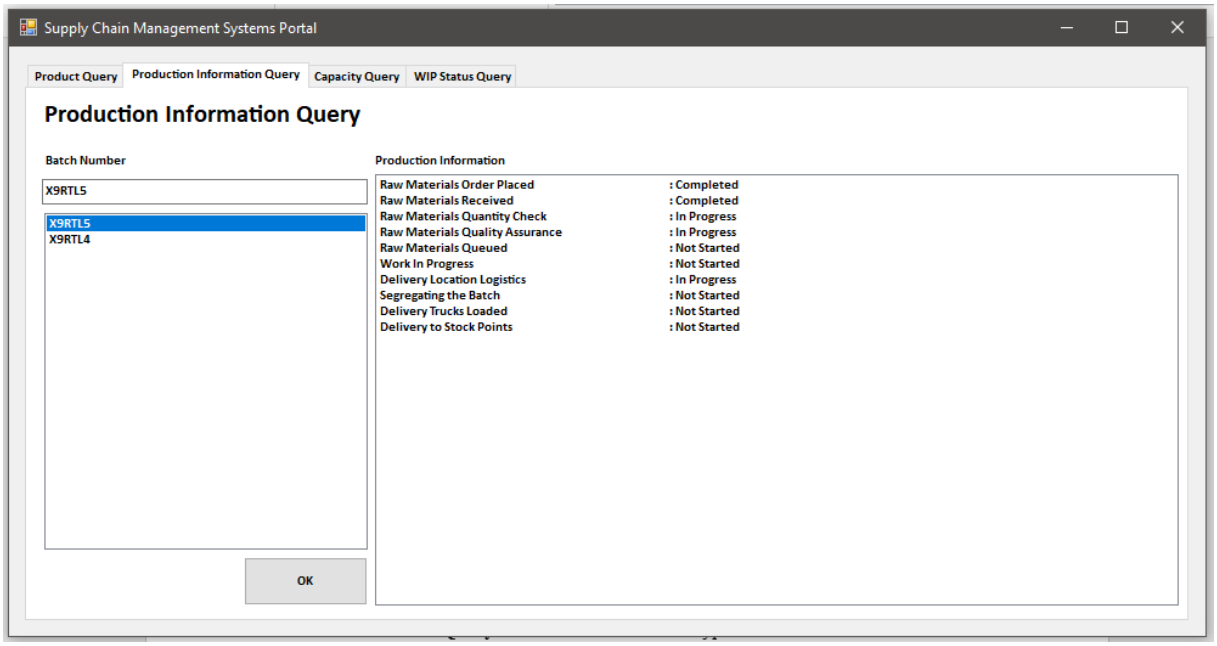

Figure 11: Production Information Query in SCM Systems Portal

9.1.3 Capacity Query:

An Upper bound is set for each stage of the production in the backend. Based on the data we get from the blockchain network about the state of each stage in the factory, we can calculate the capacity available at each stage and display it accordingly. It is shown in Figure 12

