Project Report (Paper-CSCC43)

MCA (IV Semester)

Subject: Software Project Management

LetterOfCreditNetwork on Blockchain

Department of Computer Science,
Jamia Millia Islamia,
(A Central University),
New Delhi, 110025.

Submitted By:

Md. Tauseef Alam(25).

Mohammad Faiz Ali(26).

Under Supervision

Dr. S.M.K Quadri

CERTIFICATE

This is to certify that the project entitled,
" **LetterOfCreditNetwork on Blockchain**" has been done by:-
Md Tauseef Alam and **Mohammad Faiz Ali** of Master of Computer Application (M.C.A) during semester IV from **Jamia Millia Islamia University** under the supervision of **Dr. S.M.K Quadri.**

Dr. S.M.K Quadri.

ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

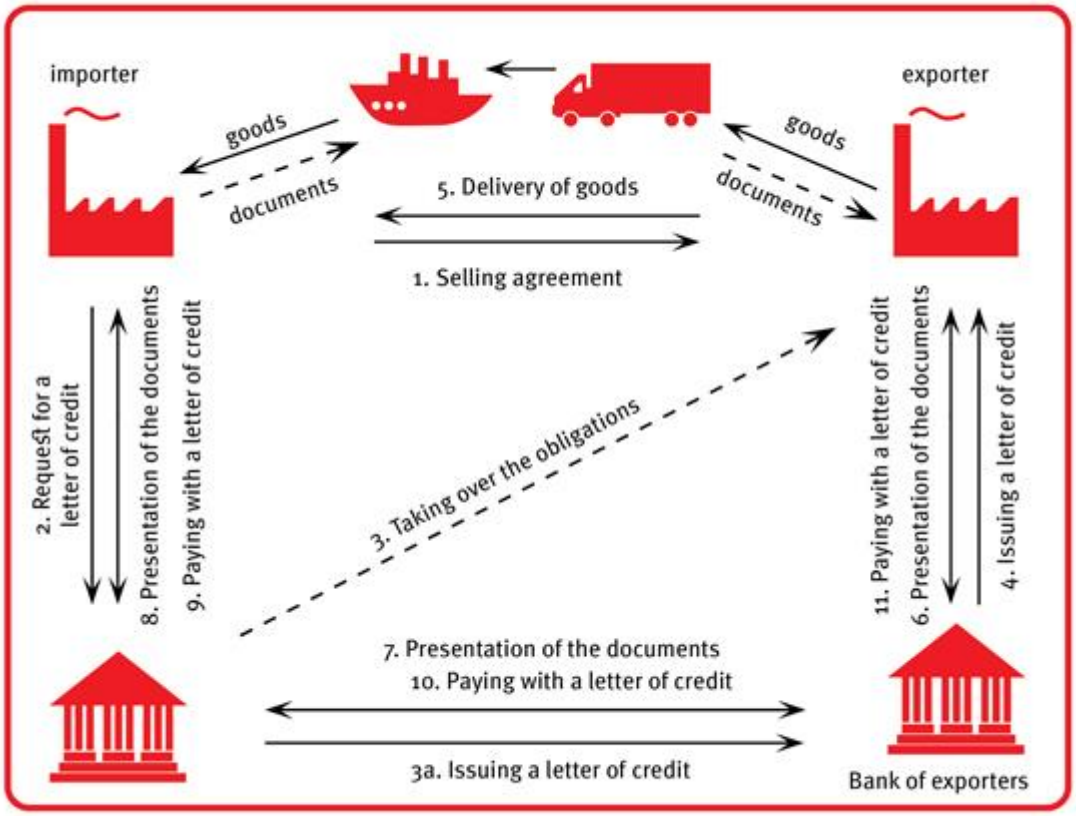
The Project was jointly undertaken by **Md Tauseef Alam** and **Mohammad Faiz Ali** as their 4th Semester Software Engineering Project, under the able guidance and supervision of **Dr. S.M.K Quadri**. Our primary thanks goes to him, who poured over every inch of our project with painstaking attention and helped us throughout the working of the project. It's our privilege to acknowledge our deepest sense of gratitude to him for his inspiration which has helped us immensely. We are extremely grateful to him for his unstilted support and encouragement in the preparation of this project. We would also thank the coursera and IBM team for their online guidance throughout the project.

TABLE OF CONTENTS

1. INTRODUCTION

A **letter of credit** (**LC**), also known as a **documentary credit** or **bankers commercial credit**, or **letter of undertaking** (**LoU**), is a payment mechanism used in international trade to provide an economic guarantee from a creditworthy bank to an exporter of goods. Letters of credit are used extensively in the financing of international trade, where the reliability of contracting parties cannot be readily and easily determined. Its economic effect is to introduce a bank as an underwriter, where it assumes the credit risk of the buyer paying the seller for goods.

Currently constrained by costs & the time to execute.



2. PROJECT GOALS

The **LetterOfCreditNetwork on Blockchain** shall :

1. Increase speed of execution (less than 1 day)
2. Vastly reduced cost
3. Reduced risk,
   e.g. currency fluctuations
4. Value added services,
   e.g. incremental payment

3.PROJECT EXECUTION

3.1 Problem Definition

A **letter of credit** (**LC**), also known as a **documentary credit** or **bankers commercial credit**, or **letter of undertaking** (**LoU**), is a payment mechanism used in international trade to provide an economic guarantee from a creditworthy bank to an exporter of goods. Letters of credit are used extensively in the financing of international trade, where the reliability of contracting parties cannot be readily and easily determined. Its economic effect is to introduce a bank as an underwriter, where it assumes the credit risk of the buyer paying the seller for goods. Currently execution is time consuming and takes lot of cost. There is also risk of fraud.

A **blockchain** originally **block chain** is a growing list of records, called *blocks*, which are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data (generally represented as a Merkle tree).

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

• Blockchain provides common ledger for letters of credit.
• Allows all counter-parties to have the same validated record of transaction and fulfilment.

3.2 MODELLING

3.3.1 Process Model

A process model for software engineering is chosen based on the nature of the project and

application, the methods and tools to be used, and the controls and deliverables that are required. The model is used to build the "LetterOfCreditNetwork on Blockchain" software is "The waterfall model".

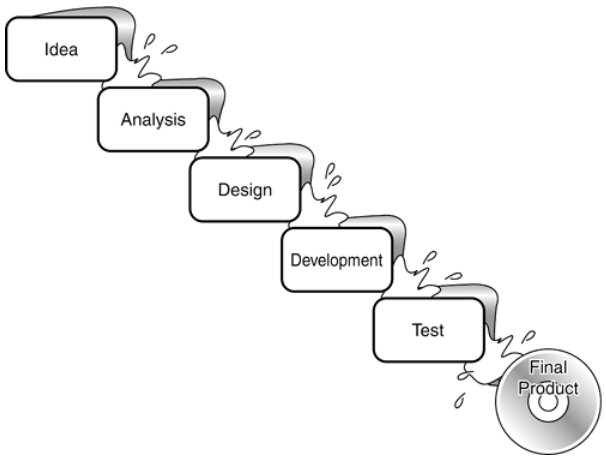


Fig.1.2 Water fall Model

The water fall model is a software development model in which a systems development is viewed as flowing downwards through the phases of the system development process. The waterfall methodology is powerful, précised, and thorough. It has a number of phases that have to be implemented in a sequential manner.
The phases which come under the waterfall model are as follows:-

1.Requirement Analysis
2.Design
3.Implementation
4.Testing
5.Maintenance

Advantages:

1.Good for large projects
2.Waterfall suits a principled approach to design
3.Waterfall divides the project into manageable areas
4.Waterfall separates the logical and physical

3.4. SYSTEM ANALYSIS AND SPECIFICATION

The design of a system is essentially a blueprint or a plan for a solution for the system. A design methodology is a systematic approach to creating a design approach, a system is viewed as a transformation function, transforming the inputs to the desired outputs.
The design process for software systems often has two levels. At the first level the focus is on

deciding which modules are needed for the system, the specifications of these modules, and how the modules should be interconnected. This is what is called the system design or top-level design. In the second level, the internal design of the modules, or how the specifications of the module can be satisfied, is decided. This design level is often called design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding.

3.4.1 Architectural Design

For a function-oriented design, the design can be represented graphically
by structure charts. The structure of a program is made up of the modules of that program together with the interconnections between modules. The structure chart of a program is a graphic representation of its structure. In a structure chart a module is represented by a box with the module name written in the box.
During design, **Structured Design Methodology** aims to control and influence the structure of the final program. The aim is to design a system so that programs implementing the design would have a hierarchical structure, with functionally cohesive modules and as few interconnection between modules as possible.
The overall strategy is to identify the input and output streams and the primary transformations that have to be performed to produce the output. High level modules are then created to perform these major activities ,which are later refined. There are four major steps in this strategy:

3.4.1.1. Restate the problem as a Data Flow Diagram
3.4.1.2 Identify the Most Abstract Input (MAI) and Most Abstract Output(MAO) Data Elements
3.4.1.3 First-level factoring
3.4.1.4 Factoring of input, output and transform branches

3.4.2 Component-level Design

Component-level design establishes the algorithm detail required to manipulate data structures, effect communication between software components via their interfaces, and implement the processing algorithms allocated to each component. Component-level design, also called procedural design, occurs after data, architectural, and interface designs have been established. The intent is to translate the design model into operational software. But the level of abstraction of the existing design model is relatively high, and the abstraction level of the operational program is low.

PSEUDOCODE: (Models(.cto))

```
namespace org.example.loc
```

```
// ENUMS
enum LetterStatus {
  o AWAITING_APPROVAL
  o APPROVED
  o SHIPPED
  o RECEIVED
  o READY_FOR_PAYMENT
  o CLOSED
  o REJECTED
}

// ASSETS
asset LetterOfCredit identified by letterId {
  o String letterId
  --> Customer applicant
  --> Customer beneficiary
  --> Bank issuingBank
  --> Bank exportingBank
  o Rule[] rules
  o ProductDetails productDetails
  o String [] evidence
  --> Person [] approval
  o LetterStatus status
  o String closeReason optional
}

// PARTICIPANTS
participant Bank identified by bankID {
  o String bankID
  o String name
}

abstract participant Person identified by personId {
  o String personId
  o String name
  o String lastName optional
  --> Bank bank
}

participant Customer extends Person {
  o String companyName
}

participant BankEmployee extends Person {
}

// CONCEPTS
concept Rule {
  o String ruleId
  o String ruleText
```

```
}

concept ProductDetails {
  o String productType
  o Integer quantity
  o Double pricePerUnit
}

// TRANSACTIONS + EVENTS
transaction InitialApplication {
  o String letterId
  --> Customer applicant
  --> Customer beneficiary
  o Rule[] rules
  o ProductDetails productDetails
}

event InitialApplicationEvent {
  --> LetterOfCredit loc
}

transaction Approve {
  --> LetterOfCredit loc
  --> Person approvingParty
}

event ApproveEvent {
  --> LetterOfCredit loc
  --> Person approvingParty
}

transaction Reject {
  --> LetterOfCredit loc
  o String closeReason
}

event RejectEvent {
  --> LetterOfCredit loc
  o String closeReason
}

transaction SuggestChanges {
  --> LetterOfCredit loc
  o Rule[] rules
  --> Person suggestingParty
}

event SuggestChangesEvent {
  --> LetterOfCredit loc
  o Rule[] rules
```

```
  --> Person suggestingParty
}

transaction ShipProduct {
  --> LetterOfCredit loc
  o String evidence
}

event ShipProductEvent {
  --> LetterOfCredit loc
}

transaction ReceiveProduct {
  --> LetterOfCredit loc
}

event ReceiveProductEvent {
  --> LetterOfCredit loc
}

transaction ReadyForPayment {
  --> LetterOfCredit loc
}
event ReadyForPaymentEvent {
  --> LetterOfCredit loc
}

transaction Close {
  --> LetterOfCredit loc
  o String closeReason
}

event CloseEvent {
  --> LetterOfCredit loc
  o String closeReason
}

// TRANSACTIONS FOR SETUP
transaction CreateDemoParticipants {
}
```

