A Major Project On

Cryptosystem for medicine supply chain management using Hyperledger-Fabric.

Submitted in partial fulfillment of the requirements for the award of the

Bachelor of Technology

In

Department of Computer Science and Engineering

By

Prashant Sharma	**17241A05Y3**
Abhishek Bhandwalkar	**17241A05U2**
Srinivas Golla	**17241A05U7**
Shaik Asif	**18245A0568**
G. Bhanu Teja	**18245A0567**

Under the Esteemed guidance of

Dr. K. Kavitha. Professor

Department of Computer Science and Engineering

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY (Approved by AICTE, Autonomous under JNTUH, Hyderabad, Bachupally, Kukatpally, Hyderabad-500090

CERTIFICATE

This is to certify that the major project entitled "**Cryptosystem for medicine supply chain management using Hyperledger-Fabric**." is submitted by **Prashant Sharma (17241A05Y3), Abhishek Bhandwalkar (17241A05U2), Srinivas Golla (17241A05U7), Shaik Asif (18245A0568), G. Bhanu Teja (18245A0567)** in partial fulfillment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during academic year 2020-2021.

INTERNAL GUIDE

Dr. K. Kavitha

(Professor)

HEAD OF THE DEPARTMENT

Prof. Dr. K. MADHAVI

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

There are many people who helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we would like to express our deep gratitude towards our internal guide **Dr. K. Kavitha, Prof.** Department of CSE for his support in the completion of our dissertation. We wish to express our sincere thanks to **Dr. K. Madhavi, HOD, Department of CSE** and to our principal **Dr. J. Praveen** for providing the facilities to complete the dissertation. We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

Prashant Sharma (17241A05Y3)

Abhishek Bhandwalkar (17241A05U2)

Srinivas Golla (17241A05U7)

Shaik Asif (18245A0568)

G. Bhanu Teja (18245A0567)

DECLARATION

We hereby declare that the industrial major project entitled **"Cryptosystem for medicine supply chain management using Hyperledger-Fabric"** is the work done during the period from **1th March 2021 to 10th June 2021** and is submitted in the partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad).The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

Prashant Sharma

(17241A05Y3)

Abhishek Bhandwalkar

(17241A05U2)

Srinivas Golla

(17241A05U7)

Shaik Asif

(18245A0568)

G. Bhanu Teja

(18245A0567)

ABSTRACT

The increasing costs and the splintered nature of the supply chain in healthcare create various challenges. The medical industry needs expert solutions that can streamline supply chain operations and processes cost-effectively. Dominant healthcare players are exploring blockchain technologies to attain efficiencies and gain greater control over their supply chain. In this project, we present the current state of the medical supply chain and compile the benefits and hurdles of the distributed organization and supervision of supply chains management. Concentrating on healthcare, we discuss the applicability of hyper ledger fabric and how we harness its power to create changes in the manual way of supply chain management in healthcare and come up with a secure and safer solution to prevent competitors from gaining confidential information and making the process of record storing absolutely innovative.

TABLE OF CONTENTS

CHAPTER -1

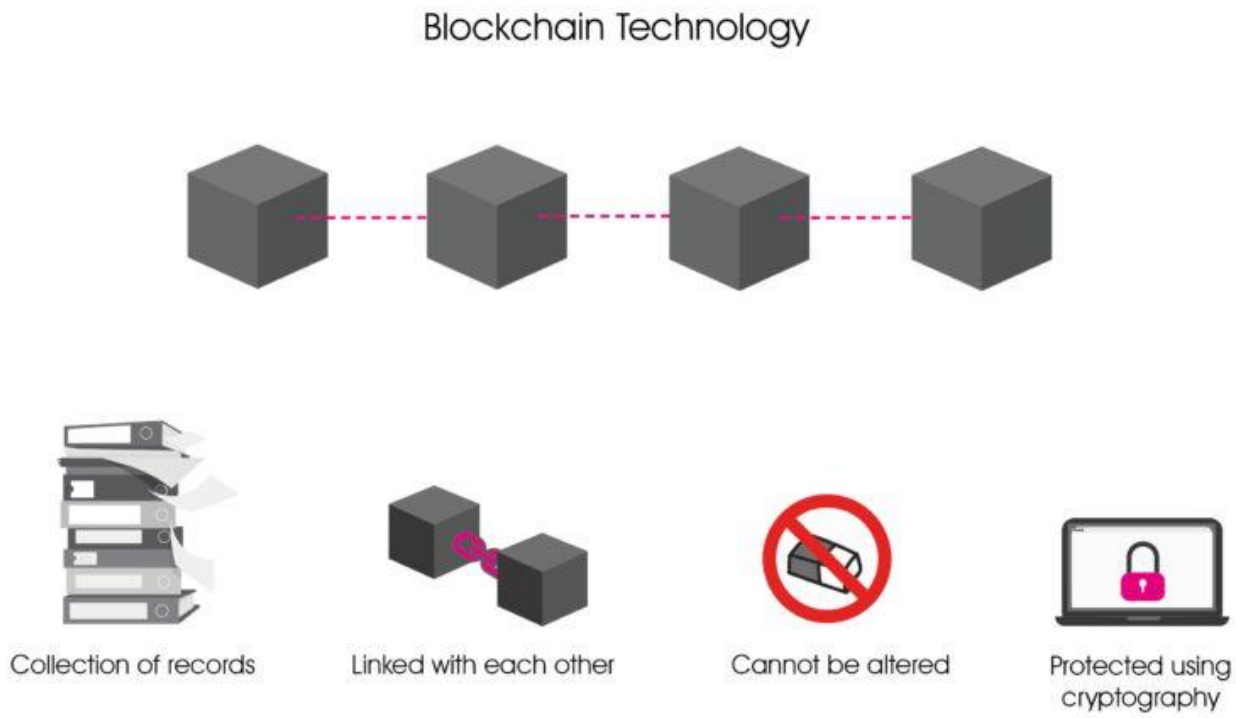
1. INTRODUCTION

In usual terms, a blockchain is an immutable transaction ledger for a distributed network maintained within *peer nodes*. The nodes keep a copy of this ledger by employing transactions validated by a *consensus protocol*, classified into blocks incorporating a hash that connects each block to the previous block.

A blockchain is a digital log of transactions that is duplicated and distributed across the blockchain's complete network of computer systems. Each block on the chain contains a number of transactions, and whenever a new transaction occurs on the blockchain, a record of that transaction is added to the ledger of each participant. Distributed Ledger Technology is a decentralized database that is administered by various people (DLT).

Blockchain is a sort of distributed ledger technology in which transactions are recorded using a hash, which is an immutable cryptographic signature.

The famous blockchain application which is the king of cryptocurrency is Bitcoin. Ethereum, an alternative cryptocurrency, takes a different path, integrating various features as Bitcoin and computing *intelligent contracts* and create a platform for distributed applications. Bitcoin and Ethereum fall into the *public permissionless* blockchain technology. These are networks that are public and open to everyone, where members interact anonymously.



Source: https://www.mondaq.com/india/fin-tech/1044486/frequently-asked-questions-about-cryptocurrency-and-blockchain-technology

As bitcoin, Ethereum and a few other derivatives grew, the interest in implementing the underlying technology of blockchain, distributed application platform and distributed ledger to more innovative *enterprise* use cases eventually increased. However, many of these enterprise use cases need unique performance characteristics that permissionless blockchain technologies are presently unable to address. In many cases, identity becomes a hard requirement, as in financial transactions where KYC and Anti-money laundering regulations should be followed.

For the use of enterprises, specific requirements have to be considered:

- Participants have to be identified
- Networks need to get *permissioned*
- High transaction throughput performance
- Transaction confirmation with low latency
- Business transactions with privacy and confidentiality

Initial blockchain platforms are currently getting *adapted* in enterprise use. Hyperledger Fabric was *designed* for enterprise application from the outset. The below paragraph explains how Hyperledger fabric differentiates itself from the various blockchain platforms present in its architectural decisions.

Hyperledger Fabric means a platform for distributed ledger solutions with a modular architecture that delivers huge confidentiality, resiliency, flexibility, and scalability. It has been designed to promote pluggable implementations of various components and support the complexity and intricacies of the economic ecosystem. The Linux Foundation established the Hyperledger design in 2015 to develop cross-industry blockchain technology platforms. It promotes a collaborative approach in developing blockchain technologies through a community process by intellectual property rights which promote open development and vital standards over time. Just like other blockchain derivatives has a ledger which makes use of smart contracts and is a system through which all the transactions are managed by the participants.

