

Decentralised Graded System: A Blockchain Application

A Project Report

*submitted in partial fulfillment of the requirement
for the award of the degree of*

Bachelor of Technology

in

COMPUTER ENGINEERING

by

Master Utkarsh Bist

20140653

Miss. Poonam Rajabhau More

20140627

Under the guidance of

Dr. Arvind W. Kiwelekar



DEPARTMENT OF COMPUTER ENGINEERING
DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY
Lonere-402103, Tal. Mangaon, Dist. Raigad (MS) INDIA

2017-2018

Certificate

The report entitled **Decentralised Graded System: A Blockchain Application** submitted by **Master Utkarsh Bist (20140653)** and **Miss. Poonam Rajabhau More (20140627)** is approved for the partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Engineering.

(Dr. Arvind W. Kiwelekar)

Guide

Department of Computer Engineering

(Dr. Arvind W. Kiwelekar)

Head

Department of Computer Engineering

External Examiner(s)

1. _____ (Name: _____)

2. _____ (Name: _____)

Place: Dr. Babasaheb Ambedkar Technological University, Lonere.

Date:

Acknowledgements

This work is just not an individual contribution still its completion. We take this opportunity to thank all for bringing it close to the conclusion. A special thanks goes to our Guide as well as Head of Department **Dr. Arvind W. Kiwelekar** for leading me to many new insights, encouraging and teaching me how to get down to the root of the problem.

We are also grateful to **Prof. H.S.Wankhede** and the department faculty and staff members for their support.

We are also thankful to all friends and well wishers for support during the demanding period of this work. This document, in its eternity (as with everything we have or have yet to accomplish) is attributable to the my parents, whose strength and compassion are my constant inspiration.

We would also like to thank my wonderful colleagues and friends for listening my ideas, asking questions and providing feedback and suggestions for improving my ideas.

Master Utkarsh Bist (20140653)

Miss. Poonam Rajabhau More (20140627)

Abstract

The ongoing expansion of the internet has made virtually gone all online. The technology to support such expansion also evolved with it. But the one thing, that hasn't kept up with the ever expansion of the digital world is the security and proof of work of any transaction or ledger storage in digital world. For this we introduce Blockchain, with a demonstrated project - Distributed Graded System. With this system monitoring degrees, certificates, marksheets would be lot easier. While appending them illegally would be equally difficult to perform. So no more fake certificates to show. Finding true educational background is lot easier with Blockchain.

To be clear, while the blockchain contains transaction data, its not a replacement for databases, messaging technology, transaction processing, or business processes. The blockchain contains verified proof of transactions. However, while blockchain essentially serves as a database for recording transactions, its benefits extend far beyond those of a traditional database.

Contents

1	Introduction	1
1.1	Overview of Blockchain	1
1.2	Working Example of Blockchain	3
1.3	Idea of Decentralised	5
1.3.1	Birth of Decentralised Applications	5
1.4	Overview of Decentralised Graded System	6
1.5	Applying Software Engineering Approach	6
2	Literature Survey	8
2.1	Performance Analysis of Private Blockchain Platforms in Varying Workloads	8
2.2	Blockchain: Future of financial and cyber security	9
2.3	Degree Verification System	9
2.4	The Internet Blockchain: A Distributed, Tamper-Resistant Transaction Framework for the Internet	9
3	Hyperledger	11
3.0.1	Hyperledger Composer	13
3.0.2	Key Concepts of Hyperledger Composer	14
3.0.3	Hyperledger Composer Modeling Language	17
3.0.4	Transaction Processor Functions	18
3.0.5	Access Control Language	19

4	Problem Definition	20
4.1	Existing System	20
4.2	Challenges	20
4.3	Solution	21
4.4	Advantage	22
4.5	Limitations	23
5	Project	24
5.1	UML Diagrams	25
5.2	Decentralised Graded System	26
5.2.1	Participants	27
5.2.2	Asset	29
5.2.3	Transaction	30
5.2.4	Permissions	37
6	System Requirement Specification	39
6.1	Software Requirements	39
6.2	Hardware Requirements	40
7	Screenshots	41
8	Conclusion	48
	Bibliography	49

List of Figures

1.1	Blockchain stores transaction records in a series of connected blocks . . .	2
1.2	Centralised ledger	3
1.3	Blockchain decentralised ledger	4
3.1	Hyperledger Modular Umbrella Approach	12
3.2	Hyperledger platform flow	13
3.3	Hyperledger Composer	14
4.1	Inter Institution Network Currently	21
4.2	Inter Institution Network on Blockchain	22
5.1	Use-Case Diagram	25
5.2	Sequence Diagram	26
5.3	Professor Participant	27
5.4	Student Participant	28
5.5	Public Participant	29
5.6	Grade Asset	29
5.7	GradeMarkProfessor Transaction	30
5.8	GradeMarkStudent Transaction	32
5.9	Change Transaction	34
7.1	Business Network	41
7.2	Create Professor	42
7.3	Create Public	42
7.4	Create Student	43

7.5	Create Grade	43
7.6	Grade updation by Professor	44
7.7	Grade updation by Student	44
7.8	Processing State	45
7.9	State change by Professor	45
7.10	INVALID state	46
7.11	Transaction Record	46
7.12	Finalised State	47

Chapter 1

Introduction

1.1 Overview of Blockchain

Blockchain is a shared, distributed ledger that facilitates the process of recording transactions and tracking assets in a business network. An asset can be tangible a house, a car, cash, land or intangible like intellectual property, such as patents, copyrights, or branding. Virtually anything of value can be tracked and traded on a blockchain network, reducing risk and cutting costs for all involved.

Bitcoin is actually built on the foundation of blockchain, which serves as bitcoin's shared ledger. Think of blockchain as an operating system, such as Microsoft Windows or MacOS, and bitcoin as only one of the many applications that can be run on that operating system. Blockchain provides the means for recording bitcoin transactions the shared ledger but this shared ledger can be used to record any transaction and track the movement of any asset whether tangible, intangible, or digital. For example, blockchain enables securities to be settled in minutes instead of days. It can also be used to help companies manage the flow of goods and related payments, or enable manufacturers to share production logs with Original Equipment Manufacturers (OEMs) and regulators to reduce product recalls. Bitcoin and blockchain are not the same. Blockchain provides the means to record and store bitcoin transactions, but blockchain has many uses beyond bitcoin. Bitcoin is only the first use case for blockchain.

Blockchain owes its name to the way it stores transaction data in blocks that are linked together to form a chain. As the number of transactions grows, so does the blockchain. Blocks record and confirm the time and sequence of transactions, which are then logged into the blockchain, within a discrete network governed by rules agreed on by the network participants.

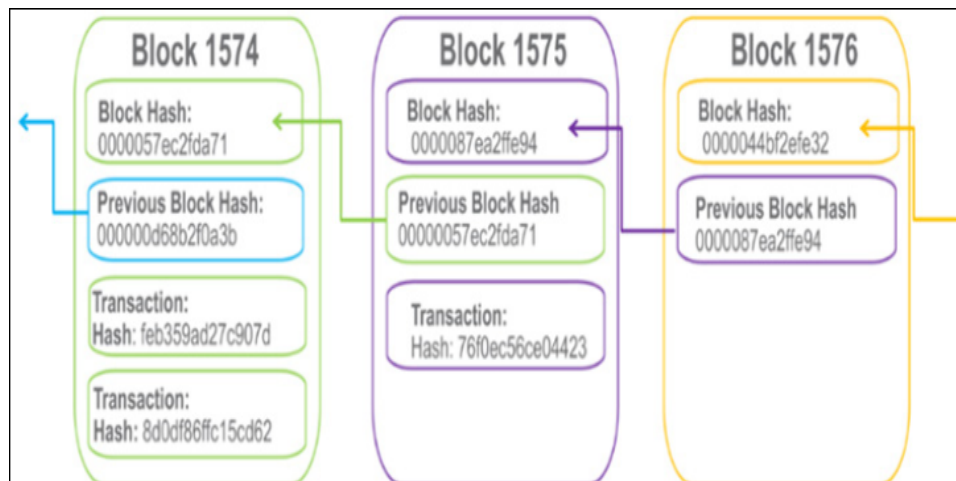


Figure 1.1: Blockchain stores transaction records in a series of connected blocks
credit: Blockchain for Dummies- IBM limited Edition- edition 2017, Page 19.

Each block contains a hash (a digital fingerprint or unique identifier), timestamped batches of recent valid transactions, and the hash of the previous block. The previous block hash links the blocks together and prevents any block from being altered or a block being inserted between two existing blocks. In this way, each subsequent block strengthens the verification of the previous block and hence the entire blockchain. The method renders the blockchain tamper-evident, lending to the key attribute of immutability.

1.2 Working Example of Blockchain

From a cruising altitude, a blockchain might not look that different from things you're familiar with, say Wikipedia.

With a blockchain, many people can write entries into a record of information, and a community of users can control how the record of information is amended and updated. Likewise, Wikipedia entries are not the product of a single publisher. No one person controls the information.

Descending to ground level, however, the differences that make blockchain technology unique become more clear. While both run on distributed networks (the internet), Wikipedia is built into the World Wide Web (WWW) using a client-server network model. A user (client) with permissions associated with its account is able to change Wikipedia entries stored on a centralized server.

