|  |  |
| --- | --- |
| A PROJECT   REPORT | |
| On Implementation of Blockchain and  Hyperledger  (for proof-of-concept)  As  SUMMER-TRAINING  PROJECT INTERNSHIP  Under the guidance of  Dr.Swapnil Shrivastva  (Principal Technical Officer,  CDAC, Knowledge Park,  Bengaluru -560038 ) |
| July 12, 2018  Been Submitted by:  Akshay Sharma, Sushant Sharma, (NIT Hamirpur) |



CANDIDATE’S DECLARATION

I hereby certify that the work being presented in the report entitled as ―6th Semester Summer training, in the partial fulfillment of the requirements for the award of degree of Bachelor of technology and submitted in the department of Computer Science and Engineering of the National Institute of Technology, Hamirpur, is an authentic record of my own work carried out during the period 1ST June 2017 – 17TH July 2017 under the supervision of Dr. Swapnil Shrivastva, Principal Technical Officer, CDAC Bengaluru.

The matter presented in the report has not been submitted by me for the award of any degree of this or any other institute /University.

AKSHAY SHARMA

15537

3rd Year -CSE (NIT Hamirpur)

sharmakshay27@gmail.com

CANDIDATE’S DECLARATION

I hereby certify that the work being presented in the report entitled as ―6th Semester Summer training, in the partial fulfillment of the requirements for the award of degree of Bachelor of technology and submitted in the department of Computer Science and Engineering of the National Institute of Technology, Hamirpur, is an authentic record of my own work carried out during the period 1ST June 2017 – 17TH July 2017 under the supervision of Dr. Swapnil Shrivastva, Principal Technical Officer, CDAC Bengaluru.

The matter presented in the report has not been submitted by me for the award of any degree of this or any other institute /University.

SUSHANT SHARMA

15MI541

3rd Year – Integrated CSE (NIT Hamirpur)

sharma.sushant813@gmail.com

ACKNOWLEDGEMENT

It is my pleasure to be indebted to number of people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior, and acts during the course of study.

I would like to express our sincere gratitude and heartfelt thanks to Dr. Swapnil Shrivastava, Principal Technical Officer, CDAC Bengaluru for their unflinching support and guidance, valuable suggestions and expert advice. Their words of wisdom and expertise in subject matter were of immense help throughout the duration of this project.

I also take the opportunity to thank [Dr. Kamlesh Dutta](http://nith.ac.in/~kd/) and all the faculty of Department of Computer Science and Engineering, NIT HAMIRPUR for helping us by providing necessary knowledge base and resources.

I would like to thank my parents for their moral support and my friends with whom I shared my day-to-day experience and received lots of suggestions that improved my quality of work.

Table of Contents

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Title** | **Page no.** |
| 1. | Abstract | 6 |
| 2. | Introduction | 7 |
| 3. | Concepts learning | 9 |
| 4. | Platform for Implementation | 20 |
| 5. | Project Domain | 29 |
| 6. | Code Implementation | 33 |
| 7. | Code Execution | 42 |
| 8. | Conclusion | 60 |
| 9. | Future Scope | 61 |
| 10. | References | 63 |

Chapter 1  
Abstract

Blockchain is a chain of blocks that contains information. This technique originally was developed in 1991 by a group of researchers and was originally used to timestamp digital documents so it isn’t possible to backdate then or to taper with them. However, it went mostly unused until it was used by Satoshi Nakamoto in 2009 to create digital currency Bitcoin. It was first of its kind to be free from double spending problem without need of any trusted authority or central server. Blockchain is a distributed ledger that completely open to anyone. It has an interesting property that once some data has been recorded in blockchain it becomes it becomes very difficult to subject it to unintentional or intentional change. It is basically because it makes use of hashing and is distributed. There isn’t any central entity managing the while network of blocks. Instead of using a central entity to manage, blockchain make use of P2P network and everyone is allowed to join the network. When someone joins the network, he gets full copy of the blockchain and in future node can use to verify that everything is still in order. Blockchain has its social perception closely related to bitcoin and other cryptocurrencies but it has far more applications rather than cryptocurrencies alone. One of the recent advancement is smart contracts. These contracts are simply programs stored in blocks and can used to automatically exchange coins based on certain conditions. Potential of blockchain is unimaginable. Currently, the largest internet service provider in India (Reliance Jio) is also planning to launch its digital currency in India. As a distributed ledger blockchain reduces the cost involved in verifying transactions, and by removing the need of third parties such as banks to complete the transactions, technology also lowers the cost of networking, therefore allowing several applications. Currently it is widely used in cryptocurrencies like Bitcoin, Ethereum, Bitcoin cash and the list goes on and on. But applications of Blockchain goes beyond our imagination

Chapter 2  
Introduction

A blockchain is an electronic ledger of digital records, events, or transactions that are cryptographically hashed, authenticated, and maintained through a distributed network of participants using a group consensus protocol. Much like a checkbook is a ledger of one’s personal financial transactions, with each entry indicating the details of a transaction the blockchain is a complete listing of all transactions, whether financial or otherwise. However, the blockchain is distributed among thousands of computers precisely called nodes with a process for validating transactions that utilizes a group consensus protocol. Making an addition to a blockchain ledger requires the approval of the network at large making retrospective changes essentially impossible. Blockchain technology’s most disruptive aspect is its ability to eliminate the need for third-party intermediaries in some transactions. The technology is based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need for a trusted third party. Because many industries rely upon guarantors, authenticators and trusted third parties, blockchain technology is likely to be extremely disruptive. However, Blockchain technology itself is non-controversial and has worked flawlessly over the years and is being successfully applied to both financial and non-financial world applications. Last year, Marc Andreessen, the doyen of Silicon Valley’s capitalists, listed the blockchain distributed consensus modelas the most important invention since the Internet itself. Johann Palychata from BNP Paribas wrote in the Quintessence magazine that bitcoin’s blockchain, the software that allows the digital currency to function should be considered as an invention like the Blockchain Technology steam or combustion engine that has the potential to transform the world of finance and beyond.

In year 2008, an individual or group writing under the name of Satoshi Nakamoto published a paper entitled “Bitcoin: A Peer-To-Peer Electronic Cash System”. This paper described a peer-to-peer version of the electronic cash that would allow online payments to be sent directly from one party to another without going through a financial institution. Bitcoin was the first realization of this concept. Now word cryptocurrencies is the label that is used to describe all networks and mediums of exchange that uses cryptography to secure transactions-as against those systems where the transactions are channeled through a centralized trusted entity. The author of the first paper wanted to remain anonymous and hence no one knows Satoshi Nakamoto to this day. A few months later, an open source program implementing the new protocol was released that began with the Genesis block of 50 coins. Anyone can install this open source program and become part of the bitcoin peer-to-peer network. It has grown in popularity since then. The popularity of the Bitcoin has never ceased to increase since then. The underlying Blockchain technology is now finding new range of applications beyond finance.

\*\*\*References: [4],[5],[6]

Chapter 3  
Concept Learning

“Blockchain can do for business what the internet did for communication.”

There are following three things which caused these concepts like Blockchains to rise. That are:

* Business Networks
* Wealth
* Markets
* Business Networks benefit from proper connectivity which includes participants (like *customers, suppliers, banks, partners*), regulatory boundary and cross geography.
* Wealth is generated and controlled by the flow of goods and services across business network in transactions and contracts.
* Markets are central to this process. They can be private(like *supply chain financing, bonds*) as well as public (like *fruit market, car auction*)

# **3.1. Basics**

## **3.1.1. Assets**

Assets are those things which gets transferred from one participant to another and builds value. There can be 2 fundamental types of assets:

1. Tangible (e.g., *a house*)
2. Intangible (e.g., *a mortgage*):

Intangible further subdivides into:

i) Financial (e.g., *bond*)

ii) Intellectual (e.g., *patents*)

iii) Digital (e.g., *music*)

Note: Cash is also an asset which has property of anonymity.

## **3.1.2. Ledgers**

Ledger is the record system for a business. Business will have multiple ledgers for multiple business networks in which they participate.

## **3.1.3. Transaction**

Transaction simply is an asset being transferred onto or off the ledger.

E.g., *Balbir gives a car to Chirag (simple)*.

## **3.1.4. Contracts**

Contracts are the conditions for transaction to occur.

E.g., *If Chirag pays Balbir money, then car passes from Balbir to Chirag*

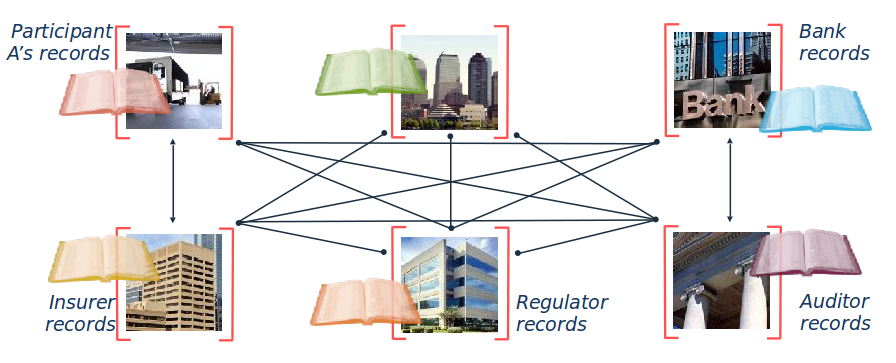
*(simple).*

*If car won’t start, funds do not pass to Balbir (as decided by third party arbitrator) (complex).*

# **3.2. Problem and its solution.**

# **3.2.1. Problem:**

Every participant had its own ledger which was inefficient, expensive and vulnerable.

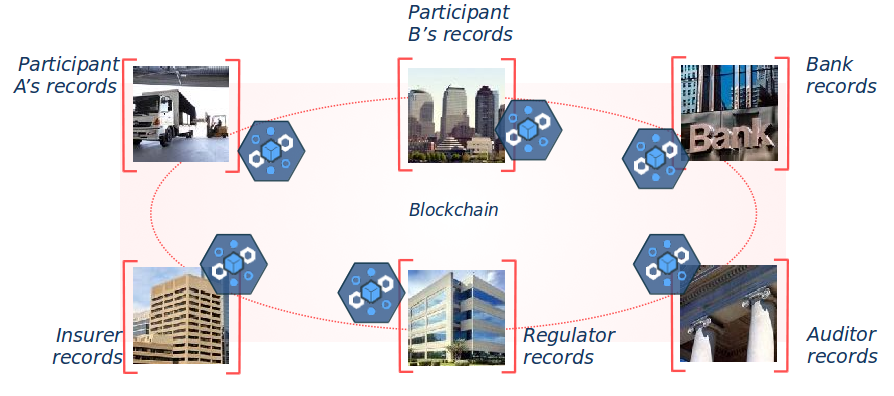


[fig. 3(a)]

# **3.2.2. Solution:**

As the solution to this problem, Blockchain gained its importance in the market as well as in technical world. It gave world a trusted distributed ledger with shared business processes.

# **3.3. Blockchain**

It is basically a shared replicated, permissioned ledger with consensus, provenance, immutability and finality.

[Fig. 3(b)]

\*\*\*References- [1]

## **3.3.1. Definition**

### **Wikipedia:**

A blockchain, originally block chain, is a continuously growing list of records, called blocks, which are linked and secured using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data (generally represented as a Merkle tree root hash). By design, a blockchain is resistant to modification of the data. It is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way". For use as a distributed ledger, a blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and validating new blocks. Once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks, which requires consensus of the network majority.

Blockchains are secure by design and exemplify a distributed computing system with high Byzantine fault tolerance. Decentralized consensus has therefore been achieved with a blockchain.

\*\*\*References: [3]

## **3.3.2. Features of Blockchain**

1. Shared Ledger: It is the shared system of record, been shared between participants and keeps record of all transactions across business network. Participants have own copy through replication. And It is permissioned, so that participants see only appropriate transactions.
2. Smart Contract: These are encoded in programming language and are simply the business rules which are implied by the contract. These are embedded in the Blockchain and executed with the transaction. They are verifiable and signed. E.g., *Defines contractual conditions under which corporate Bond transfer occurs.*
3. Privacy: The ledger is shared, but participants require privacy. Participants need appropriate confidentiality between subsets of participants in which Identity not to be linked to a transaction. Transactions need to be authenticated. Cryptography central to these processes.
4. Trust: The ledger is a trusted source of information. Participants endorsed transactions. Basically, it is business network which decides who will endorse transactions and endorsed.

