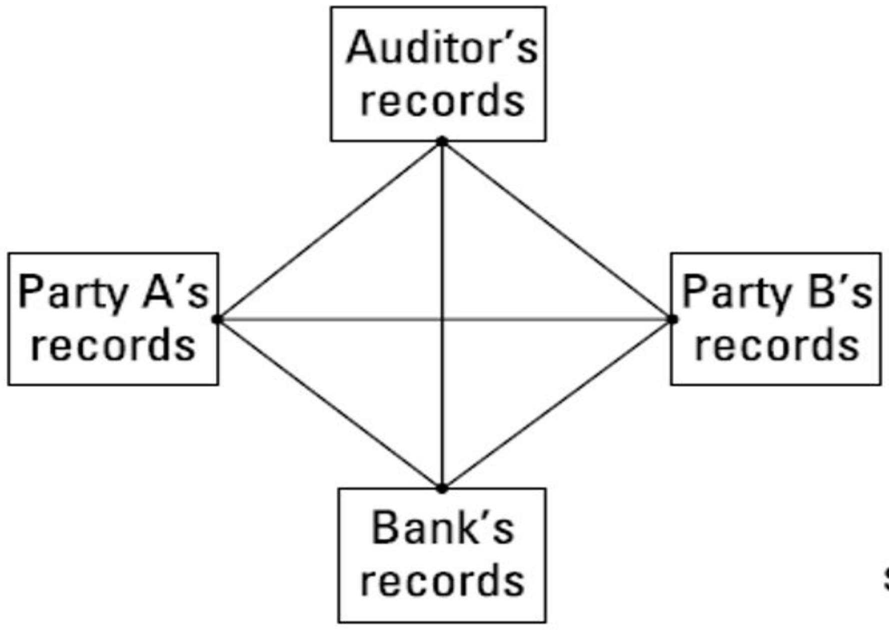
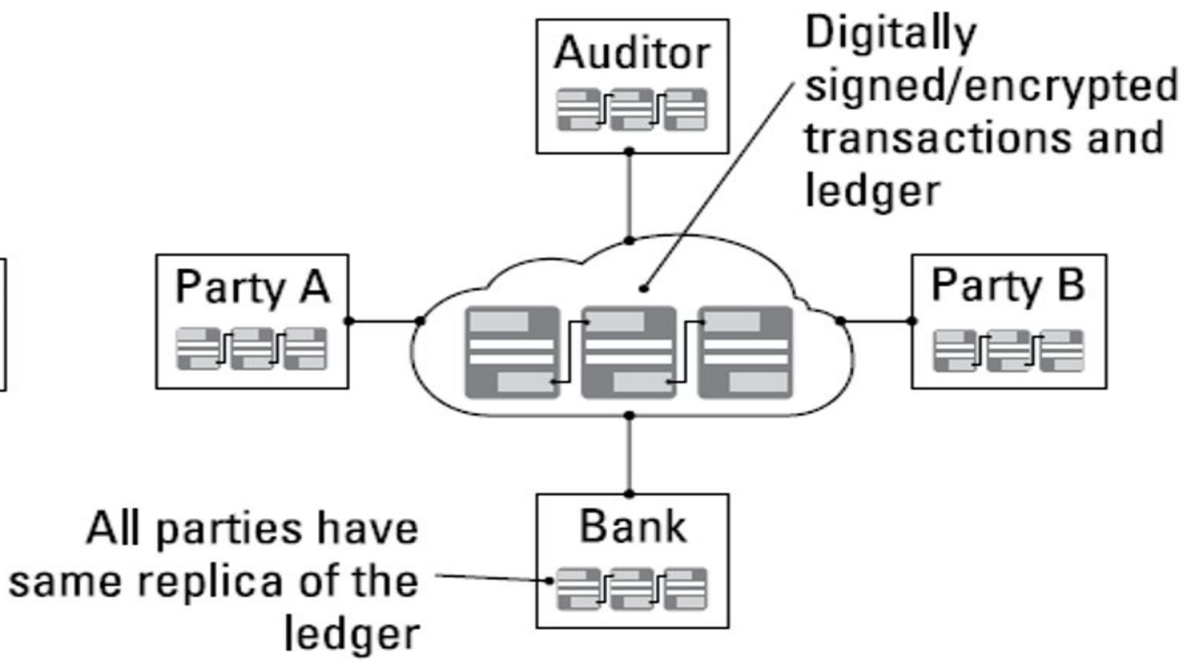
Trading Platform using Hyperledger Fabric

Presented by (Team-6) : Vamshi Krishna Shanagonda
Sai Jeevan Teegala
Akhila Katukuri

Maintenance of Ledgers:

- The shortcomings of current transaction systems




Auditor
Digitally signed/encrypted transactions and ledger
Party A
Party B
Bank
All parties have same replica of the ledger

What is Blockchain Technology

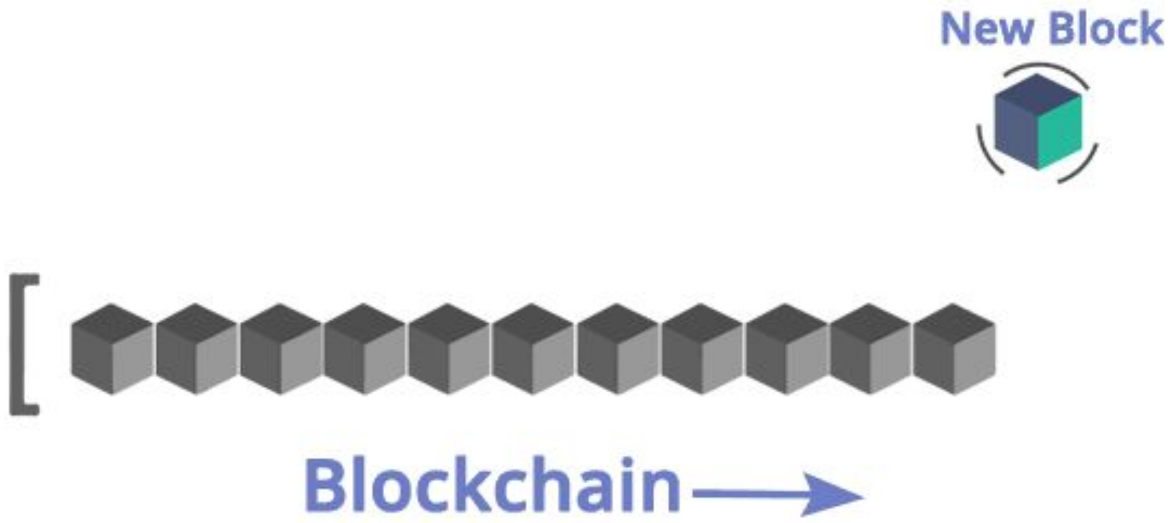
- Blockchain is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system.
- Blockchain is a distributed ledger technology (DLT) that allows data to be stored globally on thousands of servers – while letting anyone on the network see everyone else's entries in near real-time.
- A blockchain is a digital record of transactions. The name comes from its structure, in which individual records, called blocks, are linked together in single list, called a chain. Blockchains are used for recording transactions made with cryptocurrencies, such as Bitcoin, and have many other applications.