The Hyperledger Fabric is different from other blockchain derivatives is because it is permissioned and private. Rather than an open, permissionless system which allows unknown identities to participate in the network, Hyperledger Fabric network members enroll by a trusted **Membership Service Provider (MSP)**.

It provides various pluggable options. Ledger data is stored in many formats. Consensus mechanisms allow you to swap in and out, supporting different MSPs

The Fabric further offers the facility to design **channels**, allowing participants to build separate ledger of transactions. It is a crucial option for networks where each participant becomes a competitor and does not want each transaction, they make a special price for some participant and not others, for example, known to each participant. If 2 participants form a channel, then the participants and no others got copies of the ledger for that particular channel.

Source: https://cdn-images-1.medium.com/max/770/1*EBdlbLK7xC1kxLen8GLMVA.png

In the current world of technical evolution. Medical supply chain management is a field that is constantly under radar due to increased failures in medical supply chain management due to improper exchange of information. To deal with this situation, we have provided a stable solution by harnessing the power of Hyperledger Fabric, which is an industrial solution for developing decentralized applications. Our solution addresses the problem by deploying all the transaction on a ledger. That brings transparency in the communication.

The Hyperledger fabric has the power to encrypt important medical records and also keep track of the drugs. More importantly it creates a sense of trust among the users as the data is safe and highly encrypted.

1.0.1. IMPORTANT DEFINITIONS

Ledger: It is the current state of the business as a journal of transaction. A ledger consists of two different parts, a **world state** and a **blockchain**. It is kept with all peers and also stored at a subset of the orderers. With the context of an orderer, we refer to the Ledger as an order ledger. In opposite, with the context of peer, it is referred to as a peer ledger. PeerLedger varies from an OrdererLedger where peers locally have a bitmask indicating valid transactions from invalid ones.

World State: Any chain code has its dedicated World state and blockchain. It is a database that holds the current ledger state. These states are expressed in terms of key-value pairs. They hold the fact of the business object. It can be created, update and delete. When the application submits the transaction, and when the point of committing the valid transaction arrives, it first gets committed in the World state and then updated in the ledger. The state has a version number. Whenever the state is updated, the version number changes. When a ledger gets created, the world state is empty. Any transaction representing a change in the world state gets recorded. This means a world state can be generated at any time from the blockchain. The peers always maintain the state.

Chain Code: The transaction logic that manages the object's lifecycle, which is included in the world state, is called a smart contract. It is deployed on the network by packaging into a chain code. The Smart contract is delineated in a chain code. We can define multiple contracts in a single chain code. When we deploy a chain code, the available smart contracts in the chain code are made obtainable to the application. *Smart contracts* are programs that domain-specific and are meant for specific business processes.

In contrast, chain codes are containers of the group of similar smart contracts for instantiation and installation. Each chain code has an attached endorsement policy that applies to all smart contracts which are under it. This recognises which organisation must approve a transaction produced by Smart contract in order to consider it valid. So, in short, every smart contract has an endorsement policy attached to it. So, to summarise, each smart contract is associated with an endorsement policy. It can call other contracts within a particular channel or across different channels.

1.0.2. ROLES THAT GOVERN THE HYPERLEDGER FABRIC:

- **Client**: They are applications which act on behalf of a person to submit transactions over the network. The client makes use of the Fabric SDK to communicate with the network. The client interacts with the SDK to Read/Write the data into a Fabric blockchain and an in-state database. The client is also issued a certificate to ensure that a valid client initiates a transaction over the network
- **Peer**: It is responsible for committing the transactions and maintaining the copy and state of the ledger. The ordering service sends ordered state updates to the peer in blocks and thereby maintains the state and the ledger. Thereby it can have a unique endorser role. The unique function of an *endorsing peer* happens concerning a distinct chain code and consists of *endorsing* a transaction before it has been committed.
- **Ordered/Ordering Service**: The orderer accepts the endorsed transactions, thereby ordering them into blocks and delivers them to the committing peers.

1.0.3. TYPES OF PEERS:

- **Endorsing Peer**: Endorsing peers are a particular sort of committing peers who have an added task in endorsing a given transaction. These peers endorse the transaction request that comes from the client. All endorsing peer owns the image of the smart contract installed and also have a ledger with it. The endorser stimulates the transaction. These transactions are executed basing on the smart contract over the personal copy of the ledger and generate the Read/Write sets and are sent to the clients. The transactions are not committed to the ledger while the transaction takes place.
- **Committing Peer:** Peers which commit the block received from the orderer in the copy of their blockchain. It contains the list of transactions where committing peer validate each transaction, mark it as either valid or invalid, and commit to the block. All transaction is committed to blockchain for audit purpose.

- **Anchor Peer:** As the Fabric network is extended across multiple organizations, we need peers to communicate across an organization. Now all peers cannot do this, but these peers are authorized to do so as they are special and are nothing but anchor peers. They are defined in the channel configuration.
- **Leading Peer:** Leader peers communicate or broadcast messages from the orderer to different peers in the same organization. The peers use the Gossip protocol to make sure that all peers receive the message. They cannot communicate across an organization. Assume a Leading peer is not responding or is out of network, we have the freedom to select a leading peer from the available lot by randomly choosing or by the process of voting

1.0.4. CHANNEL

There can be multiple channels associated with a fabric network. Due to this, organizations can make use of the same network separating multiple blockchains. Only peers can view the transaction generated by any other peer in a channel. In other words, the network is partitioned by the channel in order to provide visibility to stakeholders only. Only the channel members are involved in consensus.

On the contrary, other peers of the network cannot see the transactions over the channel. Multiple ledgers can be maintained by peers. Furthermore, a member can stay connected to various channels.

The configtx.yaml maintains the configuration of the channel. Using this file, we create a channel by generating a channel. Tx file. The channel is instantiated using a chain code, whereas the same chain code is installed on the participating peers. The communication between all the peers is made possible because of all configurations present in the channel. It holds the information like the list of peers and who are anchor, leader, endorsing peers. When the client uses SDK for communication, the SDK's first thing is getting a list of endorsing peers to which the transaction request must be sent. The SDK then transmits transaction requests to the peers. Peers participating in multiple channels go ahead and commit/simulate their transactions to various ledgers. Certain ordering services are also part of the channels.

1.1. RATIONALE

In a rapidly developing industrial world, the vital aspect of an organization is supply chain management. Being able to observe and follow the progress of commodities increases production and transparency in product manufacturing.

We have proposed a system specifically dedicated to tracking the progress of medicines from production to the point it reaches the customer. Technologies deployed in the development of this project involve Hyperledger-Fabric, Docker, Frameworks of JavaScript. This project is based on the platform of private blockchain Hyperledger Fabric, a project initiated by the Linux community in 2016 to provide business solutions using blockchain.

1.2. GOAL

The main goal of the project is to provide stable solution for transparent communication in the field of supply chain management in the field of pharmaceuticals. By harnessing the technology of private blockchain called Hyperledger fabric which provides industrial solutions to build decentralized applications.

1.3. OBJECTIVE

- Developing transparent communications between departments or users.
- Providing features for development of private channels.
- Tracking the progress of medicine production
- Allocating permissions to users over the network.
- No cost of transactions over the network.

1.4. METHODOLOGY

What Is Supply Chain Management (SCM)?

The administration of the movement of goods and services is referred to as supply chain management, and it encompasses all procedures that transform raw materials into finished items. It entails actively simplifying a company's supply-side processes in order to increase customer value and achieve a competitive advantage in the market.

SCM refers to providers' efforts to design and operate supply chains that are as efficient and cost-effective as possible. Supply chains encompass everything from manufacturing to product creation, as well as the information systems required to coordinate these activities.

What is the Process of Supply Chain Management?

Typically, SCM aims to centralize or link a product's manufacturing, shipment, and distribution. Companies can reduce extra expenses and deliver items to customers faster by optimizing the supply chain. Internal inventories, internal manufacturing, distribution, sales, and company vendor inventories are all under tighter supervision.

SCM is founded on the concept that practically every product that reaches the market is the result of the efforts of multiple businesses that make up a supply chain. Despite the fact that supply chains have been around for a long time, most businesses have only recently recognized them as a valuable addition to their operations.