\*\*\*References: [1]

## **3.3.3. How does it work?**

The blockchain technology is probably the best invention since the internet itself. It allows value exchange without the need for trust or for a central authority. Imagine you and I bet $50 on tomorrow’s weather in San Francisco. I bet it will be sunny, you that it will rain. Today we have three options to manage this transaction:

• We can trust each other. Rainy or sunny, the losing one will give $50 to the winner. If we are friends, this could be a good way of managing it. However, friends or strangers, one can easily not pay the other.

• We can turn the bet into a contract. With a contract in place both parties will be more prone to pay, however, should any of the two decide not to pay, the winner will have to pay additional money to cover legal expenses and the verdict might take a long time. Especially for a small amount of cash, this doesn’t seem the optimal way of

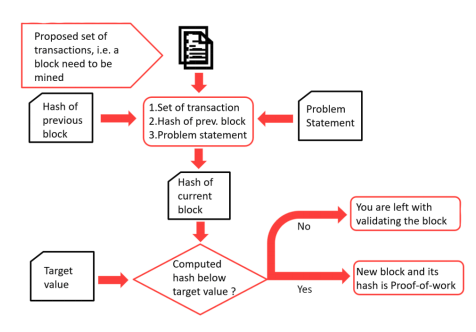
managing the transaction.

• We can involve a neutral third party. Each of us gives $50 to a third party, she then will give the total amount to the winner. But hey, she could also run.

Both trust and contract aren’t optimal solutions: we can’t trust strangers and enforcing a contract requires time and money. The blockchain technology is interesting because it offers us a third option which is secure, quick and cheap. Blockchain allows us to write a few lines of code, a program running on the blockchain, to which both of us send $50. This program will keep the $100 safe and check tomorrow’s weather automatically on several data sources. Sunny or rainy it will transfer automatically the whole amount to the winner. Each party can check the contract logic, and once it’s running on the blockchain it can’t be changed or stopped. This effort can be quite too high for a $50 bet but imagine when selling a house or a company.

## **3.3.4. How Mining works in Blockchain?**

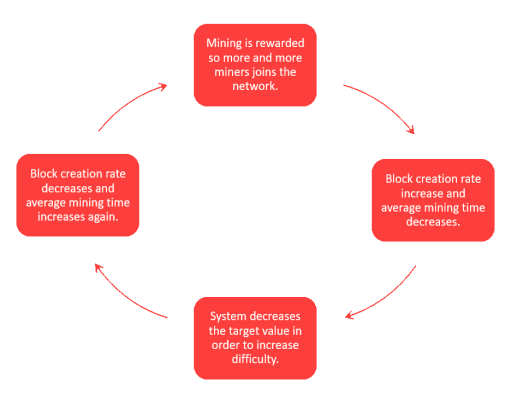
A block is constituted by several pending transactions broadcasted to global blockchain network. Every 10 minutes or so specialized computers called miners collect a few hundreds of transactions and combine them in a block. Miners will now process the new block to reach the consensus of how the new block will look like. At this stage, all miners start to work on solving the Proof-of-work of the problem. Consider a member of network wo find the hash of the proposed block below the threshold value then that miner wins the game and all other miners are left with validating the block. When more than 51% of the members have added the block to their copy of the blockchain then block is validated and considered definitive.



[fig. 3(c)]

If the computed hash isn’t below the threshold value and no other miner yet calculated the hash then it goes back to the problem statement of the proof-of-work and follows the algorithm for every miner.

Since, mining is rewarded so more and more miners joins the network. As the number of miners grows, processing power of the network grows and the block creation rate increases and block creation time decreases. This leads to a problem. In case of Bitcoin, mining is rewarded with Bitcoins that are not yet in circulation and the number of bitcoins is fixed at 21 million. If this happens at a greater pace, then there will be nothing left to reward the miners and at a smaller time span more bitcoins needs to be added to the system. By adding more and more bitcoins, it will lead to inflation in the cryptocurrency industry and it will no longer be different from current centralized financial system. To counter this, we have the system by which we maintain the balance between the block creation rate and the increasing processing power due to increasing number of miners in the networks.



[fig. 3(d)]

Here proof-of-work acts as a limiting factor. When block creation rate rises to a certain value, then problem statement in proof of work become more complex or the target value of the problem statement decreases which increases the difficulty and to generate a hash less than the target value becomes difficult. This leads to increase in block creation time and thus it balances the whole system.

\*\*\*References: [4],[5],[6]

## **3.3.5. Applications**

Instead of cryptocurrencies, blockchain has a vast range of applications to transform the future of current economic, social, internet status. Some of them are listed below.

### **3.3.4.1. Financial services**

Traditional systems tend to be unmanageable, error-prone and slow. Intermediaries are often needed to mediate the process and resolve conflicts. Naturally, this costs stress, time, and money. In contrast, users find the blockchain cheaper, more transparent, and more effective. Small wonder that a growing number of financial services are using this system to introduce innovations, such as smart bonds and smart contracts. The former automatically pays bondholders their coupons once certain preprogrammed terms are met. The latter are digital contracts that self-execute and self-maintain, again when terms are met. Example of blockchain financial services are:

1. Asset Management: Trade Processing and Settlement

2. Insurance: claim and processing

3. Payments: Cross-Border Payments

### **3.3.4.2. Smart Contracts**

Smart contracts are digital which are embedded with an if-this-then-that (IFTTT) code, which gives them self-execution. In real life, an intermediary ensures that all parties follow through on terms. The blockchain not only waives the need for third parties, but also ensures that all ledger participants know the contract details and that contractual terms implement automatically once conditions are met.

You can use smart contracts for all sort of situations, such as financial derivatives, insurance premiums, property law, and crowd funding agreements, among others. Examples of blockchain Smart contracts.

are:

1. Blockchain Healthcare

2. Blockchain Music

3. Voting system in Blockchain

### **3.3.4.3. Blockchain as Identity**

Blockchain protects your identity by encrypting it and protecting it from spammers and other marketing schemes. Some examples of blockchain identity are:

1. Passport

2. Birth, Death and Wedding certificates

3. Personal ID/Security ID

It’s important to note that for the blockchain to work, the node-to-node network must be motivated and agree to work under ethical standards. Once, and only if, these standards are adhered to, the blockchain could become a powerful tool for improving business, conducting fair trade, democratizing the global economy, and helping support more open and fair societies.

\*\*\*References:[3], [4],[5],[6]

3.4. Hyperledger

## **3.4.1. Definition**

Hyperledger (or the Hyperledger project) is an umbrella project of open source blockchains and related tools, started in December 2015 by the Linux Foundation, to support the collaborative development of blockchain-based distributed ledgers.

\*\*\*References: [2]

## **3.4.2. Hyperledger Fabric (a Hyperledger framework)**

Hyperledger Fabric is the first project to move to active status within the Linux Foundation’s Hyperledger, emerging as the de-facto standard for enterprise blockchain platforms like the IBM Blockchain Platform and others. Through open source and open governance, Hyperledger Fabric features innovative new capabilities hardened for use by businesses so they can usher in a new era of trust, transparency and accountability.

#### **Permissioned network**

#### Network

Establish decentralized trust in a network of known participants rather than a public network with no identity.

#### **Confidential transactions**

Confidential

Expose only the data you want to share to the parties you want to share it with.

#### **Pluggable architecture**

Pluggable

Tailor the blockchain to industry needs with a pluggable architecture rather than a one size fits all approach.

#### **Easy to Get Started**

Get Started

Program smart contracts in the languages your team works in today instead of learning custom languages and architectures.

\*\*\*References: [12]

## **3.4.3. Hyperledger Composer ( A Hyperledger tool)**

Hyperledger Composer is a set of collaboration tools for building blockchain business networks that make it simple and fast for business owners and developers to create smart contracts and blockchain applications to solve business problems. Built with JavaScript, leveraging modern tools including node.js, npm, CLI and popular editors, Composer offers business-centric abstractions as well as sample apps with easy to test develops processes to create robust blockchain solutions that drive alignment across business requirements with technical development.

Blockchain package management tooling contributed by IBM. Composer is a user-facing rapid prototyping tooling, running on top of Hyperledger Fabric, which allows the easy management of Assets (data stored on the blockchain), Participants (identity management, or member services) and Transactions (Chaincode a.k.a Smart Contracts, which operate on Assets on the behalf of a Participant). The resulting application can be exported as a package (a BNA file) which may be executed on a Hyperledger Fabric instance, with the support of a Node.js application (based on the Loopback application framework) and provide a REST interface to external applications.

Composer provides a GUI user interface "Playground" for the creation of applications, and therefore represents an excellent starting point for Proof of Concept work.

\*\*\*References: [12]

Chapter 4

Platform for Implementation

For implementing the learned concepts, the next task was choosing the platform or the developing environment which itself took a very considerable amount of time. There are many developing environment for applying concepts of Blockchain and Hyperledger. Out of them following two are the ones best suited for development:

1. Bluemix (by IBM)
2. Hyperledger Composer Playground

Following are the complete details of process of installing the both. One can choose either as a Hyperledger developing environment.

# **4.1. Bluemix**

## **4.1.1. Setting up your development environment - all clients**

▪ You will need Docker installed on your workstation to use the local development environment. To install Docker, please follow the instructions at the following links:

▪▪ OSX: https://docs.docker.com/docker-for-mac/install/#download-docker-for-mac

▪▪ Windows: https://docs.docker.com/docker-for-windows/install/

▪▪ Ubuntu: https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/

▪▪ Debian: https://docs.docker.com/engine/installation/linux/docker-ce/debian/

▪▪ CentOS: https://docs.docker.com/engine/installation/linux/docker-ce/centos/

▪▪ Fedora: https://docs.docker.com/engine/installation/linux/docker-ce/fedora/

▪ curl is used to manage the developer set up

▪▪ it’s installed as part of current releases of OS X

▪▪ if you don’t have it and are on a Linux system, use your package manager   
 to install it

▪▪ for example: Ubuntu: *sudo apt-get install curl*

## **4.1.2. Set up your development environment - Windows**

▪ Microsoft Windows is not a natively supported operating system for Hyperledger Composer, as of September, 2017. To run this tutorial on your Windows (V7 or higher) operating system, we will install VirtualBox and then install Ubuntu 16.04 as your machine image. We will then use the Ubuntu installation exec to get everything onto your computer.

▪ Step 1: download a 64-bit Ubuntu 16.04 image

▪▪ Go here: https://www.ubuntu.com/download

▪▪ click on the Ubuntu Desktop option - this is a large download and will take  
 30+ minutes

▪ Step 2: download and install VirtualBox

▪▪ Go here: https://www.virtualbox.org/wiki/Downloads

▪▪ And click on Windows Hosts. This will start the download of the VirtualBox  
 installer

▪ Step 3: Go to your downloads folder and run the VirtualBox …. .exe installer.

▪▪ Take the defaults

▪ Step 4: Start Virtual Box

▪▪ Click on the “new” icon

▪▪ Take all the defaults EXCEPT memory.

▪▪ Give your virtual machine at least 2Gb rather than the 1Gb default

▪ Step 5: Start your new virtual machine

▪▪ Give it the ISO file you downloaded in step 1

▪ Step 6: After the machine starts, select the “Install Ubuntu Desktop” option

▪▪ Select options to update the installation and to install 3rd party code

Step 7: Follow the steps for an Ubuntu installation on the following pages.

## **4.1.3. Setting up your development environment: Linux: Ubuntu**

▪ Open a terminal window on your Ubuntu Linux and enter the following command:

▪▪ *sudo apt-get install -y curl*

▪ After curl has been installed, copy the following as a single line into a terminal window and press enter

*curl -H 'Accept: application/vnd.github.v3.raw' https://raw.githubusercontent.com/rddill-IBM/ZeroToBlockchain/master/setup\_Ubuntu\_Part\_1.sh | bash*

▪▪ This should be copied and pasted onto a single line in your terminal   
 window. Press enter after copying.

▪ This will install all required software up through docker.

▪ You will then need to reboot your system prior to executing the following command:

*curl -H 'Accept: application/vnd.github.v3.raw' https://raw.githubusercontent.com/rddill-IBM/ZeroToBlockchain/master/setup\_Ubuntu\_Part\_2.sh | bash*

▪▪ This will complete the installation of the hyper ledger images and the   
 supporting tools.