Applications of Blockchain

- The first major blockchain innovation was bitcoin, a digital currency experiment.
- The second innovation was called blockchain, which was essentially the realization that the underlying technology that operated bitcoin could be separated from the currency and used for all kinds of other interorganizational cooperation.

- Advantages
  - Fast
  - International
  - Easy accounting
  - Weighs nothing
  - Cheap

Blockchain Structure



4 Key Concepts of Blockchain

Distributed shared ledger

Cryptography

Consensus

Smart contracts

Blockchain Architecture

- Revolutionary Technology
  - TCP/IP, HTTP, Cloud Computation, Big Data, IoT, FinTech…
- Blockchain versions:
  - Blockchain 1.0
    - Bitcoin
    - Programmable Money
  - Blockchain 2.0
    - Ethereum
    - Smart Contract
  - Blockchain 3.0…
    - Non-Financial Uses
- It is also used in many other applications.

Blockchain vs. Current Banking System

- **Decentralized System**
  - The Blockchain system follows a decentralized approach when compared to banks and financial organizations which are controlled and governed by Central or Federal Authorities.
  - Here, everyone who is involved with the system holds some power.
- **Public Ledgers**
  - The ledger which holds the details of all transactions which happen on the Blockchain, is open and completely accessible to everyone who is associated with the system.
  - Even though the complete ledger is publicly accessible, the details of the people involved in the transactions remains completely anonymous.
- **Verification of Every Individual Transaction**
  - Every single transaction is verified by cross-checking the ledger and the validation signal of the transaction is sent after a few minutes.
  - Through the usage of several complex encryption and hashing algorithm, the issue of double spending is eliminated.
- **Low or No Transaction Fees**
  - These transaction fees are however relatively quite less when compared to the fees implied by banks and other financial organizations.
  - If a transaction needs to be completed on priority then an additional transaction fees can be added by the user so as to have the transaction verified on priority.

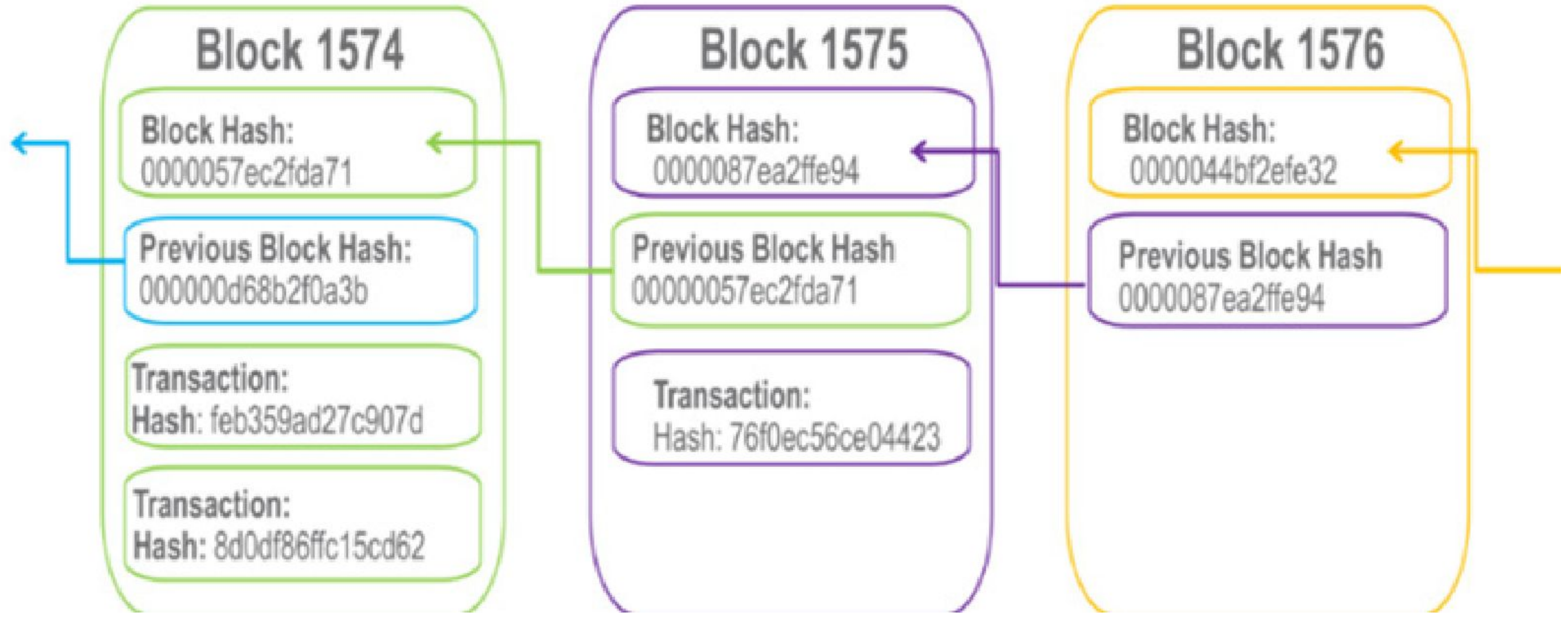
The key business benefits:

- Time efficient
- Cost efficient
- Tighter security
- Enhanced privacy
- Improved auditability
- Increased operational efficiency

Building trust with blockchain

- Distributed and sustainable
- Secure, private, and indelible
- Transparent and auditable
- Consensus-based and transactional
- Orchestrated and flexible