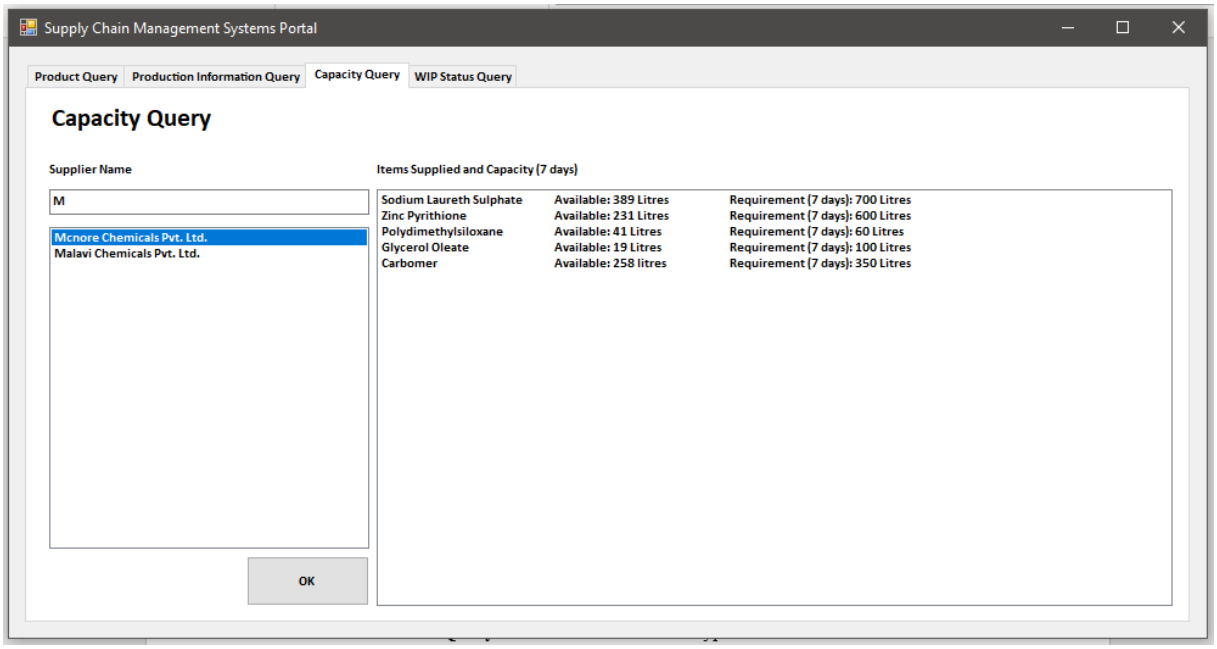

Figure 12: Capacity Query in SCM Systems Portal

9.1.4 WIP Status Query:

We need a search bar to type in the batch number. Results will be displayed on the right. If the users want to know more details of the batch number that are being worked inside the factory, they use this tool. This tool specifically gives details only regarding the factory operations and shows the status for each stage for that specific batch number. It is shown in Figure 13

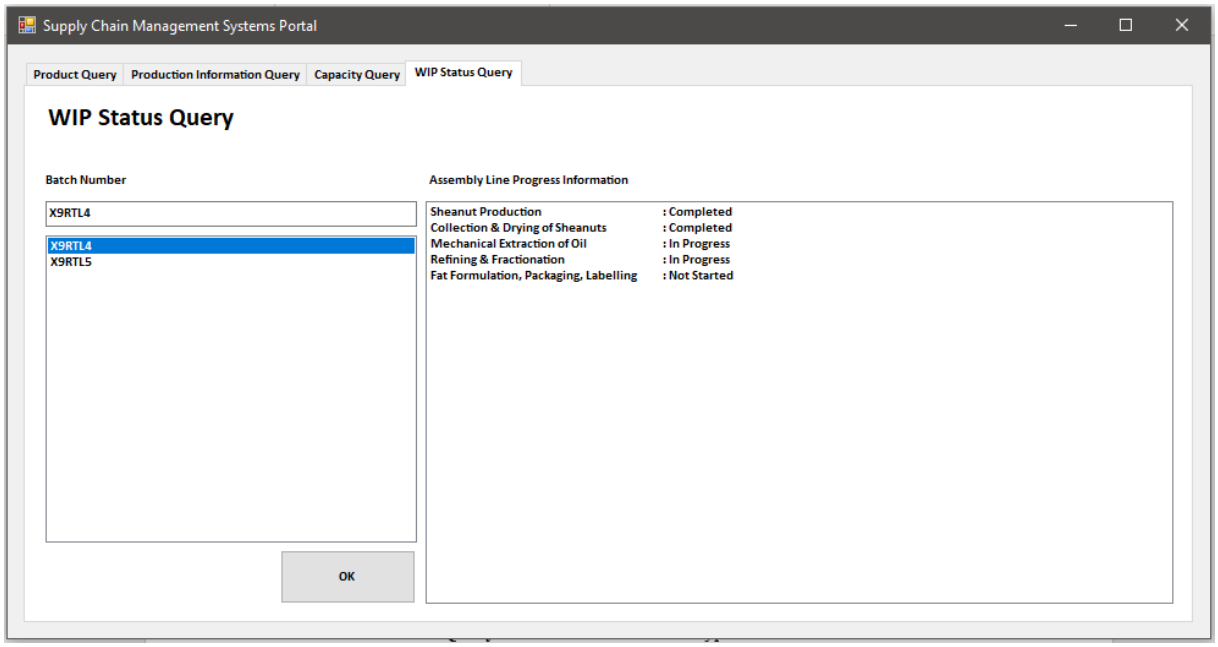

Figure 13: WIP Status Query in SCM Systems Portal

9.2 Advanced Planning and Scheduling (APS):

This tool is the core of the application as it does the calculations for the query in the Systems portal. It retrieves the data and makes the calculations and use analytics tools to extrapolate the charts. It is shown in Figure 14 how the APS queries support the SCM Systems Portal