▪ This command will go through the following steps, informing you of each step in the process

▪▪ Update the apt-get repositories and upgrade current software

▪▪ Ensure that the base development environment is installed

▪▪ Check for the presence of git and install it if it’s missing

▪▪ Check for the presence of NodeJS version 6 (Required for hyper ledger) and   
 install it if it’s missing

▪▪ Install the NodeJS SDK for Hyperledger composer

▪▪ Install the VSCode editor

▪▪ Install the Hyperledger fabric docker images

▪▪ Install the fabric tools and update your .bash\_profile

▪▪ Install Hyperledger composer platform-specific binaries

## **4.1.4. Setting up your development environment: Linux: Ubuntu (Optional)**

▪ If you don’t want to automatically install and update all of the tools, then execute the following three commands, instead:

▪▪ *curl -H ‘Content:application/vnd.github.v3.raw’ https://raw.githubusercontent.com/rddill-IBM/ZeroToBlockchain/master/setup\_Ubuntu\_Part\_1.sh >> setup\_Ubuntu\_Part\_1.sh*

▪▪ *sudo chmod +x setup\_Ubuntu\_Part\_1.sh*

▪▪ *./setup\_Ubuntu\_Part\_1.sh -h*

▪ the -h option will list the commands you use to prevent selected installation actions from taking place, for example, to do everything except install git and NodeJS V6, you would type:

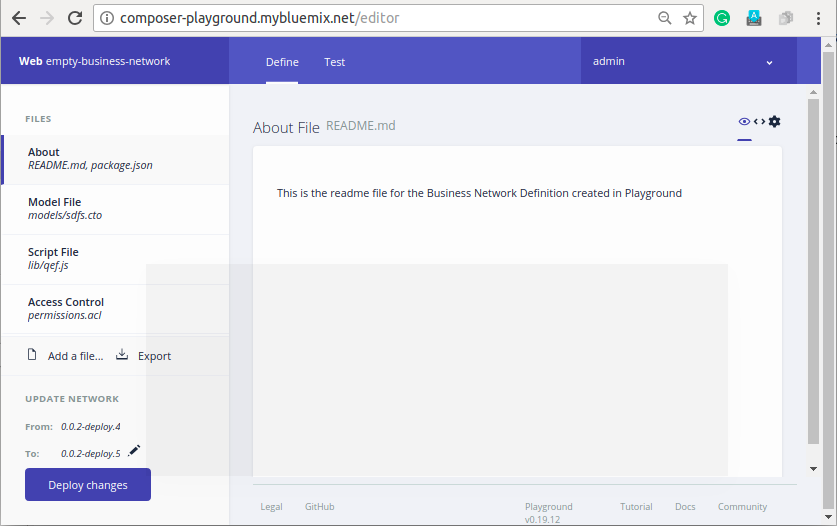
▪▪ *./setup\_Ubuntu\_Part\_1.sh -g false -n false*

\*\*\*References: [1]

# **4.2. Hyperledger Composer Playground**

In case one finds difficulties in above mentioned installation process of Bluemix, the alternate solution to it is - Hyperledger Composer Playground.

Hyperledger Composer is an extensive, open development toolset and framework to make developing blockchain applications easier with primary goal to accelerate time to value and make it easier to integrate blockchain applications with the existing business systems. It can be used to rapidly develop use cases and deploy a blockchain solution in weeks rather than months. It allows you to model your business network and integrate existing systems and data with your blockchain applications.

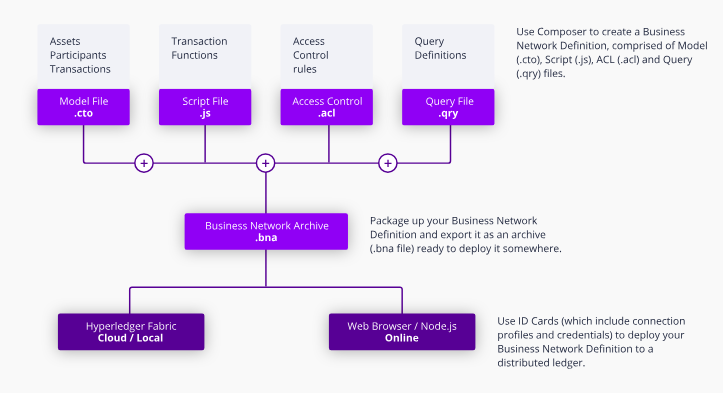


[ Fig 4(a) ]

Hyperledger Composer supports the existing Hyperledger Fabric blockchain infrastructure and runtime, which supports pluggable blockchain consensus protocols to ensure that transactions are validated according to policy by the designated business network participants.

Everyday applications can consume the data from business networks, providing end users with simple and controlled access points.

We used Hyperledger Composer to quickly model our business network, containing our assets and the transactions related to them; assets are tangible or intangible goods, services, or property. As part of our business network model, we defined the transactions which can interact with assets. Business networks also include the participants who interact with them, each of which can be associated with a unique identity, across multiple business networks.



[ Fig 4(b) ]

## **4.2.1. Hyperledger Composer Environment**

You can experience Hyperledger Composer with our browser-based UI called Hyperledger Composer Playground. Playground is available as a hosted version (no install necessary) or a local install (good for editing and testing sample business networks offline).

Developers who want to use Hyperledger Composer's full application development capabilities should install the Developer Tools.

1. Hosted version of Hyperledger Composer Playground (shown in fig 4(a)):

- To try Hyperledger Composer right away, follow this link up:

<https://composer-playground.mybluemix.net/>

1. Offline local version of Hyperledger Composer Playground:

For installing Hyperledger Composer on a local computer, follow these steps:

i) Install prerequisites from the following link:

<https://hyperledger.github.io/composer/latest/installing/installing-prereqs.html>

ii) Install developing environment from the following link:

<https://hyperledger.github.io/composer/latest/installing/development-tools.html>

iii) Update the developing environment from following link:

<https://hyperledger.github.io/composer/latest/installing/update-dev-env.html>

* For an architectural overview of a typical solution built with Composer, follow up the following link: <https://hyperledger.github.io/composer/latest/introduction/solution-architecture.html>

## **4.2.2. Key Concepts in Hyperledger Composer**

Hyperledger Composer is a programming model containing a modeling language, and a set of APIs to quickly define and deploy business networks and applications that allow participants to send transactions that exchange assets.

### **Blockchain State Storage**

All transactions submitted through a business network are stored on the blockchain ledger, and the current state of assets and participants are stored in the blockchain state database. The blockchain distributes the ledger and the state database across a set of peers and ensures that updates to the ledger and state database are consistent across all peers using a consensus algorithm.

### **Connection Profiles**

Hyperledger Composer uses Connection Profiles to define the system to connect to. A connection profile is a JSON document the is part of a business network card. These profiles are usually provided by the creator of the system they refer to and should be used to create business network cards in order to be able to connect to that system.

### **Assets**

Assets are tangible or intangible goods, services, or property, and are stored in registries. Assets can represent almost anything in a business network, for example, a house for sale, the sale listing, the land registry certificate for that house, and the insurance documents for that house may all be assets in one or more business networks.

Assets must have a unique identifier, but other than that, they can contain whatever properties you define. Assets may be related to other assets or participants.

### **Participants**

Participants are members of a business network. They may own assets and submit transactions. Participant types are modeled, and like assets, must have an identifier and can have any other properties as required. A participant can be mapped to one or multiple identities.

### **Identities**

An identity is a digital certificate and private key. Identities are used to transact on a business network and must be mapped to a participant in the business network. A single identity is stored in a business network card and if that identity has been mapped to a participant, it allows the user of that business network card to transact on a business network as that participant.

### **Business Network cards**

Business network cards are a combination of an identity, a connection profile, and metadata, the metadata optionally containing the name of the business network to connect to. Business network cards simplify the process of connecting to a business network and extend the concept of an identity outside the business network to a 'wallet' of identities, each associated with a specific business network and connection profile.

### **Transactions**

Transactions are the mechanism by which participants interact with assets. This could be as simple as a participant placing a bid on an asset in an auction, or an auctioneer marking an auction closed, automatically transferring ownership of the asset to the highest bidder.

### **Queries**

Queries are used to return data about the blockchain world-state. Queries are defined within a business network and can include variable parameters for simple customization. By using queries, data can be easily extracted from your blockchain network. Queries are sent by using the Hyperledger Composer API.

### **Events**

Events are defined in the business network definition in the same way as assets or participants. Once events have been defined, they can be emitted by transaction processor functions to indicate to external systems that something of importance has happened to the ledger. Applications can subscribe to emitted events through the composer-client API.

### **Access Control**

Business networks may contain a set of access control rules. Access control rules allow fine-grained control over what participants have access to what assets in the business network and under what conditions. The access control language is rich enough to capture sophisticated conditions declaratively, such as "only the owner of a vehicle can transfer ownership of the vehicle". Externalizing access control from transaction processor function logic makes it easier to inspect, debug, develop and maintain.

### **Historian registry**

The historian is a specialized registry which records successful transactions, including the participants and identities that submitted them. The historian stores transactions as Historian Record assets, which are defined in the Hyperledger Composer system namespace.

\*\*\*References: [13]]

Chapter 5  
Project Domain

5.1. Agriculture Market

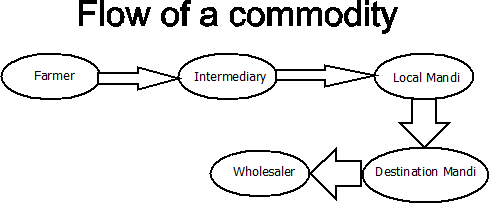
India is an agricultural country and one third population depends on the agricultural sector directly or indirectly. Agriculture remains as the main stray of the Indian economy since times immemorial. Indian agriculture contribution to the national gross domestic product (GDP) is about 25 per cent. With food being the crowning need of mankind, much emphasis has been on commercializing agricultural production. For this reason, adequate production and even distribution of food has of late become a high priority global concern.

Agricultural marketing is mainly the buying and selling of agricultural products. In earlier days when the village economy was more or less self-sufficient the marketing of agricultural products presented no difficulty as the farmer sold his produce to the consumer on a cash or barter basis.

Today's agricultural marketing has to undergo a series of exchanges or transfers from one person to another before it reaches the consumer. There are three marketing functions involved in this, i.e., assembling, preparation for consumption and distribution. Selling on any agricultural produce depends on some couple of factors like the demand of the product at that time, availability of storage etc. The products may be sold directly in the market or it may be stored locally for the time being. Moreover, it may be sold as it is gathered from the field or it may be cleaned, graded and processed by the farmer or the merchant of the village. Sometime processing is done because consumers want it, or sometimes to conserve the quality of that product. The task of distribution system is to match the supply with the existing demand by whole selling and retailing in various points of different markets like primary, secondary or terminal markets.

Most of the agricultural products in India are sold by farmers in the private sector to moneylenders (to whom the farmer may be indebted) or to village traders. Products are sold in various ways. For example, it might be sold at a weekly village market in the farmer's village or in a neighboring village. If these outlets are not available, then produce might be sold at irregularly held markets in a nearby village or town, or in the mandi.

In our very basic model of agriculture market, there will be a farmer who will have the produce or commodity. Intermediaries will be bidding for it and one paying highest will get the commodity. Then this intermediary sells the commodity to Local Mandi where processing of food takes place. On the order by Destination Mandis, the Local market pays the transportation cost and sells the commodity to Destination Mandi after it has been processed completely. Now from this Destination Mandi various Wholesalers buy the commodity. Following figure shows the flow diagram of above.