PSEUDOCODE: (AccessControl(.acl))

```
rule ParticipantsSeeSelves {
        description: "Let participants see themselves"
        participant(p): "org.hyperledger.composer.system.Participant"
        operation: ALL
    resource(r): "org.hyperledger.composer.system.Participant"
        condition: (r.getIdentifier() == p.getIdentifier())
        action: ALLOW
}
```

```
rule ParticipantsSeeBanks {
        description: "Let participants see themselves"
        participant: "org.hyperledger.composer.system.Participant"
        operation: READ
    resource: "org.example.loc.Bank"
        action: ALLOW
}

rule SeeOtherCustomers {
          description: "Let Customers see other Customers"
        participant: "org.example.loc.Customer"
        operation: READ
    resource: "org.example.loc.Customer"
        action: ALLOW
}

rule CustomerSeeBankEmployee {
          description: "Let Customers see their BankEmployees"
        participant(p): "org.example.loc.Customer"
        operation: READ
    resource(r): "org.example.loc.BankEmployee"
        condition: (r.bank.getIdentifier() == p.bank.getIdentifier())
        action: ALLOW
}

rule BankEmployeeSeeCustomer {
          description: "Let BankEmployees see their Customers"
        participant(p): "org.example.loc.BankEmployee"
        operation: READ
    resource(r): "org.example.loc.Customer"
        condition: (r.bank.getIdentifier() == p.bank.getIdentifier())
        action: ALLOW
}

rule BankEmployeeSeeBankEmployee {
          description: "Let BankEmployees see their colleagues"
        participant(p): "org.example.loc.BankEmployee"
        operation: READ
    resource(r): "org.example.loc.BankEmployee"
        condition: (r.bank.getIdentifier() == p.bank.getIdentifier())
        action: ALLOW
}

rule CustomerMakeApplication {
          description: "All customers can submit an InitialApplication transaction"
        participant: "org.example.loc.Customer"
        operation: CREATE
    resource: "org.example.loc.InitialApplication"
        action: ALLOW
}

rule CustomerCreateLOC {
          description: "All customers can create a LetterOfCredit asset"
        participant: "org.example.loc.Customer"
        operation: CREATE
    resource: "org.example.loc.LetterOfCredit"
        transaction: "org.example.loc.InitialApplication"
        action: ALLOW
}

rule CustomerViewLetterOfCredit {
          description: "All customers can view letters of credit they are involved
with"
        participant(p): "org.example.loc.Customer"
```

```
        operation: READ
    resource(r): "org.example.loc.LetterOfCredit"
    condition: (p.getIdentifier() === r.applicant.getIdentifier() || p.getIdentifier()
=== r.beneficiary.getIdentifier())
        action: ALLOW
}


rule BankEmployeeViewLetterOfCredit {
          description: "All bank employees can view letters of credit their bank is
involved with"
        participant(p): "org.example.loc.BankEmployee"
        operation: READ
    resource(r): "org.example.loc.LetterOfCredit"
    condition: (p.bank.getIdentifier() === r.issuingBank.getIdentifier() ||
p.bank.getIdentifier() === r.exportingBank.getIdentifier())
        action: ALLOW
}

rule CustomerApproveApplication {
          description: "All customers can submit an Approve transaction for an LoC they
are involved with"
        participant(p): "org.example.loc.Customer"
        operation: CREATE
    resource(r): "org.example.loc.Approve"
    condition: (p.getIdentifier() === r.loc.applicant.getIdentifier() ||
p.getIdentifier() === r.loc.beneficiary.getIdentifier())
        action: ALLOW
}

rule BankEmployeeApproveApplication {
          description: "All bank employees can submit an Approve transaction for an LoC
their bank is involved with"
        participant(p): "org.example.loc.BankEmployee"
        operation: CREATE
    resource(r): "org.example.loc.Approve"
    condition: (p.bank.getIdentifier() === r.loc.issuingBank.getIdentifier() ||
p.bank.getIdentifier() === r.loc.exportingBank.getIdentifier())
        action: ALLOW
}

rule CustomerAddApproval {
          description: "All customers can add their approval to a Letter of Credit they
are involved with"
        participant(p): "org.example.loc.Customer"
        operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
        transaction(t): "org.example.loc.Approve"
    condition: (p.getIdentifier() === r.applicant.getIdentifier() || p.getIdentifier()
=== r.beneficiary.getIdentifier())
        action: ALLOW
}

rule BankEmployeeAddApproval {
          description: "All bank employee can add their approval to a Letter of Credit
their bank is involved with"
        participant(p): "org.example.loc.BankEmployee"
        operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
        transaction(t): "org.example.loc.Approve"
    condition: (p.bank.getIdentifier() === r.issuingBank.getIdentifier() ||
p.bank.getIdentifier() === r.exportingBank.getIdentifier())
        action: ALLOW
}
```

```
rule CustomerSubmitSuggestChanges {
          description: "All customers can submit a SuggestChanges transaction to a
Letter of Credit they are involved with"
        participant(p): "org.example.loc.Customer"
        operation: CREATE
    resource(r): "org.example.loc.SuggestChanges"
    condition: (p.getIdentifier() === r.loc.applicant.getIdentifier() ||
p.getIdentifier() === r.loc.beneficiary.getIdentifier())
        action: ALLOW
}

rule BankEmployeeSubmitSuggestChanges {
          description: "All bank employees can submit a SuggestChanges transaction to a
Letter of Credit their bank is involved with"
        participant(p): "org.example.loc.BankEmployee"
        operation: CREATE
    resource(r): "org.example.loc.SuggestChanges"
    condition: (p.bank.getIdentifier() === r.loc.issuingBank.getIdentifier() ||
p.bank.getIdentifier() === r.loc.exportingBank.getIdentifier())
        action: ALLOW
}

rule CustomerSuggestChanges {
          description: "All customers can submit a SuggestChanges transaction to a
Letter of Credit they are involved with"
        participant(p): "org.example.loc.Customer"
        operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
        transaction(t): "org.example.loc.SuggestChanges"
    condition: (p.getIdentifier() === r.applicant.getIdentifier() || p.getIdentifier()
=== r.beneficiary.getIdentifier())
        action: ALLOW
}

rule BankEmployeeSuggestChanges {
          description: "All bank employee can add their approval to a Letter of Credit
their bank is involved with"
        participant(p): "org.example.loc.BankEmployee"
        operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
        transaction(t): "org.example.loc.SuggestChanges"
    condition: (p.bank.getIdentifier() === r.issuingBank.getIdentifier() ||
p.bank.getIdentifier() === r.exportingBank.getIdentifier())
        action: ALLOW
}

rule CustomerAddChanges {
          description: "All customers can update a LetterOfCredit with their suggested
rules if they are involved in it"
        participant(p): "org.example.loc.Customer"
        operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
        transaction(t): "org.example.loc.SuggestChanges"
    condition: (p.getIdentifier() === r.applicant.getIdentifier() || p.getIdentifier()
=== r.beneficiary.getIdentifier())
        action: ALLOW
}

rule BankEmployeeAddChanges {
          description: "All bank employees can update a LetterOfCredit with their
suggested rules if their bank is involved in it"
        participant(p): "org.example.loc.Customer"
        operation: UPDATE
```

```
    resource(r): "org.example.loc.LetterOfCredit"
        transaction(t): "org.example.loc.SuggestChanges"
    condition: (p.bank.getIdentifier() === r.issuingBank.getIdentifier() ||
p.bank.getIdentifier() === r.exportingBank.getIdentifier())
        action: ALLOW
}

rule CustomerRejectApplication {
          description: "All customers can submit a Reject transaction for an LoC they
are involved with"
        participant(p): "org.example.loc.Customer"
        operation: CREATE
    resource(r): "org.example.loc.Reject"
    condition: (p.getIdentifier() === r.loc.applicant.getIdentifier() ||
p.getIdentifier() === r.loc.beneficiary.getIdentifier())
        action: ALLOW
}

rule BankEmployeeRejectApplication {
          description: "All bank employees can submit a Reject transaction for an LoC
their bank is involved with"
        participant(p): "org.example.loc.BankEmployee"
        operation: CREATE
    resource(r): "org.example.loc.Reject"
    condition: (p.bank.getIdentifier() === r.loc.issuingBank.getIdentifier() ||
p.bank.getIdentifier() === r.loc.exportingBank.getIdentifier())
        action: ALLOW
}

rule CustomerMarksAsRejected {
          description: "All customers can update a LetterOfCredit they are involved
with with a REJECTED status"
        participant(p): "org.example.loc.Customer"
        operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
        transaction(t): "org.example.loc.Reject"
    condition: (p.getIdentifier() === r.applicant.getIdentifier() || p.getIdentifier()
=== r.beneficiary.getIdentifier())
        action: ALLOW
}

rule BankEmployeeMarksAsRejected {
          description: "All bank employees can update a LetterOfCredit their bank is
involved with with a REJECTED status"
        participant(p): "org.example.loc.BankEmployee"
        operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
        transaction(t): "org.example.loc.Reject"
    condition: (p.bank.getIdentifier() === r.issuingBank.getIdentifier() ||
p.bank.getIdentifier() === r.exportingBank.getIdentifier())
        action: ALLOW
}

rule BeneficiaryShipProduct {
    description: "The beneficiary send a transaction to mark a letter of credit as
relating to goods that have been shipped"
    participant(p): "org.example.loc.Customer"
    operation: CREATE
    resource(r): "org.example.loc.ShipProduct"
    condition: (p.getIdentifier() === r.loc.beneficiary.getIdentifier())
    action: ALLOW
}

rule BeneficiaryMarkAsShippedProduct {
```

```
    description: "The applicant can mark a letter of credit as relating to goods that
have been shipped"
    participant(p): "org.example.loc.Customer"
    operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
    transaction(t): "org.example.loc.ShipProduct"
    condition: (p.getIdentifier() === r.beneficiary.getIdentifier())
    action: ALLOW
}

rule ApplicantReceiveProduct {
    description: "The applicant send a transaction to mark a letter of credit as
relating to goods that have been received"
    participant(p): "org.example.loc.Customer"
    operation: CREATE
    resource(r): "org.example.loc.ReceiveProduct"
    condition: (p.getIdentifier() === r.loc.applicant.getIdentifier())
    action: ALLOW
}

rule ApplicantMarkAsReceivedProduct {
    description: "The applicant can mark a letter of credit as relating to goods that
have been received"
    participant(p): "org.example.loc.Customer"
    operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
    transaction(t): "org.example.loc.ReceiveProduct"
    condition: (p.getIdentifier() === r.applicant.getIdentifier())
    action: ALLOW
}

rule IssuingBankReadyForPayment {
    description: "The issuing bank employee can state the letter is ready for payment"
    participant(p): "org.example.loc.BankEmployee"
    operation: CREATE
    resource(r): "org.example.loc.ReadyForPayment"
    condition: (p.bank.getIdentifier() === r.loc.issuingBank.getIdentifier())
    action: ALLOW
}

rule IssuingBankMarkReadyForPayment {
    description: "The issuing bank employee can mark the letter as ready for payment"
    participant(p): "org.example.loc.BankEmployee"
    operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
    transaction(t): "org.example.loc.ReadyForPayment"
    condition: (p.bank.getIdentifier() === r.issuingBank.getIdentifier())
    action: ALLOW
}

rule ExportingBankCloseLetter {
    description: "The exporting bank employee can close the letter"
    participant(p): "org.example.loc.BankEmployee"
    operation: CREATE
    resource(r): "org.example.loc.Close"
    condition: (p.bank.getIdentifier() === r.loc.exportingBank.getIdentifier())
    action: ALLOW
}

rule ExportingBankMarkLetterClosed {
    description: "The exporting bank employee can mark the letter as closed"
    participant(p): "org.example.loc.BankEmployee"
    operation: UPDATE
    resource(r): "org.example.loc.LetterOfCredit"
```

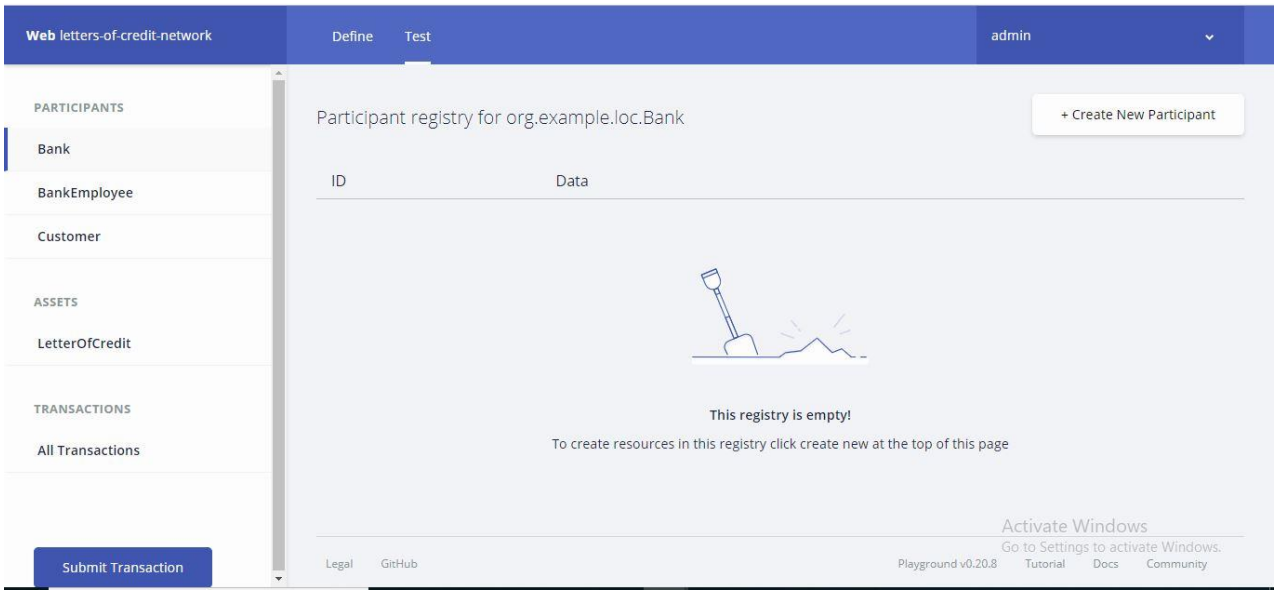
```
    transaction(t): "org.example.loc.Close"
    condition: (p.bank.getIdentifier() === r.exportingBank.getIdentifier())
    action: ALLOW
}

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

rule System {
    description: "Grant all full access to system resources"
    participant: "org.**"
    operation: ALL
    resource: "org.hyperledger.composer.system.**"
    action: ALLOW
}
```

3.5 SYSTEM DESIGN

At the first visit of the LetterOfCreditNetwork on Blockchain site, the user interacts with the system via the interface below.