Whenever a user accesses the Wikipedia page, they will get the updated version of the 'master copy' of the Wikipedia entry. Control of the database remains with Wikipedia administrators allowing for access and permissions to be maintained by a central authority.

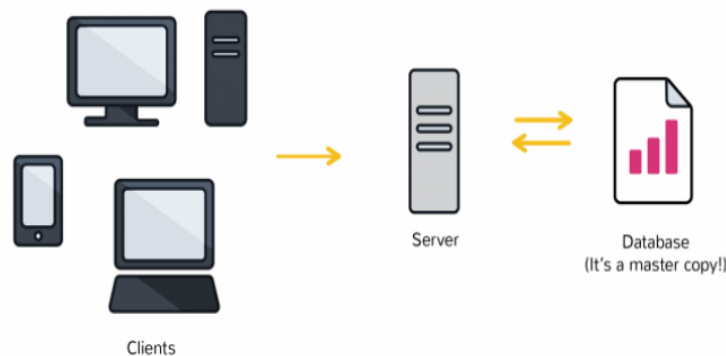


Figure 1.2: Centralised ledger

credit: <https://www.coindesk.com/information/what-is-blockchain-technology/>

Wikipedia's digital backbone is similar to the highly protected and centralized databases that governments or banks or insurance companies keep today. Control of centralized

databases rests with their owners, including the management of updates, access and protecting against cyber-threats.

The distributed database created by blockchain technology has a fundamentally different digital backbone. This is also the most distinct and important feature of blockchain technology.

Wikipedia's 'master copy' is edited on a server and all users see the new version. In the case of a blockchain, every node in the network is coming to the same conclusion, each updating the record independently, with the most popular record becoming the de-facto official record in lieu of there being a master copy. Transactions are broadcast,



Figure 1.3: Blockchain decentralised ledger

credit: <https://www.coindesk.com/information/what-is-blockchain-technology/>

and every node is creating their own updated version of events. It is this difference that makes blockchain technology so useful - It represents an innovation in information registration and distribution that eliminates the need for a trusted party to facilitate digital relationships.

Yet, blockchain technology, for all its merits, is not a new technology.

Rather, it is a combination of proven technologies applied in a new way. It was the particular orchestration of three technologies (the Internet, private key cryptography and a protocol governing incentivization) that made bitcoin creator Satoshi Nakamoto's idea so useful.

1.3 Idea of Decentralised

By design, the blockchain is a decentralized technology.

Anything that happens on it is a function of the network as a whole. Some important implications stem from this. By creating a new way to verify transactions aspects of traditional commerce could become unnecessary. Stock market trades become almost simultaneous on the blockchain, for instance or it could make types of record keeping, like a land registry, fully public. And decentralization is already a reality.

A global network of computers uses blockchain technology to jointly manage the database that records Bitcoin transactions. That is, Bitcoin is managed by its network, and not any one central authority. Decentralization means the network operates on a user-to-user (or peer-to-peer) basis. The forms of mass collaboration this makes possible are just beginning to be investigated.

1.3.1 Birth of Decentralised Applications

As the concept is still in its infancy, there might not be one definition of what a Dapp is. However, there are noticeable common features of Dapps:

- (1) Open Source: Ideally, it should be governed by autonomy and all changes must be decided by the consensus, or a majority, of its users. Its code base should be available for scrutiny.
- (2) Decentralized: All records of the applications operation must be stored on a public and decentralized blockchain to avoid pitfalls of centralization.
- (3) Incentivized: Validators of the blockchain should be incentivized by rewarding them accordingly with cryptographic tokens.
- (4) Protocol: The application community must agree on a cryptographic algorithm to show proof of value. For example, Bitcoin uses Proof of Work (PoW) and Ethereum is currently using PoW with plans for a hybrid PoW/Proof of Stake (PoS)⁵ in the future.

If we adhere to the above definition, the first Dapp was in fact Bitcoin itself. Bitcoin is an implemented blockchain solution that arose from problems revolving around centralization and censorship. One can say Bitcoin is a self-sustaining public ledger that allows efficient transactions without intermediaries and centralized authorities.

1.4 Overview of Decentralised Graded System

With the ever evolving technology growth, the proof of someone's education qualification can easily be tempered with. We often come across , especially in political environments, questions about the candidates true education backgrounds. Its not at all difficult for people to forge fake marksheet, certificate,degree or what not. And at the same time its equally difficult to cross check the true credential of their presented certificates. All you need is a few corrupt minds. These faking of certificates is possible only because each and every educational institute keep separate centralized database. Not only it dents the transparency, but also the process of retrieving ones true credentials.

One of the possible and feasible solution is to maintain a common database in a cloud. Maintain as well as verify it though blockchain hashing technique and proof of work. The way it will work is first the faulty will upload marksheet in the cloud which can be viewed by all student plus the authority authorized to view in only read data. A copy of the cloud data will be stored in there local database. Any change in cloud data will instantaneous will change the data stored in each institute database. By this, all will know what is appended. If legal, all will give the proof of work or else the the illegal change would be undone.

1.5 Applying Software Engineering Approach

The whole software engineering approach is applied while implementing the project. The detailed 4 phases of software engineering are applied to the application. Following phases of software engineering are applied to the project:

I. Requirement Gathering: In this phase we had collected all the information about

our project considering the end user. The whole information gathering process is done considering the intended audience as the end user or the user who will going to use the system. The survey about which language to use , which server to use, which database system to be use is done in the requirement gathering stage.

- II. Design: After requirement gathering the next stage is design phase. In design phase the detailed design process is carried out. In designing phase all UML diagram of the project are drawn such as data ow, component, deployment, class diagram, software architecture of the system is drawn to understand the overall flow of the system.
- III. Coding: The coding is the phase which have to carried out when detailed designing of the project is complete. So after design phase the coding is carried out according to the design. Before the coding it is compulsory to choose the right programming language in which coding is to done.
- IV. Testing: The important phase in the software engineering process is the testing phase. The lot of errors are occurred during the testing phase of the system. The testing is done to check whether the system works as intended or it giving any unexpected bug. The testing can be carried out using any testing method.

Chapter 2

Literature Survey

2.1 Performance Analysis of Private Blockchain Platforms in Varying Workloads

The paper titled Performance Analysis of Private Blockchain Platforms in Varying Workloads by Suporn Pongnumkul, Chaiyaphum Siripanpornchana and Suttipong Thajchayapong performance analysis of two popular private blockchain platforms, Hyperledger Fabric and Ethereum (private deployment), to assess the performance and limitations of these state-of-the-art platforms. Many industries have become interested in adopting blockchain in their IT systems, but scalability is an often-cited concern of current blockchain technology. Therefore, the goals of this preliminary performance analysis are twofold. First, a methodology for evaluating a blockchain platform is developed. Second, the analysis results are presented to inform practitioners in making decisions regarding adoption of blockchain technology in their IT systems. The experimental results, based on varying number of transactions, show that Hyperledger Fabric consistently outperforms Ethereum across all evaluation metrics which are execution time, latency and throughput.

2.2 Blockchain: Future of financial and cyber security

The paper titled Blockchain: Future of financial and cyber security by Sachchidanand Singh and Nirmala Singh explains Blockchain as a decentralized ledger used to securely exchange digital currency, perform deals and transactions. Each member of the network has access to the latest copy of encrypted ledger so that they can validate a new transaction. Blockchain ledger is a collection of all Bitcoin transactions executed in the past. Basically, it's a distributed database which maintains a continuously growing tamper proof data structure blocks which holds batches of individual transactions. The completed blocks are added in a linear and chronological order. Each block contains a timestamp and information link which points to a previous block.

2.3 Degree Verification System

The paper titled "Degree Verification System" points out the flaws about the current Degree verification system. The process is lengthy and often not really efficient enough. The system starts with contacting with the student section of the respective institution and clear any issues present. confirm the details of the degree holder. Verification without checking your records and your verification is incorrect, you are responsible for reordering and paying any fees associated with your verification order. These kind of system often is confidential keeping the details of degree stored securely in database of college.

2.4 The Internet Blockchain: A Distributed, Tamper-Resistant Transaction Framework for the Internet

The paper titled The Internet Blockchain: A Distributed, Tamper-Resistant Transaction Framework for the Internet by Adishesu Hari and T.V. Lakshman proposes the

use of a blockchain based mechanism to secure the Internet BGP and DNS infrastructure. While the blockchain has scaling issues to be overcome, the key advantages of such an approach include the elimination of any Public Key Infrastructure (PKI) - like root of trust, a verifiable and distributed transaction history log, multi-signature based authorizations for enhanced security, easy extensibility and scriptable programmability to secure new types of Internet resources and potential for a built in cryptocurrency. A tamper resistant DNS infrastructure also ensures that it is not possible for the application level PKI to spoof HTTPS traffic.

Chapter 3

Hyperledger

Hyperledger is an open source collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration, hosted by The Linux Foundation, including leaders in finance, banking, Internet of Things, supply chains, manufacturing, and Technology.

Hyperledger does not support Bitcoin or any other cryptocurrency. But the platform is thrilled by blockchain technology. Not since the Web itself, the website tells, has a technology promised broader and more fundamental revolution than blockchain technology. Blockchains has the potential to build a new generation of transactional applications that establishes trust, accountability, and transparency at their core while streamlining business processes and legal constraints.