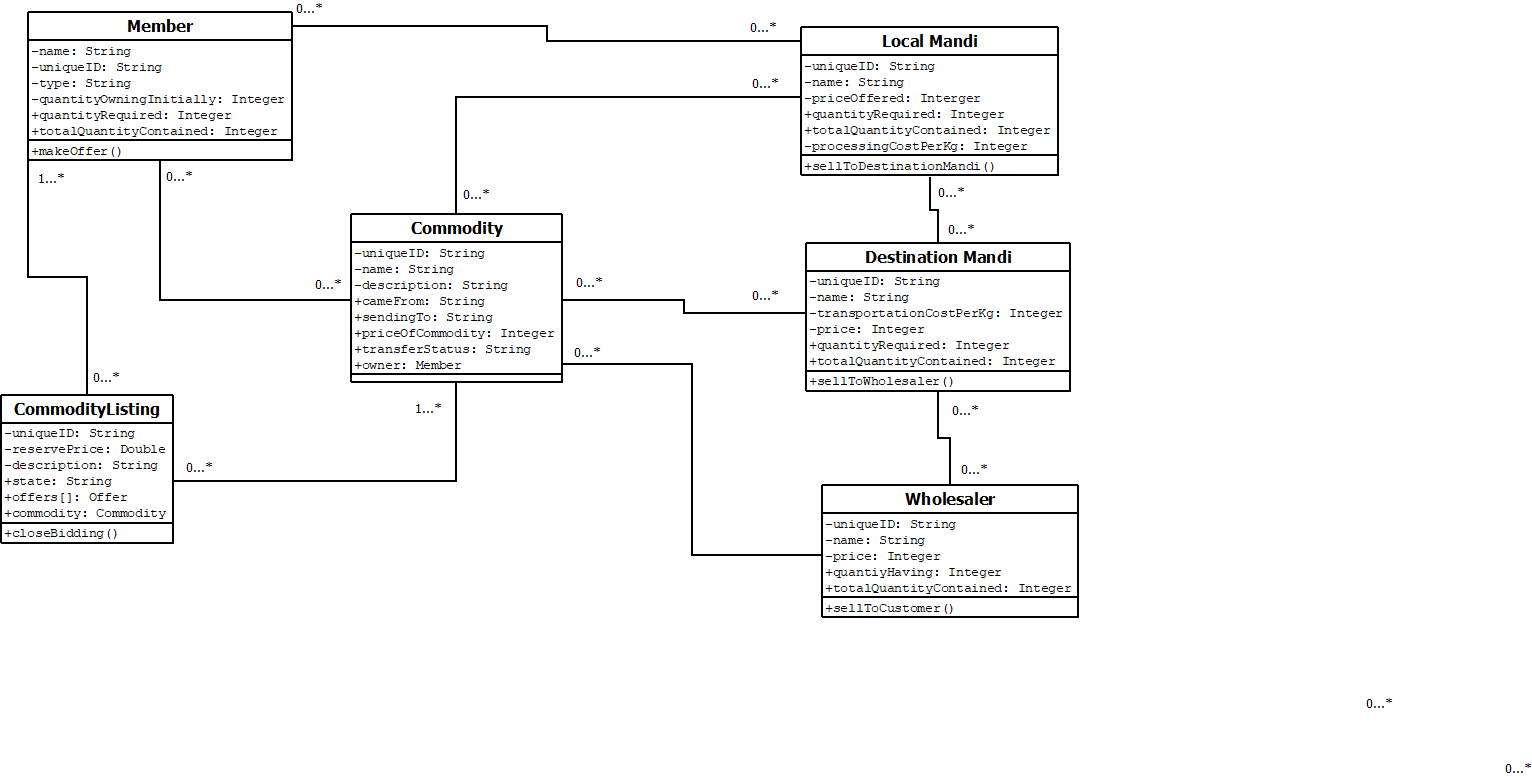
[Fig.5(a)]

5.2. Class Diagram

Class diagram for above flow diagram is given below. It have 4 participant classes (same for Farmer and Intermediary) and 2 asset classes.

Participants – Member, Local Mandi, Destination Mandi, Wholesaler.

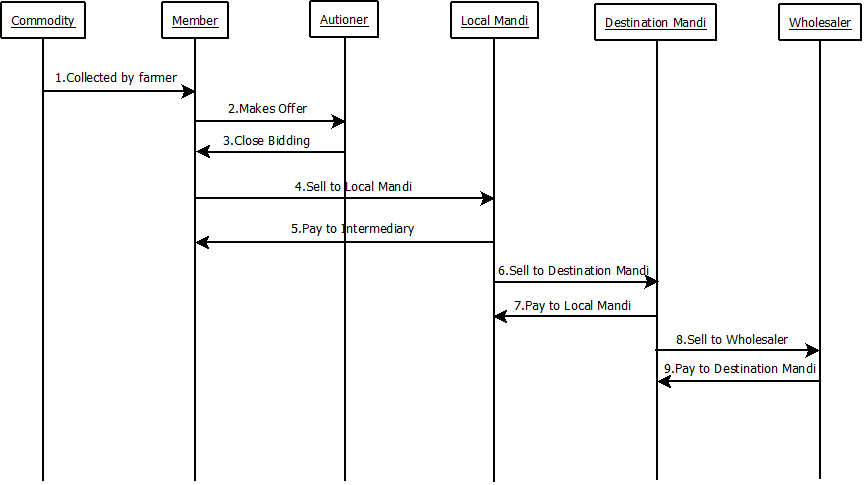
Assets- Commodity, CommodityListing.



[Fig.5(b)]

5.3. Sequence Diagram

Following shows the sequence of events that will take between various actors. Note that creation of objects has been presumed.



[Fig.5(c)]

Chapter 6  
Code Implementation

The coding is done in 3 basic parts or files which are as follows:

1. Model File (.cto file) - for definition
2. JavaScript File (.JS file) - for logic behind transactions
3. Access Control File (.acl file) - for access permissions

# **6.1. Model File:**

It contains .cto\* format file. It is used for the purpose of definition of participants, assets, transactions or any other class which needed to be defined. Everything which is been written in this file is in cto modelling language. This file consists of 3 major components:

1. Participants {Members (farmer or intermediary), Local Mandi, Destination Mandi, Wholesaler}
2. Assets {Commodity, Commodity Listing}
3. Transactions {Making bidding offer, closing the bid, selling to local mandi, selling to destination mandi, selling to wholesaler}

## **6.1.1. Participants**:

We had 4 classes of participants in our program. They are as follows:

1. Member (It is a common class for both farmers and intermediaries.)
2. LocalMandi
3. DestinationMandi
4. Wholesaler

### **6.1.1.1. Member**

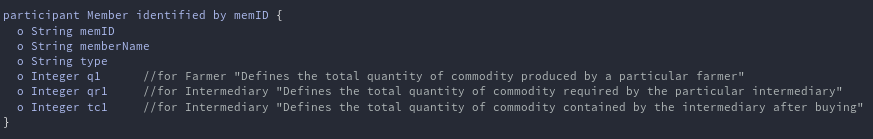
Earlier, we had 2 different classes for farmers and intermediaries but due to some kind of type errors in auction transactions, we have to get them merged into one same class i.e., Member.

Attribute “*type”* is used to distinguish between them.

For instance:- Farmer “*type*” can be “*farmer*” and for Intermediary “*type*” can be “*Intermediary”.*

“*memID”* is used to store a unique ID for every instance of participant Member.

Fig.5(a) shows the definition of participant *“*Member*”.*



[Fig. 6(a)]

### **6.1.1.2. Local Mandi**

Following is the definition of participant “LocalMandi*”*. In this, each instance of participant LocalMandi will have unique ID as “*localMandiID”* , based upon which one local mandi will be distinguished from the other.

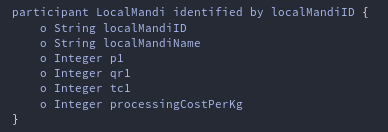
*“localMandiName”* is for storing name of the wholesaler owning that *localMandiID*.

*“p1”* is the price/kg, at which local mandi gets a particular commodity.

*“qr1”* is the quantity required by local mandi.

*“tc1*” is the total quantity owned by local mandi at any instant.

“*processingCostPerKg”* is the processing cost per Kg that will occur before selling it to any other mandi.



[Fig. 6(b)]

### **6.1.1.3. Destination Mandi**

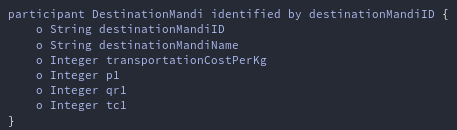
Following is the definition of participant “DestinationMandi*”*. In this each instance of participant DestinationMandi will have unique ID as “*destinationMandiID”* , based upon which one destination mandi will be distinguished from the other.

*“destiantionMandiName”* is for storing name of the wholesaler owning that *destinationMandiID*.

*“p1”* is the price/kg, at which destination mandi gets a particular commodity.

*“qr1”* is the quantity required by destination mandi.

*“tc1*” is the total quantity owned by destination mandi at any instant.



[Fig. 6(c)]

### **6.1.1.4. Wholesaler**

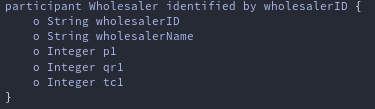
Following is the definition of participant “Wholesaler*”*. In this, each wholesaler will have unique “*wholesalerID”*, based upon which one wholesaler will be distinguished from the other.

*“wholesalerName”* is for storing name of the wholesaler owning that *wholesalerID*.

*“p1”* is the price/kg, at which wholesaler gets a particular commodity.

*“qr1”* is the quantity required by wholesaler.

*“tc1*” is the total quantity owned by wholesaler at any instant.



[Fig. 6(d)]

## **6.1.2. Assets:**

We basically have two assets in our program. They are as follows:

1. Commodity - (which can be wheat, rice, pulses, etc. as the case may be.)
2. CommodityListing - (for maintaining list of bidders which here are intermediaries.)

### **6.1.2.1. Commodity**

Following is the definition of asset “Commodity” in which “*commodityID*” is used to distinguish any particular instance of any commodity from another.

“*cameFrom”* is used to store name of the seller of that commodity after last transaction.

“*sendingTo”* is used to store name of the buyer of that commodity after last transaction.

“*priceofcom”* is used to store current price of that commodity per Kg.

“*transferStatus”* is used to store status of any transaction whether successful or not.

Reference to participant “Member” is for knowing and storing the details of current owner of that commodity. Every time the commodity gets transferred from one participant to another, it is this variable that gets updated and stores the information of current owner accordingly.



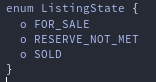
[Fig. 6(e)]

### **6.1.2.2. CommodityListing**

Following is the definition of asset “CommodityListing” in which “*listingID”* is the unique ID of any bidding list.

“*reservePrice*” is the minimum bid price which any authoritative party would decide then.

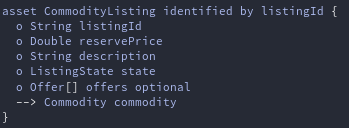
“*ListingState”* is of enum type having 3 values that are:- “FOR\_SALE”, “RESERVE\_NOT\_MET” and “SOLD”.



[Fig. 6(f)]

“*Offer[]”* will have array of instances of bidders in descending order of their bidding price.

Reference to “Commodity”- for information of commodity which is to be auctioned.



[Fig. 6(g)]

## **6.1.3. Transactions:**

There are 5 basic transactions in our code, which are as follows:

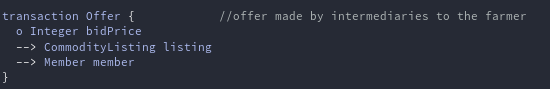
1. Offer
2. CloseBidding
3. SellToLocalMandi
4. SellToDestinationMandi
5. SellToWholesaler

### **6.1.3.1. Offer**

Following is the definition of transaction “Offer” having “*bidPrice”* for the amount any intermediary will bid for. There are 2 references in this transaction:

“CommodiyListing”- for pointing to the bidders’ list in which this offer is to be included.

“Member”- for pointing to the member that makes this bid.



[Fig. 6(h)]

### **6.1.3.2. CloseBidding**

Following is the definition of transaction “CloseBidding*”.* It has only one reference i.e., to *“Commoditylisting”* - which points to the list, of which bidding result is to be evaluated. As a result, commodity either gets sold to highest bidder or not get sold to any (i.e., in case the bidding price is less than the reserved price).



[Fig. 6(g)]

### **6.1.3.3. SellToLocalMandi**

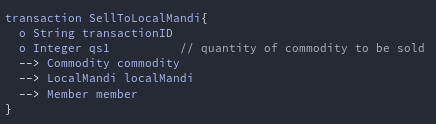
This definition of transaction “SellToLocalMandi” includes “*transactionID*” - for having unique transaction ID for the transaction.

“*qs1*” - for quantity to be sold to local mandi.

Reference to:- Commodity – is for information of commodity which is to be sold,

LocalMandi – is for information of the local mandi to which this commodity is to be sold, and

Member – is for information of the intermediary who is selling the commodity to this local mandi.



[Fig. 6(h)]

### **6.1.3.4. SellToDestinationMandi**

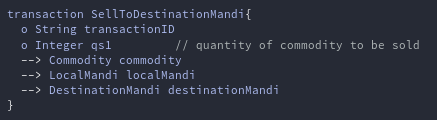
This definition of transaction “SellToDestinationMandi” includes “*transactionID*” - for having unique transaction ID for the transaction.

“*qs1*” - for quantity to be sold to destination mandi.

Reference to:- Commodity - for information of commodity which is to be sold,

DestinationMandi - for information of the destination mandi to which this commodity is to be sold, and

LocalMandi - for information of the local mandi which is selling the commodity to this destination mandi.



[Fig. 6(i)]

### **6.1.3.5. SellToWholesaler**

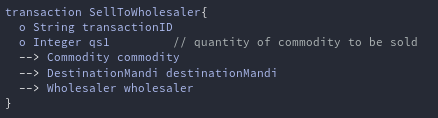
This definition of transaction “SellToWholesaler” includes:  
“*transactionID*” - for having unique transaction ID for the transaction.

“*qs1*” - for quantity to be sold to wholesaler. Reference to:

*Commodity* - for information of commodity which is to be sold,

*Wholesaler* - for information of the wholesaler to which this commodity is to be sold, and

*DestinationMandi* - for information of the destination mandi which is selling the commodity to this wholesaler.