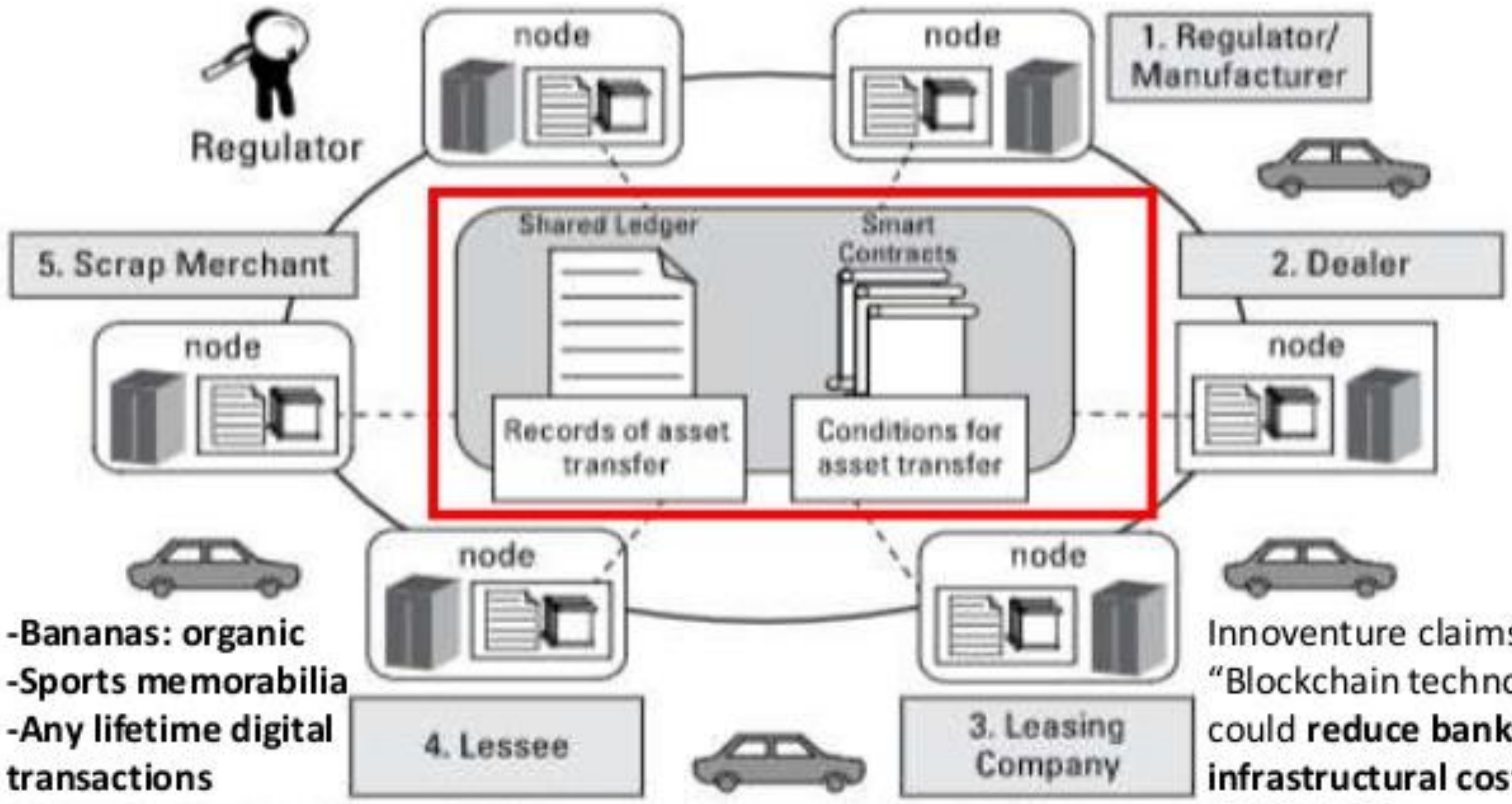
Why It’s Called “Blockchain”



Different Players in Implementation

- Blockchain user
- Regulator
- Blockchain developer
- Blockchain network operator
- Traditional processing platforms
- Traditional data sources
- Certificate authority

BlockChain use-case (dubai)



-Bananas: organic
-Sports memorabilia
-Any lifetime digital transactions

FIGURE 1-3: Tracking vehicle ownership with blockchain.

Innoventure claims that "Blockchain technology could **reduce banks' infrastructural costs by $1.520 trillion a year by 2022**" [2]

HyperLedger Introduction

Hyperledger Fabric

- The Linux Foundation founded Hyperledger in 2015
- Hyperledger Fabric is a platform for distributed ledger solutions in industrial level.
- A modular architecture - Delivers high degrees of confidentiality, resiliency, flexibility and scalability.
- It is designed to support pluggable implementations of different components, and accommodate the complexity and intricacies that exist across the economic ecosystem.
- Breaks from some other blockchain systems is that it is **private** and **permissioned**

Hyperledger Fabric - Cont.

- Like other blockchain technologies, it has a ledger, uses smart contracts, and is a system by which participants manage their transactions.
- Ledger data can be stored in multiple formats, consensus mechanisms can be switched in and out.
- Offers the ability to create channels, allowing a group of participants to create a separate ledger of transactions.
- Hyperledger is based on blockchain but its not a cryptocurrency.
- Hyperledger in general do not enforce any requirements about the hardware, network infrastructures, additional software around it, security models etc.
- Operational power: 0.5 million operations per minute where as other blockchain does only 1000.

Advantages of Hyperledger Fabric

- Permissioned membership
- Performance, scalability, and levels of trust
- Data on a need-to-know basis
- Rich queries over an immutable distributed ledger
- Modular architecture supporting plug-in components
- Protection of digital keys and sensitive data

Hyperledger Components

- **Fabric CA:**

Fabric CA stands for Certificate Authority (CA) for Hyperledger Fabric.It provides features such as: registration of identities into the user registry, issuance of Enrollment Certificates (ECerts), certificate renewal and revocation

- **Peer:**

Peer is the place where the ledger and the blockchain data is stored. You can create backup of the ledger from the peer.

- **Ordering service:**

This is the heart of consensus algorithm, its main role is to provide the order of operations before committing anything to ledger it must pass through the ordering service. It is responsible for verification, security, policy verification etc.

Hyperledger Components

- **Channel:**

Channel is a private “subnet” of communication between two or more specific network members. Channels are completely isolated,they have different ledgers, different height of blocks, policies, stories and rules.