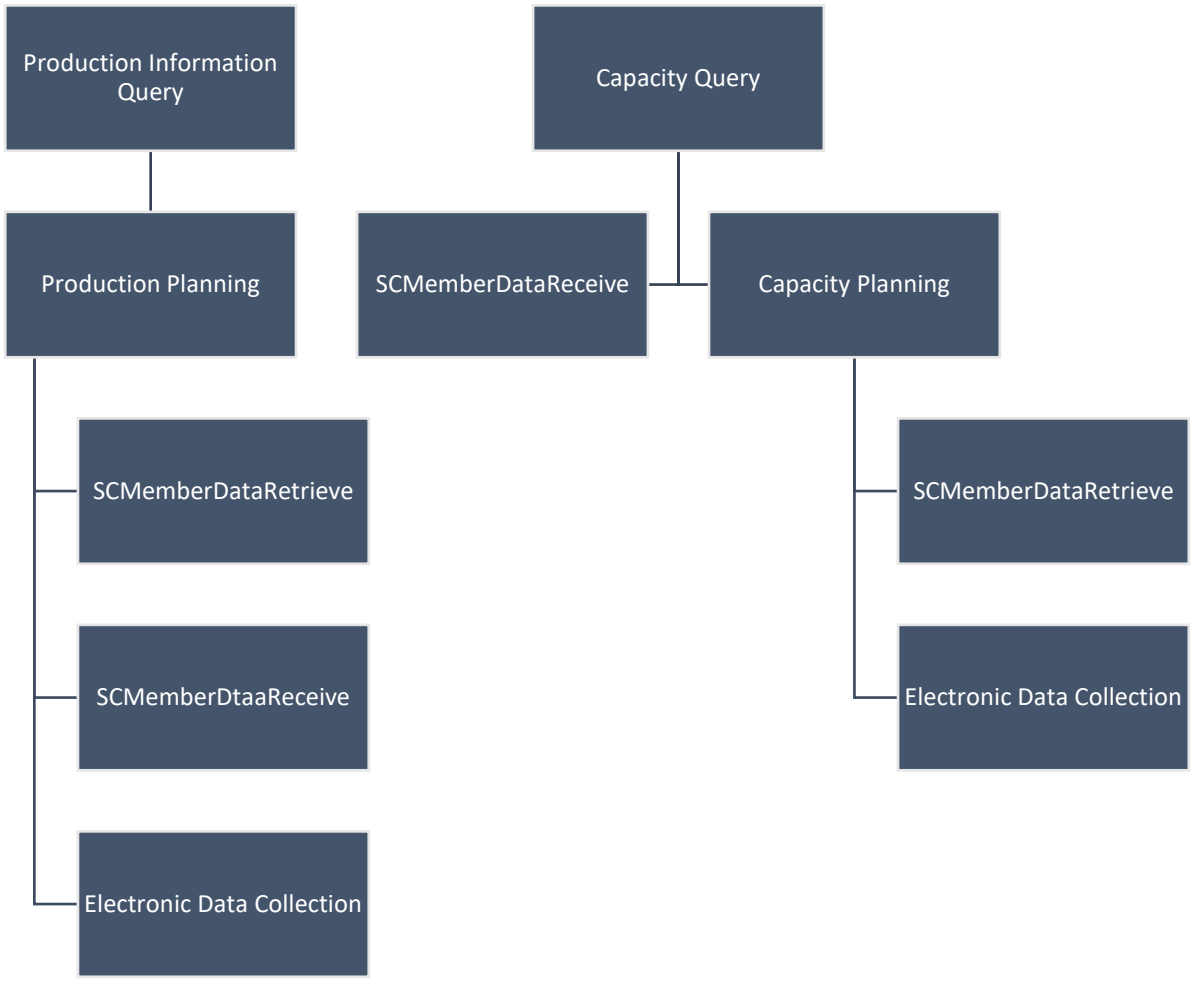

Figure 14: Applications running for the Production Information/Capacity Queries

9.2.1 Capacity Planning:

This tool is used for retrieving the information for Capacity Query. It makes calls to EDC (Electronic Data Collection) to know the status of the objects in the factory and SCMemberDataRetrieve to find out the status of the members. By collecting both the information, it does the calculation on how much capacity is available right now. It also uses Analytics tools to predict the capacity going to be available in the future.