So we have a lot of promises and we have Hyperledger. With it, the Linux Foundation aims to create an environment in which communities of software developer and companies meet and coordinate to build blockchain frameworks. The Linux Foundation founded the platform in December 2015. In February 2016 it announced the first founding members, in March 2016 ten more members joined. Today Hyperledger has an impressive list of more than 100 members. The list covers a wide scope of well know industry leaders. It includes mobility tech giants like Airbus and Daimler, IT-companies like IBM, Fujitsu, SAP, Huawei, Nokia, Intel and Samsung, financial institutions like Deutsche Brse, American Express, J.P. Morgan, BBVA, BNP Paribas and Well Fargo, as well as Blockchain startups like Blockstream, Netki, Lykke, Factom, bloq and Con-

sensys. A lot of the worlds largest companies in Tech and Finance meet at Hyperledger with some of the hottest blockchain startups.

As of October 2017, Hyperledger consists of eight projects, five of which are distributed ledger frameworks. The other three projects are modules that support these frameworks.

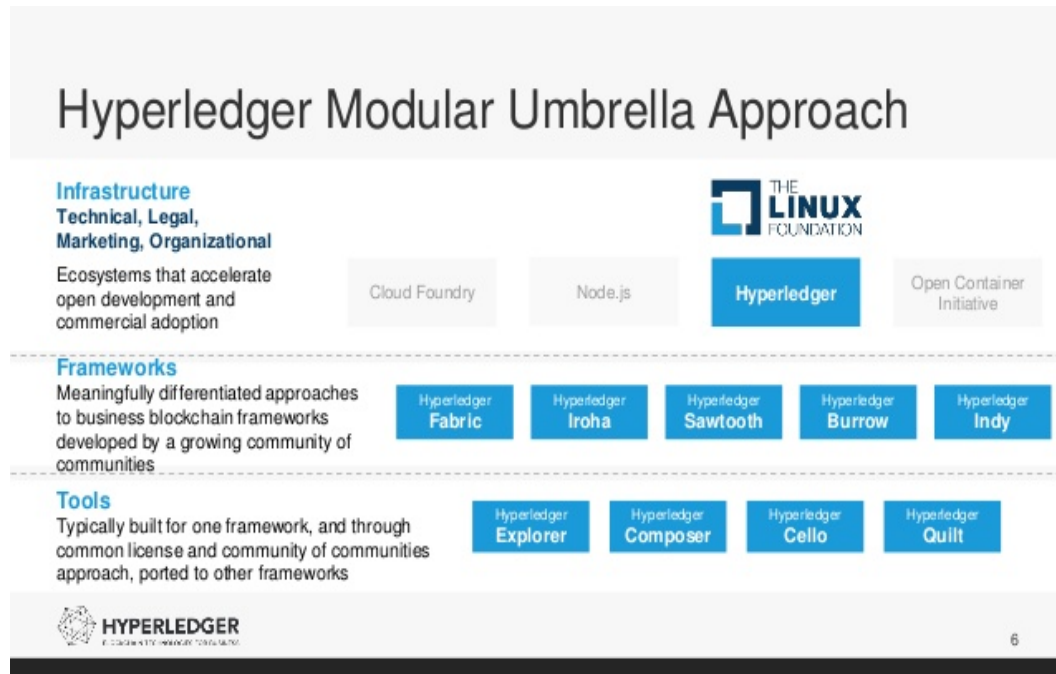


Figure 3.1: Hyperledger Modular Umbrella Approach

Flow of Hyperledger is:

- (1) Install Hyperledger Composer development tools.
- (2) Start Hyperledger Fabric.
- (3) Generate the Business Network Archive file.
- (4) Deploy the Business Network Archive using Composer Playground.

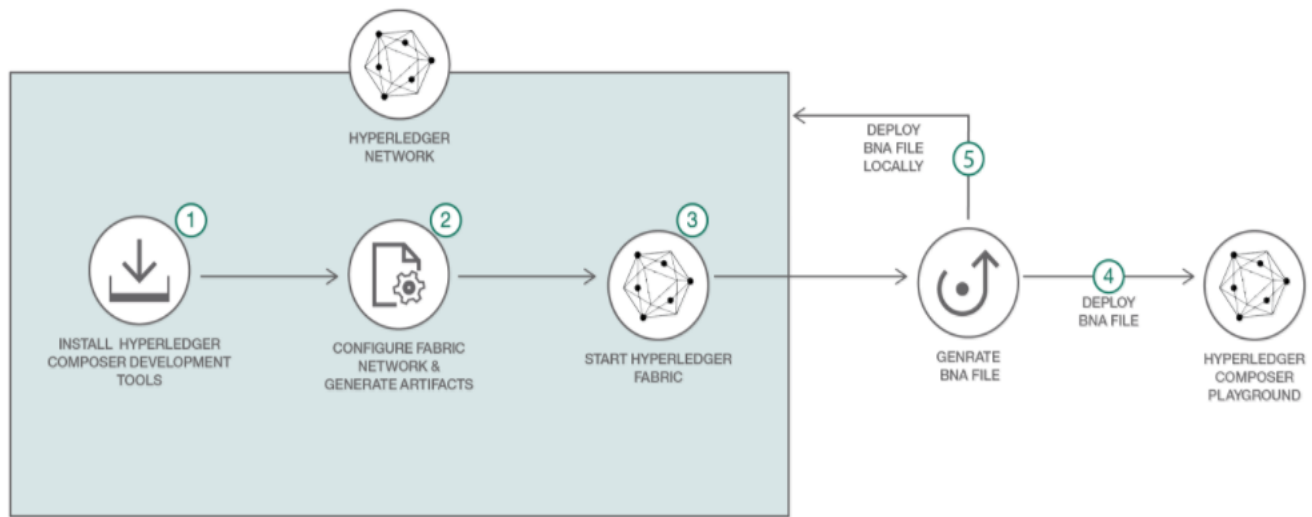


Figure 3.2: Hyperledger platform flow

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

- (4) (Alternative method) Deploy the Business Network Archive on Hyperledger Composer running locally.

3.0.1 Hyperledger Composer

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

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.

You can use Hyperledger Composer to quickly model your current business network, containing your existing assets and the transactions related to them; assets are tangible or intangible goods, services, or property. As part of your business network model, you define 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.

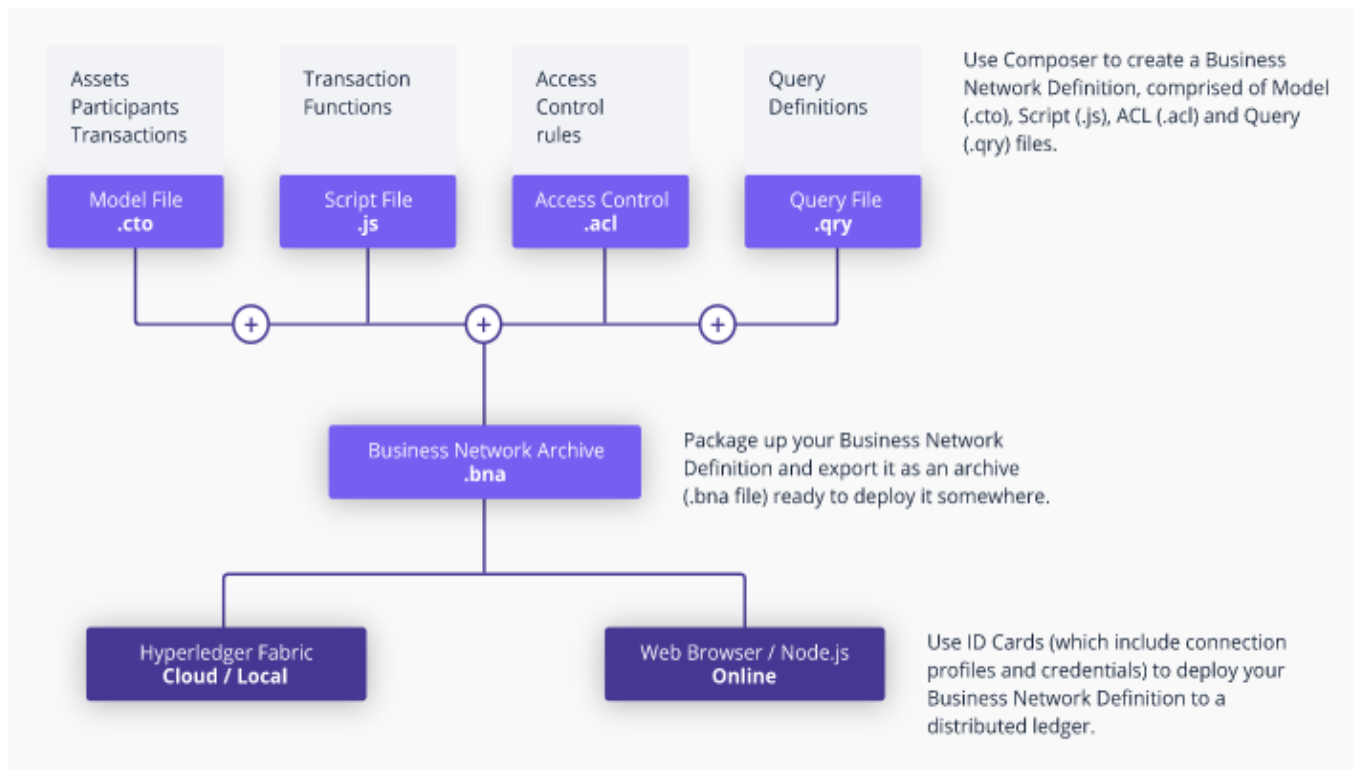


Figure 3.3: Hyperledger Composer

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

3.0.2 Key Concepts of 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 that 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.