Improvements in productivity and efficiency have a direct and long-term influence on a company's bottom line. Supply chain management keeps businesses out of the news and out of costly recalls and lawsuits.

Source: https://www.csss.es/benefits-supply-chain-management/

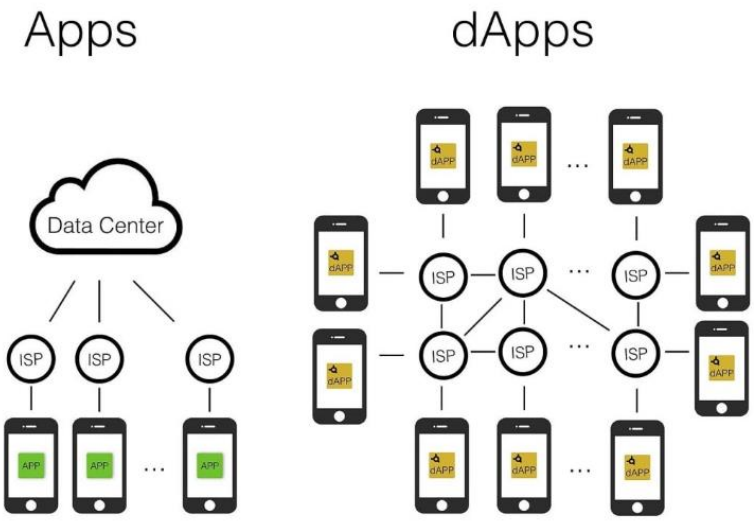
Decentralized Applications – dApps

Decentralized applications are software that interacts with the blockchain, which keeps track of the state of all network participants. Decentralized applications provide the same user interface as any other website or smartphone app today. A decentralized application's essential functionality is represented via a smart contract. Smart contracts are blockchain building blocks that process data from external sensors or events and assist the blockchain in managing the status of all network actors.

A decentralized application's frontend reflects what you see, while the backend represents all of the business logic. One or more smart contracts interact with the underlying blockchain to reflect this business logic. Decentralized storage networks like Swarm or IPFS may host the frontend as well as files like photos, videos, and audio. To render a webpage, traditional Web apps employ HTML, CSS, and javascript or something similar. This page communicates with a centralized database that holds all of the information.

A typical Web application is akin to a decentralized application. The frontend renders the page using the same technologies as the backend. It includes a "wallet" that connects to the blockchain. Cryptographic keys and the blockchain address are managed by the wallet. For user identification and authentication, a public-key infrastructure is employed. Instead of connecting to a database through an API, a wallet application activates the actions of a smart contract that communicates with a blockchain: Website that is Web3 compatible:

Front End → Smart Contract → Blockchain



Source: https://icoholder.com/blog/applying-red-box-dapp-in-the-ethereum-technology-platform/

Our approach to the system was using Hyperledger Fabric. First, all the departments in the medicinal plant were analyzed. So, at the core, we found

the following departments starting from the

- Pharmaceutical Research and Development plant
- Production department
- Testing Department
- Warehouse department
- Transportation
- Healthcare providers
- Retailers

All these departments are represented as nodes and are registered on the network. Depending upon the requirement, private channels can be developed inside each department for private communication. Once all nodes are up and running, each action performed by these nodes(department) is recorded on the ledger. We can understand this as pharmaceutical department as node once it completes its research trials a user on their behalf performs a transaction(action) which updates the ledger with the current status of the

medicine which has been shipped from research to production similarly when each department completes their work, they go on to update the ledger. This can be understood below with the block diagram**.**

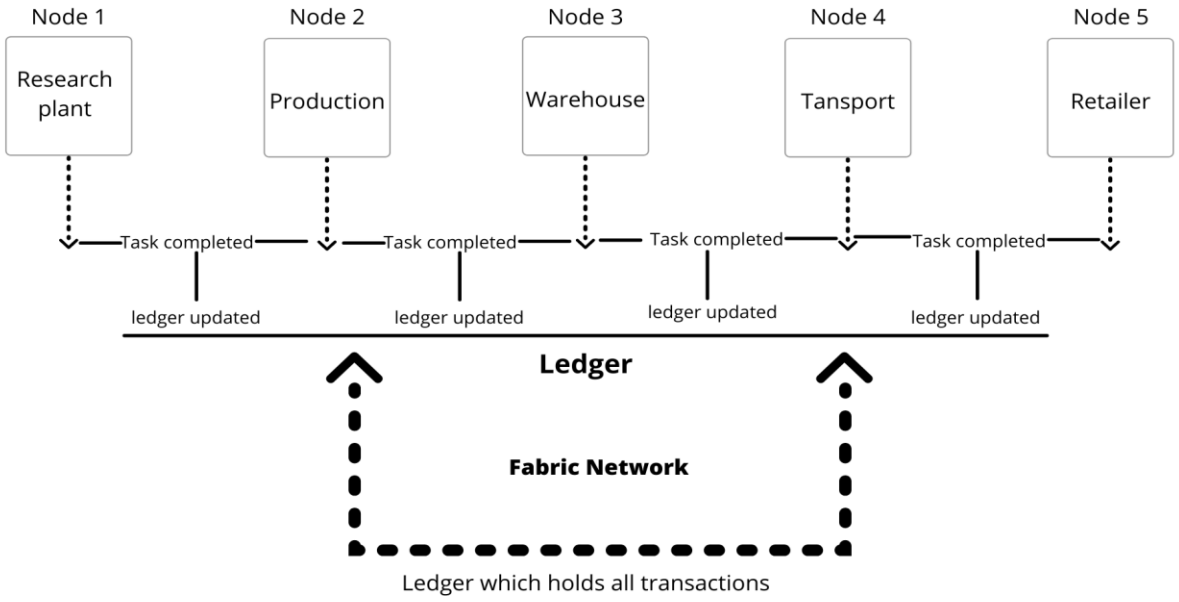


Figure 1. Methodology

1.5. PROCESS.

Each user is represented as a node over the network. As mentioned in the block diagram

Once a user performs a transaction it is committed to the ledger by the peer node. Once the peer node updates the ledger. The transaction is visible across the entire ledger, and all the nodes or users over the network can view it. Meanwhile, if users want to communicate privately and exchange information among specific people, this can be done by creating private channels. These channels provide a tremendous advantage for inter-department communication and maintain particular people's information privacy over the network.

With respect to our project when the Research plant which is also a node performs an action. The peer node commits those actions to the ledger, and the Production unit is informed via a notification from the ledger that work is completed at the research plant. Now it's the turn of the Production department to make the necessary arrangements for starting the production of medicine manufacturing. Once the Production department receives the required samples, they update the ledger with acknowledgement, and the current status of the ledger shows medicine in the production phase.

Inside the production department, if the user wishes to communicate with the testing team without getting the information updated on to the ledger. They can do this by creating a private channel adding the required people to it. So, a transaction committed in the private channel does not appear on the Global ledger of the network. The below diagram shows the working of the private channel.

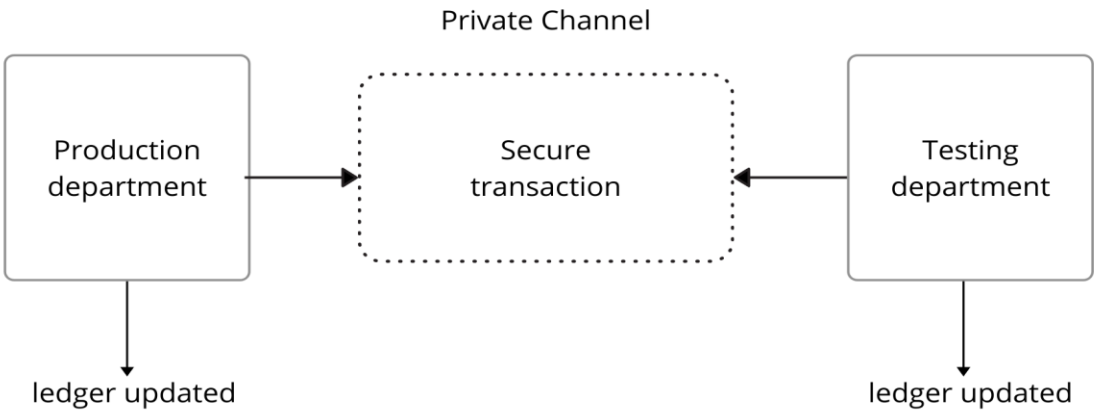


Figure 2. The ledger updating process

Similarly, transaction between other Depatments take place and these transactions are recorder on global ledger which is visible to all the authorized users.