- **Chaincode:**

A chaincode typically handles business logic agreed to by members of the network, so it similar to a “smart contract”. All your business logic is inside the chaincode.

Hyperledger Composer

- 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
- Extensive
- Open development toolset
- Framework to make developing Blockchain applications easier.

Hyperledger Composer

Composer mainly consists of these components:

- Business Network Archive (BNA)
- Composer Playground
- Composer REST Server

Composer Components

- **Business Network Archive** composer allows us to package a few different files and generate an archive which can then be deployed onto a Fabric network. To generate this archive, we need:

Network Model is a definition of the resources present in the network. These resources include Assets, Participants, and Transactions. We will come back to these later.
Business Logic contains all Logic for the transaction functions.
Access Control Limitations contains various rules which define the rights of different participants in the network. This includes, but is not limited to, defining what Assets the Participants can control.
Query File is a set of queries which can be run on the network. These can be thought of as similar to SQL queries.

Composer Components

- **Composer Playground** is a web based user interface that we can use to model and test our business network. Playground is good for modelling simple Proofs of Concept, as it uses the browser’s local storage to simulate the blockchain network. However, if we are running a local Fabric runtime and have deployed a network to it, we can also access that using Playground. In this case, Playground isn’t simulating the network, it’s communicating with the local Fabric runtime directly.
- **Composer REST Server** is a tool which allows us to generate a REST API server based on our business network definition. This API can be used by client applications and allows us to integrate non-blockchain applications in the network.

Project Implementation

Prerequisites required:

- Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
- Docker Engine: Version 17.03 or higher
- Docker-Compose: Version 1.8 or higher
- Node: 6.x (note versions 7 and higher are not supported)
- npm: v3.x or v5.x
- git: 2.9.x or higher
- Python: 2.7.x
- nvm and Apple Xcode (for Mac)
- Hyperledger Composer Extension for VSCode.

Trading Platform

- Basically trading platform is a place where different users who own trading cards come to interchange their cards with others.
- For setting up a trading cards network different traders who own trading cards of Baseball, Football, and Cricket players, will be able to trade cards among themselves.
- This can be done by using the set up of REST API server to allow client side software to interact with our business network.
- We can download the fabric runtime directly using the terminal through npm and start it up. Then we generate a **Peer Admin** card which is basically YOU the developer. Participants in a Fabric network can have business network cards, analogous to real life business cards. Fabric is a base layer for private blockchains to build upon.
- The holder of the Peer Admin business card has the authority to deploy, delete, and manage business networks on this Fabric runtime.
- Then we use commands in NPM to generate boilerplate business networks among other things and generates the empty template network structure .

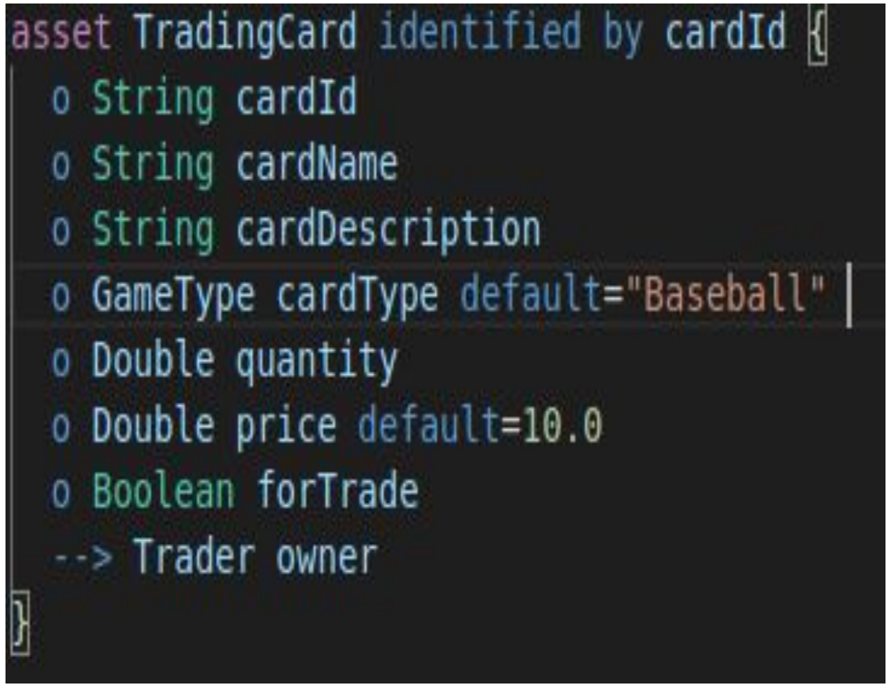
Designing the business network

- The first and most important step towards making a business network is identifying the resources present. We have four resource types in the modeling language:
    1. Assets
    2. Participants
    3. Transactions
    4. Events
- For our trading-cards-platform , we will define an asset type which is the Trading Card , a participant who is the Trader , a transaction of Trading Cards among themselves and an event of Trade Notification when transaction occurs.
- Once we open the generated template files if we open the .cto file which is the modeling file. This contains the specification for our asset Trading Card. All assets and participants need to have a unique identifier for them which we specify in the code, and in our case, it’s IDs.