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 net-

work 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 specialised registry which records successful transactions, including the participants and identities that submitted them. The historian stores transactions as `HistorianRecord` assets, which are defined in the Hyperledger Composer system namespace.

3.0.3 Hyperledger Composer Modeling Language

Hyperledger Composer includes an object-oriented modeling language that is used to define the domain model for a business network definition.

A Hyperledger Composer CTO file is composed of the following elements:

- (1) A single namespace. All resource declarations within the file are implicitly in this namespace.
- (1) A set of resource definitions, encompassing assets, transactions, participants, and events.
- (3) Optional import declarations that import resources from other namespaces.

Your organization namespace is defined in the namespace line of your model (.cto) file, and all resources created are implicitly part of this namespace.

As well as defining new classes of asset, participant, event, and transaction, there is a system namespace which contains the base definitions of asset, event, participant, and transaction. These base definitions are abstract types which are implicitly extended by all assets, events, participants, and transactions.

In the system namespace definitions, asset and participant have no required values. Events and transactions are defined by an `eventId` or `transactionId` and a timestamp.

The system namespace also includes definitions of registries, historian records, identities, and a number of system transactions.

3.0.4 Transaction Processor Functions

A Hyperledger Composer Business Network Definition is composed of a set of model files and a set of scripts. The scripts may contain transaction processor functions that implement the transactions defined in the Business Network Definition's model files.

Transaction processor functions are automatically invoked by the runtime when transactions are submitted using the `BusinessNetworkConnection` API.

Decorators within documentation comments are used to annotate the functions with metadata required for runtime processing.

Each transaction type has an associated registry storing the transactions.

Transaction processor structure

The structure of transaction processor functions includes decorators and metadata followed by a JavaScript function, both parts are required for a transaction processor function to work.

The first line of comments above a transaction processor function contains a human readable description of what the transaction processor function does. The second line must include the `@param` tag to indicate the parameter definition. The `@param` tag is followed by the resource name of the transaction which triggers the transaction processor function, this takes the format of the namespace of the business network, followed by the transaction name. After the resource name, is the parameter name which will reference the resource, this parameter must be supplied to the JavaScript function as an argument. The third line must contain the `@transaction` tag, this tag identifies the code as a transaction processor function and is required.

```
/**
 * A transaction processor function description
 * @param {org.example.basic.SampleTransaction} parameter-name A human
   description of the parameter
```

```
* @transaction
*/
function transactionProcessor(parameter-name) {
    //Do some things.
}
```

3.0.5 Access Control Language

Develop includes an access control language (ACL) that provides declarative access control over the elements of the domain model. By defining ACL rules you can determine which users/roles are permitted to create, read, update or delete elements in a business network's domain model.

Develop differentiates between access control for resources within a business network (business access control) and access control for network administrative changes (network access control). Business access control and network access control are both defined in the access control file (.acl) for a business network.

Network access control uses the system namespace, which is implicitly extended by all resources in a business network; and grants or denies access to specific actions as defined below, and is intended to allow for more nuanced access to certain network-level operations.

Chapter 4

Problem Definition

4.1 Existing System

With traditional methods for recording transactions- marksheet, degree or certificates and tracking assets, institutions on a network keep their own ledgers and other records, as shown in the figure 3.1 . This traditional method can be expensive, partially because it involves intermediaries that charge fees for their services. Its clearly inefficient and prone to mishandling due to delays in executing agreements and the duplication of effort required to maintain numerous ledgers. Its also vulnerable because if a central system is compromised, due to fraud, cyberattack, or a simple mistake, the entire inter-institution network is affected.

4.2 Challenges

Throughout history, instruments of trust, such as minted coins, paper money, letters of credit, and banking systems, have emerged to facilitate the exchange of value and protect buyers and sellers. Important innovations, including telephone lines, credit card systems, the Internet, and mobile technologies have improved the convenience, speed, and efficiency of transactions while shrinking and sometimes virtually eliminating the distance between buyers and sellers. Still, many business transactions remain inefficient,

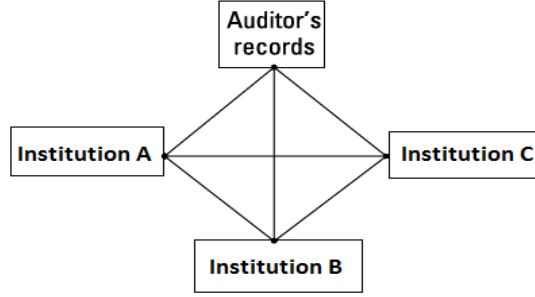


Figure 4.1: Inter Institution Network Currently

expensive, and vulnerable. Such problems also exists in the traditional graded system.

- (1) The time between transaction and settlement can be long.
- (2) Duplication of effort and the need for third-party validation and/or the presence of intermediaries add to the inefficiencies.
- (3) Fraud, cyberattacks, and even simple mistakes add to the cost and complexity of doing business, and they expose all institution member in the network to risk if a central system is compromised.
- (4) Auditing of institutions , which often involves considerable paperwork and a time-consuming vetting process.

4.3 Solution

The figure 4.1 represents institution networks that uses blockchain. The blockchain architecture gives participants the ability to share a ledger that is updated, through peer-to-peer replication, every time a transaction occurs. Peer to- peer replication means that each participant (node) in the network acts as both a publisher and a subscriber. Each node can receive or send transactions to other nodes, and the data is synchronized across

the network as it is transferred.

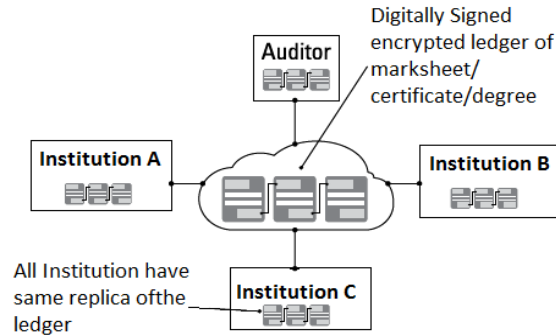


Figure 4.2: Inter Institution Network on Blockchain

The participants in both transaction (Traditional or Revised) systems are the same. What has changed is that the transaction record is now shared and available to all parties.

4.4 Advantage

- (1) Consensus: For a transaction to be valid, all participants must agree on its validity.
- (2) Provenance: Participants know where the asset came from and how its ownership has changed over time.
- (3) Immutability: No participant can tamper with a transaction after its been recorded to the ledger. If a transaction is in error, a new transaction must be used to reverse the error, and both transactions are then visible.
- (4) Finality: A single, shared ledger provides one place to go to determine the ownership of an asset or the completion of a transaction.

4.5 Limitations

- (1) Large Energy Consumption: The Bitcoin blockchain network's miners are attempting 450 thousand trillions solutions per second in efforts to validate transactions, using substantial amounts of computing power.
- (2) Cultural adoption: Blockchain represents a complete shift to a decentralized network which requires the buy-in of its users and operators.
- (3) Cost: Blockchain offers tremendous savings in transaction costs and time but the high initial capital costs could be deterrent.
- (4) Integration Concerns: Blockchain applications offers solutions that require significant changes to, or complete replacement of, existing systems. In order to make the switch, companies must strategize the transition.

Chapter 5

Project

To resolve the problem discussed in earlier chapter through blockchain, a sample construction of a simple distributed Graded System has been created resolving all the below problem definations:

- (1) All, including Professor and Student, should have the authourity to update any student marks.
- (2) Marks Updated by Professor should be marked as Valid transaction.
- (3) Marks Updated by Student should be marked as Valid transaction only by Professor.
- (4) After a particular time period set by Professor, the marks has to be submitted and no more updation by professor or student can take place.
- (5) All transaction ever happen has to be recorded with a timestamp for future references.