1.6 ROLES AND RESPONSIBILITY.

ROLE	NAME	RESPONSIBILTY
Front end development	Abhishek Bhandwalkar	Worked on building a responsive and interactive user interface using Bootstrap4. Technologies used to develop UI are HTML, CSS, JavaScript, Bootstrap4
Network	Prashant Sharma	Worked on building the network, hosting ledger network and managing the insertion, deletion and updation in the network. Technologies used node.js, Hyperledger fabric, express.js

Back-end development	Golla Srinivas	Worked on connecting the Front end with hyperleger fabric network by creating APIs and managing POST, GET, PUT requests between Flask and Node.js. Technologies used Flask, node.js, jinja, express.js.
Testing	Shaik Asif	Worked on Testing the network with different corner cases
Testing	G. Bhanu Teja	Worked on testing the network with different corner cases.

1.7. MARKET POTENTIAL

By deploying this architecture model pharmaceuticals will be able to make their production process streamlined and will be able to reduce the cost due to lack in communication.

1.8. INNOVATION

The idea behind this project is to give the pharmaceuticals to give a clear and transparent view of the point of manufacturing and position of the shipment when it is out for delivery. It will give them the ability to track the distribution of medicine among the customers in case of counterfeiting the medicine.

1.9. USEFULNESS

The evident and prominent use case of the Hyperledger Fabric is the healthcare/pharmaceutical sector. Pharmaceuticals can use the fabric to create a blockchain network that has total control of the supply chain.

Each drug that leaves the manufacturing unit gets registered to the supply chain according to our writing standards. This information is shared with the shareholders and also with the end-users to track its movement. This approach removes counterfeit drugs from the chain. The solution that we provide can offer many benefits.

By using the DLT network, payments get faster instead of waiting for months/years together. The Hyperledger Fabric encrypts all the information which is stored in the records and also track the drugs. More importantly, it provides transparency which the users require to have faith and trust with the stakeholders.

Another critical use case of our project is that it provides end to end traceability. The pharma leads will have the ability to track the medicine in whichever state it might be. By simply typing in the medicine id on the front, we will see whether the medicine is in the production stage or testing stage or is out for delivery. Moreover, a simple scan of the barcode will tell us the pinpoint location of the medicine when it is en route for delivery. We would be able to deliver the drug to the hospitals quicker, thereby preventing drug wastage.

The supply chain leverages the power of the fabric for practical improvements in the efficiency of the process, adding to it auditable tracking and checking for malicious behavior. Paperwork is one of the reasons which takes up half of the transport cost. This can be avoided entirely by using fabric records, which will store all the information. Even mislabeling the dosage and other human errors can be avoided. The info can be retrieved by simple API calls, which returns a JSON response.

An article from PWC tells that 2 per cent of the world's economy comes from the sale of counterfeit drugs. Now our code can help avoid that by providing accountability, tradability, traceability and transparency. By better mapping and visualization, we can improve the operational efficiency of the manufacturing process.

It allows the customers to understand the process more effectively. It helps them engage with the brand and the process of manufacturing. Another significant aspect in which the Hyperledger fabric helps is tokenizing an asset into various shares representing ownership. The project helps in verifying the authenticity of data points, thereby ensuring real-time updates. It enables transparent verification of legal documents, certifications and other documents. Because of the transparency involved in the fabric, it allows efficient recalls. Making use of blockchain in various conventional manufacturing processes results in an increase in trade volumes.

CHAPTER - 2

2. REQUIREMENT ENGINEERING

2.1. FUCNTIONAL REQUIREMENTS

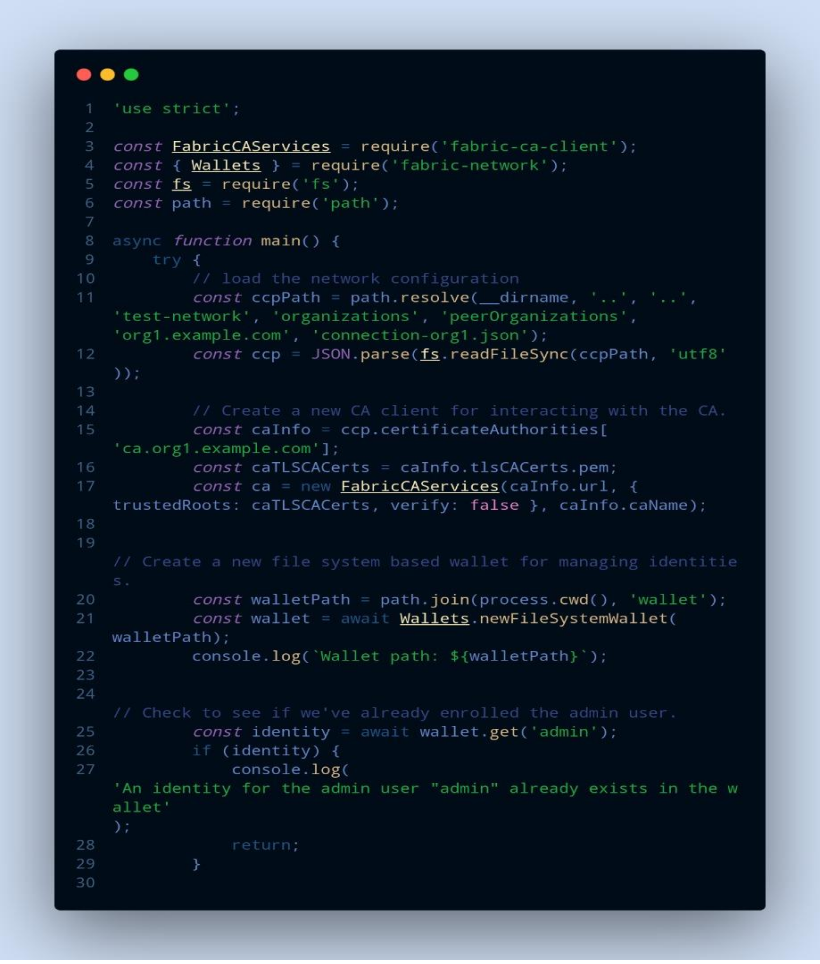
a. **Front End for User Interface:** Usually, for the front end, we use HTML; “HTML which is a markup language for all documents that are intended to be view in web browser. HTML elements are the components which makes HTML pages. Images and other objects, such as interactive forms, can be embedded in the produced page using HTML techniques. HTML allows you to

create organized documents by indicating structural semantics for text elements like headers, paragraphs, lists, links, quotations, and other elements. Tags, which are written in angle brackets, separate HTML elements. Tags like and introduce content to the page directly. It can be assisted by Cascading Style Sheets (CSS) and scripting languages such as JavaScript. CSS (Cascading Style Sheets) specifies how documents are displayed on computers, in print, or even in their spoken languages. Since its inception in 1994, the W3C has actively advocated using style sheets on the Internet. CSS (Cascading Style Sheets) are a simple and effective way to declare numerous properties for HTML tags. You can specify a variety of style properties for an HTML element using CSS. A colon separates the name and value of each property (:). A semi-colon separates each property declaration. A script is a short piece of software that may be used to make your website more interactive. A script could, for example, create a pop-up alert box message or a dropdown menu. This script could be written in JavaScript or Visual Basic for Applications (VBScript). Any programming language can create numerous little routines known as event handlers, which may then be triggered using HTML attributes. Most web developers nowadays utilize JavaScript and related frameworks solely; most significant browsers do not even support VBScript".

b. **Hyperledger Fabric: "** Hyperledger Fabric is designed to serve as a basis for building modular applications and solutions. Plug-and-play components, such as consensus and membership services, are possible with Hyperledger Fabric. Its modular and adaptable architecture caters to a wide range of industry applications. It takes a novel method to the consensus that allows for scalability while maintaining anonymity. Hyperledger Fabric uses the Ordering Service to accept authorized transactions, place blocks in a specific order, and send blocks to the network's committing peers. The blockchain network's state database and ledger are stored on NFS (Network File System). Before each request is completed, the fabric-ca client verifies a peer's identity and approves the transaction request."