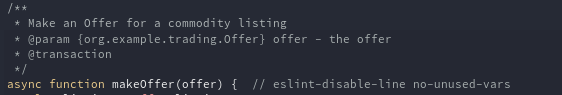
[Fig. 5(j)]

# **6.2. JavaScript file:**

It contains the logic behind every transactions which are been called during code execution. Complete code of which is given in the git-hub link: <https://github.com/sharmakshay27/KisaanPower>   
or <https://github.com/sushants1997/KisaanPower>

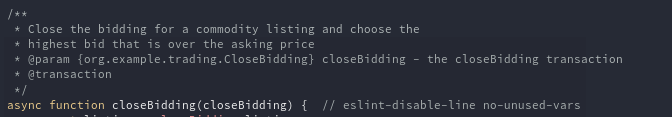
Whenever any particular transaction is being used, then its respective corresponding functions are called - with parameters as specified in respective transactions. E.g.,

* For making an Offer transaction, *async function makeOffer(offer)* is called.



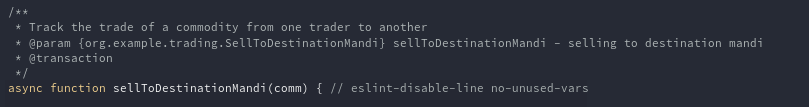
[Fig. 6(k)]

* Similarly, for making a CloseBidding transaction, this *async function closedBidding(closeBidding)* is called.



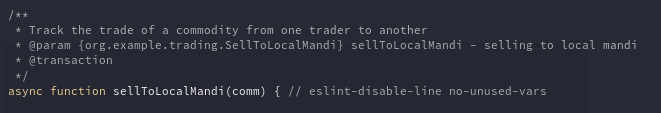
[Fig. 6(l)]

* Similarly, for making a SellToDestinationMandi transaction, this *async function sellToDestinationMandi(comm)* is called*.*



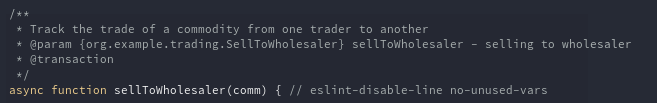
[Fig. 6(m)]

* Similarly, for making a SellToLocalMandi transaction, this *async function sellToLocalMandi(comm)* is called.



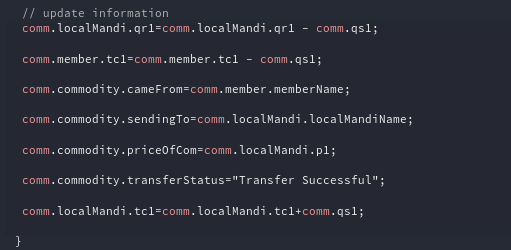
[Fig. 6(n)]

* Similarly, for making a SellToWholesaler transaction, this *async function sellToWholesaler(comm)* is called.



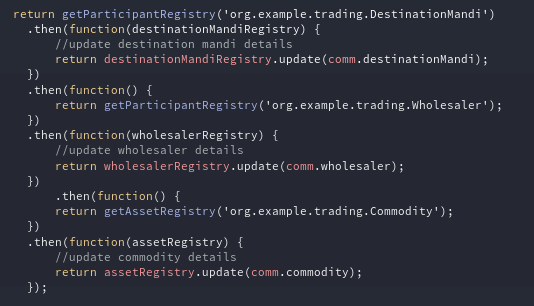
[Fig. 6(o)]

* The following part of the JS code updates the necessary details of participants and assets, only after transaction gets successful.



[Fig. 6(p)]

* The following part of JS code is for updating of values in the registry with the new values of attributes of participants and assets, after the transaction gets successful.



[Fig. 6(q)]

Chapter 7  
Code Exeution

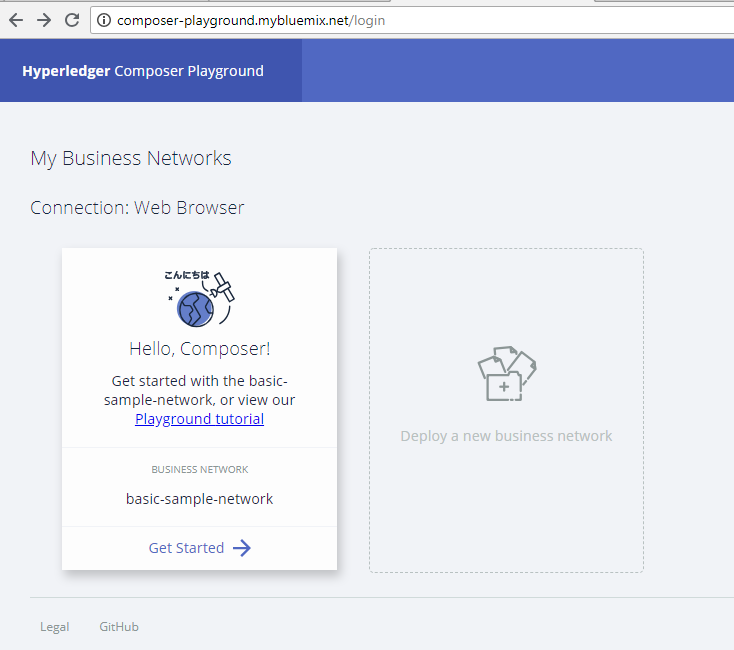
After having the network been made, the next thing is to deploy it. For this following steps are needed to be followed:

1. Step 1: Open the playground and necessary files.  
   1.1) Delete all the sample files that it has as default.  
   1.2) From local storage  
    1.2.1) Add Model file (.cto\*)  
    1.2.2) Add JavaScript file (.js\*)  
    1.2.3) Add Access Control file (.acl\*)
2. Step 2: Submit Transactions.

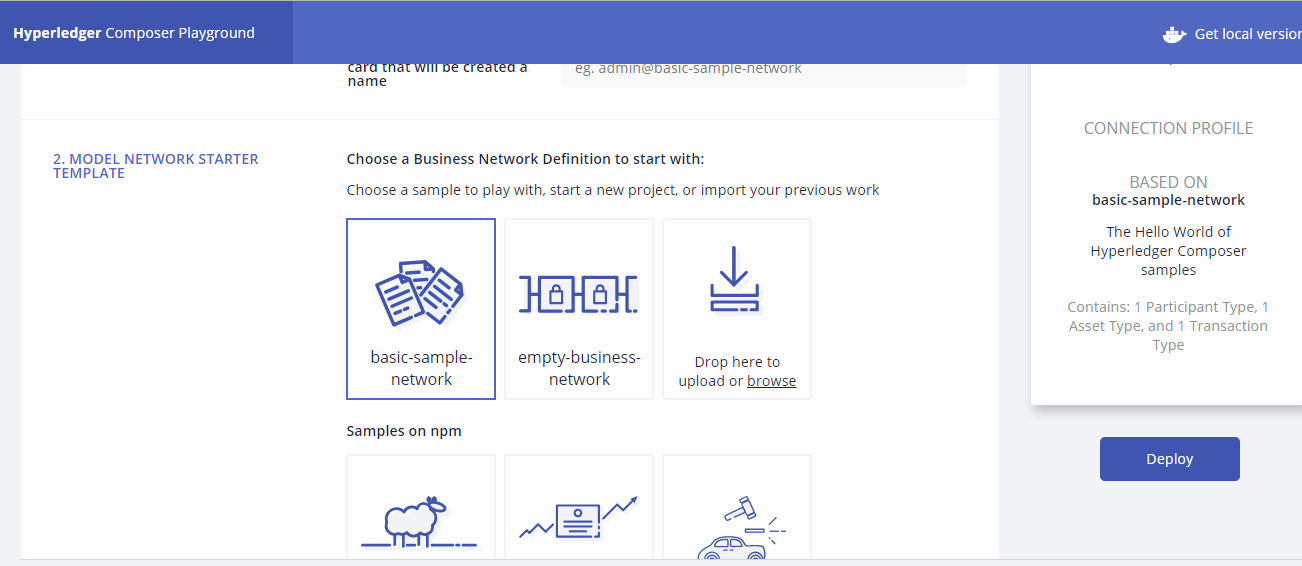
i.e., Follow these:

For Execution of the program follow the following steps:

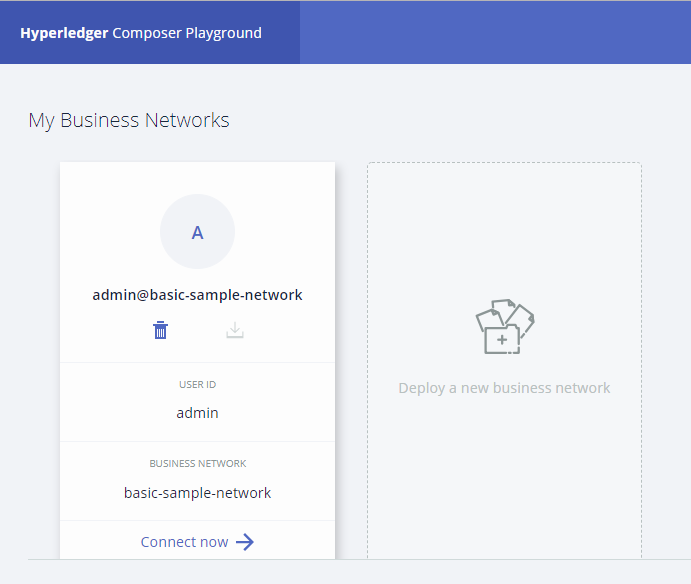
1. Open the Hyperledger playground. Click on Deploy a new business network.



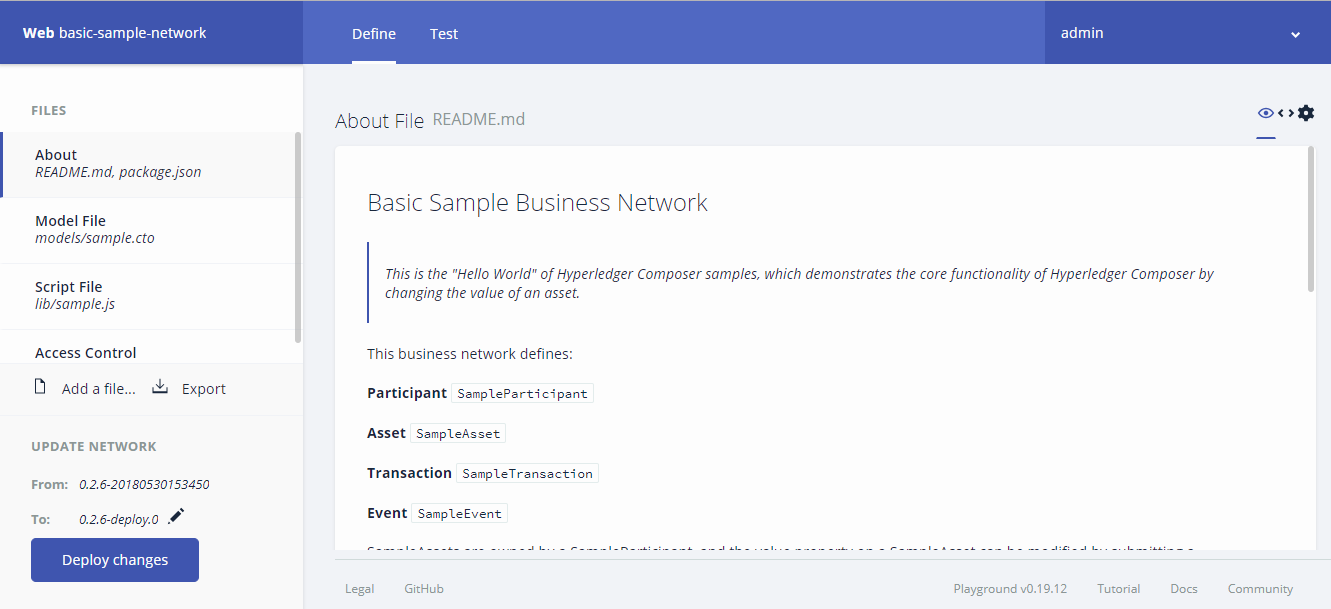
1. Click on Basic Sample Network and then click on Deploy button.