5.1 UML Diagrams

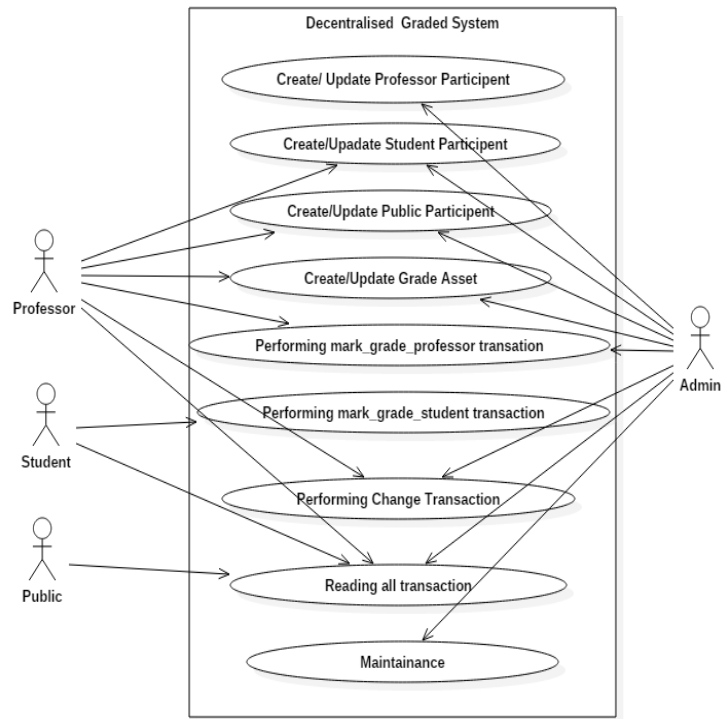


Figure 5.1: Use-Case Diagram

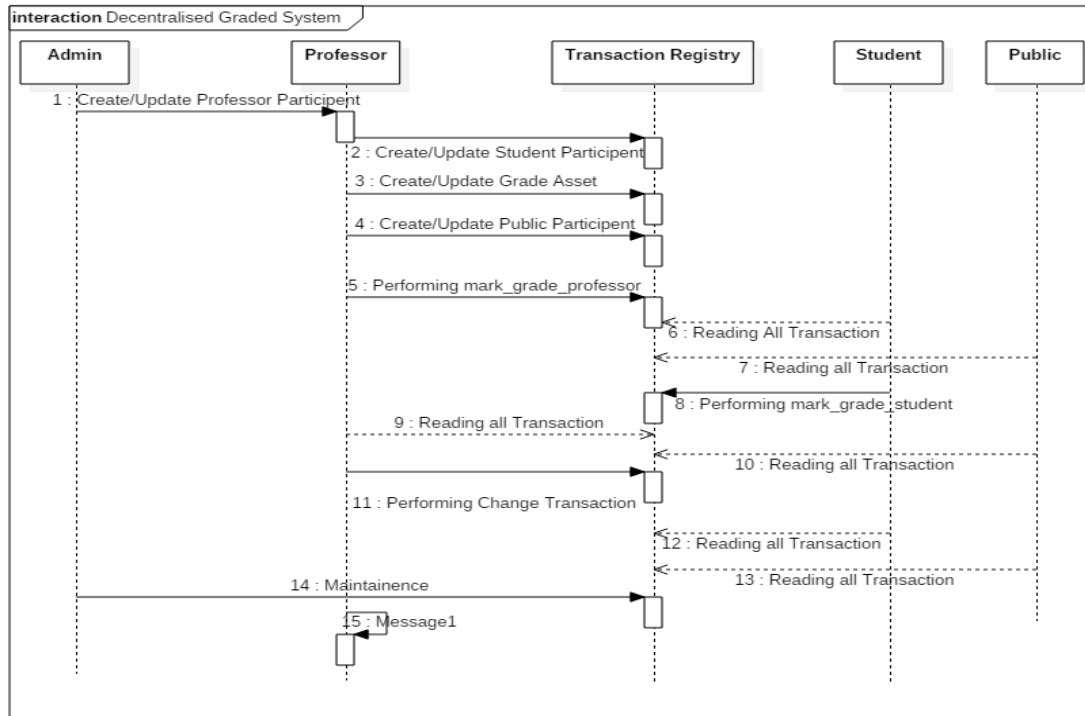


Figure 5.2: Sequence Diagram

5.2 Decentralised Graded System

- (1) Participants: Professor, Student, Public
- (2) Assets: Grade
- (3) Transaction: MarkGradePprofessor, MarkGradeStudent, change

5.2.1 Participants

Professor

```
participant Professor identified by professor_id
{
  o String professor_id
  o String first_name
  o String last_name
  o Uni university
  o String test optional
}
```

Figure 5.3: Professor Participant

- (1) ProfessorID : uniquely identifies Professor
- (2) First Name : Tells you the initials of the professor
- (3) Last Name : Tells you the surname of the professor
- (4) University : The university he is associate with
- (5) Test : Tests the avaiability of the professor

Student

- (1) StudentID : uniquely identifies Student
- (2) First Name : Tells you the initials of the student
- (3) Last Name : Tells you the surname of the student
- (4) University : The university he is associate with

```

participant Student identified by student_id
{
  o String student_id
  o String first_name
  o String last_name
  o Uni university
  o exam exam
  o State state
  o String last_grade
  o String[] grades
  o String test optional
}

```

Figure 5.4: Student Participant

- (5) Exam : What exam marks is this.
- (6) State : Current State of marks Validity The possible States are : VALID, INVALID, PROCESSING, FINALISED
- (7) LastGrade : The latest grade granted to the student
- (8) Grades : Array of all the grades granted to the student in chronological order.
- (9) Test : Tests the availability of the student

Public

Public participants can only READ the transactions.

- (1) PublicID : uniquely identifies the person
- (2) First Name : Tells you the initials of the person
- (3) Last Name : Tells you the surname of the person
- (4) Description : The reason he has been granted public ID status

```
participant Public identified by public_id
{
  o String public_id
  o String first_name
  o String last_name
  o String Description
}
```

Figure 5.5: Public Participant

5.2.2 Asset

Grade

```
asset Grade identified by grade_id
{
  o String grade_id
  o exam exam
  o Uni university
  o State state
  o String grade_num
  --> Student student
}
```

Figure 5.6: Grade Asset

- (1) GradeID : uniquely identifies Grade
- (2) exam : What exam marks is this

- (3) University : The university it is associate with
- (4) State : Current State of marks Validity
- (5) GradeNum: The value of the grade
- (6) Student : ID of the student, the garde is associate with

5.2.3 Transaction

GradeMarkProfessor

```
transaction mark_grade_professor
{
  --> Professor professor
  o marksheet grade_new
  --> Student student
  --> Grade grade
}
```

Figure 5.7: GradeMarkProfessor Transaction

- (1) Professor : ProfessorID who is updating the grade
- (2) GradeNew : Value of the Grade
- (3) Student : StudentID whose grade is being updated
- (4) Grade : GradeID of the grade asset

The following is the consensus code for the above transaction.

```
/**
 * Marks the grade of an exam
 * @param {org.acme.model.mark_grade_professor}mark_grade_professor - the grade
 *   to be processed
 * @transaction
 */

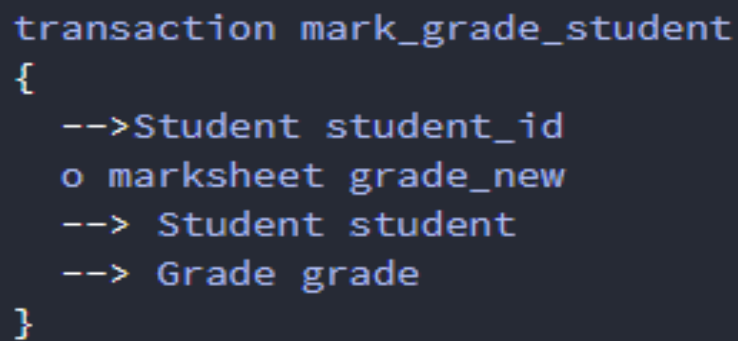
async function mark_grade_professor(pro)
{
  if(pro.student.state === 'FINALISED')
  {
    throw new Error('MARKS HAS BEEN SUBMITTED');
  }
  else
  {
    if (pro.grade.student !== pro.student)
    {
      throw new Error('IDs not matched');
    }
    else
    {
      pro.professor.test = 'OK';
      pro.student.state = 'VALID';
      pro.grade.state = 'VALID';
      pro.grade.grade_num = pro.grade_new;
      pro.student.last_grade = pro.grade_new;
      pro.student.grades.push(pro.grade_new);

      const ar = await getParticipantRegistry('org.acme.model.Student');
      await ar.update(pro.student);

      const dr = await getAssetRegistry('org.acme.model.Grade');
```

```
        await dr.update(pro.grade);
    }
}
}
```

GradeMarkStudent



```
transaction mark_grade_student
{
    -->Student student_id
    o marksheet grade_new
    --> Student student
    --> Grade grade
}
```

Figure 5.8: GradeMarkStudent Transaction

- (1) StudentID : StudentID who is updating the grade
- (2) GradeNew : Value of the Grade
- (3) Student : StudentID whose grade is being updated
- (4) Grade : GradeID of the grade asset

The following is the consensus code for the above transaction.

```
/**
 * Marks the grade of an exam
 * @param {org.acme.model.mark_grade_student}mark_grade_student - the grade to
 * be processed
```



```

@transaction
*/

async function mark_grade_student(stu)
{
  if(stu.student.state === 'FINALISED')
  {
    throw new Error('MARKS HAS BEEN SUBMITTED');
  }
  else
  {
    if (stu.grade.student !== stu.student)
    {
      throw new Error('IDs not matched');
    }
    else
    {
      stu.student_id.test = 'OKAY';
      stu.student.state = 'PROCESSING';
      stu.grade.state = 'PROCESSING';
      stu.grade.grade_num = stu.grade_new;
      stu.student.last_grade = stu.grade_new;
      stu.student.grades.push(stu.grade_new);
    }
    const ar = await getParticipantRegistry('org.acme.model.Student');
    await ar.update(stu.student);

    const dr = await getAssetRegistry('org.acme.model.Grade');
    await dr.update(stu.grade);
  }
}