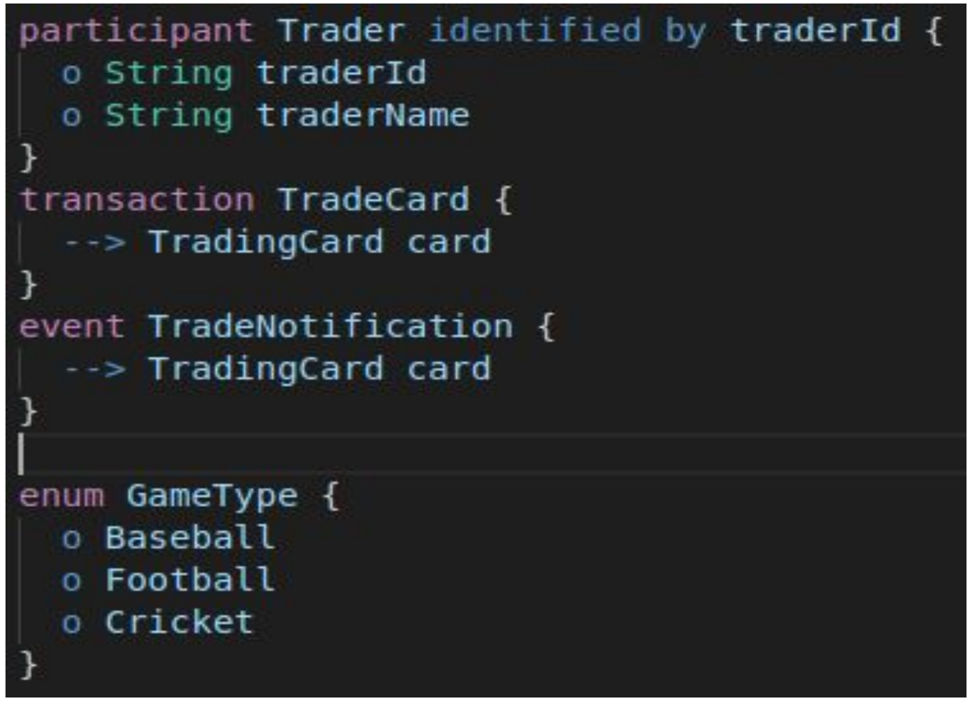
Designing the business network

- Also, our asset has a gametype property for the card. In our project, no Trading card can have a game type other than Baseball, Football, or Cricket
- Now, we also specify our Trader participant resource type.
- We have a participant type Trader and they’re uniquely identified by their IDs.
- By adding a reference to our trading cards we can use this reference to point it to their owners, so we know who the card belongs to. This can be achieved by using the pointer reference.

Code



```
asset TradingCard identified by cardId {
  o String cardId
  o String cardName
  o String cardDescription
  o GameType cardType default="Baseball"
  o Double quantity
  o Double price default=10.0
  o Boolean forTrade
  --> Trader owner
}
```



```
participant Trader identified by traderId {
  o String traderId
  o String traderName
}
transaction TradeCard {
  --> TradingCard card
}
event TradeNotification {
  --> TradingCard card
}

enum GameType {
  o Baseball
  o Football
  o Cricket
}
```

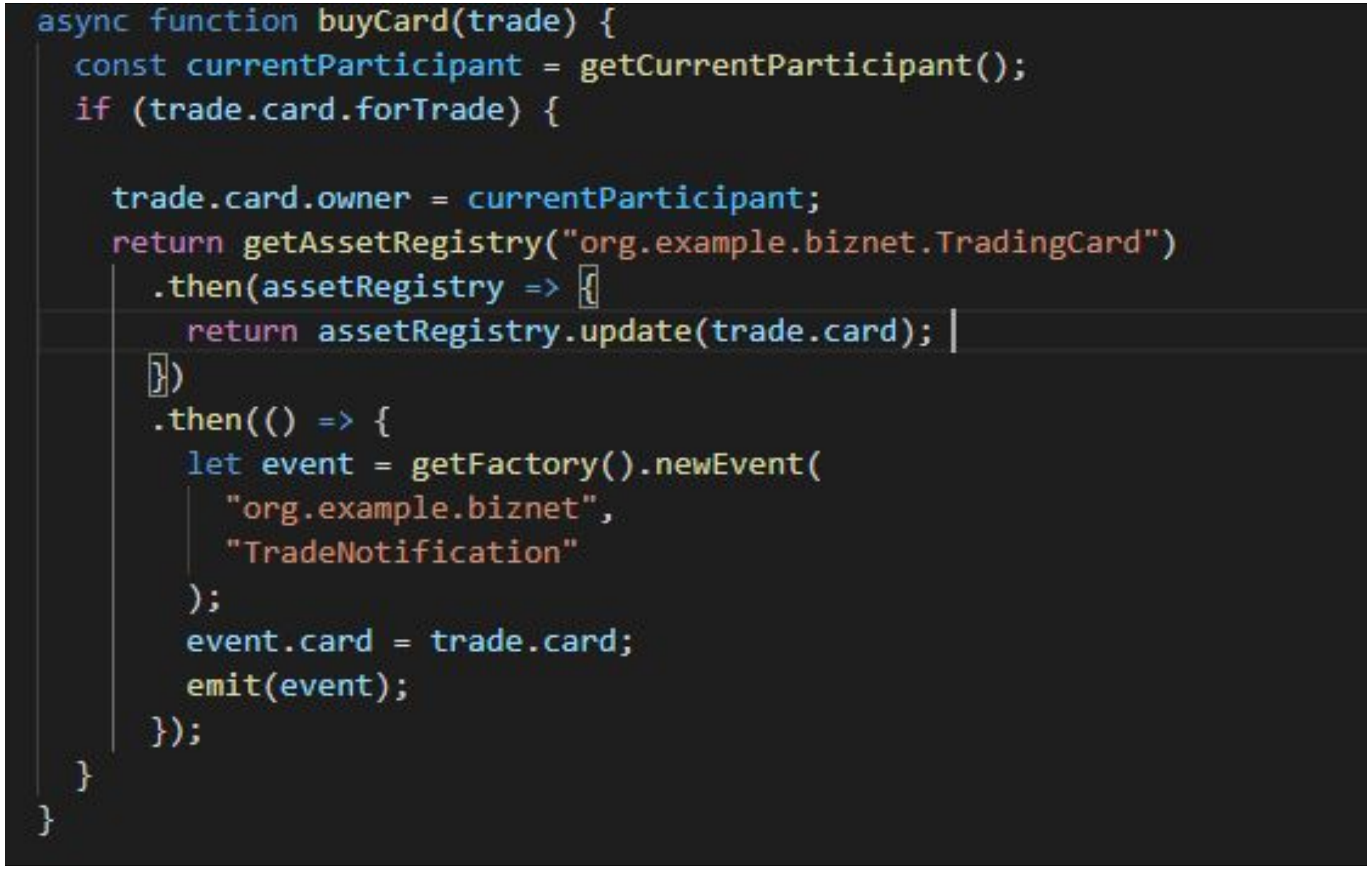
Adding logic for our transactions

- We add logic behind the transaction function using a Javascript logic file, which basically says to start the transaction event when only when the card is ready for trade and other user is trying to but it. Then, it fires off the trade notification event for that card.

Generating a Business Network Archive (BNA):

- To make an archive file for our business network so we deploy it on our local Fabric runtime. This can we done using their respective commands in the terminal.