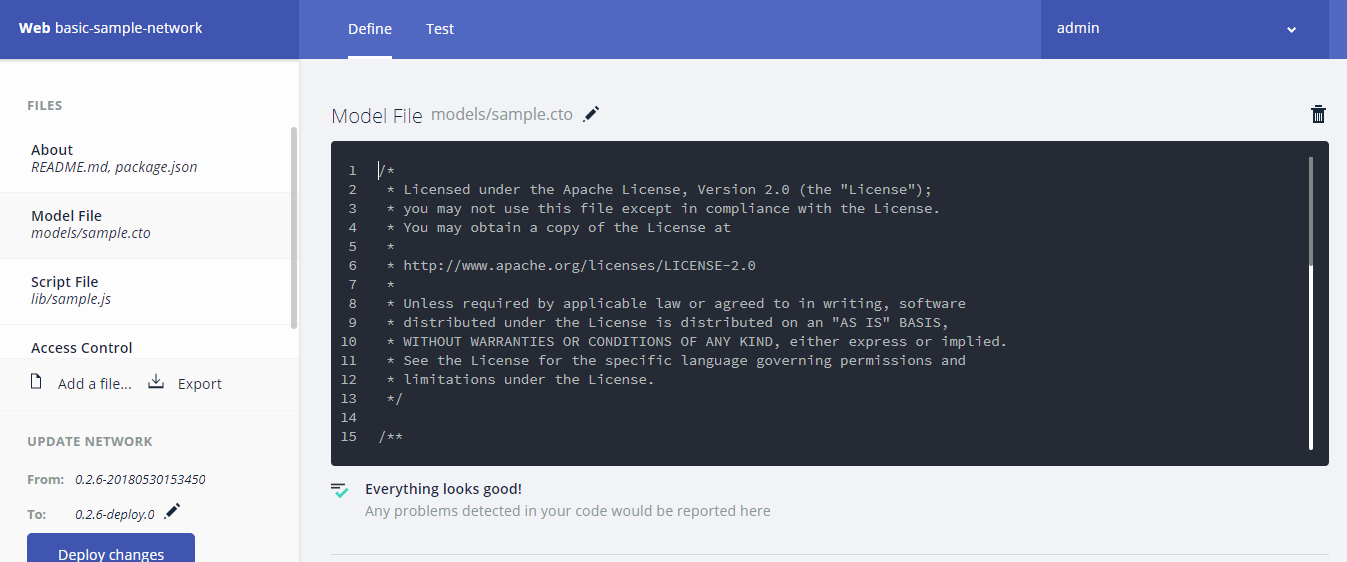
1. You will get this window. Click on Connect now.

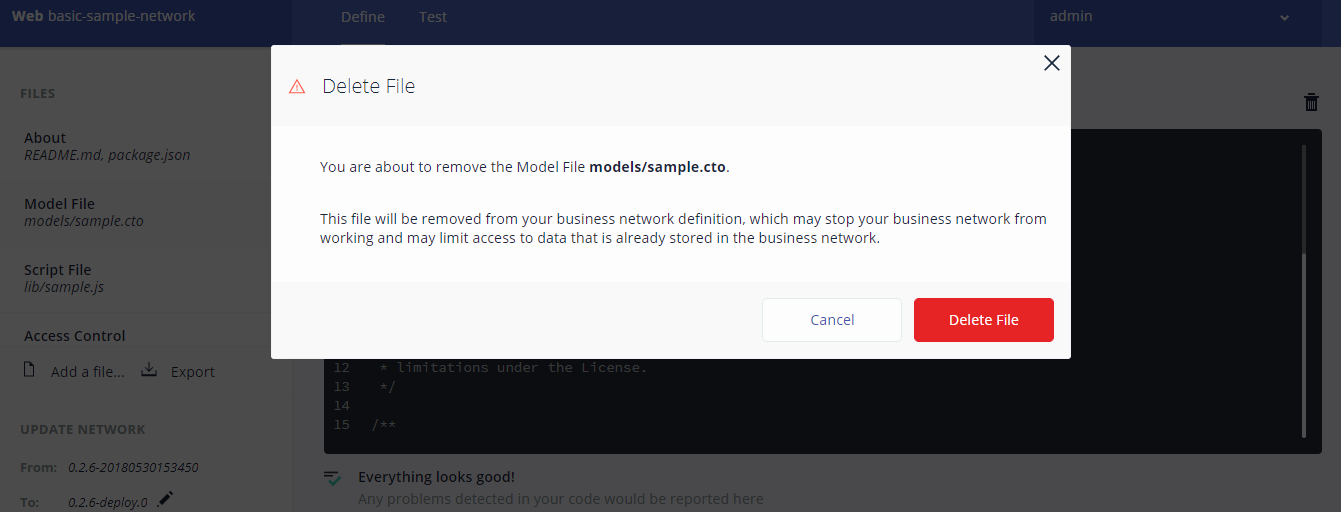


1. You will the following Screen, Delete all the default files which you can see on your left hand side.

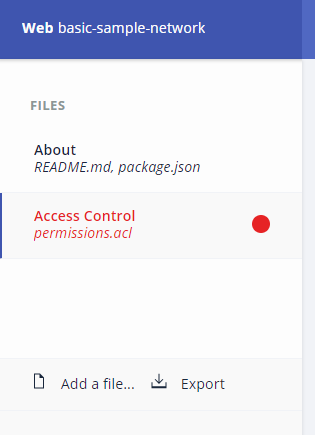


* 1. Click on delete icon on top right and delete when prompted with the warning message.

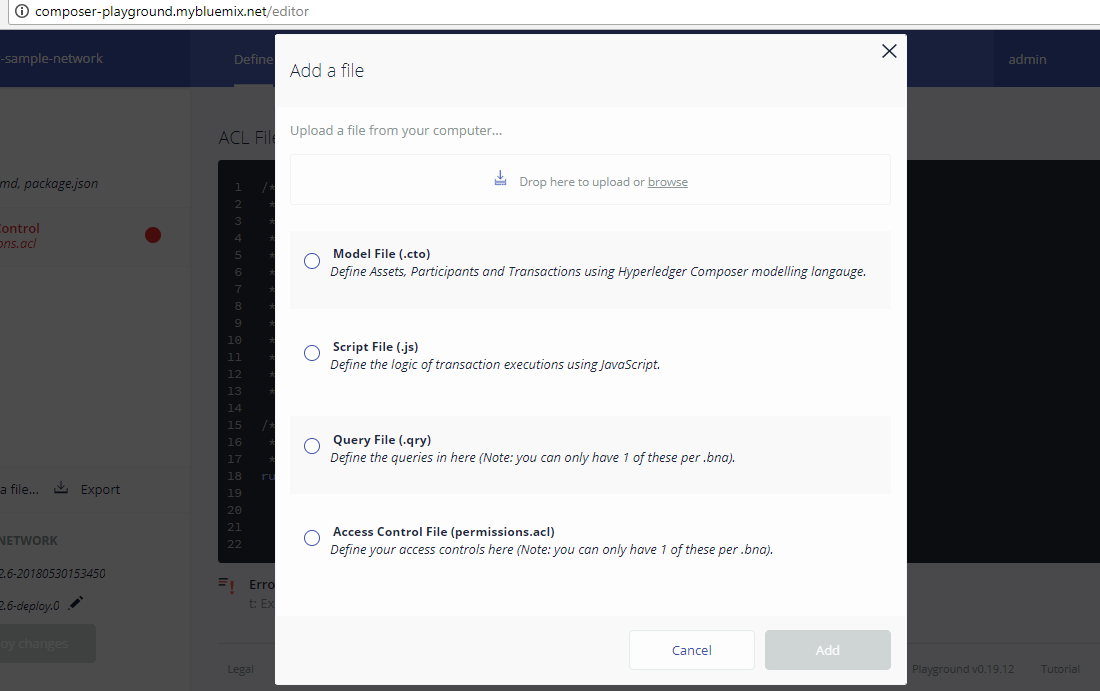




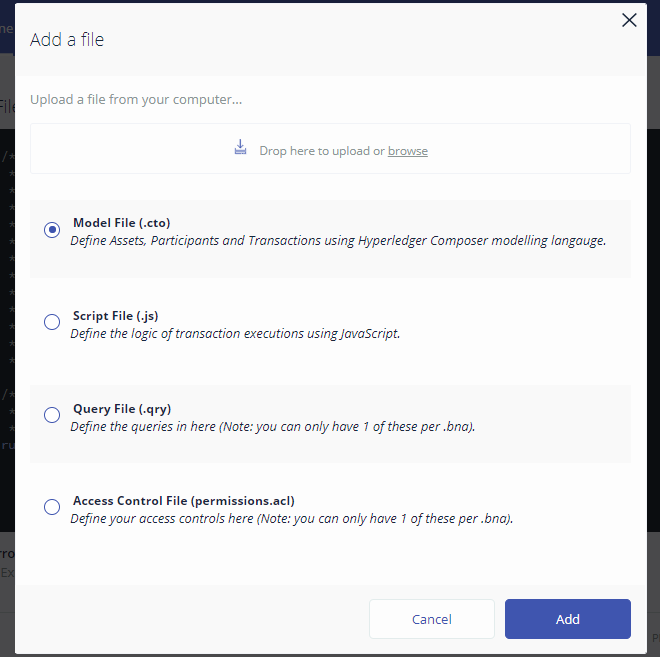
Note: You won’t be able to delete access file. So leave it.



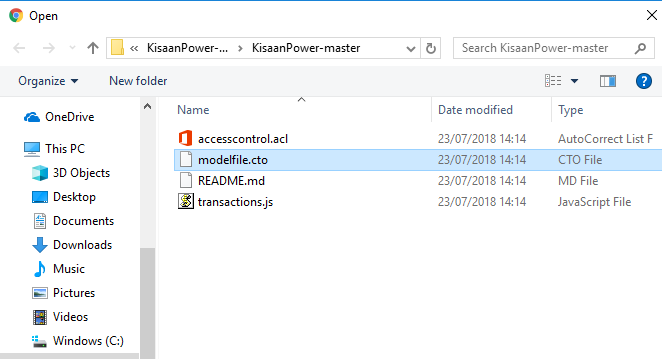
1. Upload the files accordingly from your PC.



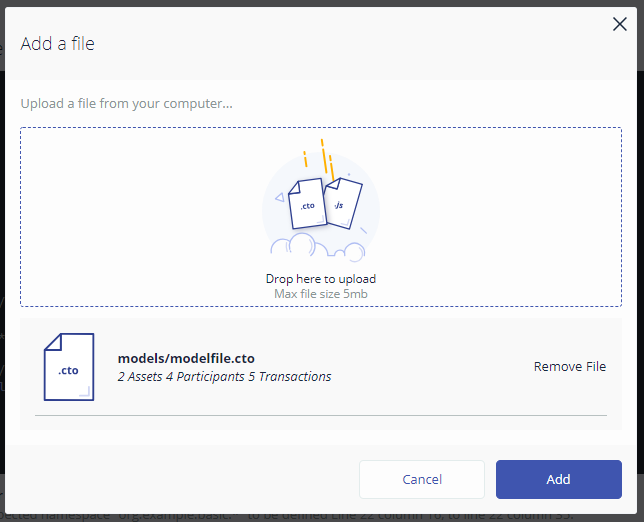
* 1. Check the Model file dialog box and click on browse button.



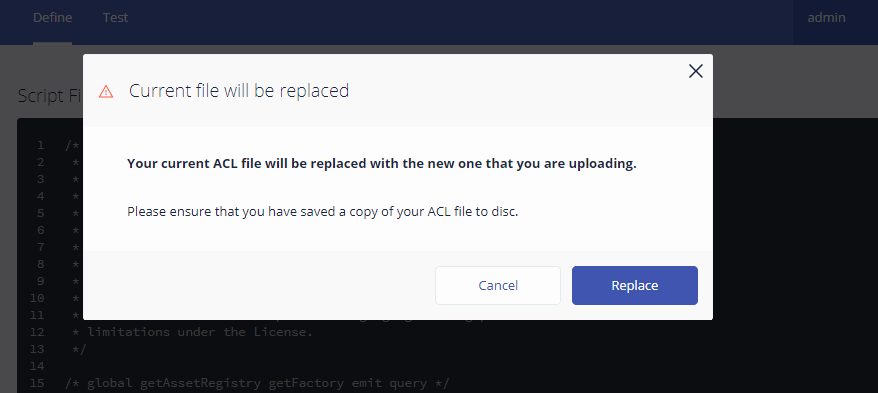
* 1. Select the required model file.