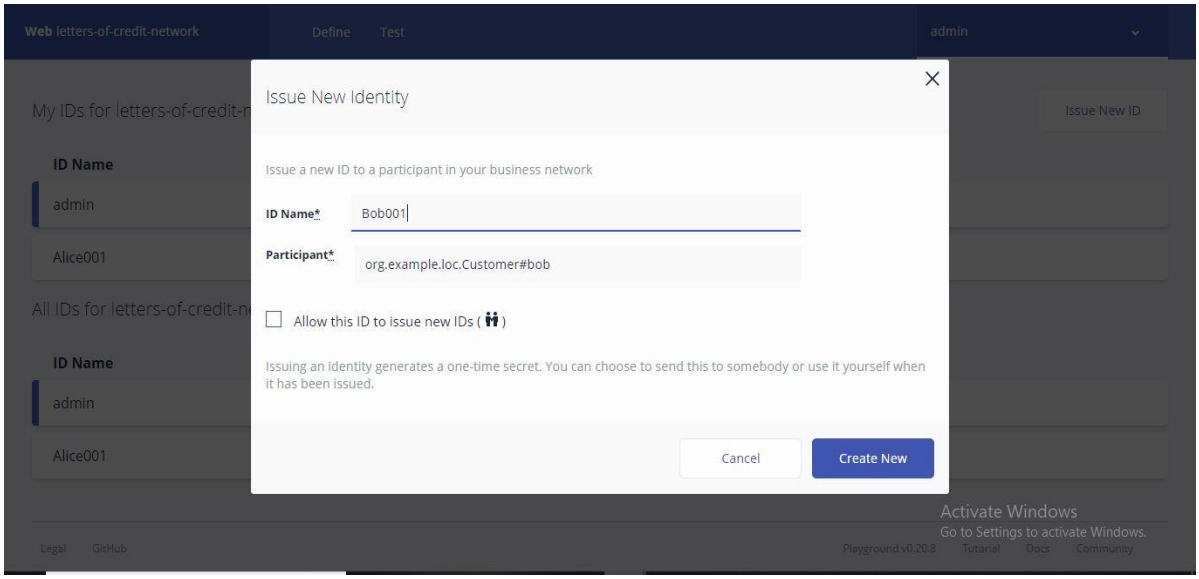
3.5 System Design

First page (Homepage)

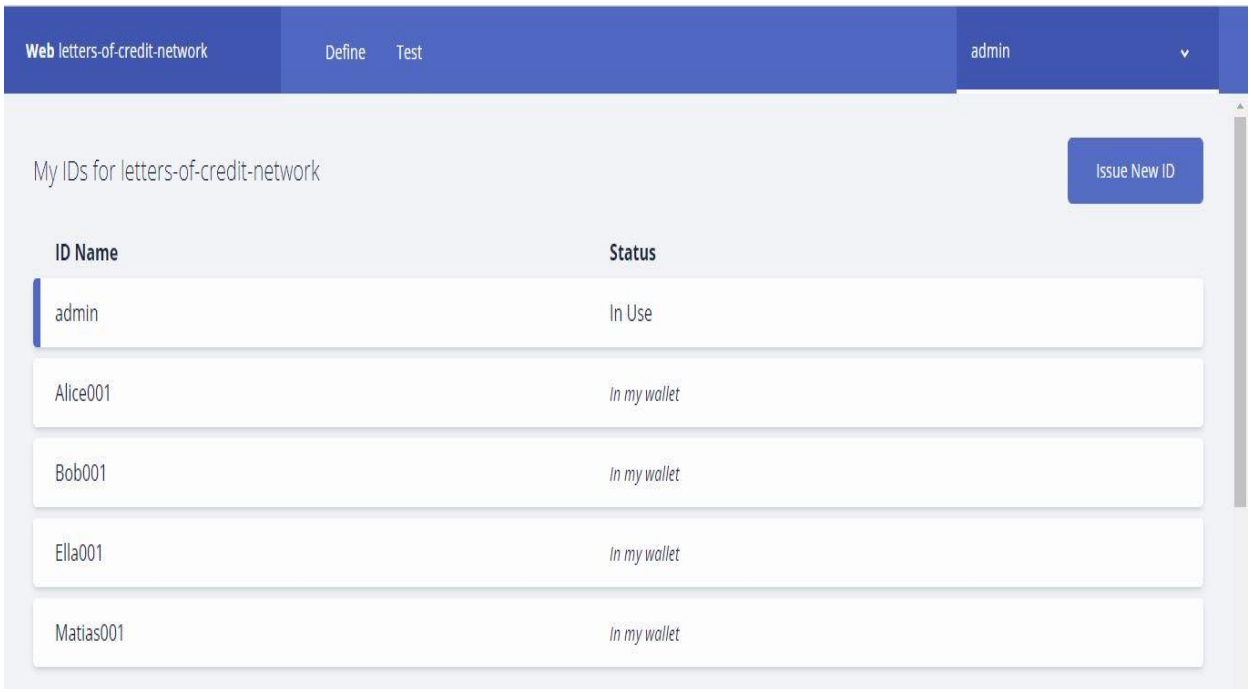


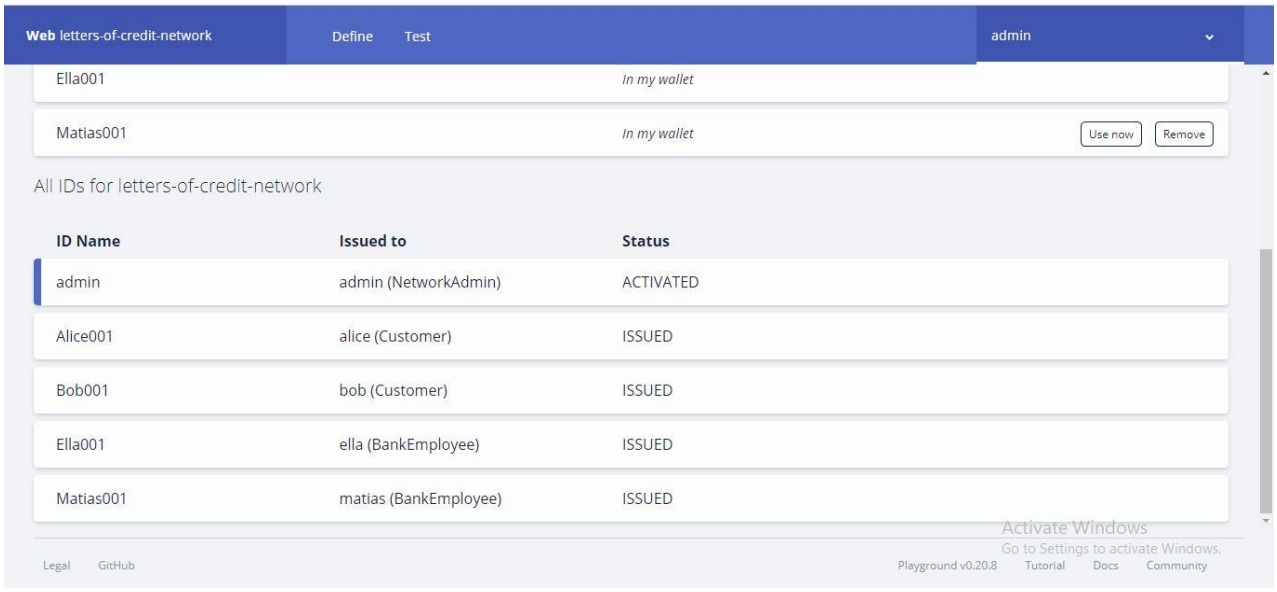
Navigate to the ID registry and generate IDs for:

```
org.example.loc.Customer#alice
org.example.loc.Customer#bob
org.example.loc.BankEmployee#matias
org.example.loc.BankEmployee#ella
```



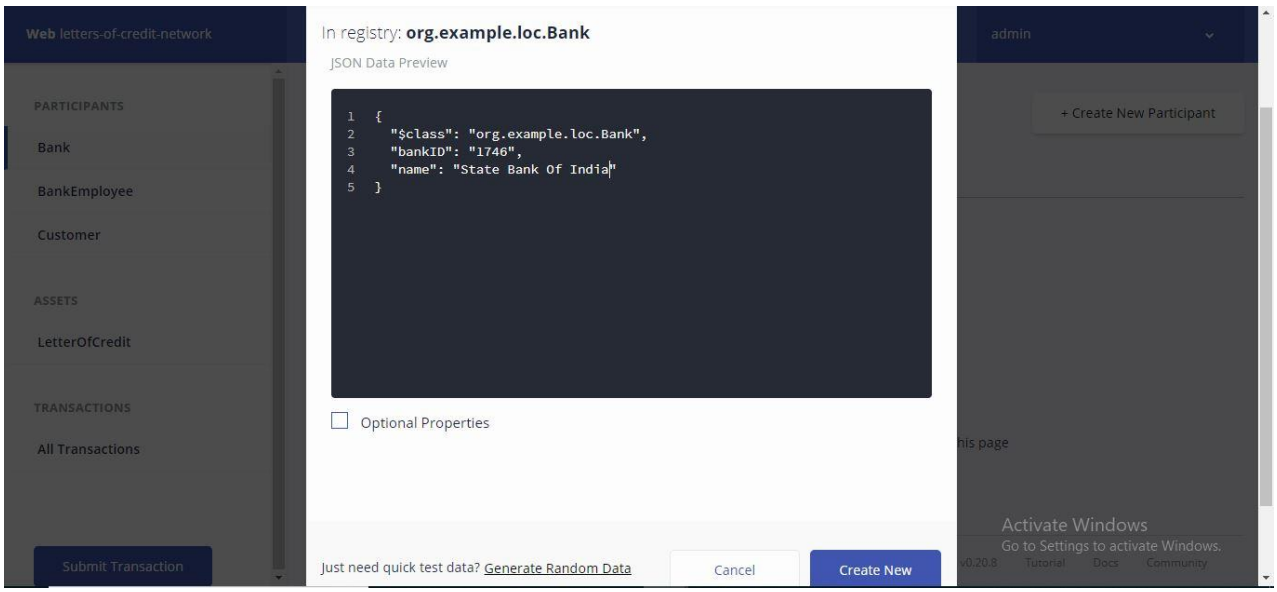
The above screen shows creating id registry for Bob as Bob001.