Code



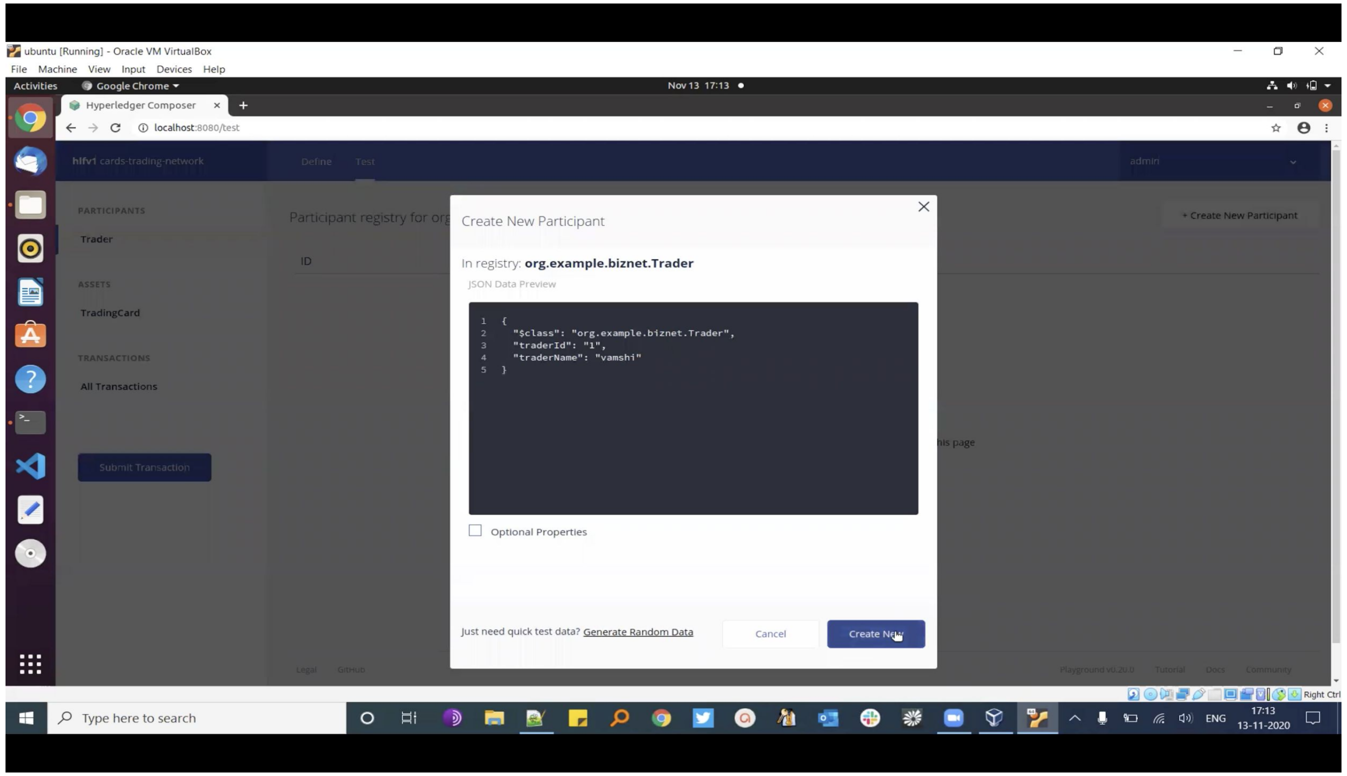
```
async function buyCard(trade) {
  const currentParticipant = getCurrentParticipant();
  if (trade.card.forTrade) {

    trade.card.owner = currentParticipant;
    return getAssetRegistry("org.example.biznet.TradingCard")
      .then(assetRegistry => {
        return assetRegistry.update(trade.card);
      })
      .then(() => {
        let event = getFactory().newEvent(
          "org.example.biznet",
          "TradeNotification"
        );
        event.card = trade.card;
        emit(event);
      });
  }
}
```

Testing our Business Network

- Start Composer Playground to interact with it and the playground opens in the browser:
- Then connect to the trading platform. This is the page is where we can make changes to our code, deploy those changes to upgrade our network, and export business network archives.
- This page contains details to add new participants as the Trader and then we can also add new card along with their details and makes transactions.
- To make client-side software for users to provide them a seamless experience, they don’t even have to necessarily know about the underlying blockchain technology. For that we have the composer-rest-server module to help us with just that. This can be achieved by connecting to an existing business network with the card admin network card, and connect to an existing REST API as well.

Product Results

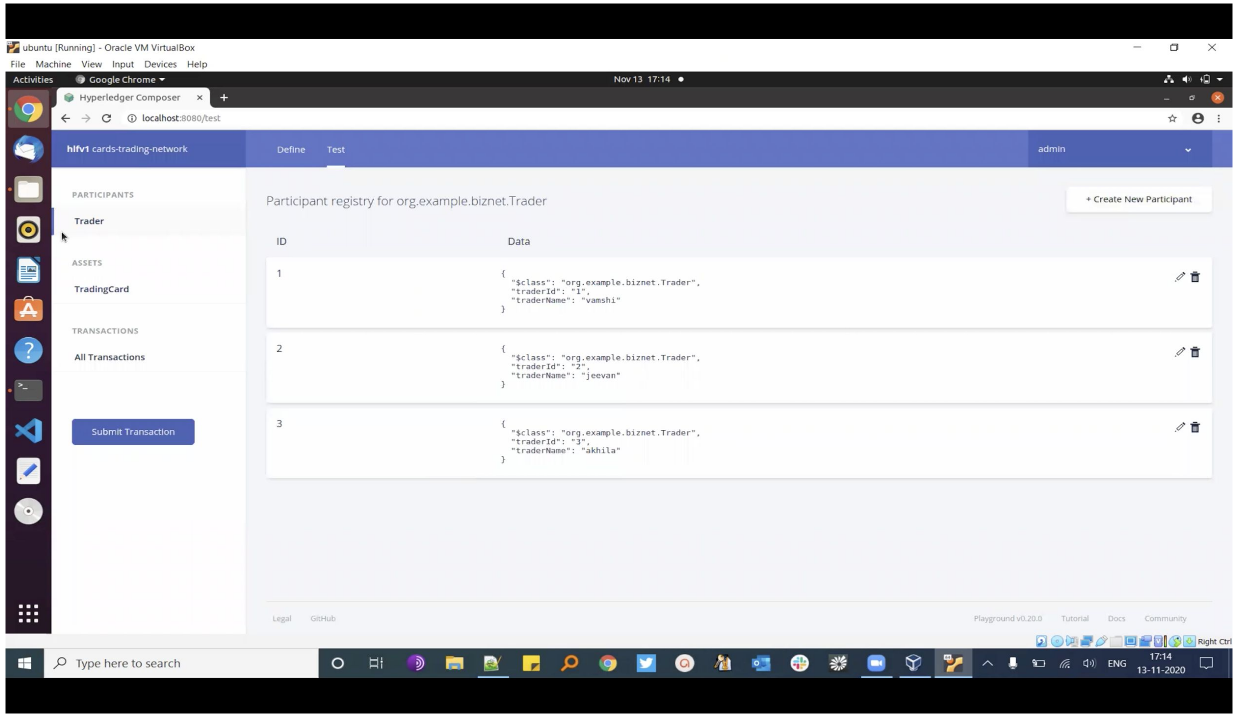
Select Trader from Participants, click on Create New Participant near the top right, and make a new Trader similar to this:



```
{
  "$class": "org.example.biznet.Trader",
  "traderId": "1",
  "traderName": "vamshi"
}
```