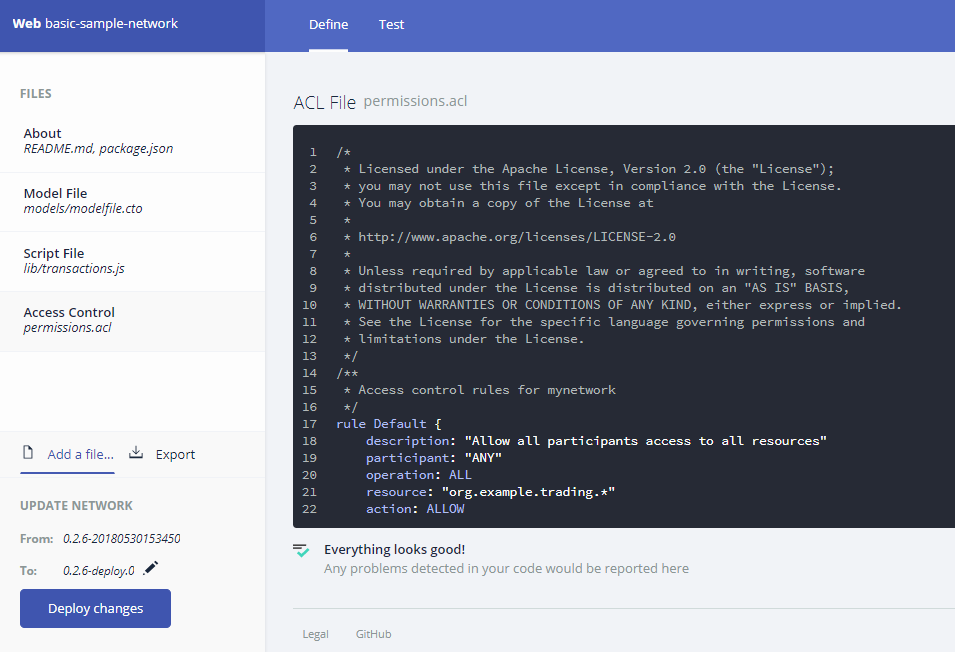
* 1. Click add.



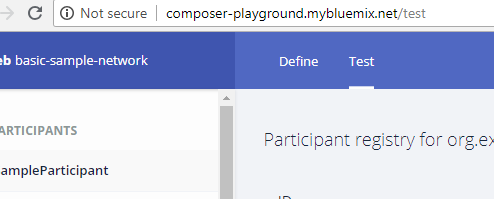
* 1. Similarly do for other files too.
  2. Note: When you will add access control file. You will be prompted with warning message to replace. Click on replace.



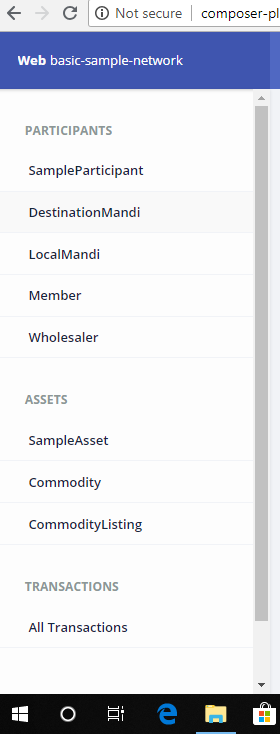
* 1. Click on Deploy changes at the left bottom corner of the window.



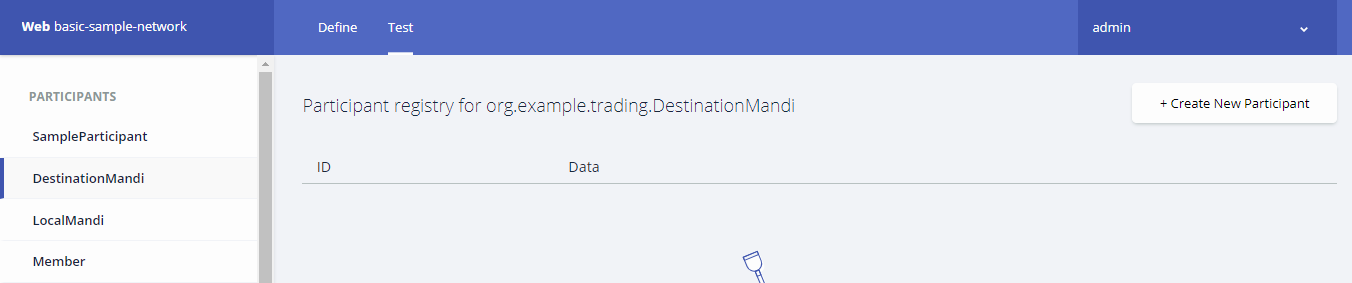
1. Click on Test.



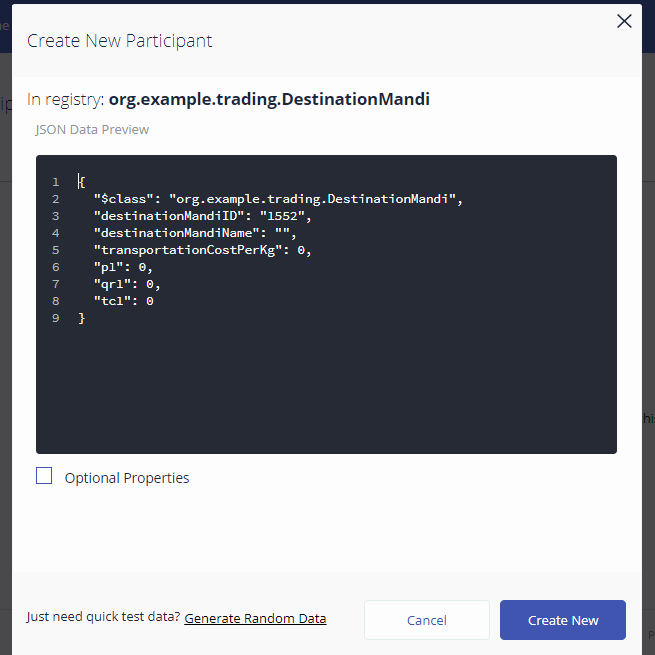
You will get the following options on left side of the window.



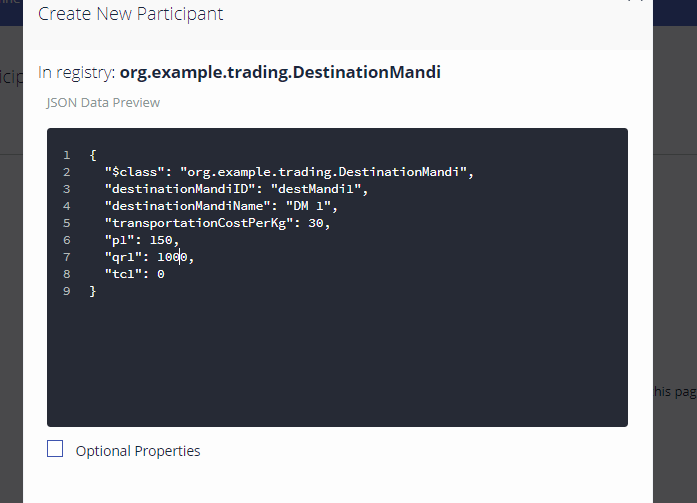
* 1. Click on any participant like DestinationMandi. And then click on +Create New Participant button on top right corner of your screen.

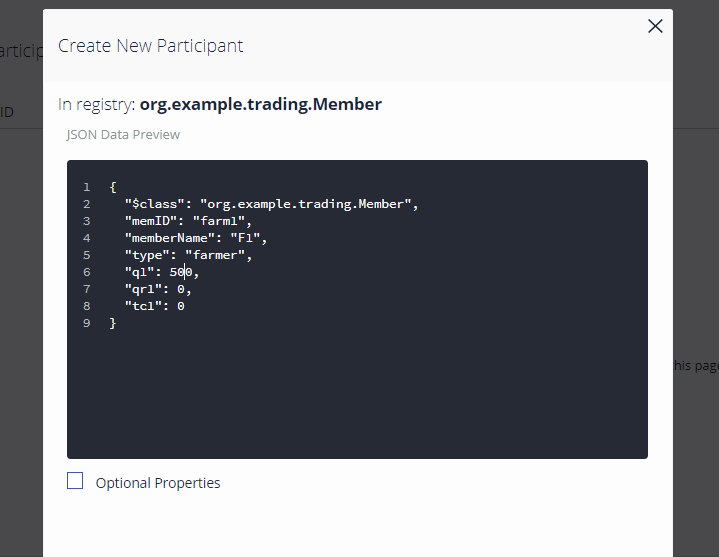


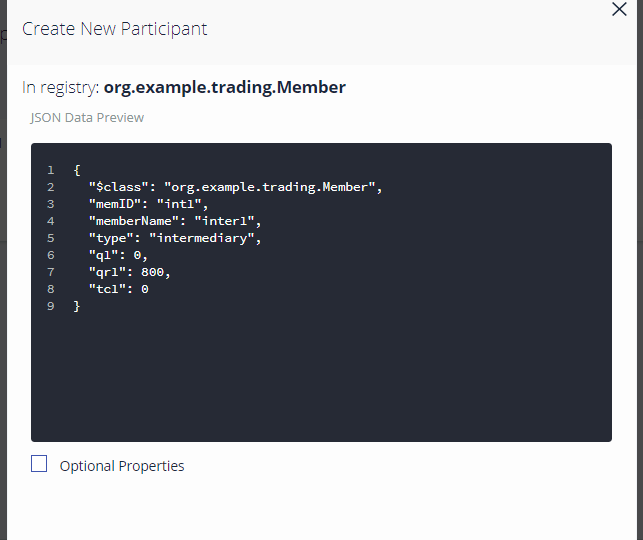
* 1. Following window will get opened. Fill the Required details and follow the same for all other participants as well as assets.

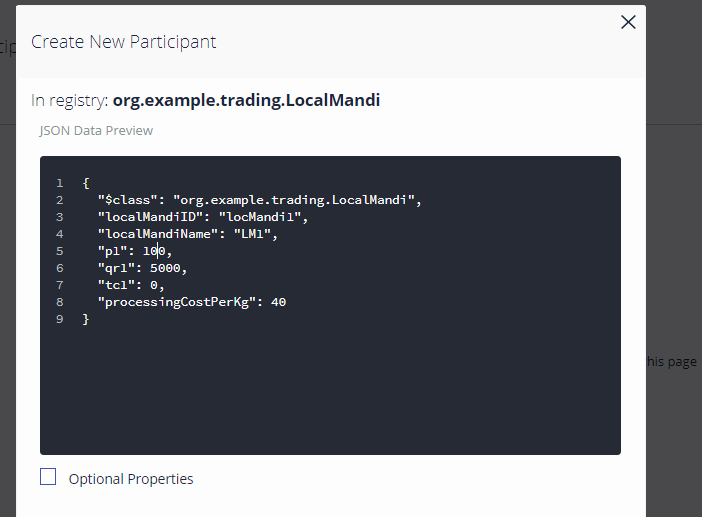


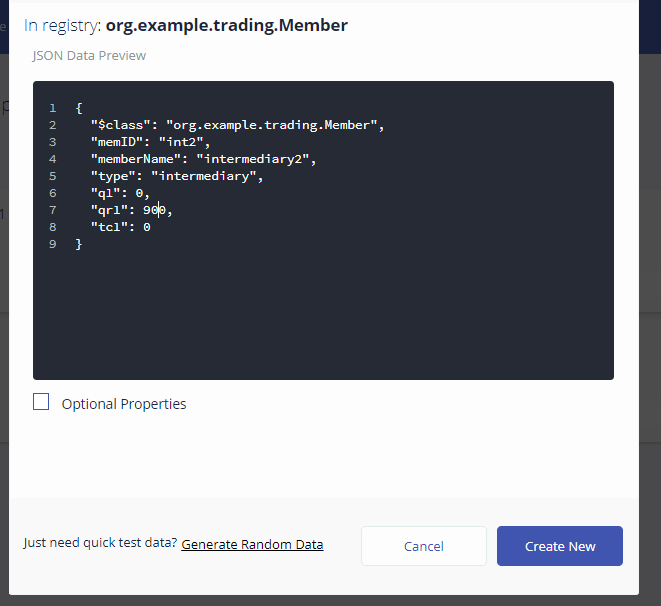
1. Filling the required details as shown in following figures. You can fill the details of your own choice too. (Note this is just an example.)