9.2.2 Production Planning:

This tool is used for retrieving information for Production Information Query. It sends data to SCMemberDataReceive and EDC after planning the production based on the data received from Capacity Planning and SCMemberDataRetrieve. Therefore, this tool has Capacity Planning as dependency.

9.3 Manufacturing Execution System (MES)

Manufacturing Execution System is the tool which the first line workers (assembly line) workers use to record the happenings in the factory. It is the deepest layer that we're going to develop and the rest of the tools run on the information that we receive from here.

9.3.1 Dispatching:

This tool is used to record when the product is dispatched and suggesting the dispatch time to minimize the wait time based on the status of queue at the factory. It also manages how all the different raw materials are transported according to the sequence of the requirement in the factory. This tool only has a text box to type in the product id and a radio button to send the information. This happens at each stage of production to notify that the product has crossed certain stages. It is shown in Figure 15

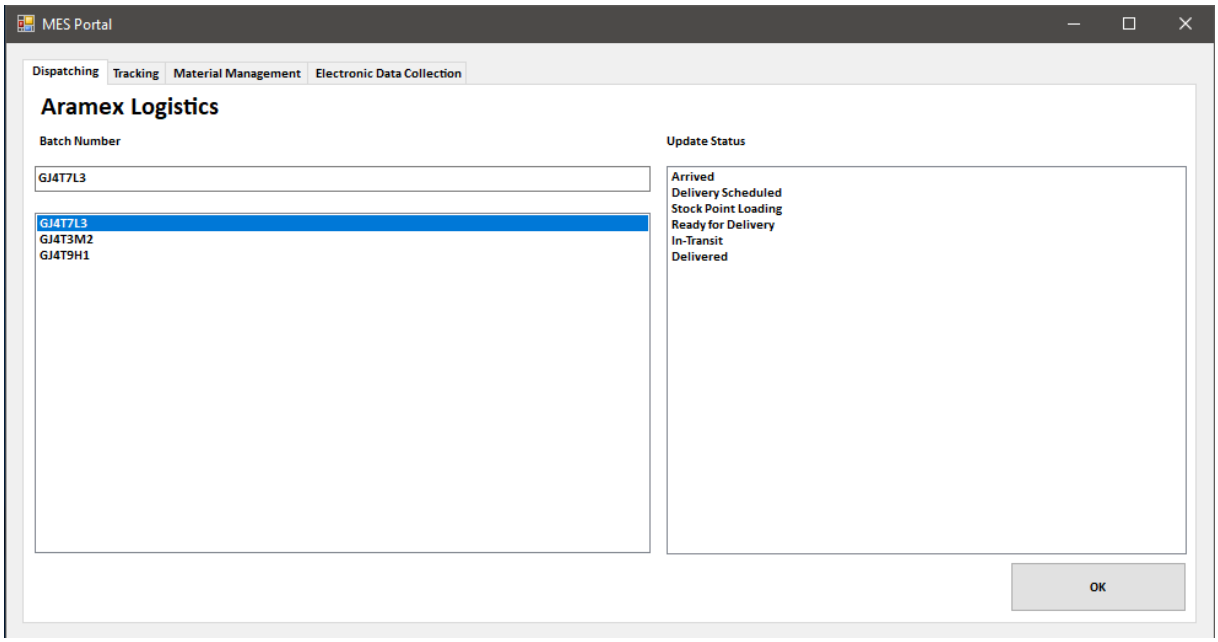

Figure 15: Dispatch Information Update in MES Portal

9.3.2 Tracking:

Tracking is the same as dispatching tool but it comes into action when the product already starts in the assembly line in the factory. From the point it enters the product station until it leaves. Therefore, this tool also needs a text box to enter the product id and a radio button. This is present at each stage of assembly line to notify it has crossed certain stages shown in Figure 16