```

Change

```
transaction change
{
  --> Professor id
  --> Student student
  --> Grade grade
  o State state
}
```

Figure 5.9: Change Transaction

- (1) Professor : ProfessorID who is changing the state
- (2) Student : StudentID whose grade is being updated
- (3) Grade : GradeID of the grade asset
- (4) State : The value of the state

The following is the consensus code for the above transaction.

```
/**
 * Marks the grade of an exam
 * @param {org.acme.model.change}change - the grade to be processed
 * @transaction
 */

async function change(ch)
{
  if (ch.grade.student !== ch.student)
```

```

{
    throw new Error('IDs not matched');
}
else
{
    if(ch.student.state === 'FINALISED')
    {
        throw Error("THE STATE IS FINALISED. CANNOT CHANGE ANYTHING NOW.");
    }
    else
    {
        if(ch.state === 'INVALID')
        {
            if(ch.student.state === 'INVALID')
            {
                throw Error("THE STATE IS ALREADY IN INVALID STATE.");
            }
            else
            {
                {
                    ch.student.last_grade = ' ';
                    ch.grade.grade_num = ' ';
                }
            }
        }
        else if(ch.state === 'FINALISED')
        {
            if(ch.student.state === 'INVALID')
            {
                throw Error("First VALID the marks");
            }
        }
        else if(ch.student.state === 'PROCESSING')
        {
            throw Error("First VALID the marks");
        }
    }
}

```

```

    else
    {
        ch.student.state = 'FINALISED';
        ch.grade.state = 'FINALISED';
    }
}
else if (ch.state === 'VALID')
{
    if(ch.student.state === 'VALID')
    {
        throw Error("THE STATE IS ALREADY IN VALID STATE");
    }
    else
    {
        ch.student.state = 'VALID';
        ch.grade.state = 'VALID';
    }
}
}

const ar = await getParticipantRegistry('org.acme.model.Student');
await ar.update(ch.student);

const dr = await getAssetRegistry('org.acme.model.Grade');
await dr.update(ch.grade);
}
}

```

5.2.4 Permissions

```
rule NetworkAdminUser {
    description: "Grant business network administrators full access to user
        resources"
    participant: "org.hyperledger.composer.system.NetworkAdmin"
    operation: ALL
    resource: "**"
    action: ALLOW
}

rule Student {
    description: "Grant Student to only update mark_grade_student Transaction"
    participant: "org.acme.model.Student"
    operation: ALL
    resource: "org.acme.model.mark_grade_student"
    action: ALLOW
}

rule Stud {
    description: "Grant Student, Professor, Public Participant to read"
    participant: "org.acme.model.*"
    operation: READ
    resource: "**"
    action: ALLOW
}

rule Professor {
    description: "Grant Student to only update mark_grade_professor
        Transaction"
    participant: "org.acme.model.Professor"
    operation: ALL
    resource: "org.acme.model.mark_grade_professor"
```

```
        action: ALLOW
    }

    rule Professorchange {
        description: "Grant Student to only update change Transaction"
        participant: "org.acme.model.Professor"
        operation: ALL
        resource: "org.acme.model.change"
        action: ALLOW
    }

    rule NetworkAdminSystem {
        description: "Grant business network administrators full access to system
            resources"
        participant: "org.hyperledger.composer.system.NetworkAdmin"
        operation: ALL
        resource: "org.hyperledger.composer.system.**"
        action: ALLOW
    }
}
```

Chapter 6

System Requirement Specification

6.1 Software Requirements

- (1) Operating System : Any Open-source OS
- (2) Technology : Hyperledger
- (3) Web Technologies : HTML, JavaScript, CSS
- (4) IDE : VSCode Editor
- (5) Web Server : Apache/ Tomcat
- (6) Network : LAN
- (7) Database : Cloud (local for demo)

6.2 Hardware Requirements

- (1) Processor : i3
- (2) Speed : 1.1 GHz
- (3) RAM : 1GB
- (4) Hard Disk : 5 GB
- (5) Key Board : Standard Windows Keyboard

Chapter 7

Screenshots

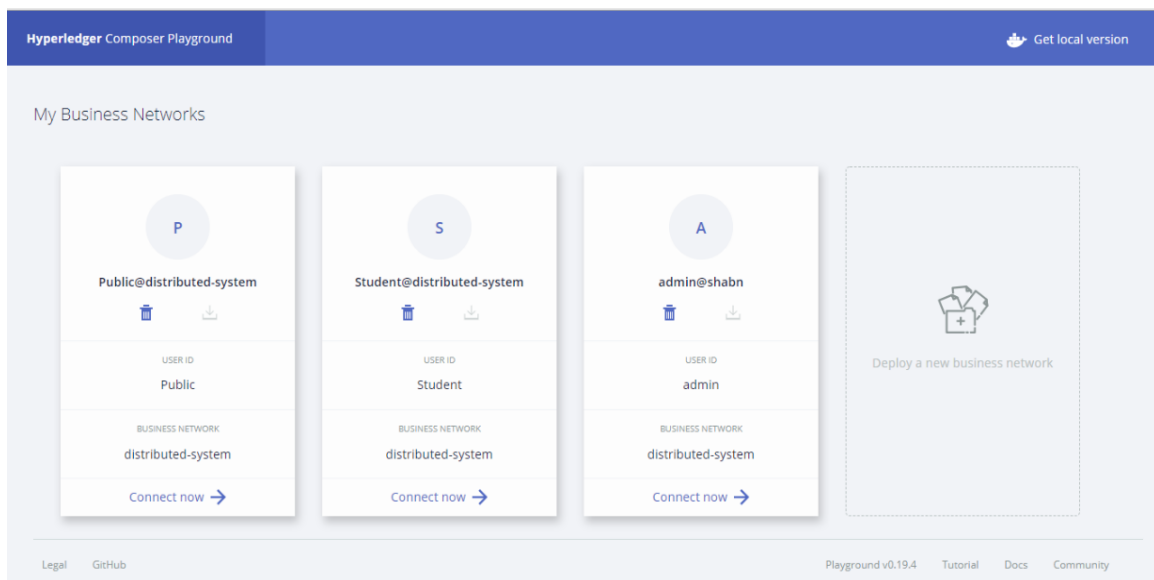


Figure 7.1: Business Network

This is the viewing options created by admin using there identities wallet. Each Student, Professor or public member will have a separate viewing window connected through blockchain blocks. Each viewing will have only that much of power that has been decided in ACL.

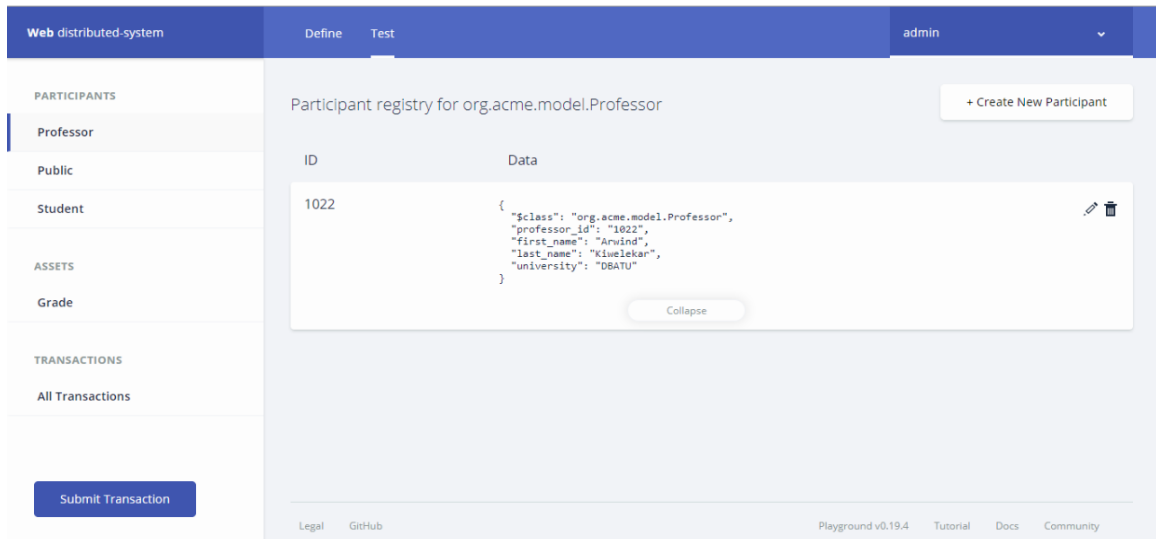


Figure 7.2: Create Professor

Admin will create Professor identity. It will be uniquely identified by professorid.



Figure 7.3: Create Public

Admin and professor can create Public identity. It will be uniquely identified by publicid.