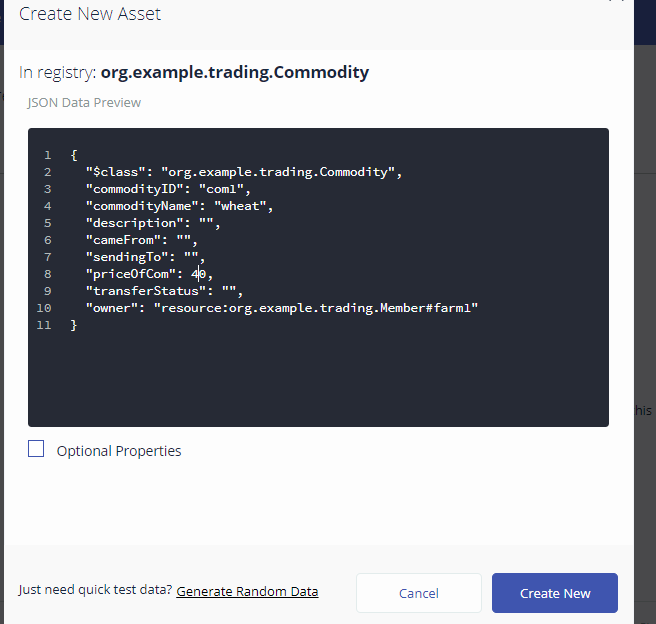


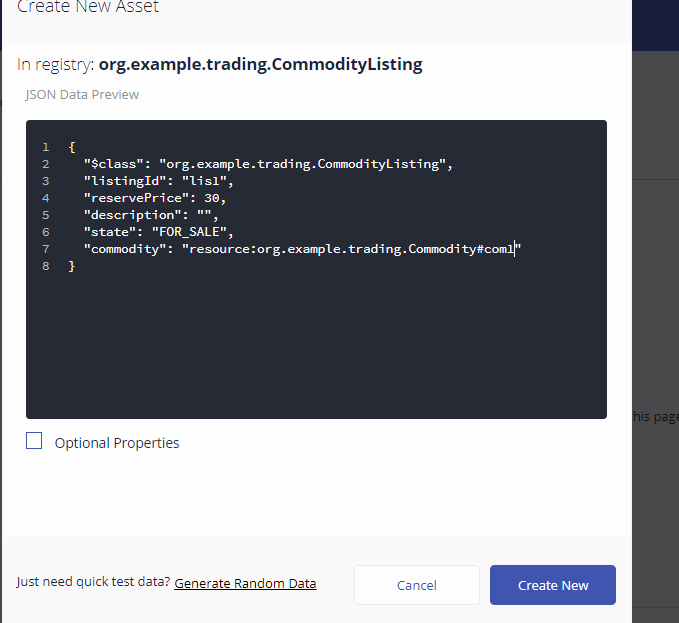




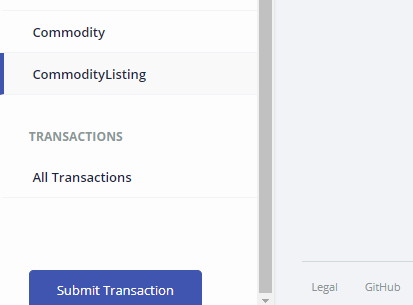




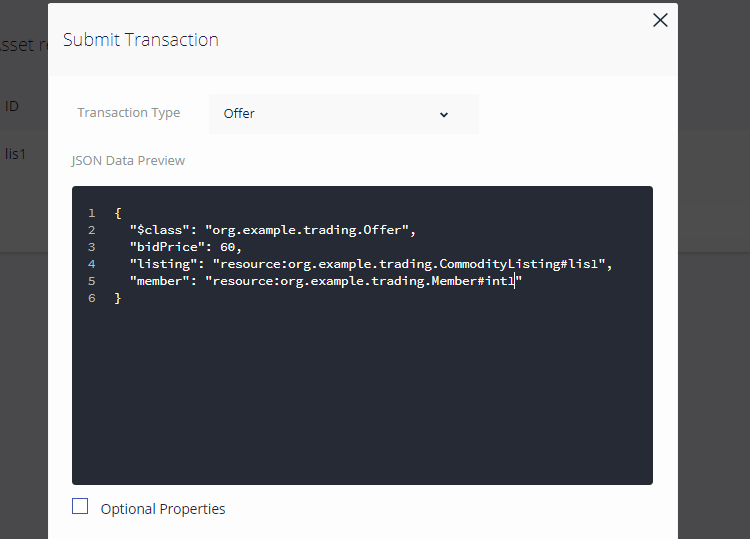


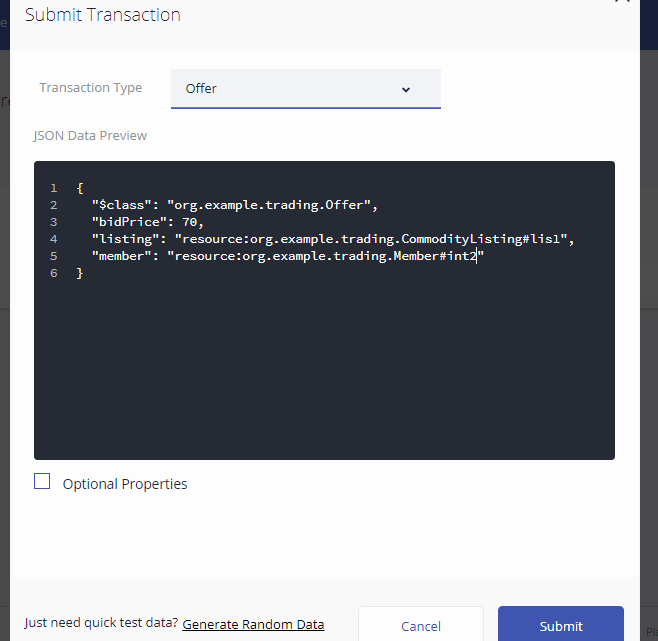


1. Once all participants and assets been defined initially. Click on submit transaction button at bottom left corner of window.

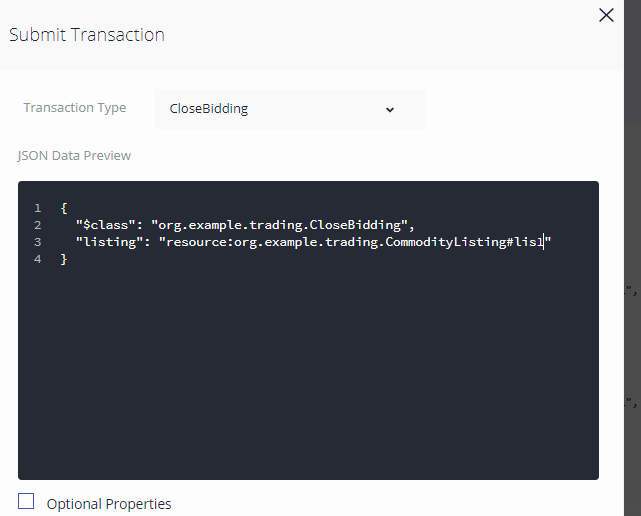


* 1. Select transaction type as “Offer” and filled the required details.

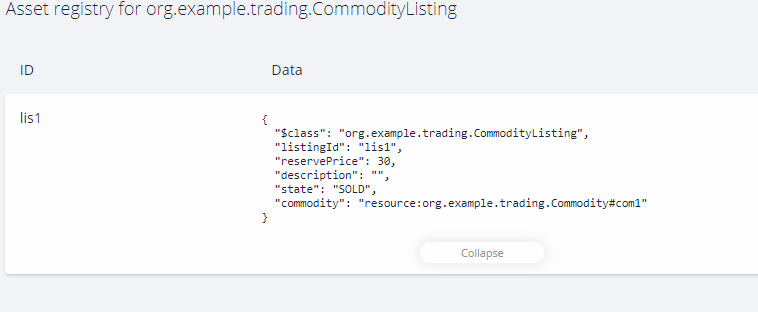




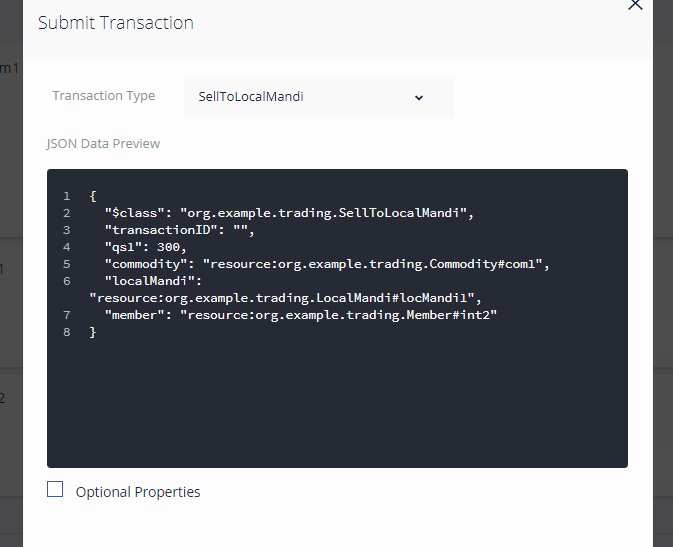
* 1. After successfully updating offer transaction, We need to close the bidding and decide the result of our auction. For this we have to select CloseBidding transaction type.

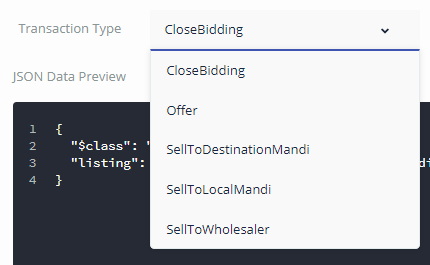


After Submitting you will find the state as “SOLD”.



* 1. Similarly other transactions can be done.





Follow above steps for all the transactions and then see result on clicking on any asset or participant. You can also view all transactions in the end.

Chapter 8  
Conclusion

From above code and successful execution of it, it is clear that concepts of Blockchain and Hyperledger can be applied to our agricultural sector too. Transactions in agricultural sector can be recorded in Blockchains to maintain Hyperledger which will bring transparency into the systems. With evolvement of this project many new aspects can be touched bringing out what is best for the farmers. It will surely give power along with awareness to the farmers which will help to improve corporate marketing among farmers. This implementation can be served as a proof of concept in many applications.

Chapter 9  
Future Scope

Further there are many possibilities for the improvement of this project. Currently it is the basic model by which proof of concept can be given. It lays down the platform working on which many great heights can be achieved in agriculture sector especially when we talk about things like ‘bringing transparency into the system’.   
  
In our very basic model, we have 1 commodity of 1 single quality, 1 farmer, 2 intermediaries bidding for the commodity, 1 local mandi, 1 destination mandi, 1 wholesaler. Note that with increase in functionality model will tend to become complex.

This can be increased as:-

1. 1 or more commodities with each one further subdivided into different categories on the basis of qualities. Price of different quality be different.
2. 1 or more farmers with each having number of different commodities of different quality to sell.
3. 2 or more intermediaries bidding for different commodities (say potato, tomato, etc.) of different qualities (like quality A (best quality), then quality B, and then quality C – for potatoes itself). Bidding can be first done for commodity of highest quality and then of lower. In case commodity of a particular quality runs out, then intermediaries which are still in demand of that commodity will increase its bidding price of their next demand so that it gets something in the end. E.g., Suppose there are 3 Intermediaries : Inter-1, Inter-2, and Inter-3 with initial case as:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Bidding Price for A | Requirement of A | Bidding Price for B | Requirement of B | Bidding Price for C | Requirement of C |
| Inter-1 | 60 | 100 | 40 | 110 | 20 | 210 |
| Inter-2 | 80 | 200 | 60 | 330 | 40 | 400 |
| Inter-3 | 70 | 150 | 50 | 250 | 30 | 500 |

After completion of auction of quality A, if there are intermediaries so left which didn’t get the commodity then its bidding price for quality B should be increased so that it gets at least something in the end. Therefore, new bidding list could be as:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Bidding Price for A | Requirement of A | Bidding Price for B | Requirement of B | Bidding Price for C | Requirement of C |
| Inter-1 | 75 | 100 | 62\* | 110 | 36 | 210 |
| Inter-2 | 80 | 0 | 60 | 330 | 40 | 400 |
| Inter-3 | 78 | 50 | 58 | 250 | 39 | 500 |

1. 2 or more Local Mandis can be there, with auction process among different local mandis and intermediaries too.
2. 2 or more Destination Mandis taking commodities from different local mandis and auctioning to wholesalers.
3. 2 or more Wholesalers, with auction process among different wholesalers and destination mandis too.
4. Smart contracts can be created like more profit will be there if group of 10 farmers combine and move to next closest Mandi.
5. Access rights can be given accordingly.
6. Queries can be made to get the information like average price per commodity, max selling of which commodity, etc. Based upon which different valuable decisions can be made.

Chapter 10  
References

1. <https://github.com/rddill-IBM/ZeroToBlockchain>
2. <https://en.wikipedia.org/wiki/Hyperledger>
3. <https://en.wikipedia.org/wiki/Blockchain>
4. <https://www.blockchain.com/>
5. <https://coinmarketcap.com/>
6. <https://www.forbes.com/lists/2006/18/06f2000_The-Forbes-2000_Rank.html>
7. <https://www.coursera.org/learn/blockchain-basics>
8. <https://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/crse0401.html>
9. <https://blockgeeks.com/guides/what-is-blockchain-technology/>
10. <https://medium.com/digitalcrafts/programming-blockchain-in-javascript-8051cab272da>
11. <https://www.tutorialspoint.com/javascript/>
12. <https://www.hyperledger.org/projects/composer>