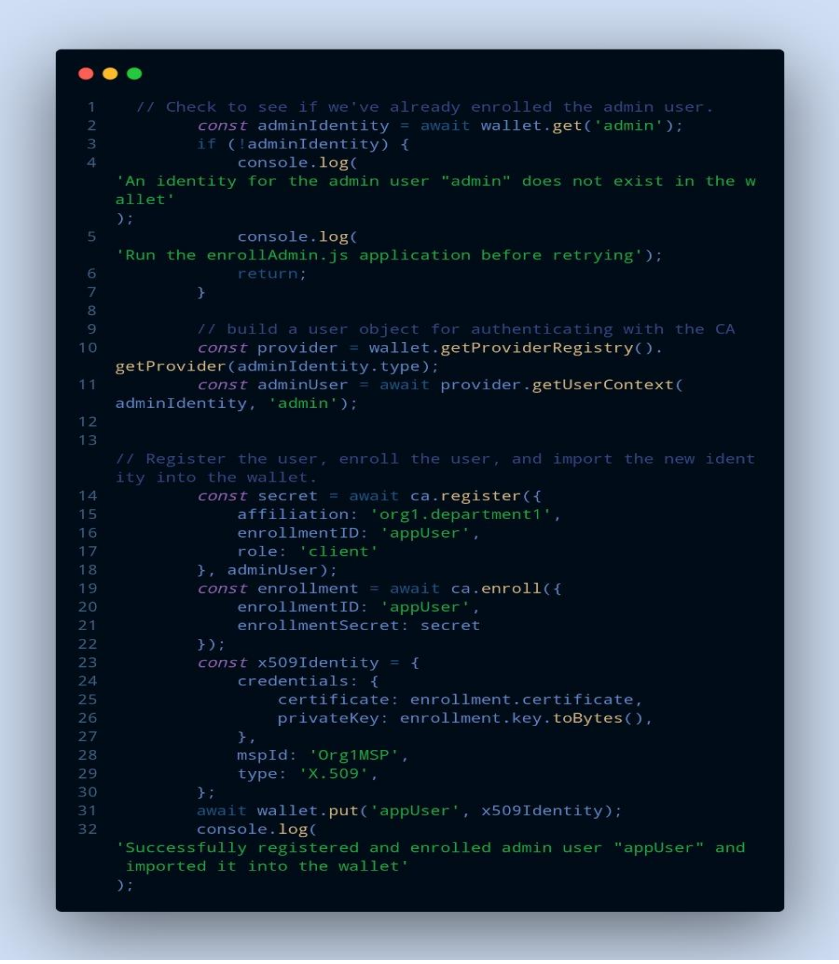
```
// Enroll the admin user, and import the new identity into the wallet.
        const enrollment = await ca.enroll({ enrollmentID: 'admin', enrollmentSecret: 'adminpw' });
        const x509Identity = {
            credentials: {
                certificate: enrollment.certificate,
                privateKey: enrollment.key.toBytes(),
            },
            mspId: 'Org1MSP',
            type: 'X.509',
        };
        await wallet.put('admin', x509Identity);
        console.log(
'Successfully enrolled admin user "admin" and imported it into the wallet'
);

    } catch (error) {
        console.error(`Failed to enroll admin user "admin": ${error}`);
        process.exit(1);
    }
}

main();
```

Figure 3. Enrolling the admin user.



```
'use strict';

const FabricCAServices = require('fabric-ca-client');
const { Wallets } = require('fabric-network');
const fs = require('fs');
const path = require('path');

async function main() {
    try {
        // load the network configuration
        const ccpPath = path.resolve(__dirname, '..', '..', 'test-network', 'organizations', 'peerOrganizations', 'org1.example.com', 'connection-org1.json');
        const ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'));

        // Create a new CA client for interacting with the CA.
        const caInfo = ccp.certificateAuthorities['ca.org1.example.com'];
        const caTLSCACerts = caInfo.tlsCACerts.pem;
        const ca = new FabricCAServices(caInfo.url, { trustedRoots: caTLSCACerts, verify: false }, caInfo.caName);


// Create a new file system based wallet for managing identities.
        const walletPath = path.join(process.cwd(), 'wallet');
        const wallet = await Wallets.newFileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);


// Check to see if we've already enrolled the admin user.
        const identity = await wallet.get('admin');
        if (identity) {
            console.log('An identity for the admin user "admin" already exists in the wallet');
            return;
        }

```

Figure 4. Enrolling admin extension



```
// Check to see if we've already enrolled the admin user.
        const adminIdentity = await wallet.get('admin');
        if (!adminIdentity) {
            console.log(
'An identity for the admin user "admin" does not exist in the w
allet'
);
            console.log(
'Run the enrollAdmin.js application before retrying');
            return;
        }

        // build a user object for authenticating with the CA
        const provider = wallet.getProviderRegistry().
getProvider(adminIdentity.type);
        const adminUser = await provider.getUserContext(
adminIdentity, 'admin');


// Register the user, enroll the user, and import the new ident
ity into the wallet.
        const secret = await ca.register({
            affiliation: 'org1.department1',
            enrollmentID: 'appUser',
            role: 'client'
        }, adminUser);
        const enrollment = await ca.enroll({
            enrollmentID: 'appUser',
            enrollmentSecret: secret
        });
        const x509Identity = {
            credentials: {
                certificate: enrollment.certificate,
                privateKey: enrollment.key.toBytes(),
            },
            mspId: 'Org1MSP',
            type: 'X.509',
        };
        await wallet.put('appUser', x509Identity);
        console.log(
'Successfully registered and enrolled admin user "appUser" and
 imported it into the wallet'
);
```

Figure 5. User registration over the network.

c. **API to connect front end with Hyperledger fabric:** Hyperledger Fabric is written in node.js. So, we have used express.js to create APIs. " *Express* is a Node.js web application framework that offers a comprehensive collection of functionalities for building web and mobile applications. It makes it easier to create Node-based Web apps quickly. Some of the core features of the Express framework are listed below. Allows middleware to reply to HTTP requests to be put up. Defines a routing table for doing various actions based on HTTP Method and URL. Allows you to render HTML pages dynamically by supplying variables to templates." Our front end uses Flask to create APIs, which is written in Python. "Flask is a Python-based microweb framework. It is referred to as a microframework because it does not necessitate using any specific tools or libraries. It does not have a database abstraction layer, form validation, or other components that rely on third-party libraries to do typical tasks. On the other hand, extensions can be used to add application

functionalities as if they were built into Flask itself. Object-relational mappers, form validation, upload handling, different open authentication protocols, and other framework-related tools all have extensions". Express.js sends data to a URL, and Flask request data from the URL and these requested data is then displayed on the screen. To make HTML more interactive with Flask - jinja, a web template engine for Python. "Jinja is a templating engine that is quick, expressive, and extendable. The template has special placeholders that allow you to write code that looks like Python syntax". After that, data is supplied to the template in order to render the final document.

2.1.1 INTERFACE REQUIREMENT

1. Screen-1: Home Page: Our home page has one input field, and a submit button. An input field takes the ID of the medicine required to display its details. If the user provides the ID and the submit button is clicked, then a POST request is sent, and the response is displayed on the screen below the input field. At the top left of the home screen, there is a button to navigate to the admin page where insertion, updating, and display all the medicines present take place.

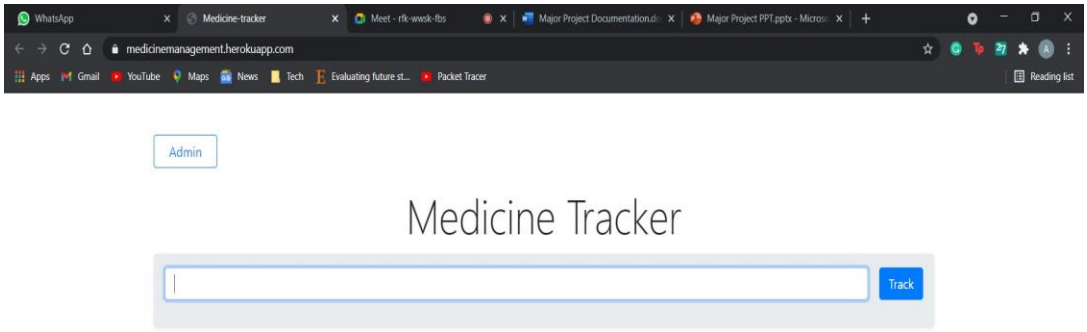


Figure 6. Screen1(Input ID)

2. Screen-2: Admin Page: Admin page has three buttons used to insert a new medicine, update the status of the given medicine, and display all the medicine present until now. When the admin clicks on the add medicine button, a form is displayed below the button where details of the new medicine that is to be added are filled. This form has five input fields. The first field takes the ID of the medicine to be added, and the second field takes the name of the medicine, the third field takes the price of the medicine, the fourth field takes state, and the fifth field takes the current status of the medicine. After all the details are filled and the admin clicks on the submit button, then a POST request is sent where the medicine details are added to the network, and a response is received after successfully added. While the user clicks on the change status button, a form is displayed below the button where details of the ID of medicine and the current status of medicine are filled. This form contains two fields. The first field takes the ID of the medicine whose status is to be changed, and the second field takes the status to be updated. After all the requirements are provided, a POST request is sent where the medicine status is updated in the network, and a response is received after successfully updated. When the admin clicked on display all medicine, the GET request is sent, and the response contains all the details in JSON format. After parsing the JSON data, this data is made to display on the screen. At the top left of the admin window, there is a button to navigate to the home page. When the admin wants to check the details of a particular medicine, he can click on the home page button.