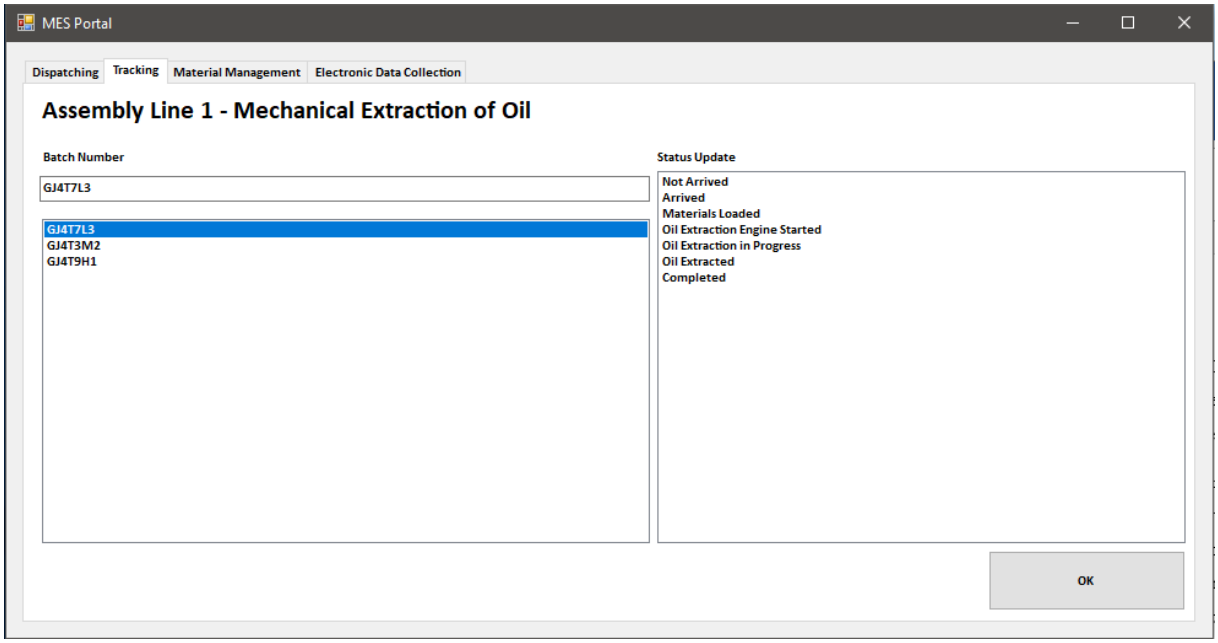

Figure 16: Tracking Information Update in MES Portal

9.3.3 Material Management:

As the tracking tool updates the status of each product, material management tool takes that data and calculates the remaining materials, semi-finished goods and alerts the dispatching tool and EDC tool to order the raw materials to ensure no delay in production shown in Figure 17