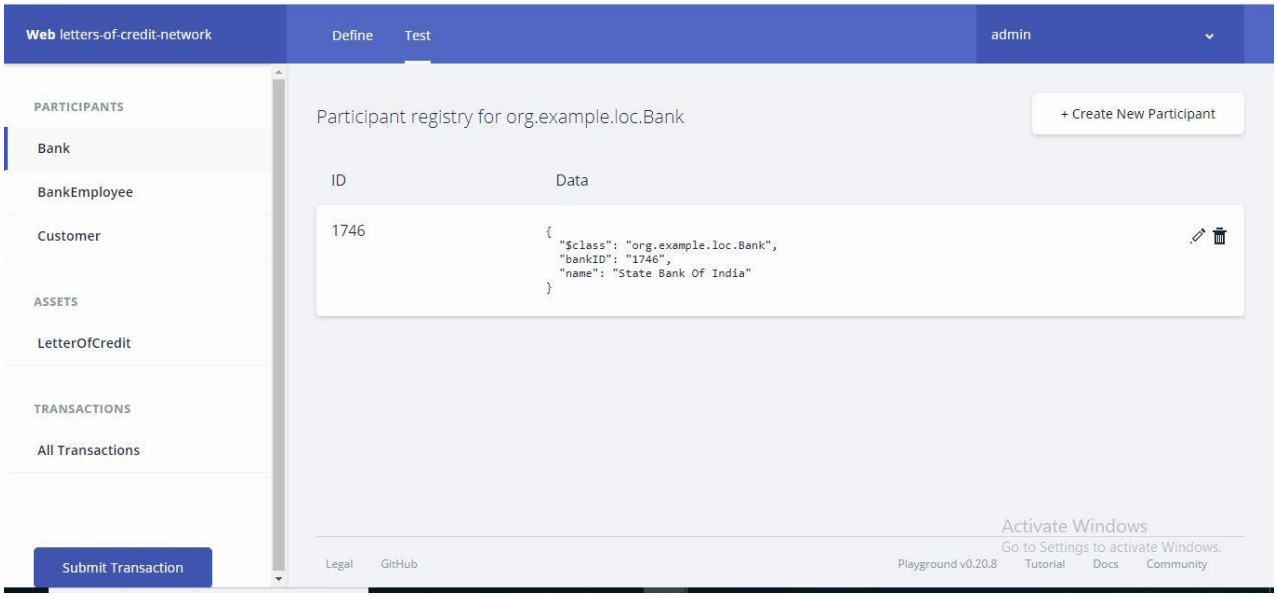




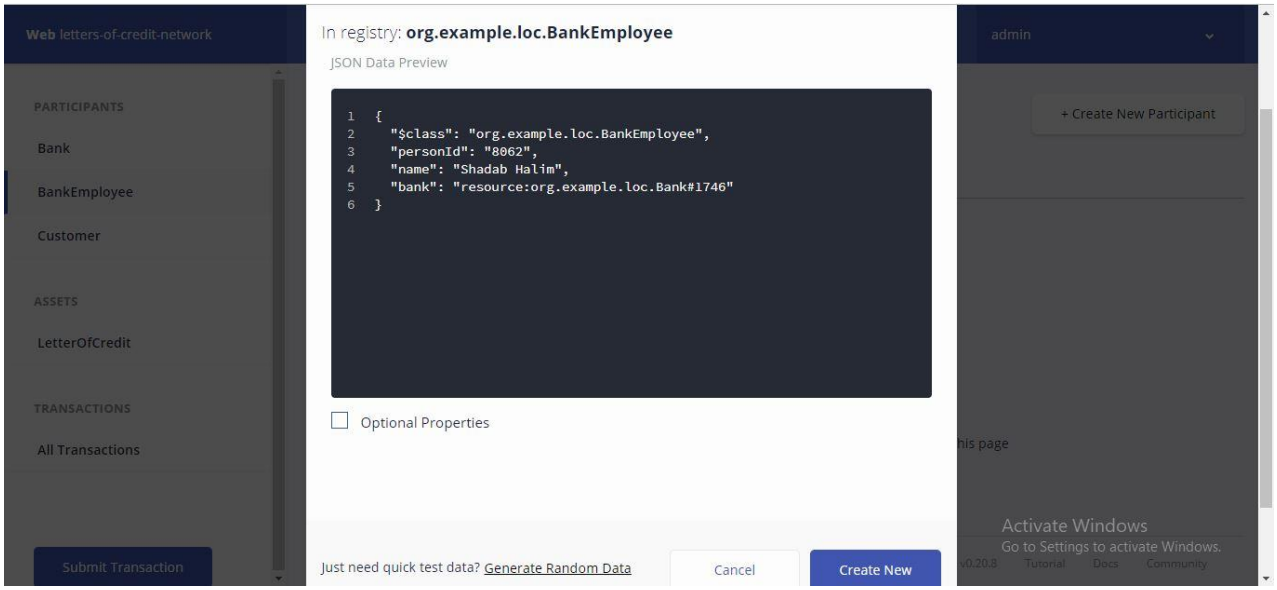
The above screen shows all the Id registered.

Now, we create Participants in the Letter of credit Network.

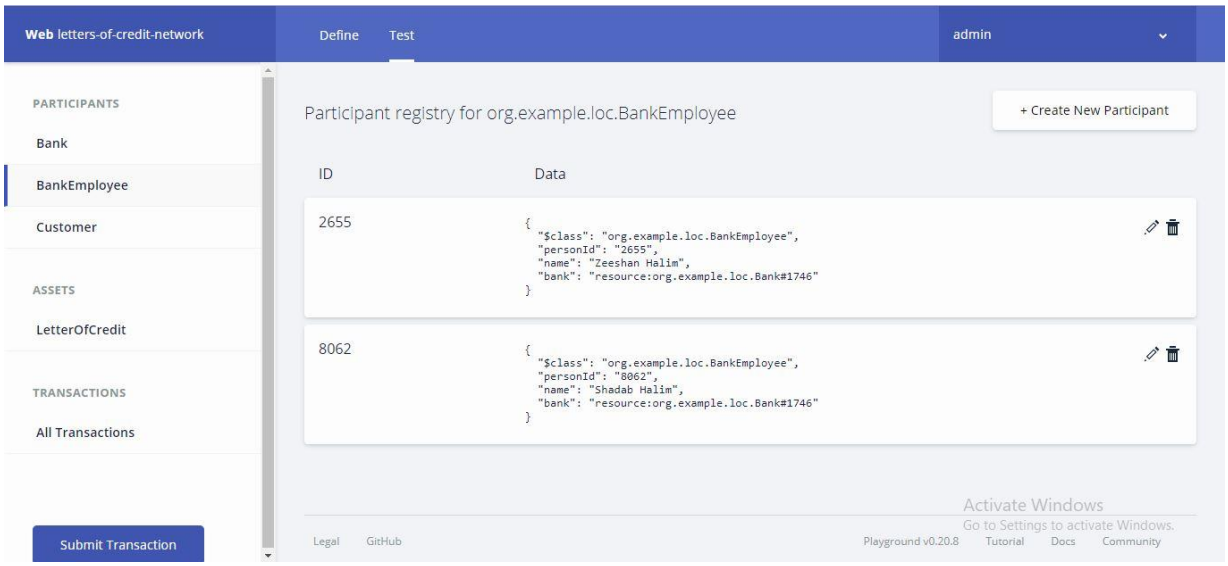




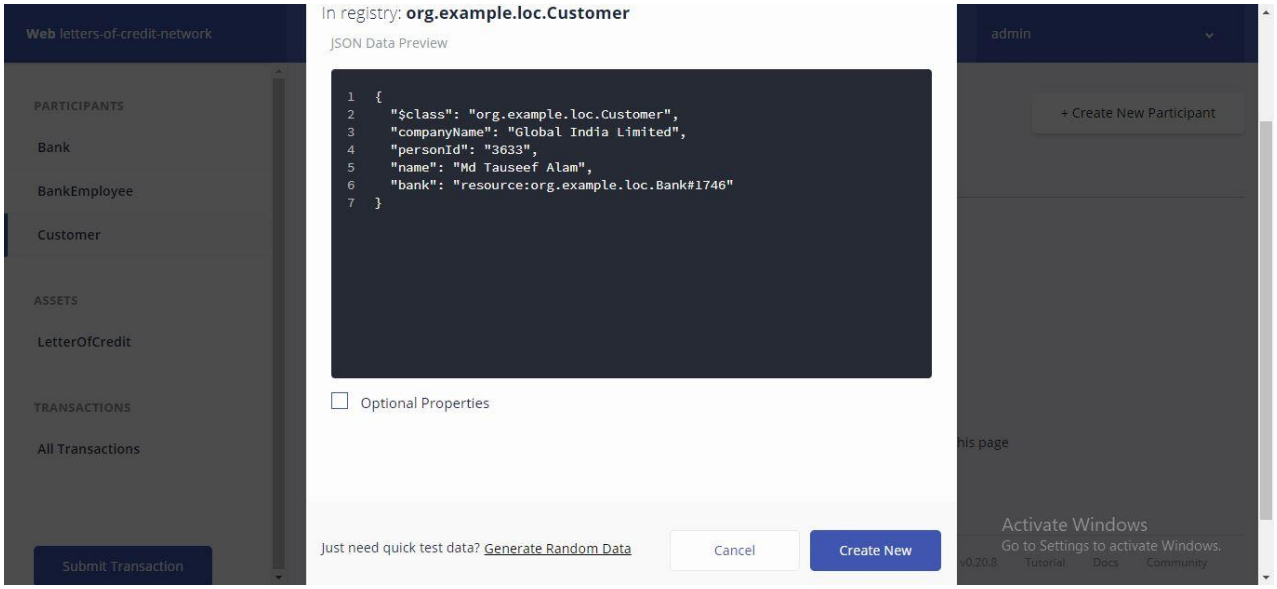
Bank Involved In Network.



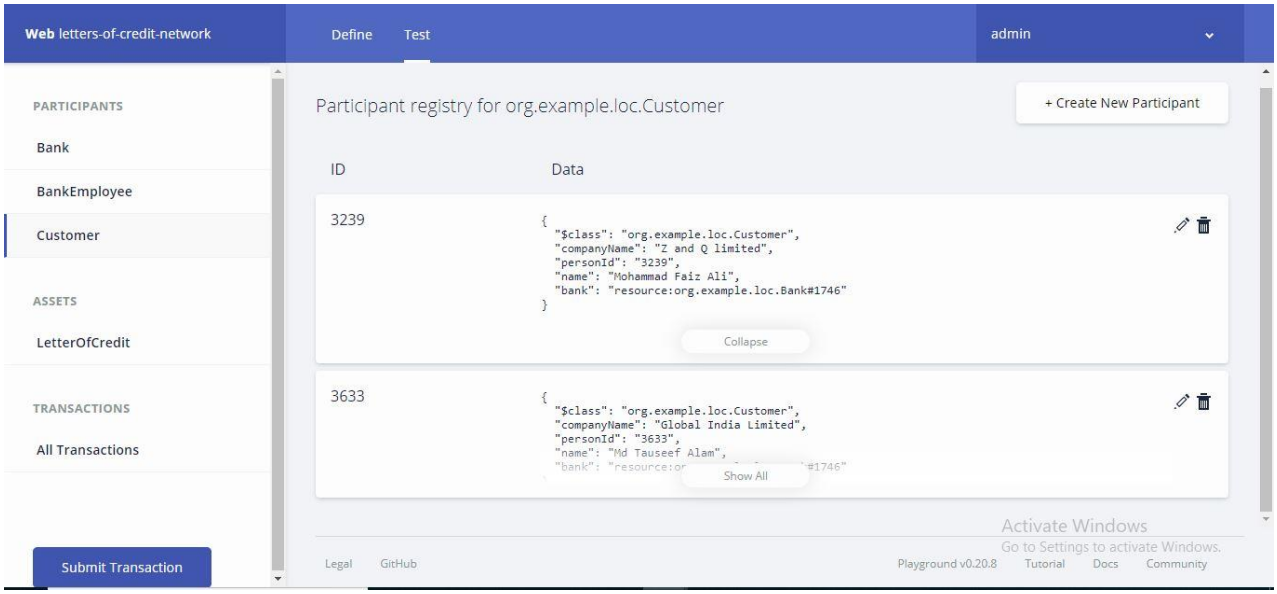
.....



Bank Employees In The Network.