Fig 1: Creating New participant

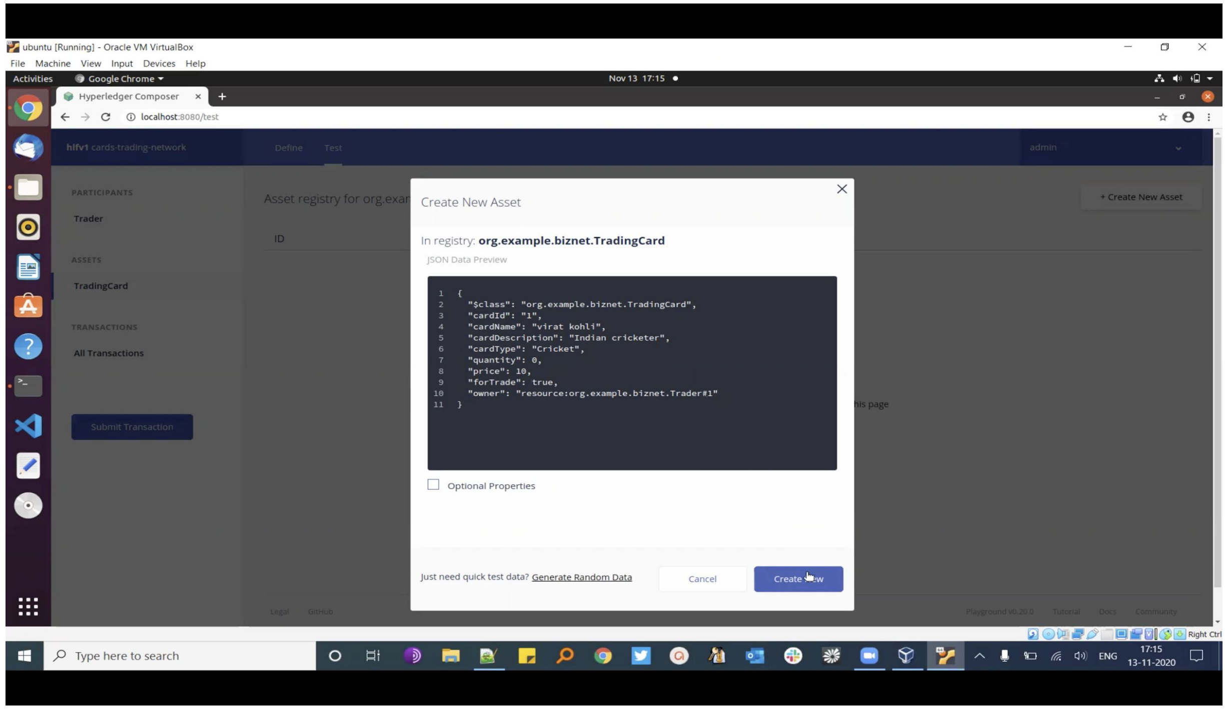
Go ahead and create a couple more Traders. Here are what our three traders look like with the names Vamshi, Jeevan, and Akhila.



Participant registry for org.example.biznet.Trader

ID	Data
1	`{ "$class": "org.example.biznet.Trader", "traderId": "1", "traderName": "vamshi" }`
2	`{ "$class": "org.example.biznet.Trader", "traderId": "2", "traderName": "jeevan" }`
3	`{ "$class": "org.example.biznet.Trader", "traderId": "3", "traderName": "akhila" }`

Fig 2: Created Traders



Create New Asset

In registry: **org.example.biznet.TradingCard**

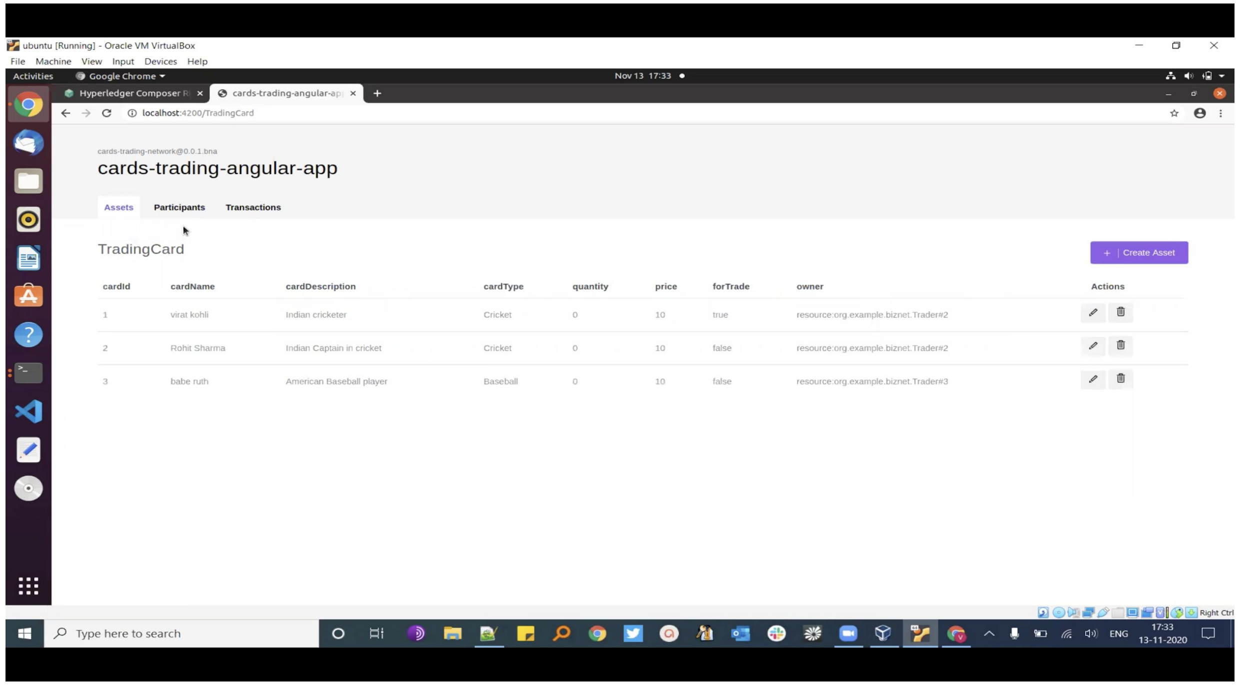
JSON Data Preview

```
{
  "$class": "org.example.biznet.TradingCard",
  "cardId": "1",
  "cardName": "virat kohli",
  "cardDescription": "Indian cricketer",
  "cardType": "Cricket",
  "quantity": 0,
  "price": 10,
  "forTrade": true,
  "owner": "resource:org.example.biznet.Trader#1"
}
```