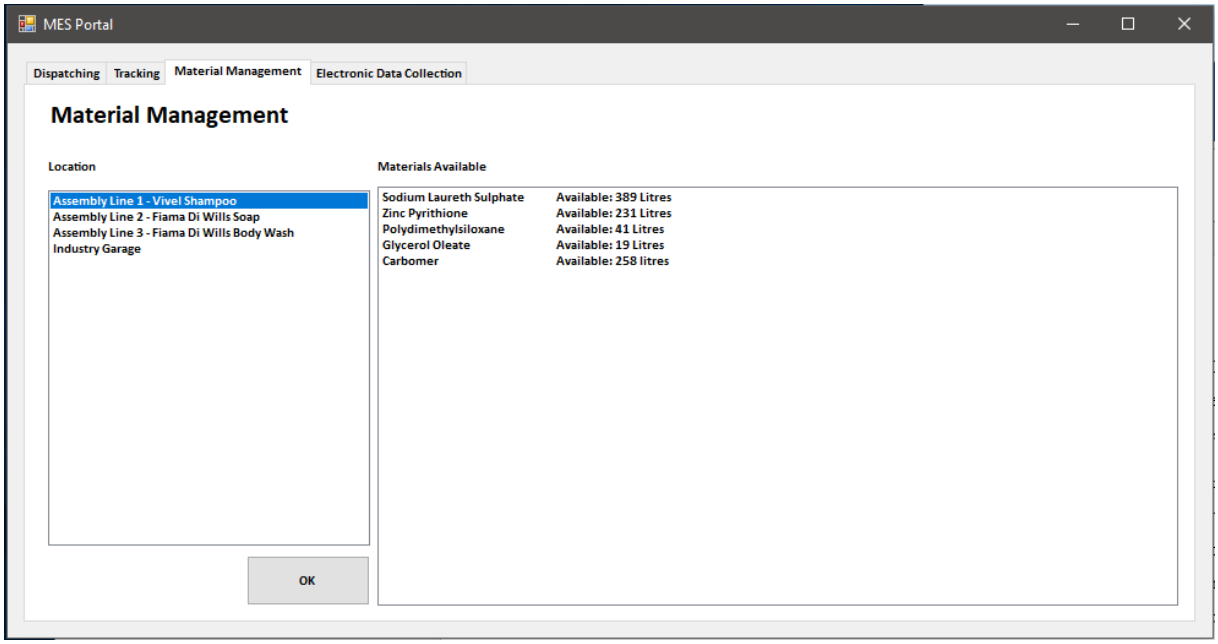

Figure 17: Material Management Information Update in MES Portal

9.3.4 Electronic Data Collection (EDC):

This tool has the status of all the objects in the factory like machines, packaging units, control systems etc., It keeps track of the status of each object and sends the data to the relevant tools when necessary to make the planning accordingly. Production Planning is also made according to the EDC. It is shown in Figure 18

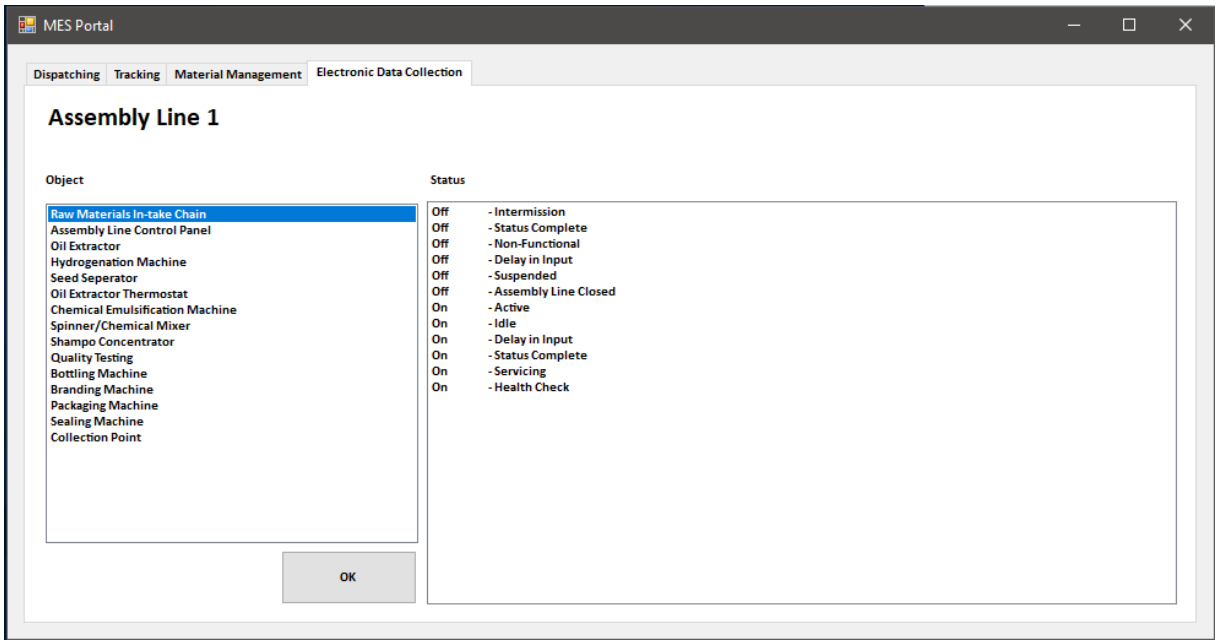

Figure 18: Electronic Data Collection Information Update in MES Portal

10. Future Scope

The Software has now been developed only for the Industry/Production area of the Supply Chain. The same method can be implemented for the Sales network that starts from the point the production is complete to the point where the customer purchases it. The backend of the application needs the development of Smart contracts for each stage. An application would be necessary at each point where the stock/products make a stop as in regional stocking points, distributors, wholesale dealers, salesmen etc.