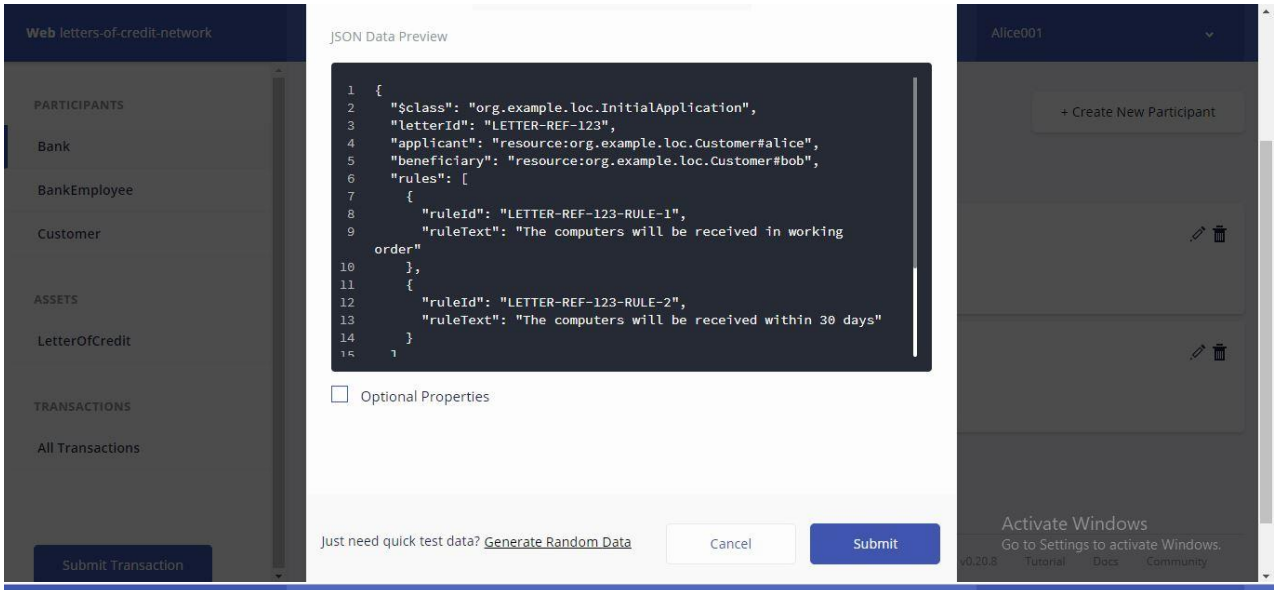
.....

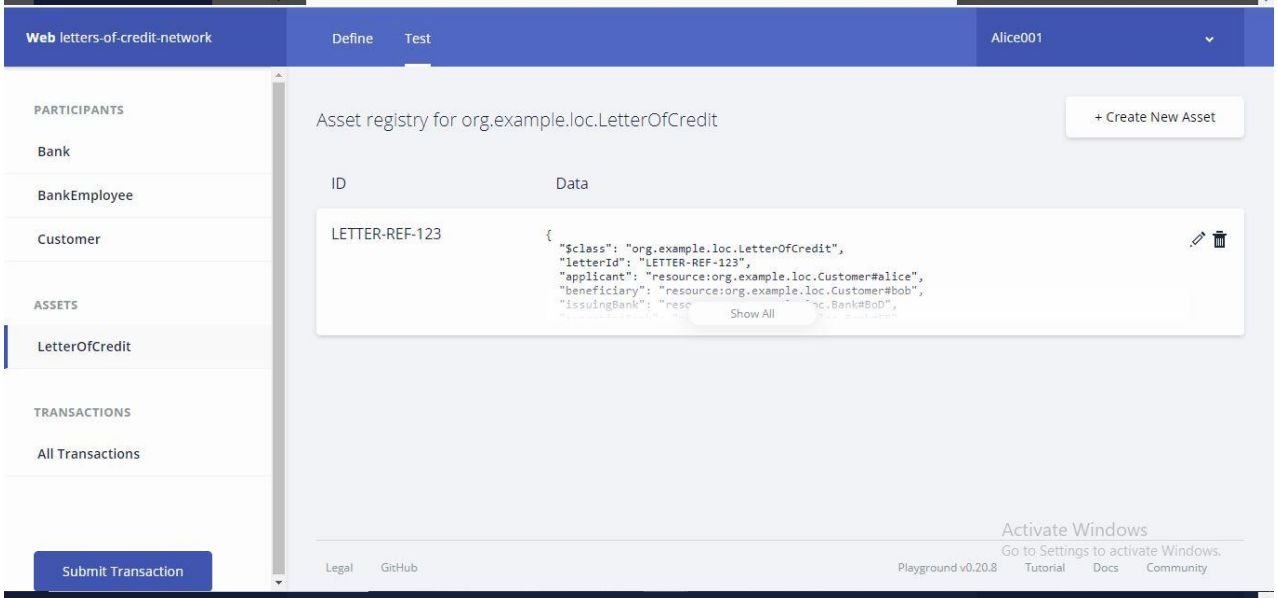


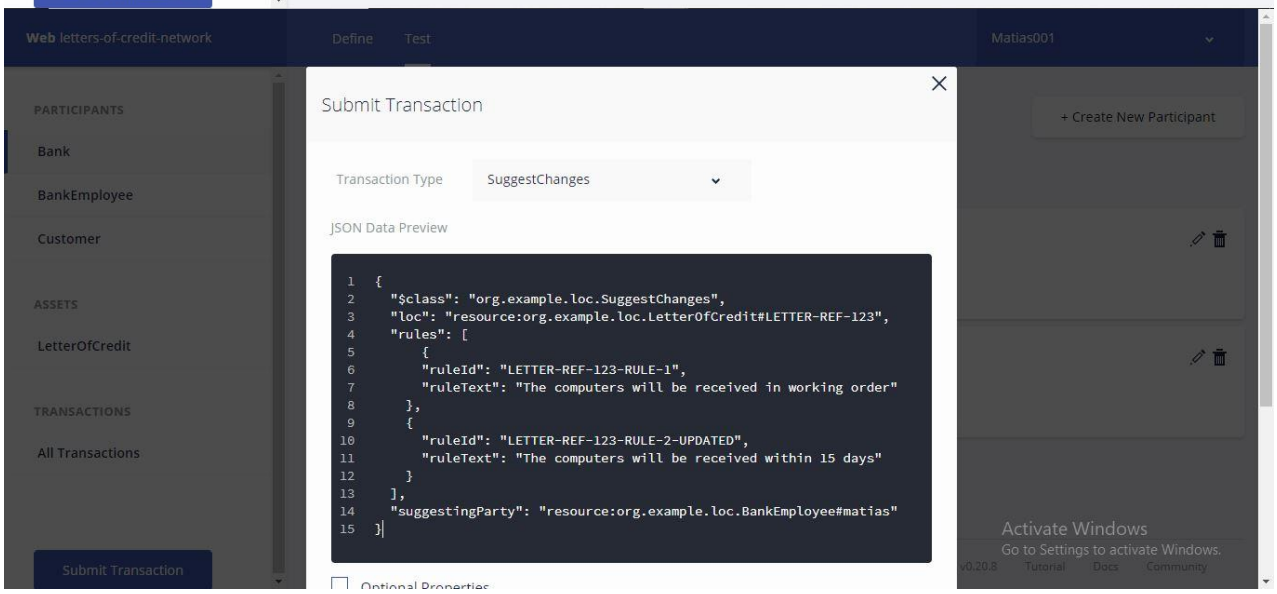
Customers In The Network.

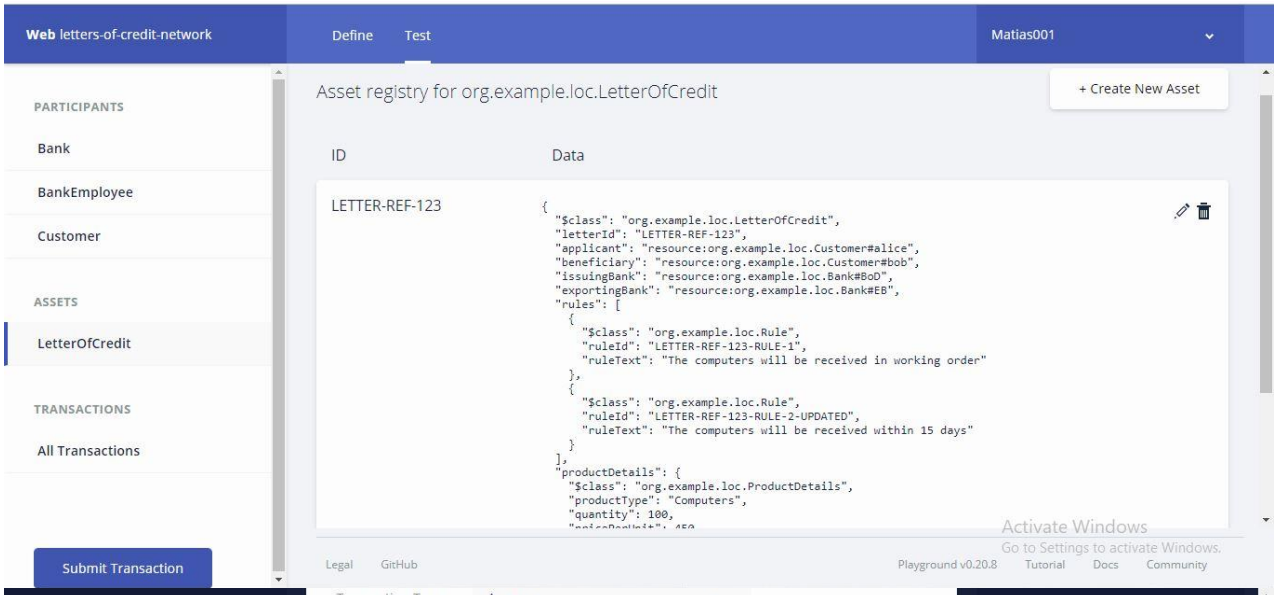
.....

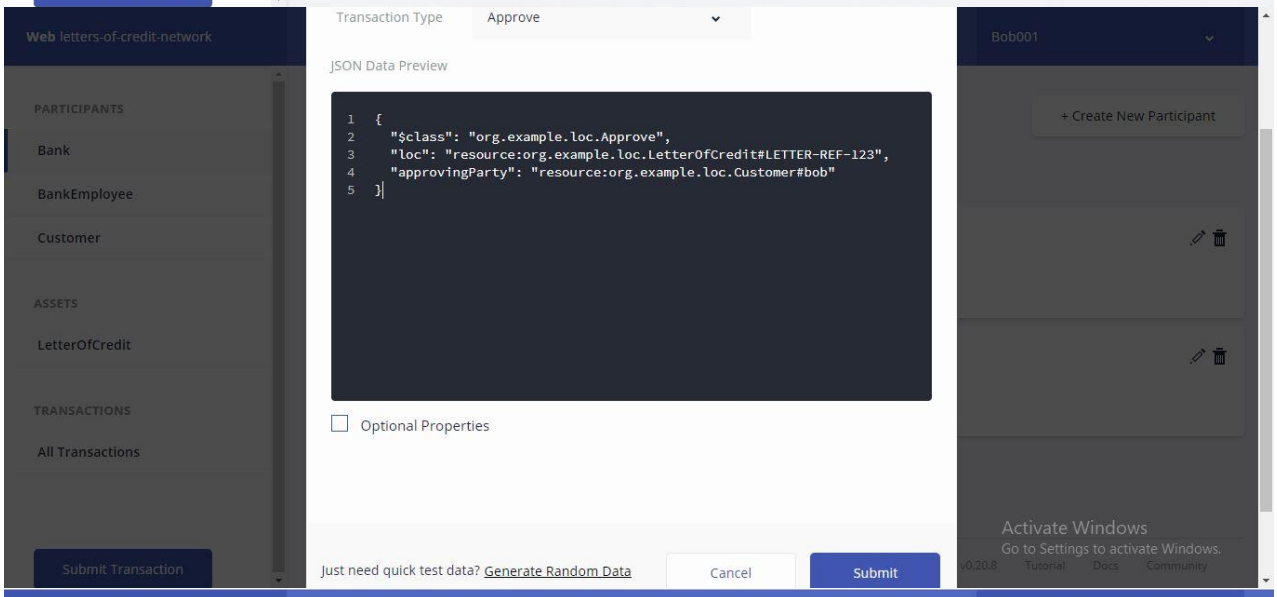
Now we will see how the letter of credit is initiated and then till it closes what happens on the blockchain network.