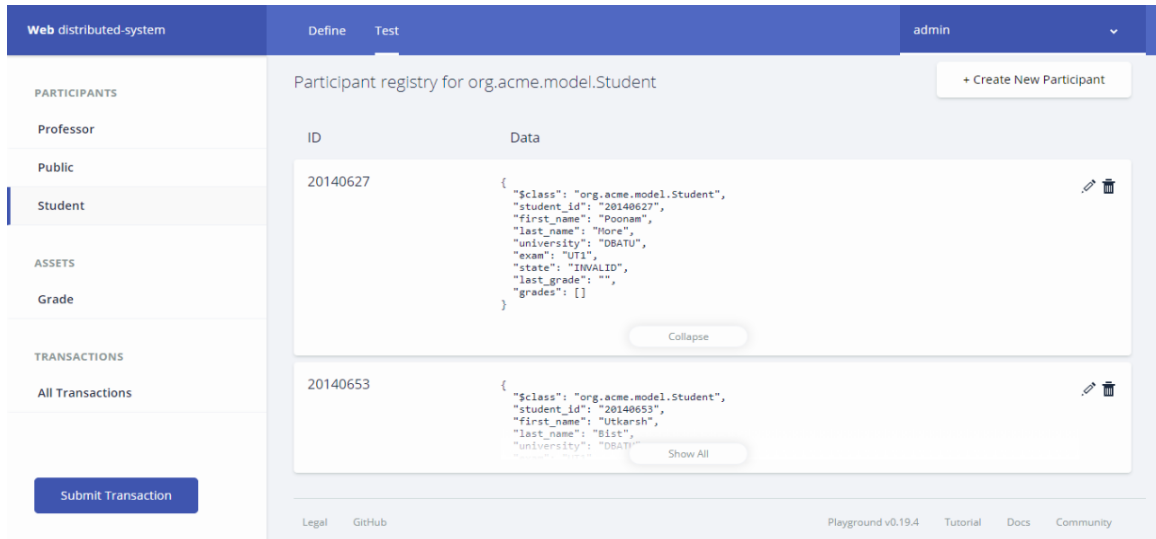


Figure 7.4: Create Student

Admin and professor can create Student identity. It will be uniquely identified by studentid.

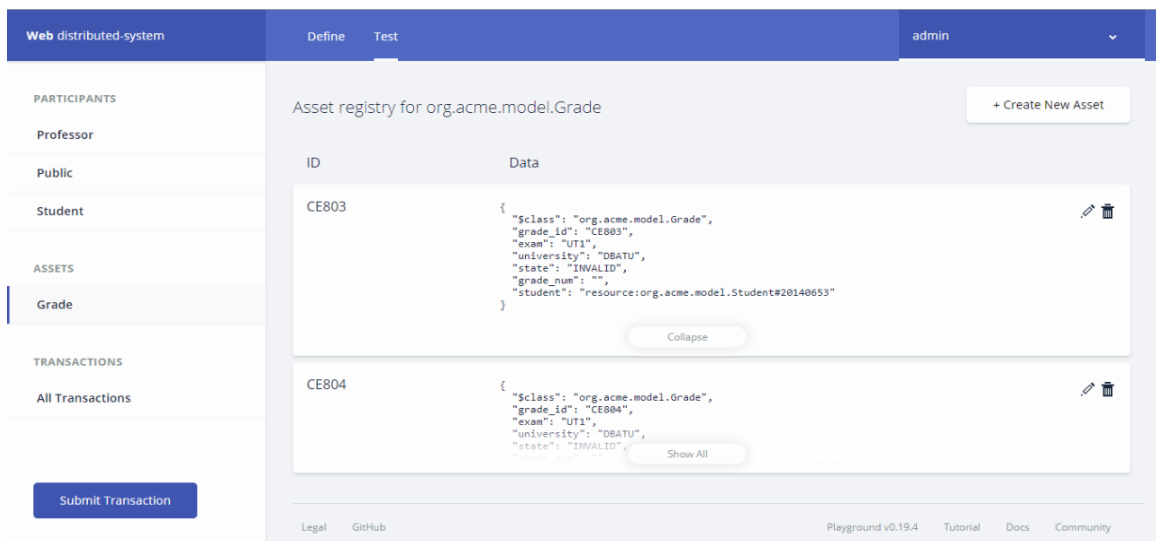


Figure 7.5: Create Grade

Admin and professor can create Grade Asset. It will be uniquely by gradeid.

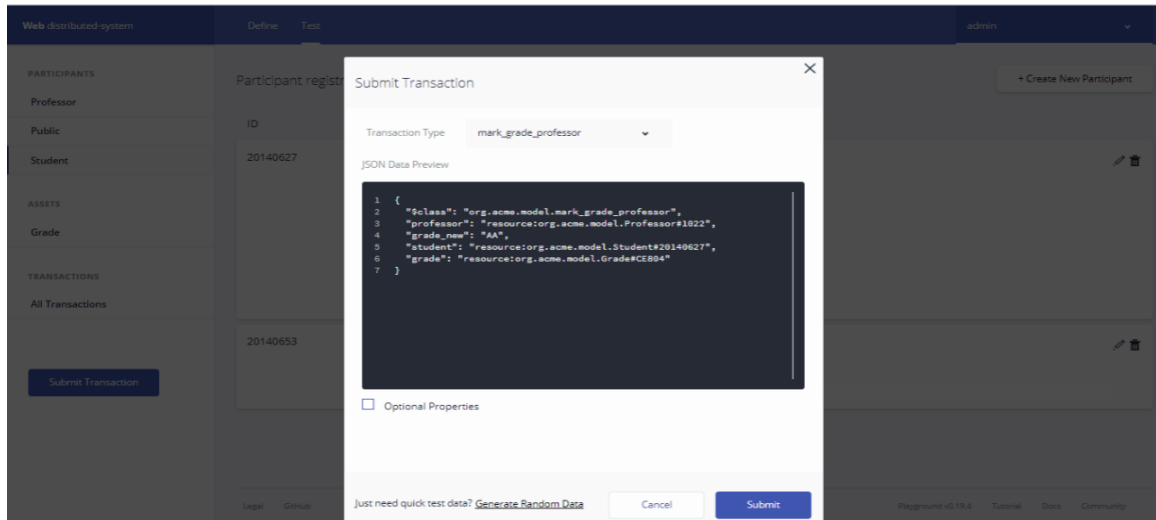


Figure 7.6: Grade updation by Professor

By filling the form on the figure, professor can perform a valid transaction and give a grade to the mentioned student.

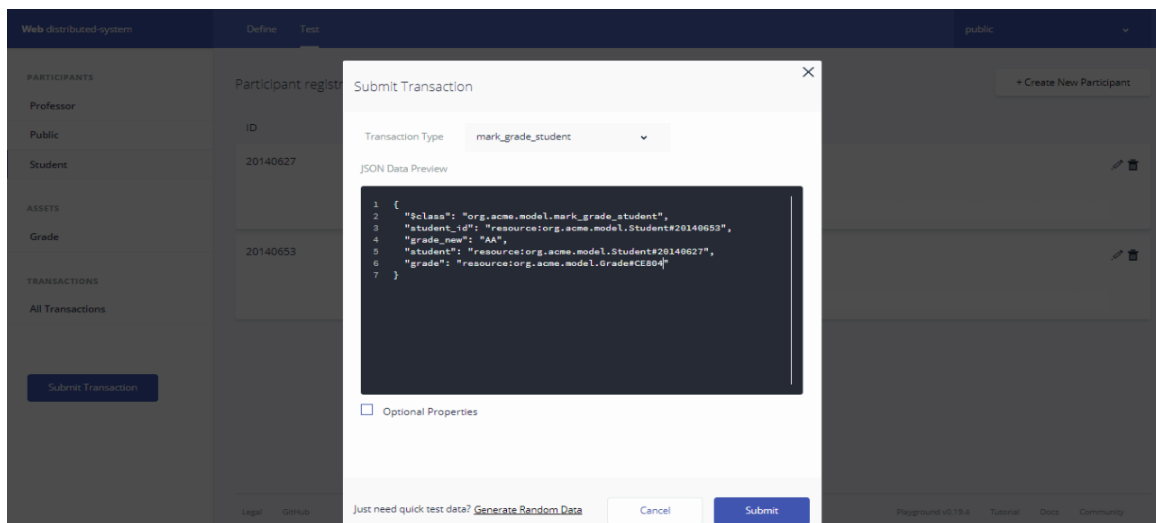


Figure 7.7: Grade updation by Student

By filling the form on the figure, student can perform a transaction that is in processing state and give a grade to the mentioned student that he feels he/she deserves.

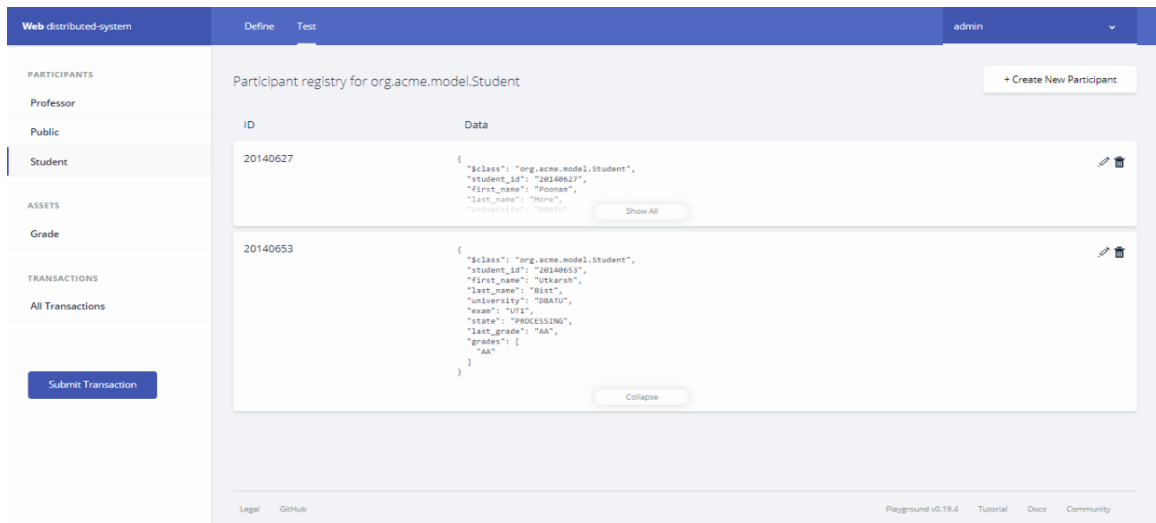


Figure 7.8: Processing State

Processing State means the grade has been updated by a student and it is waiting for a professor for any kind of validation.