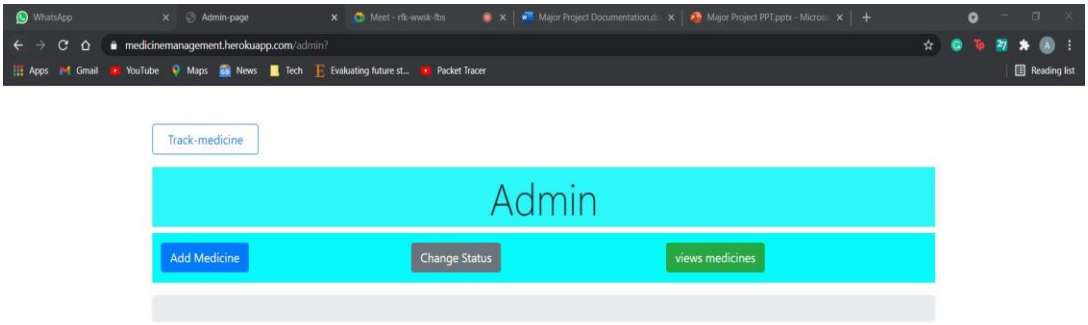


Figure 7. Admin Portal

2.2. NON-FUNCTIONAL REQUIREMENTS

The Non-Functional requirement describes a software system's quality attribute. “They assess the software system based on its responsiveness, usability, security, portability, and other non-functional criteria crucial to its success.”

"How quickly does the website load?" is an example of a non-functional demand. “Non-functional requirements that are not met can lead to systems that do not meet user needs. Non-functional Requirements allow you to place constraints or limitations on the system's architecture across several agile backlogs”. For example, when there are more than 10000 simultaneous users, the site should load in 3 seconds.

“The following is a list of non-functional resemblances. Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement”

2.2.1 EXAMPLES OF NON-FUNCTIONAL REQUIREMENTS

Some instances of non-functional requirements are as follows:

- “Can handle any inputs”.
- “The software must be portable. As a result, switching from one OS to another is not an issue”.
- “Information privacy, the export of restricted technologies, intellectual property rights, and other issues should all be investigated.”

2.2.2 ADVANTAGES OF NONFUNCTIONAL REQUIREMENT:

- “The nonfunctional requirements ensure that the software system complies with all applicable laws and regulations.”
- “They ensure the software system's dependability, availability, and performance”.
- “They ensure a positive user experience and software that is simple to use.”
- “They aid in the formulation of the software system's security policy.”

2.2.3 DISADVANTAGES OF NON-FUNCTIONAL REQUIREMENT:

- “The many high-level software sub-systems may be affected by non-functional requirements.
- They necessitate more thought during the software architecture/high-level design phase, which raises expenses.
- The majority of the time, their implementation does not correspond to the specific software sub-system.
- Once you have passed the architecture phase, it is challenging to change non-functional elements.”

CHAPTER – 3

3.ANALYSIS AND DESIGN

3.1.USE CASE DIAGRAM

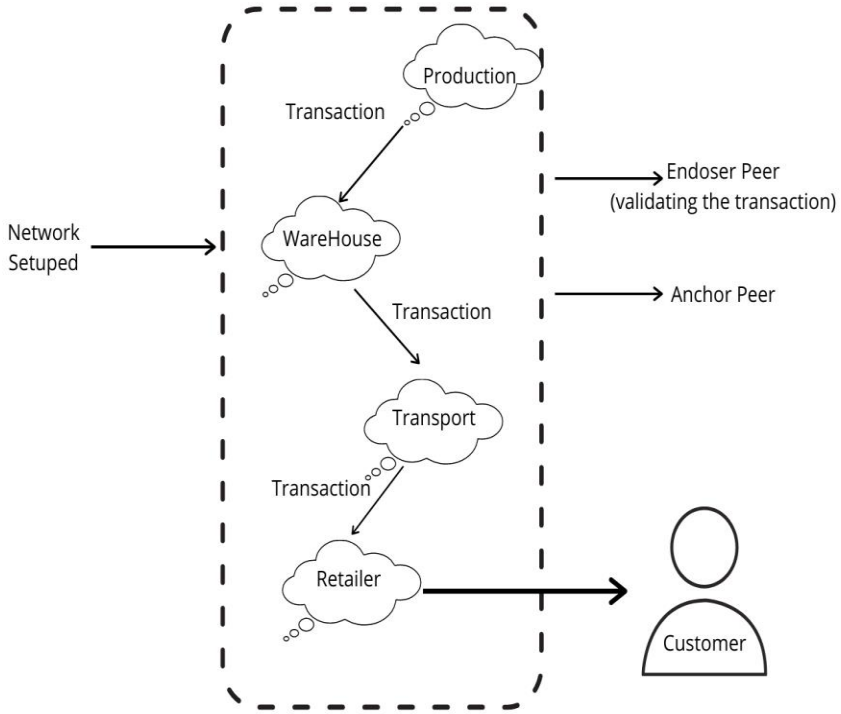


Figure 8. Use Case Diagram

The use case flow of the project is pretty simple. Initially, every department has its ledger associated with it; thus, there will be admins for every department who have access to the admin portal of the front end. Once the network is set up and ready to go, transactions can occur between all the departments. So, a medicine or a drug enters a production stage, it has its ID associated, and a hash is applied to all the other information so that the transaction is secure.

When the drug production is completed, it goes to the testing stage, where another hash is applied to secure the info, and apart from the universal drug id, it is also given a departmental id as well for internal purposes. The endorser peer comes into the picture; these are responsible for handling validating the transactions. If the transaction is valid, the medicine moves on to the next phase. If it is not valid, it simply goes back to the previous, and the validation starts again.

Now when the drug is in the warehouse and moves to the shipment stage, we will have the possibility to check the location of the drug when it is En route to the retailers. This is a beneficial feature as it avoids the counterfeiting of the drugs and is misplaced.

3.2. ACTIVITY DIAGRAM

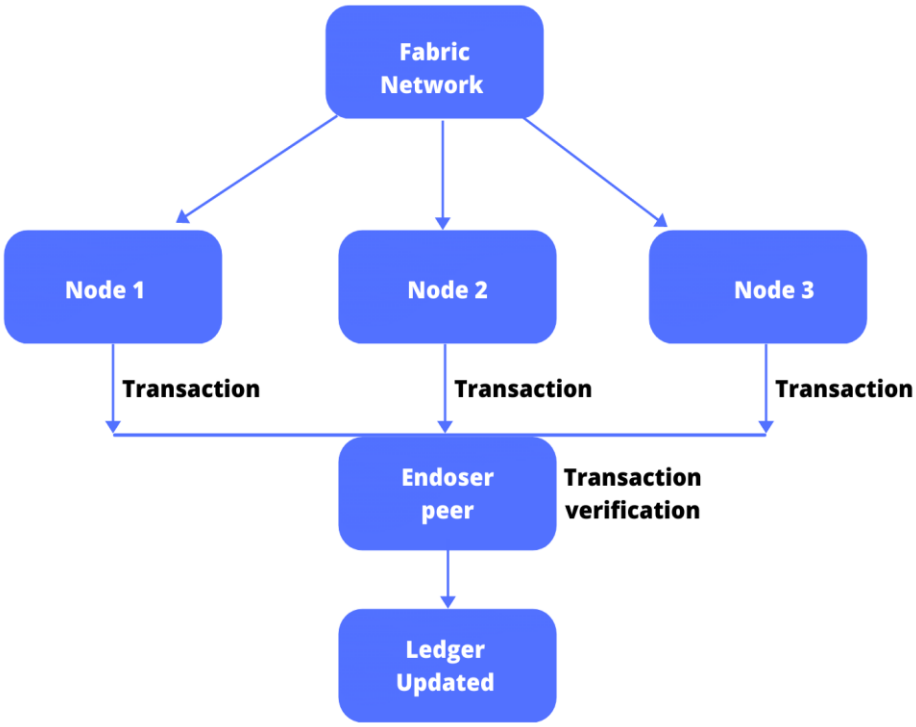


Figure 9. Activity Diagram

The class diagram of the Supply chain management application deals with all the classes present in the application structure.