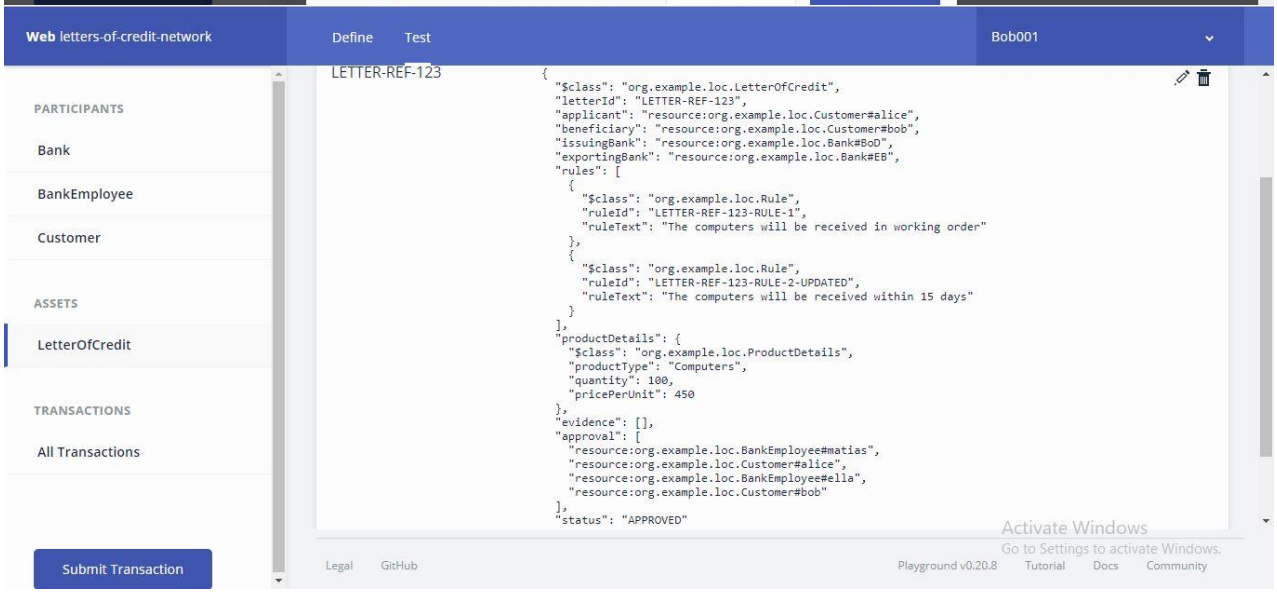

Web letters-of-credit-network
PARTICIPANTS
Bank
BankEmployee
Customer
ASSETS
LetterOfCredit
TRANSACTIONS
All Transactions
Submit Transaction
JSON Data Preview
{
"$class": "org.example.loc.InitialApplication",
"letterId": "LETTER-REF-123",
"applicant": "resource:org.example.loc.Customer#alice",
"beneficiary": "resource:org.example.loc.Customer#bob",
"rules": [
{
"ruleId": "LETTER-REF-123-RULE-1",
"ruleText": "The computers will be received in working order"
},
{
"ruleId": "LETTER-REF-123-RULE-2",
"ruleText": "The computers will be received within 30 days"
}
Optional Properties
Just need quick test data? Generate Random Data
Cancel
Submit
Alice001
+ Create New Participant
Activate Windows
Go to Settings to activate Windows.
v0.20.8
Tutorial
Docs
Community

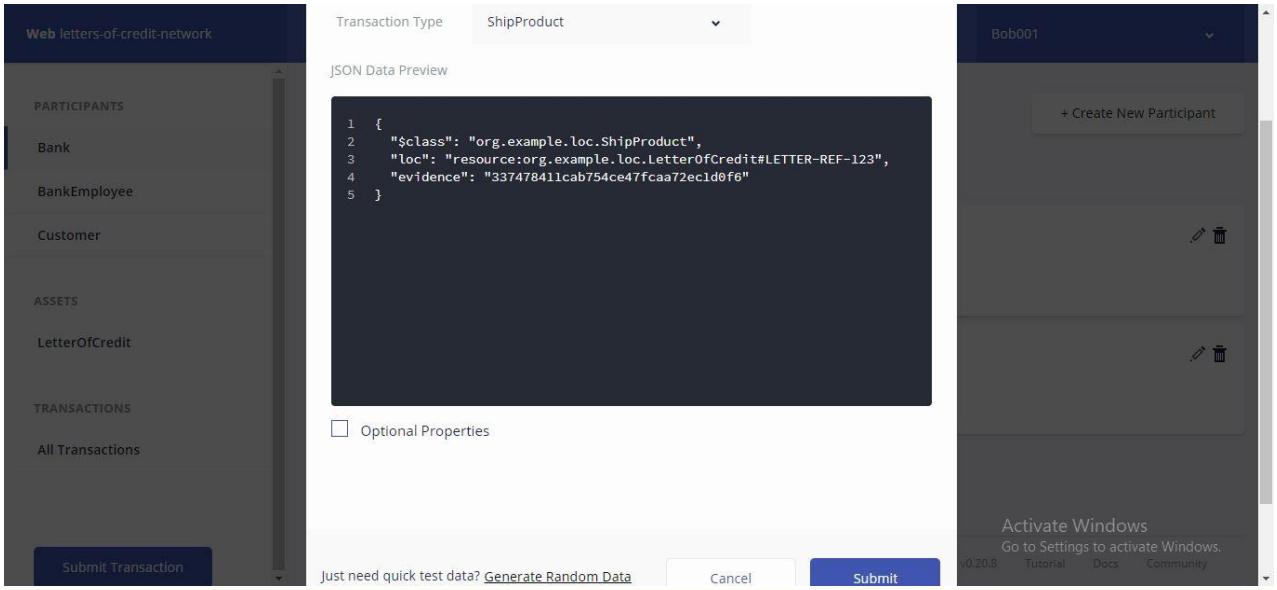

Web letters-of-credit-network
Define
Test
Alice001
PARTICIPANTS
Bank
BankEmployee
Customer
ASSETS
LetterOfCredit
TRANSACTIONS
All Transactions
Submit Transaction
Asset registry for org.example.loc.LetterOfCredit
+ Create New Asset
ID
Data
LETTER-REF-123
{
"$class": "org.example.loc.LetterOfCredit",
"letterId": "LETTER-REF-123",
"applicant": "resource:org.example.loc.Customer#alice",
"beneficiary": "resource:org.example.loc.Customer#bob",
Show All
Activate Windows
Go to Settings to activate Windows.
Legal
GitHub
Playground v0.20.8
Tutorial
Docs
Community

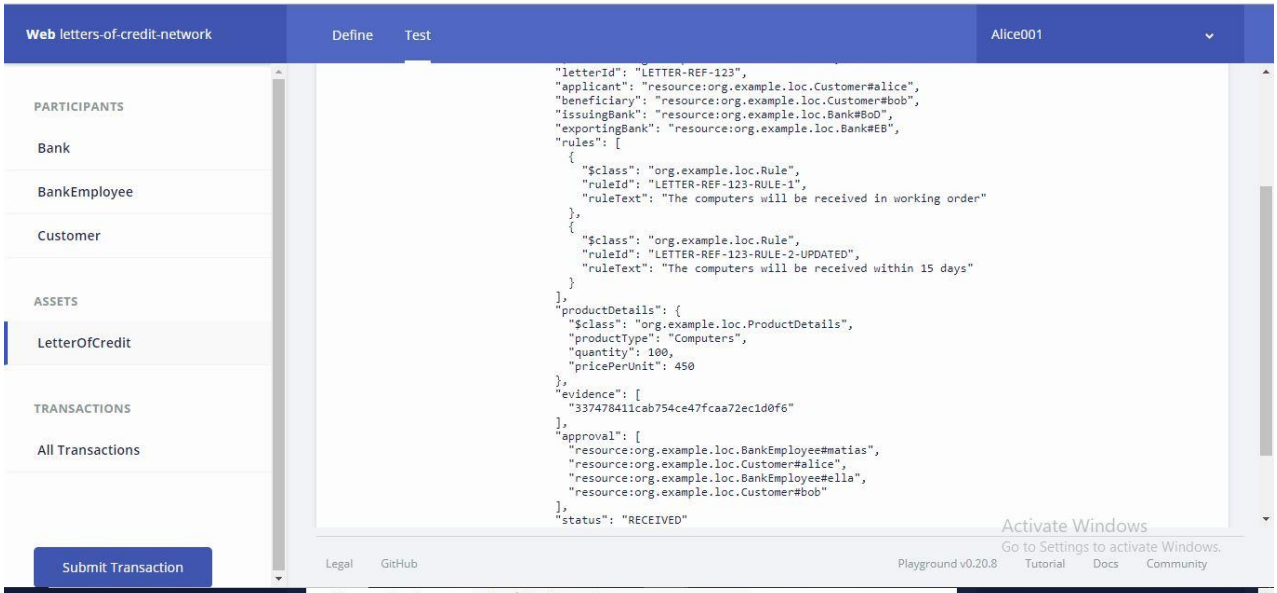

Web letters-of-credit-network
Define
Test
Matias001
PARTICIPANTS
Bank
BankEmployee
Customer
ASSETS
LetterOfCredit
TRANSACTIONS
All Transactions
Submit Transaction
Submit Transaction
Transaction Type
SuggestChanges
JSON Data Preview
{
"$class": "org.example.loc.SuggestChanges",
"loc": "resource:org.example.loc.LetterOfCredit#LETTER-REF-123",
"rules": [
{
"ruleId": "LETTER-REF-123-RULE-1",
"ruleText": "The computers will be received in working order"
},
{
"ruleId": "LETTER-REF-123-RULE-2-UPDATED",
"ruleText": "The computers will be received within 15 days"
}
],
"suggestingParty": "resource:org.example.loc.BankEmployee#matias"
}
+ Create New Participant
Activate Windows
Go to Settings to activate Windows.
v0.20.8
Tutorial
Docs
Community

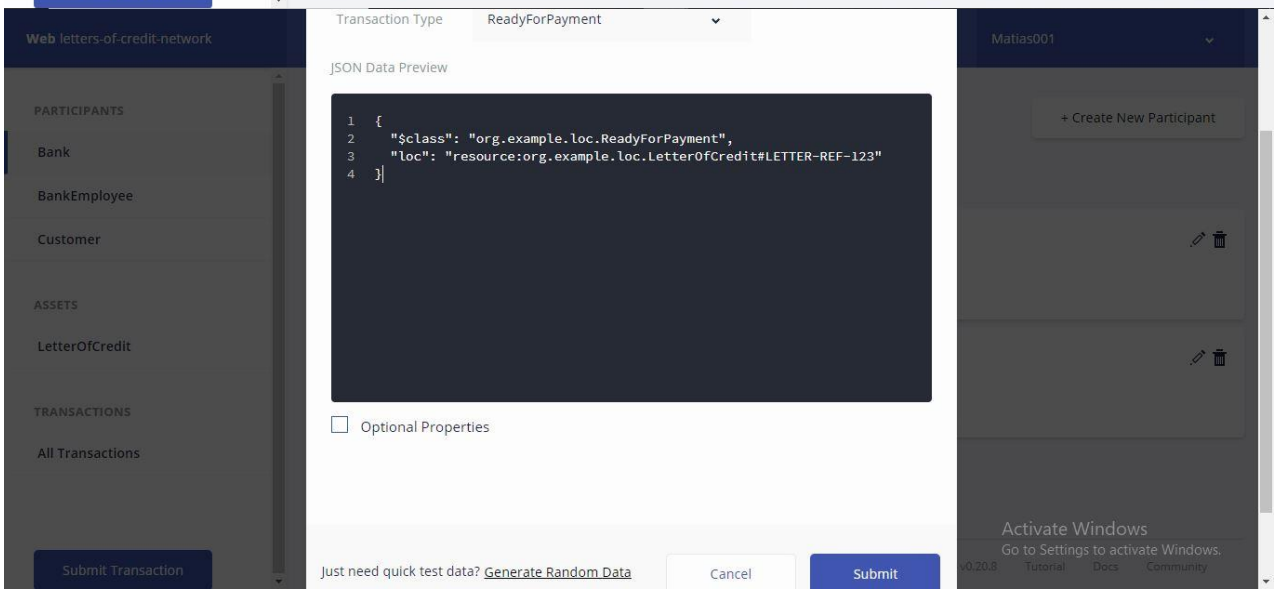

Web letters-of-credit-network
Define
Test
Matias001
PARTICIPANTS
Bank
BankEmployee
Customer
ASSETS
LetterOfCredit
TRANSACTIONS
All Transactions
Asset registry for org.example.loc.LetterOfCredit
+ Create New Asset
ID
Data
LETTER-REF-123
{
"$class": "org.example.loc.LetterOfCredit",
"letterId": "LETTER-REF-123",
"applicant": "resource:org.example.loc.Customer#alice",
"beneficiary": "resource:org.example.loc.Customer#bob",
"issuingBank": "resource:org.example.loc.Bank#BoD",
"exportingBank": "resource:org.example.loc.Bank#EB",
"rules": [
{
"$class": "org.example.loc.Rule",
"ruleId": "LETTER-REF-123-RULE-1",
"ruleText": "The computers will be received in working order"
},
{
"$class": "org.example.loc.Rule",
"ruleId": "LETTER-REF-123-RULE-2-UPDATED",
"ruleText": "The computers will be received within 15 days"
}
],
"productDetails": {
"$class": "org.example.loc.ProductDetails",
"productType": "Computers",
"quantity": 100,
Submit Transaction
Legal
GitHub
Playground v0.20.8
Tutorial
Docs
Community
Activate Windows
Go to Settings to activate Windows.