Just need quick test data? Generate Random Data

Fig 3: Creating New Asset

Now run npm install , give it a minute, and once it's all done we'll be able to load up http://localhost:4200/ and be greeted with a page similar to this:



ubuntu [Running] - Oracle VM VirtualBox
Hyperledger Composer R
cards-trading-angular-app
localhost:4200/TradingCard

cards-trading-network@0.0.1.bna

cards-trading-angular-app

Assets Participants Transactions

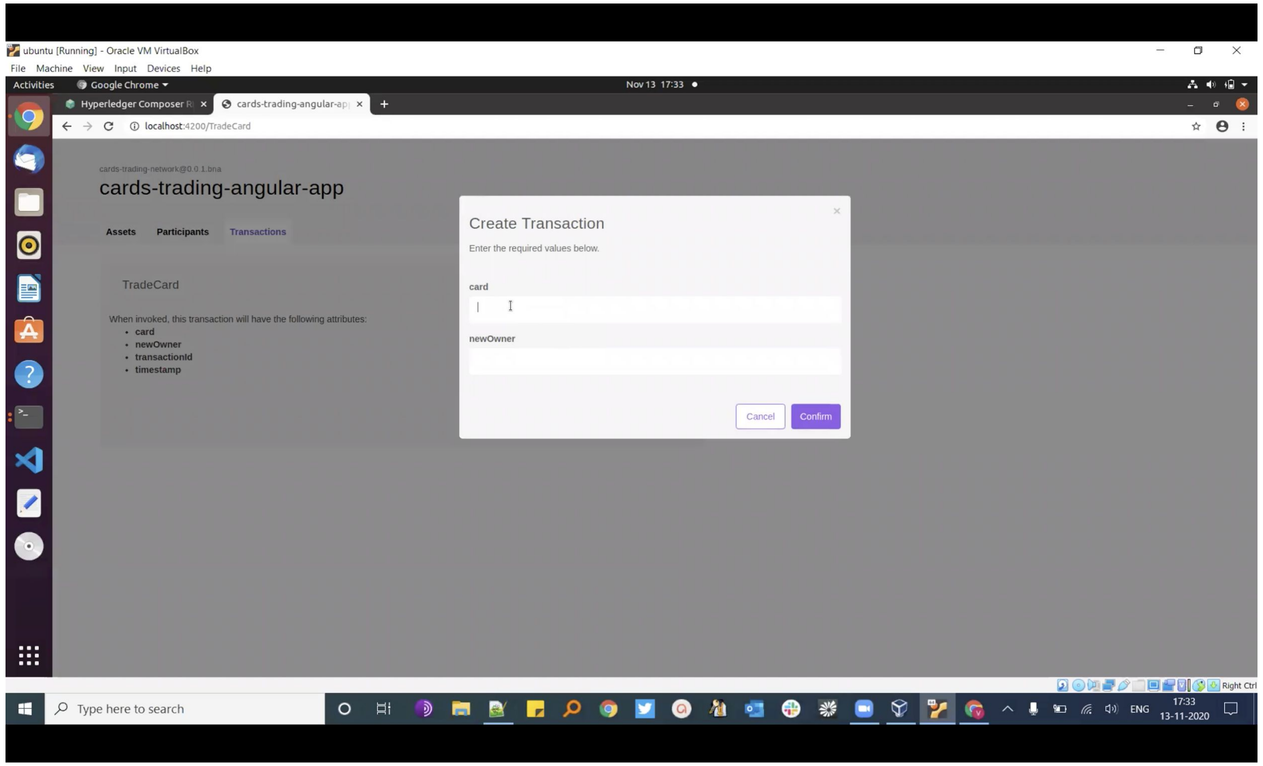
TradingCard

+ Create Asset

cardId	cardName	cardDescription	cardType	quantity	price	forTrade	owner	Actions
1	virat kohli	Indian cricketer	Cricket	0	10	true	resource:org.example.biznet.Trader#2	
2	Rohit Sharma	Indian Captain in cricket	Cricket	0	10	false	resource:org.example.biznet.Trader#2	
3	babe ruth	American Baseball player	Baseball	0	10	false	resource:org.example.biznet.Trader#3	

Fig 4: Cards trading angular app

Now invoke and create a new transaction.



cards-trading-network@0.0.1.bna

cards-trading-angular-app

TradeCard

When invoked, this transaction will have the following attributes:

- card
- newOwner
- transactionId
- timestamp

Create Transaction

Enter the required values below.

card

newOwner

Fig 5: Invoke and create transaction

References

[1] https://hyperledger.github.io/composer/v0.19/tutorials/developer-tutorial

[2] https://blog.codecentric.de/en/2018/04/blockchain-application-fabric-composer/

[3] https://ieeexplore.ieee.org/document/9034576

[4] https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf

Thank You!