Fabric network: this provides a base for all the nodes and operations which occur over the ledger.

All these transactions are recorded in the world state. Other important classes of this platform are

Nodes: these are the basic units over the network which admins use to perform transactions.

Endorser peer: This peer is used to commit and validate the transactions and add them to the ledger.

Ledger: It is a copy of all the transactions performed over the network and is present with all the admins in the network.

3.3. SYSTEM ARCHITECTURE

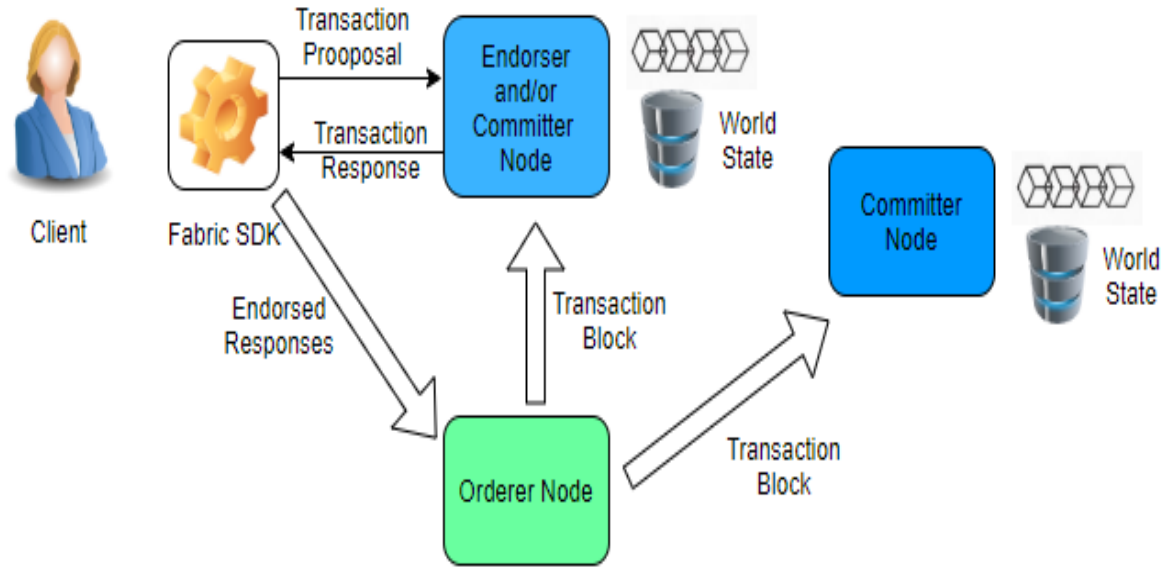


Figure 10. Architecture

Supply chain management pharmaceutical application Architecture

The architecture of this application includes a client which interacts with the fabric network through fabric SDK, which in turn holds order nodes that perform transactions, and a committer node called endorser node, which is used to commit the transaction onto the network, and which in turn updates the world state.

World state holds the proof of all the transactions which are done on the network. At each level, the authorized admin who has been provided permission to the network performs transactions. Manufactured medicines can be track down medicines from production to distribution.

CHAPTER – 4

4.IMPLEMENTATION

4.1 IMPLEMENTATION OF THE CODE

The implementation of the project includes developing a network deploying the network. Then we have developed the chaincode for our project. Chaincode is similar to smart contract in blockchain. It includes a set of all the operations which the user can perform over the network.

List of operation which we have defined in are implementation is

- Adding new Medicine
- Checking the status of medicine
- Checking all the medicine on the network.
- Change status

Once chaincode is ready we added it to the network. This chaincode decide all the actions which the application can perform. Since Hyperledger fabric is a open-source project it does not cost to execute actions in chaincode. This feature makes this industrial blockchain so unique.

```
class Medicine extends Contract {

    async initLedger(ctx) {
        console.info(
'============= START : Initialize Ledger ==========='); 
        const medi = [
            {
                ID: 'MD01',
                make: 'BharatBiotech',
                model: 'Prius',
                owner: 'Tomoko',
            },
            {
                ID: 'MD02',
                make: 'serium',
                model: 'Mustang',
                owner: 'Brad',
            },
            {
                 ID: 'MD01',
                make: 'BharatBiotech',
                model: 'Prius',
                owner: 'Tomoko'
            },
            {
                ID: 'MD01',
                make: 'BharatBiotech',
                model: 'Prius',
                owner: 'Tomoko'
            },
            {
               ID: 'MD01',
                make: 'BharatBiotech',
                model: 'Prius',
                owner: 'Tomoko'
            },
```

Figure 11. Chaincode implementation snippet

```
async queryAllmedi(ctx) {
        const startKey = '';
        const endKey = '';
        const allResults = [];
        for await (const {key, value} of ctx.stub.
getStateByRange(startKey, endKey)) {
            const strValue = Buffer.from(value).toString('utf8'
);
            let record;
            try {
                record = JSON.parse(strValue);
            } catch (err) {
                console.log(err);
                record = strValue;
            }
            allResults.push({ Key: key, Record: record });
        }
        console.info(allResults);
        return JSON.stringify(allResults);
    }

    async changeMedicineOwner(ctx, carNumber, newOwner) {
        console.info(
'============= START : changeCarOwner ===========');

        const carAsBytes = await ctx.stub.getState(carNumber);
// get the car from chaincode state
        if (!carAsBytes || carAsBytes.length === 0) {
            throw new Error(`${carNumber} does not exist`);
        }
        const car = JSON.parse(carAsBytes.toString());
        car.owner = newOwner;

```

Figure 12. Chaincode Implementation second Snippet

2ND PART OF PROJECT INCLUDES API DEVELOPMENT

We have used the Express framework to develop Api for the network. Express is a framework of NodeJS

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following is some of the core features of Express framework −

- Allows to set up middleware's to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

Additionally, Express provides support for higher number of request support in one section that means user can perform multiple APi request using the front end and he will not face network issue while loading of data. Which makes user experience smooth.

Below images show detail description for APi development in Express.

```
ar express = require('express');
var bodyParser = require('body-parser');
var app = express();
app.use(bodyParser.json());
// Setting for Hyperledger Fabric
const { Gateway,Wallets } = require('fabric-network');
const path = require('path');
const fs = require('fs');
//const ccpPath = path.resolve(__dirname, '..', '..', 'test-network', 'organizations', 'peerOrganizations', 'org1.example.com', 'connection-org1.json');

//      const ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'));


app.get('/api/queryallmedi', async function (req, res)  {
    try {
const ccpPath = path.resolve(__dirname, '..', '..', 'test-network', 'organizations', 'peerOrganizations', 'org1.example.com', 'connection-org1.json');
        const ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'));
// Create a new file system based wallet for managing identities.
        const walletPath = path.join(process.cwd(), 'wallet');
        const wallet = await Wallets.newFileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);

        // Check to see if we've already enrolled the user.
        const identity = await wallet.get('appUser');
        if (!identity) {
            console.log('An identity for the user "appUser" does not exist in the wallet');
            console.log('Run the registerUser.js application before retrying');
            return;
        }
```