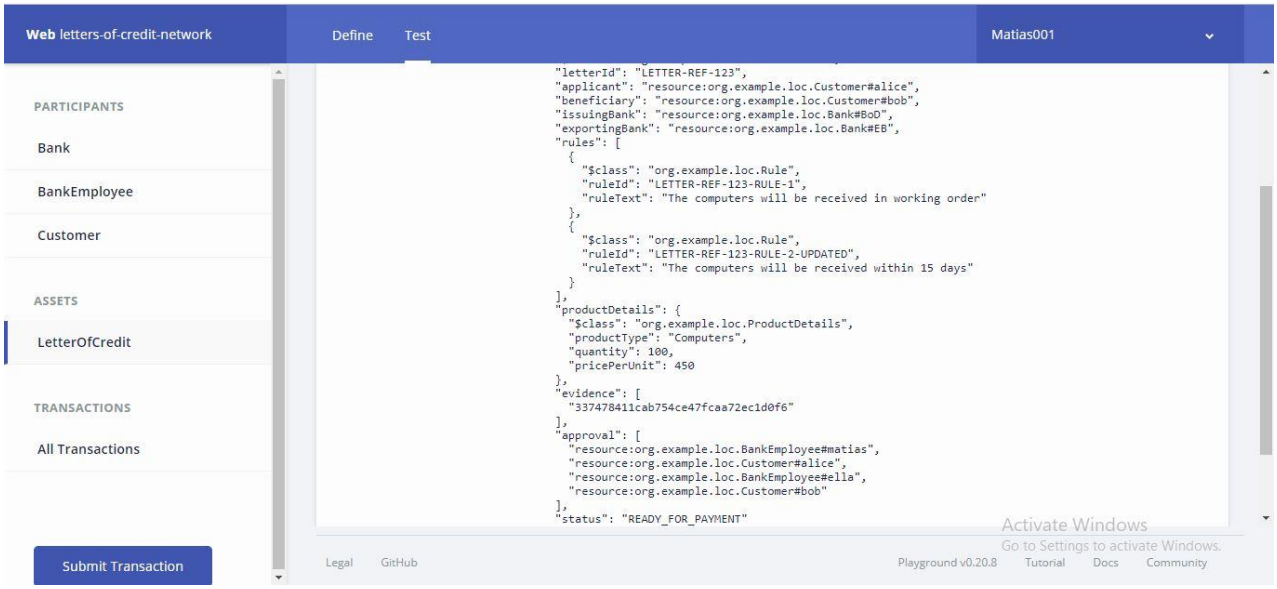

Web letters-of-credit-network
Transaction Type
Approve
Bob001
JSON Data Preview
{
"$class": "org.example.loc.Approve",
"loc": "resource:org.example.loc.LetterOfCredit#LETTER-REF-123",
"approvingParty": "resource:org.example.loc.Customer#bob"
}
Optional Properties
+ Create New Participant
Just need quick test data? Generate Random Data
Cancel
Submit
Activate Windows
Go to Settings to activate Windows.


Web letters-of-credit-network
Define
Test
Bob001
LETTER-REF-123
{
"$class": "org.example.loc.LetterOfCredit",
"letterId": "LETTER-REF-123",
"applicant": "resource:org.example.loc.Customer#alice",
"beneficiary": "resource:org.example.loc.Customer#bob",
"issuingBank": "resource:org.example.loc.Bank#BoD",
"exportingBank": "resource:org.example.loc.Bank#EB",
"rules": [
{
"$class": "org.example.loc.Rule",
"ruleId": "LETTER-REF-123-RULE-1",
"ruleText": "The computers will be received in working order"
},
{
"$class": "org.example.loc.Rule",
"ruleId": "LETTER-REF-123-RULE-2-UPDATED",
"ruleText": "The computers will be received within 15 days"
}
],
"productDetails": {
"$class": "org.example.loc.ProductDetails",
"productType": "Computers",
"quantity": 100,
"pricePerUnit": 450
},
"evidence": [],
"approval": [
"resource:org.example.loc.BankEmployee#matias",
"resource:org.example.loc.Customer#alice",
"resource:org.example.loc.BankEmployee#ella",
"resource:org.example.loc.Customer#bob"
],
"status": "APPROVED"
Submit Transaction
Legal
GitHub
Playground v0.20.8
Tutorial
Docs
Community
Activate Windows
Go to Settings to activate Windows.

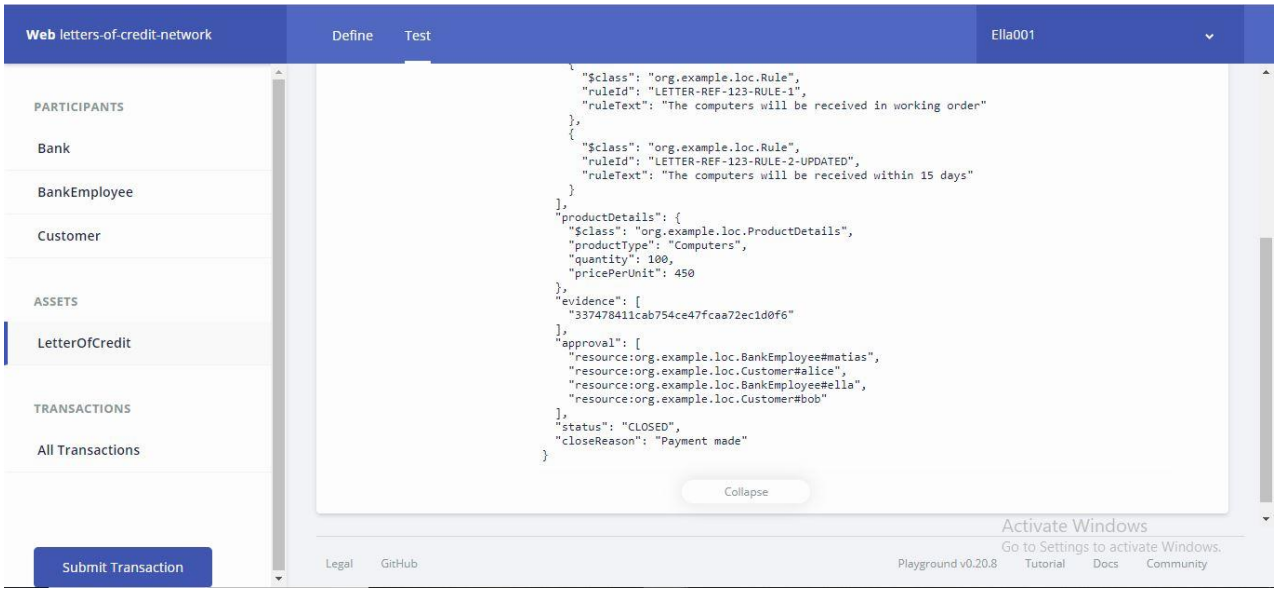

Web letters-of-credit-network
Transaction Type
ShipProduct
Bob001
JSON Data Preview
+ Create New Participant
PARTICIPANTS
Bank
BankEmployee
Customer
ASSETS
LetterOfCredit
TRANSACTIONS
All Transactions
Submit Transaction
{
"$class": "org.example.loc.ShipProduct",
"loc": "resource:org.example.loc.LetterOfCredit#LETTER-REF-123",
"evidence": "337478411cab754ce47fcaa72ec1d0f6"
}
Optional Properties
Just need quick test data? Generate Random Data
Cancel
Submit
Activate Windows
Go to Settings to activate Windows.
v0.20.8
Tutorial
Docs
Community


Web letters-of-credit-network
Define
Test
Alice001
PARTICIPANTS
Bank
BankEmployee
Customer
ASSETS
LetterOfCredit
TRANSACTIONS
All Transactions
Submit Transaction
"letterId": "LETTER-REF-123",
"applicant": "resource:org.example.loc.Customer#alice",
"beneficiary": "resource:org.example.loc.Customer#bob",
"issuingBank": "resource:org.example.loc.Bank#BoD",
"exportingBank": "resource:org.example.loc.Bank#EB",
"rules": [
{
"$class": "org.example.loc.Rule",
"ruleId": "LETTER-REF-123-RULE-1",
"ruleText": "The computers will be received in working order"
},
{
"$class": "org.example.loc.Rule",
"ruleId": "LETTER-REF-123-RULE-2-UPDATED",
"ruleText": "The computers will be received within 15 days"
}
],
"productDetails": {
"$class": "org.example.loc.ProductDetails",
"productType": "Computers",
"quantity": 100,
"pricePerUnit": 450
},
"evidence": [
"337478411cab754ce47fcaa72ec1d0f6"
],
"approval": [
"resource:org.example.loc.BankEmployee#matias",
"resource:org.example.loc.Customer#alice",
"resource:org.example.loc.BankEmployee#ella",
"resource:org.example.loc.Customer#bob"
],
"status": "RECEIVED"
Activate Windows
Go to Settings to activate Windows.
Legal
GitHub
Playground v0.20.8
Tutorial
Docs
Community


Web letters-of-credit-network
Transaction Type
ReadyForPayment
Matias001
JSON Data Preview
+ Create New Participant
PARTICIPANTS
Bank
BankEmployee
Customer
ASSETS
LetterOfCredit
TRANSACTIONS
All Transactions
Submit Transaction
{
"$class": "org.example.loc.ReadyForPayment",
"loc": "resource:org.example.loc.LetterOfCredit#LETTER-REF-123"
}
Optional Properties
Just need quick test data? Generate Random Data
Cancel
Submit
Activate Windows
Go to Settings to activate Windows.
v0.20.8
Tutorial
Docs
Community

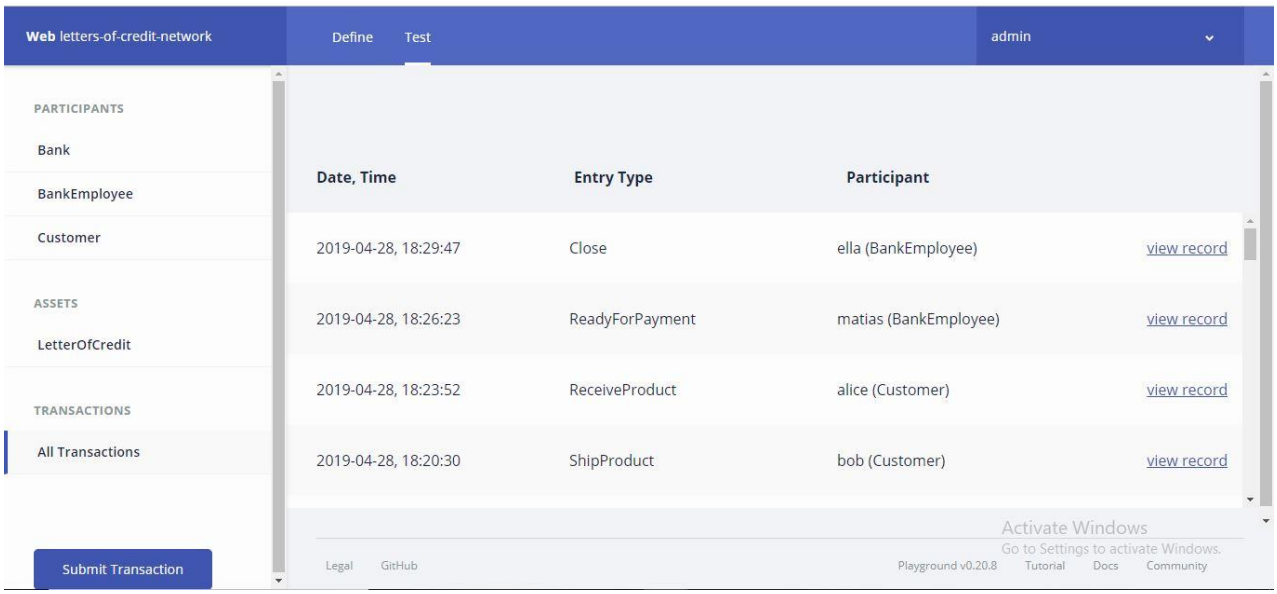


.....