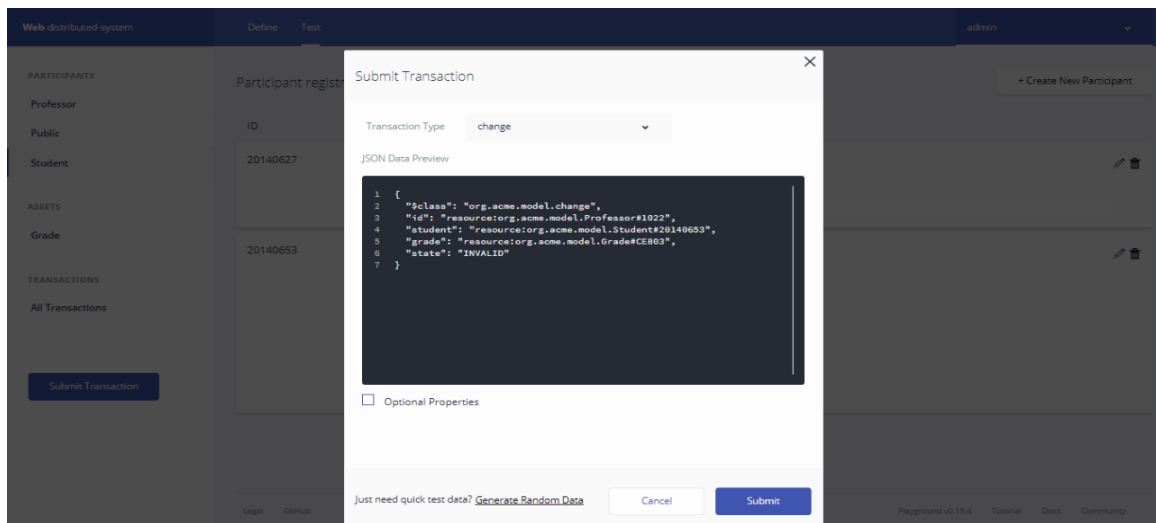


Figure 7.9: State change by Professor

By filling the form on the figure, professor can perform a state change transaction and give a suitable state: VALID, INVALID, to the student.

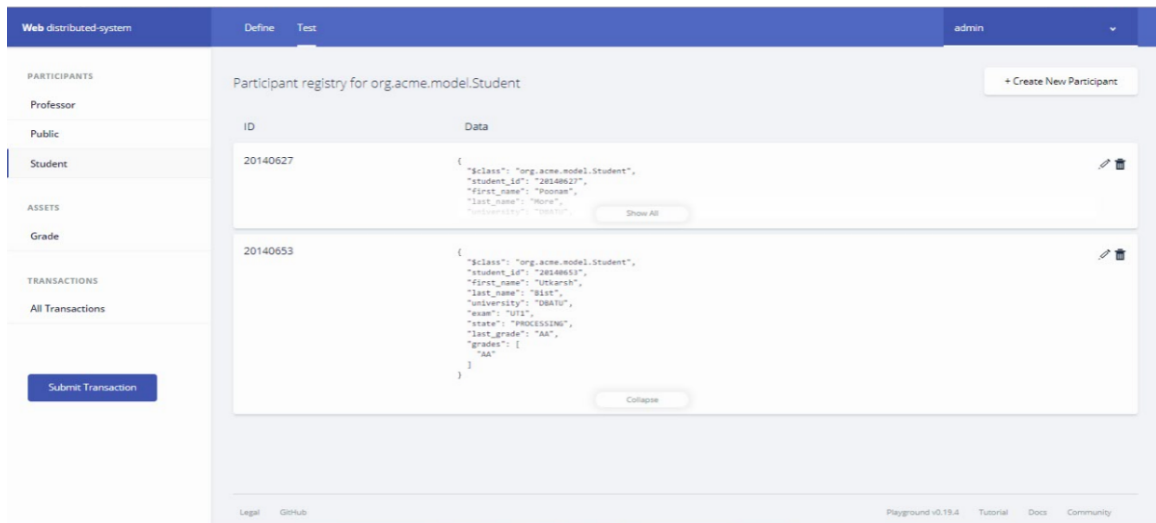


Figure 7.10: INVALID state

INVALID state cannot be FINALISED until Professor updates new grade and make it a valid Transaction.

Web distributed-system	Define	Test	admin
PARTICIPANTS			
Professor			
Public			
Student			
ASSETS			
Grade			
TRANSACTIONS			
All Transactions			
Submit Transaction			
Date, Time	Entry Type	Participant	
2018-05-03, 00:12:22	change	admin (NetworkAdmin)	view record
2018-05-03, 00:11:11	mark_grade_professor	admin (NetworkAdmin)	view record
2018-05-03, 00:10:23	change	admin (NetworkAdmin)	view record
2018-05-03, 00:08:25	mark_grade_student	admin (NetworkAdmin)	view record
2018-05-03, 00:07:11	change	admin (NetworkAdmin)	view record
2018-05-03, 00:04:36	UpdateParticipant	admin (NetworkAdmin)	view record
2018-05-03, 00:04:06	UpdateParticipant	admin (NetworkAdmin)	view record

Figure 7.11: Transaction Record

Its a snapshot of the blockchain blocks. All the activities an be monitored from here.

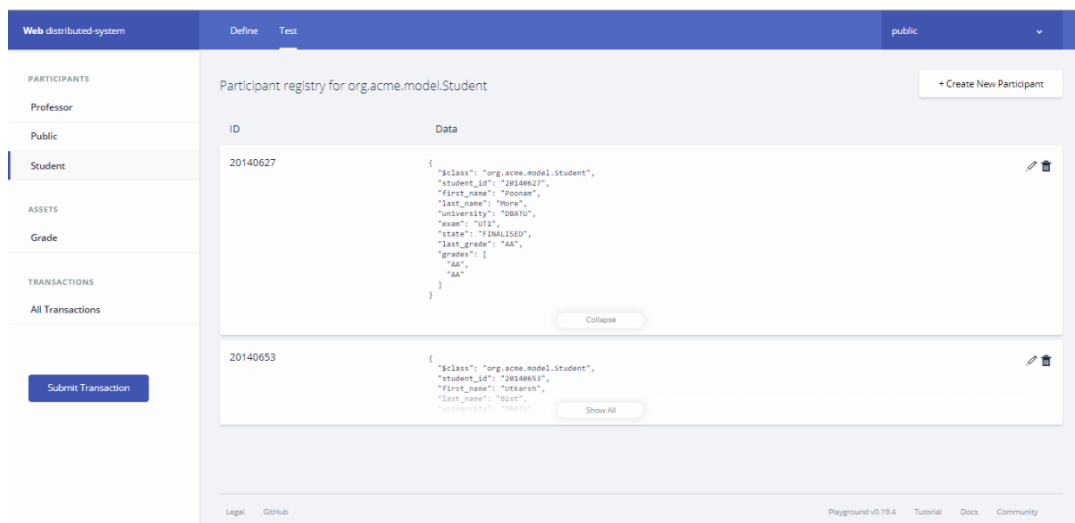


Figure 7.12: Finalised State

Once, in Finalised State, it means grades have been submitted and no more updation can take place.

Chapter 8

Conclusion

Given the incredible opportunity for decentralization, blockchain technology offers the ability to create businesses and operations that are both flexible and secure. Whether companies will succeed in deploying blockchain technology to create products and services consumers will trust and adopt remains to be seen. Nevertheless, this is definitely a space investors should watch. The demand for blockchain-based services is on the rise, and the technology is maturing and advancing at a rapid pace.

The potential applications for blockchain technology are almost without limit. At the moment, several of these applications are still either in the development stage or in beta testing. With more money being poured into blockchain-based startups, consumers should not be surprised to see blockchain services and products becoming more mainstream in the near future.

As rightly been quoted by experts:

**What Internet did for communication, Blockchain
will do for transaction!**

Bibliography

- [1] Manav Gupta, Blockchain for dummies- IBM Limited Edition, 2017.
- [2] Online Course On: "Blockchain for Business - An Introduction to Hyperledger Technologies"
<https://courses.edx.org/>
- [3] <https://developer.ibm.com/courses/all/blockchain-essentials/>
- [4] <https://hyperledger.github.io/composer/installing/getting-started-with-playground.html>