11. Conclusion

The potential of Blockchain technology is huge. In this project, we have seen that how blockchain can be used as a reliable back-end technology to manage the entire supply chain. This becomes more and more relevant as companies start growing exponentially and quality check and efficiency keeps decreasing as they grow. Blockchain can be both a cost-effective and secure way of maintaining them both.

12. References

1. Ministry of Commerce & Industry, Government of India, “Indian FMCG Industry Analysis,” *IBEF: India Brand Equity Foundation*, 2018. [Online]. Available: https://www.ibef.org/industry/fmcg-presentation.
2. “What's In Store For India's FMCG Market?,” *Nielsen India*, 25-Nov-2014. [Online]. Available: http://www.nielsen.com/in/en/insights/reports/2014/whats-in-store-for-indias-fmcg-market.html.
3. “Indian smartphone market grew 14% in 2017, shipped 124 million units, says IDC,” *Business Today India*. [Online]. Available: https://www.businesstoday.in/technology/news/india-smartphone-market-grew-14-per-cent-in-2017-shipped-124-million-units-idc/story/270626.html. [Accessed: 14-Feb-2018].
4. “Merkle tree,” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Merkle_tree.
5. Bao-Yi Tuang and Woo-Tsong Lin, “An Architecture of Supply Chain Management Systems.”
6. WittyCookie, “What are the major differences among Web 1.0, 2.0 and 3.0?,” *WittyCookie*, 05-Jun-2012. [Online]. Available: https://wittycookie.wordpress.com/2012/06/04/what-are-the-major-differences-among-web-1-0-2-0-and-3-0/.
7. D. Xiao, “The Four Layers of the Blockchain – David Xiao – Medium,” *Medium*, 22-Jun-2016. [Online]. Available: https://medium.com/@coriacetic/the-four-layers-of-the-blockchain-dc1376efa10f.
8. T. Wallet, “Fully Functioning DApp Browser for Mobile available in Trust Wallet,” *Medium*, 27-Feb-2018. [Online]. Available: https://medium.com/@trustwallet/fully-featured-dapp-browser-for-mobile-4d062a0c66cb.
9. “Blockchain,” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Blockchain.
10. J. Scutz, “A Guide to Understanding the Blockchain Architecture,” *The Market Mogul*, 01-Feb-2018. [Online]. Available: https://themarketmogul.com/blockchain-architecture-guide/.
11. “Java (programming language),” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Java_(programming_language).
12. “Node js,” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Node_js.
13. “Go (programming language),” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Go_(programming_language)#Language_tools.
14. “Geth,” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Geth.
15. “Python (programming language),” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Python_(programming_language).
16. “Solidity,” *Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/Solidity.
17. Truffle Suite, “Your Ethereum Swiss Army Knife,” *Truffle Suite*. [Online]. Available: http://truffleframework.com/.
18. “ethereumjs-testrpc,” *npm*. [Online]. Available: https://www.npmjs.com/package/ethereumjs-testrpc.
19. Truffle Suite, “Ganache,” *Truffle Suite*. [Online]. Available: http://truffleframework.com/ganache/.
20. “web3.js - Ethereum JavaScript API¶,” *web3.js - Ethereum JavaScript API - web3.js 1.0.0 documentation*. [Online]. Available: https://web3js.readthedocs.io/en/1.0/.

21. ConsenSys, “ConsenSys/abi-decoder,” *GitHub*. [Online]. Available: https://github.com/ConsenSys/abi-decoder.
22. “Environment variables for java installation,” *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/1672281/environment-variables-for-java-installation.
23. “How to Install Node.js® and NPM on Windows,” *Treehouse Blog*, 29-Dec-2015. [Online]. Available: http://blog.teamtreehouse.com/install-node-js-npm-windows.
24. Truffle Suite, “Ganache,” *Truffle Suite*. [Online]. Available: http://truffleframework.com/ganache/.
25. *Remix - Solidity IDE*. [Online]. Available: https://remix.ethereum.org/#optimize=false&version=soljson-v0.4.23 commit.124ca40d.js.