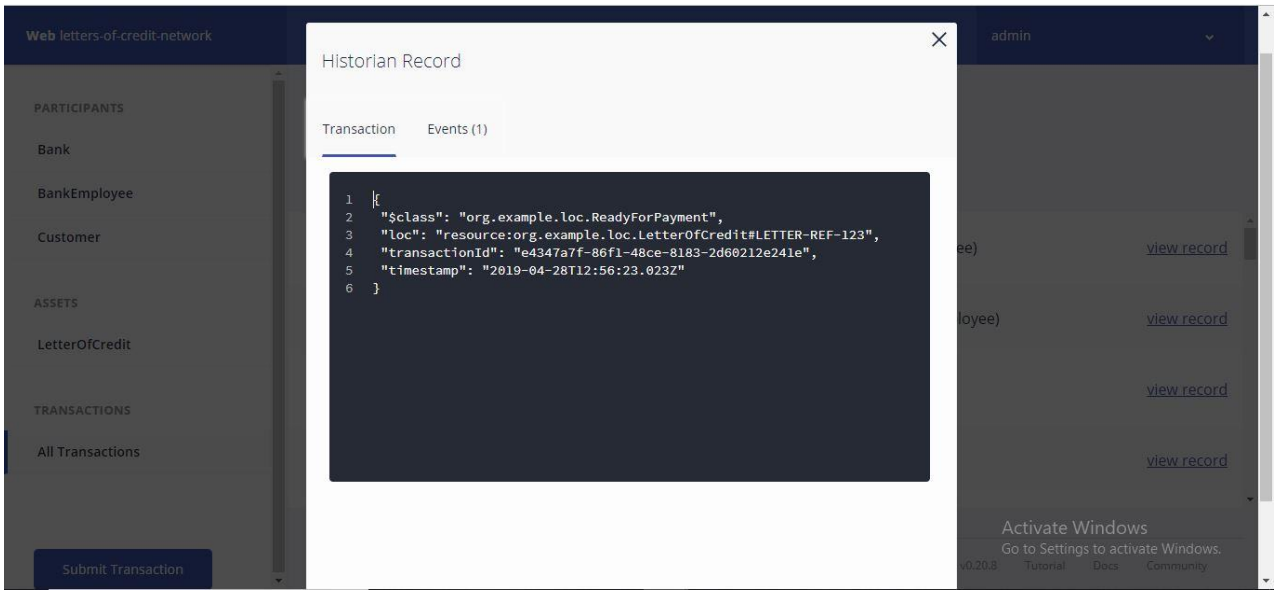


END/CLOSING OF OUR LETTER OF CREDIT.

NOW Since everything is on the blockchain how it looks we will see it(by ADMIN authority).



• • • • • • • • • • •



5. MANAGING THE PROJECT

Project management involves the planning, monitoring, and control of the people, process, and events that occur as software evolves from a preliminary concept to an operational implementation. Project managers plan, monitor, and control the work of a team of software engineers. Effective software project management focuses on the four P‟s: people, product, process, and project.

5.1 Function Points

Function-oriented software metrics use a measure of the functionality delivered b_y the application as a normalization value. Since, "functionality" cannot be measured directly; it must be derived indirectly using other direct measures. Function points are computed by completing the table 4.1. Five information domain characteristics are determined and counts are provided in the appropriate table location. Information domain values are defined in the following manner:

Number of user inputs: Each user input that provides distinct application oriented data to the software is counted. Inputs should be distinguished from inquiries, which are counted separately.

Number of user outputs: Each user output that provides application oriented information to the user is counted. In this context output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.

Number of user inquiries: An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an on-line output. Each distinct inquiry is counted.

Number of files: Each logical master file (i.e., a logical grouping of data that may be one part of a large database or a separate file) is counted.

Number of external interfaces: All machine readable interfaces (e.g., data files on storage media) that are used to transmit information to another system are counted.

Weighting factor

Measurement factors	**Count**	**Simple**	**Average**	**Complex**	
Number of user inputs	2	3	4(2)	6	8
Number of user output	2	4	5(2)	7	10
Number of user inquiries	3	3	4(3)	6	12

Number of internal logical files	1	7	10(1)	15	10
Number of external interface files	2	5	7(2)	10	14
Count total					54

Table 3.1

The *Fi* (*i* = 1 to 14) are "Complexity Adjustment Values" based on responses to the following questions:

1. Does the system require reliable backup and recovery?	5
2. Are data communications required?	5
3. Are there distributed processing functions?	3
4. Is performance critical?	4
5. Will the system run in an existing, heavily utilized operational environment?	4
6. Does the system require on-line data entry?	5
7. Does the on-line data entry require the input transaction to be built over multiple?	3
8. Are the master files updated on-line?	5
9. Are the inputs, outputs, files, or inquiries complex?	1
10. Is the internal processing complex?	1
11. Is the code designed to be reusable?	2
12. Are conversion and installation included in the design?	3
13. Is the system designed for multiple installations in different organizations?	5
14. Is the application designed to facilitate change and ease of use by the user?	5

Once these data have been collected, a complexity value is associated with each count. Organizations that use function point methods develop criteria for determining whether a particular entry is simple, average, or complex. To compute function points (FP), the following relationship is used:

FP = count total *[0.65 + 0.01 *Σ (Fi)]

= 54*(0.65 + 0.01*51)

=54*1.16

=62 (approx.)

Where count total is the sum of all FP entries obtained from Figure.

5.2 ESTIMATING EFFORTS

Barry Boehm introduced a hierarchy of software estimation models bearing the name COCOMO, for COnstructive COst MOdel. The original COCOMO model became one of the most widely used and discussed software cost estimation models in the industry. The COCOMO II application composition model uses object points.

The object point is an indirect software measure that is computed using counts of the no. of screens (user interface), reports and components likely to be required to build the application. Each object instance is classified into one of three complexity levels using criteria suggested by Boehm.

Once complexity is determined, the number of screens, reports, and components are weighted. The object point count is then determined by multiplying the original number of object instances by the weighting factor in and summing to obtain a total object point count. When component-based development or general software reuse is to be applied, the percent of reuse (%reuse) is estimated and the object point count is adjusted:

NOP = (object points) x [(100 -%reuse)/100],

where NOP is defined as new object points.

To derive an estimate of effort based on the computed NOP value, "productivity rate" must be derived.

PROD = NOP/person-month

Table 3.1 presents the productivity rate for different levels of developer experience and development environment maturity. Once the productivity rate has been determined, an estimate of project effort can be derived as

Estimated effort = NOP/PROD

Object type	**No. of objects**	**Complexity Weight**			**Count**
		Simple	Medium	Difficult	
Screen	4	1(2)	2(2)	3(1)	7
Report	2	2(2)	5(1)	8(0)	9
3GL component	2			10(2)	20
Object points sum					36

Table 3.2 Estimating object points

Data used in estimating effort are:

(1) Object points is 36(taken from table 3.2)

(2) Estimated reuse is 36%

(3) Prod is 13 (average value taken)

NOP = Object points *[(100-reuse%)/100]

= 36*[(100-35)/100]

= 24

ESTIMATED EFFORT = NOP/PROD

= 24/13

= 2 person-months

Hence estimated effort of the project is 2 person-months.

5.3 ESTIMATING SCHEDULE

Putnam and Myers suggest a set of equations derived from the software equation. Minimum development time is defined as

$t_{min} = 8.14(LOC/P)^{0.43}$ in months for t_{min} > 6 months

Since project's time period is less than 6 months, the above equation cannot be applied.

An estimation model of the form:

$$E = \left[LOC - \frac{B^{.333}}{P}\right]^3 \times \left(\frac{1}{t^4}\right) \qquad \text{(equation 1)}$$

where E = effort in person-months or person-years

t = project duration in months or years

B = "special skills factor"

P = "productivity parameter"

Calculating development time for project, using equation 1 and effort calculated in section 3.2

$$2 = \left[1000 - \frac{0.01^{.333}}{8000}\right]^3 \times \left(\frac{1}{t^4}\right)$$

$t^4 = 0.06$

Hence estimated schedule of the project is 6 months.

5.4 TIMELINE CHARTS

When creating a software project schedule, the planner begins with a set of task. If automated tools are used, the work breakdown is input as a task network or task outline. Effort, duration, and start date are then input for each task. In addition, tasks may be assigned to specific individuals. A ***timeline chart***, also called a ***Gantt chart***, is generated. A timeline chart can be developed for the entire project.

Timeline depicts a part of a software project schedule that emphasizes. All project tasks are listed in the left-hand column. The horizontal bars indicate the duration of each task. When multiple bars occur at the same time on the calendar, task concurrency is implied. The diamonds indicate milestones. Once the information necessary for the generation of a timeline chart has been input, the majority of software project scheduling tools produce *project tables*—a tabular listing of all project tasks, their planned and actual start- and end-dates, and a variety of related information.

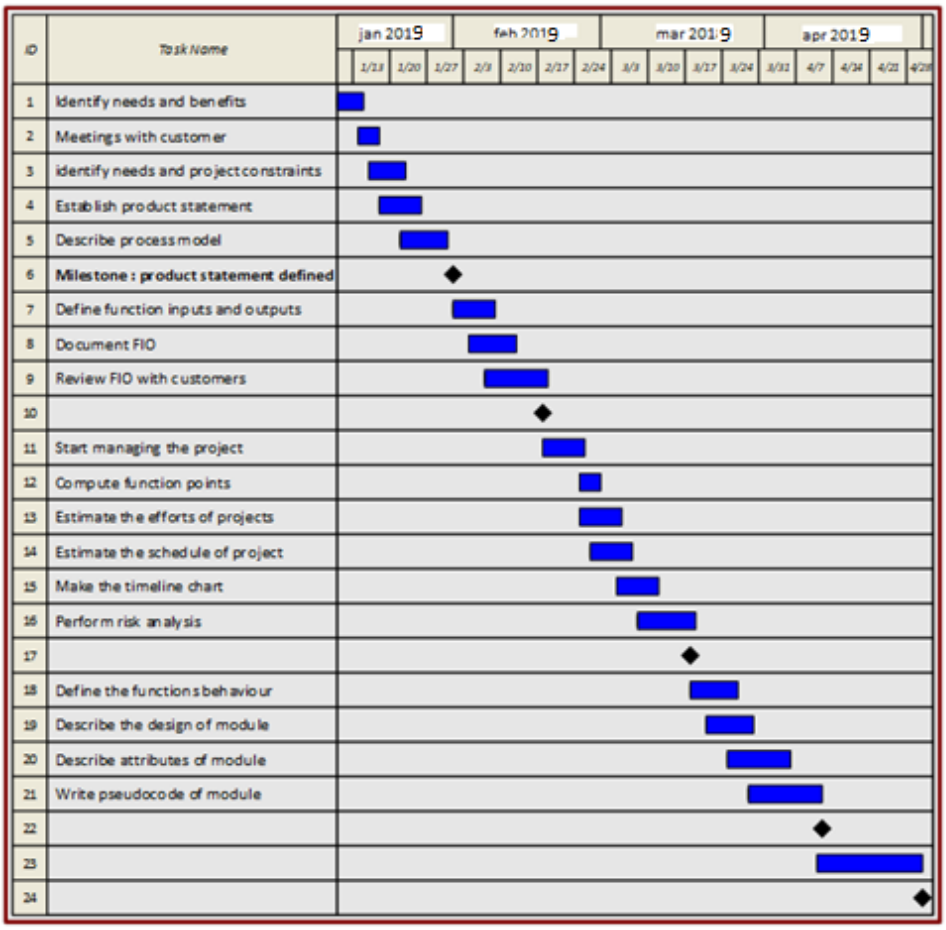


Fig.3.5 Timeline chart

6. TESTING

Testing is the process of running a system with the intention of finding errors. Testing enhances the integrity of a system by detecting deviations in design and errors in the system. Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system. Testing also adds value to the product by conforming to the user requirements.

The main purpose of testing is to detect errors and error prone areas in a system. Testing must be thorough and well-planned. A partially tested system is as bad as an untested system. And the price of an untested and under-tested system is high.

The implementation is the final and important phase. It involves user-training, system testing in order to ensure successful running of the proposed system. The user tests the system and changes are made according to their needs. The testing involves the testing of the developed system using various kinds of data. While testing, errors are noted and correctness is the mode.

7. Conclusion

Letters of Credit Network

This network tracks letters of credit from application through to closure.

Models within this business network

Participants

Customer, Bank Employee,Bank

Assets

LetterOfCredit

Transactions

InitialApplication, Approve, Reject, SuggestChanges, ShipProduct, ReceiveProduct, ReadyForPayment, Close, CreateDemoParticipants

Events

InitialApplicationEvent, ApproveEvent, RejectEvent, SuggestChangesEvent, ShipProductEvent, ReceiveProductEvent, ReadyForPaymentEvent, CloseEvent

The use of blockchain technology helped a lot to create a trust and safe and secure network. The bank (What it does(LOC)):

- Bank handling letters of credit (LOC) wants to offer them to a wider range of clients including startups
- Currently constrained by costs & the time to execute

Now why and how we use blockchain:
1.)Blockchain provides common ledger for letters of credit
2.) Allows all counter-parties to have the same validated
record of transaction and fulfillment

REFERENCES

● *Software Engineering- A practitioner's Approach by Roger S. Pressman: 6th edition McGraw Hill, 2005*

● *An Integrated Approach to Software Engineering by Pankaj Jalote: 3rd edition Springer, 2005*

- https://composer-playground.mybluemix.net/
- https://www.coursera.org/learn/ibm-blockchain-essentials-for-developers