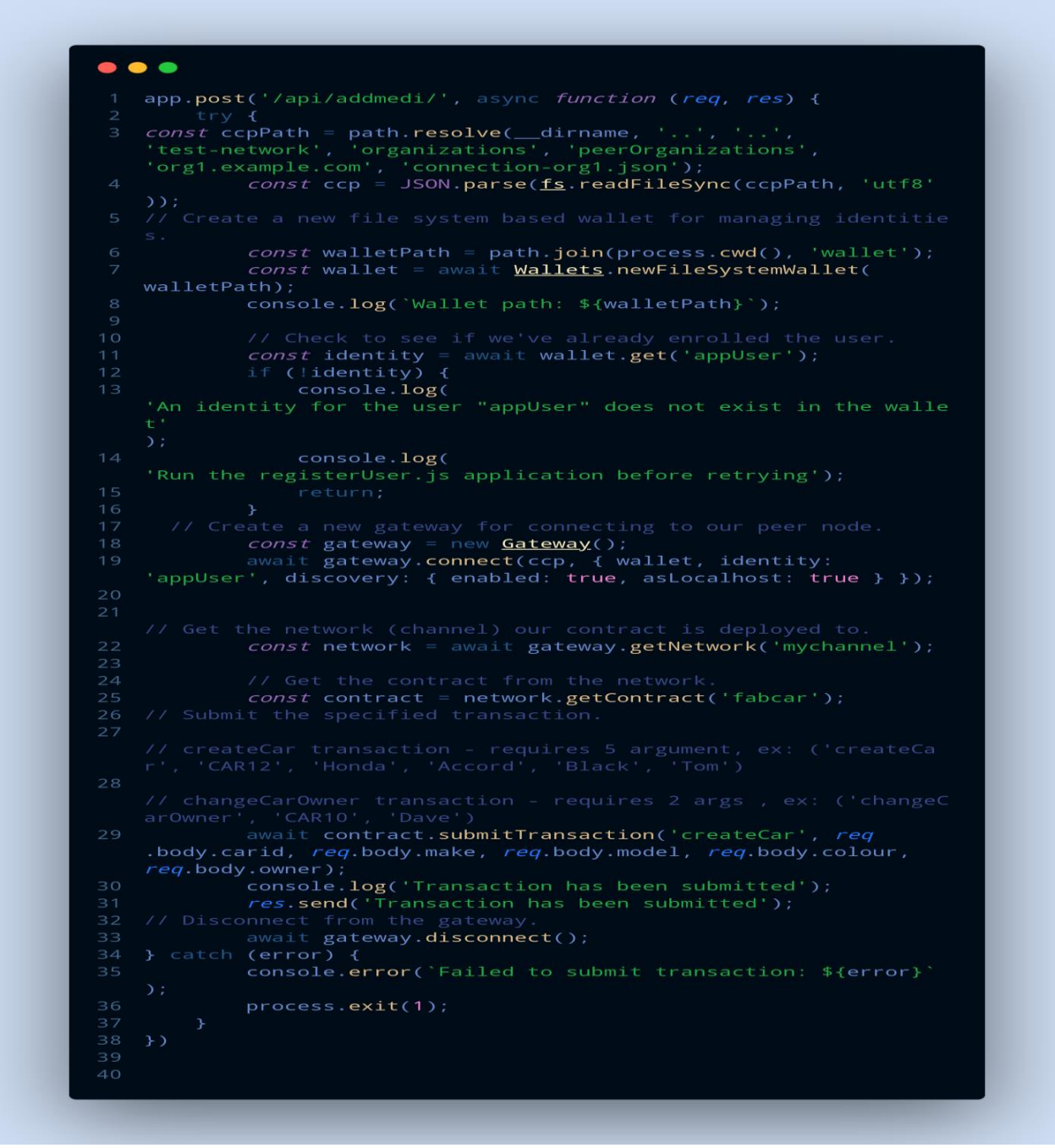
Figure 13. Api creation using express framework

```
app.get('/api/query/:car_index', async function (req, res) {
    try {
const ccpPath = path.resolve(__dirname, '..', '..',
'test-network', 'organizations', 'peerOrganizations',
'org1.example.com', 'connection-org1.json');
        const ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'
));
// Create a new file system based wallet for managing identitie
s.
        const walletPath = path.join(process.cwd(), 'wallet');
        const wallet = await Wallets.newFileSystemWallet(
walletPath);
        console.log(`Wallet path: ${walletPath}`);

        // Check to see if we've already enrolled the user.
        const identity = await wallet.get('appUser');
        if (!identity) {
            console.log(
'An identity for the user "appUser" does not exist in the walle
t'
);
            console.log(
'Run the registerUser.js application before retrying');
            return;
        }
  // Create a new gateway for connecting to our peer node.
        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity:
'appUser', discovery: { enabled: true, asLocalhost: true } });


// Get the network (channel) our contract is deployed to.
        const network = await gateway.getNetwork('mychannel');

        // Get the contract from the network.
        const contract = network.getContract('fabcar');
// Evaluate the specified transaction.

// queryCar transaction - requires 1 argument, ex: ('queryCar',
'CAR4')

// queryAllCars transaction - requires no arguments, ex: ('quer
yAllCars')
        const result = await contract.evaluateTransaction(
'queryCar', req.params.car_index);
        console.log(
`Transaction has been evaluated, result is: ${result.toString()
}`);
        res.status(200).json({response: result.toString()});
} catch (error) {
        console.error(`Failed to evaluate transaction: ${error
}`);
        res.status(500).json({error: error});
        process.exit(1);
    }
});
```

Figure 14. Get request Api implementation in express

Get APi is used to retrieve information from the network over to the front end. We are using get request to check all the medicine which are present in the network and to check their status.

Get request are also made to search medicines depending upon there Id so that process of searching becomes easier.



```
app.post('/api/addmedi/', async function (req, res) {
    try {
const ccpPath = path.resolve(__dirname, '..', '..',
'test-network', 'organizations', 'peerOrganizations',
'org1.example.com', 'connection-org1.json');
        const ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'
));
// Create a new file system based wallet for managing identitie
s.
        const walletPath = path.join(process.cwd(), 'wallet');
        const wallet = await Wallets.newFileSystemWallet(
walletPath);
        console.log(`Wallet path: ${walletPath}`);

        // Check to see if we've already enrolled the user.
        const identity = await wallet.get('appUser');
        if (!identity) {
            console.log(
'An identity for the user "appUser" does not exist in the walle
t'
);
            console.log(
'Run the registerUser.js application before retrying');
            return;
        }
  // Create a new gateway for connecting to our peer node.
        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity:
'appUser', discovery: { enabled: true, asLocalhost: true } });


// Get the network (channel) our contract is deployed to.
        const network = await gateway.getNetwork('mychannel');

        // Get the contract from the network.
        const contract = network.getContract('fabcar');
// Submit the specified transaction.

// createCar transaction - requires 5 argument, ex: ('createCa
r', 'CAR12', 'Honda', 'Accord', 'Black', 'Tom')

// changeCarOwner transaction - requires 2 args , ex: ('changeC
arOwner', 'CAR10', 'Dave')
        await contract.submitTransaction('createCar', req
.body.carid, req.body.make, req.body.model, req.body.colour,
req.body.owner);
        console.log('Transaction has been submitted');
        res.send('Transaction has been submitted');
// Disconnect from the gateway.
        await gateway.disconnect();
} catch (error) {
        console.error(`Failed to submit transaction: ${error}`
);
        process.exit(1);
    }
})
```

Figure 15. Post Api implementation

Post API is used to send data over the network we are using post API here to add new medicine to the network. We have connected one endpoint of Api to front from where the request is made and second end point of the API to the network. Where new entries are added.

4.1. SOFTWARE DETAILS

- HTML – Front end
- CSS – Front end
- Postman
- Visual Studio Code
- CouchDB
- ExpressJS
- Flask
- JavaScript
- Hyperledger Fabric
- Docker Engine: Version 17.03 or higher
- Docker-Compose: Version 1.8 or higher
- Node: 8.9 or higher (note version 9 is not supported)
- npm: v5.x
- git: 2.9.x or higher
- Python: 2.7.x
- Go version 1.12.x is required.

```
{
    "name": "fabcar",
    "version": "1.0.0",
    "description":
"FabCar application implemented in JavaScript",
    "engines": {
        "node": ">=8",
        "npm": ">=5"
    },
    "scripts": {
        "lint": "eslint .",
        "pretest": "npm run lint",
        "test": "nyc mocha --recursive"
    },
    "engineStrict": true,
    "author": "Hyperledger",
    "license": "Apache-2.0",
    "dependencies": {
        "body-parser": "^1.19.0",
        "express": "^4.17.1",
        "fabric-ca-client": "^2.2.4",
        "fabric-network": "^2.2.4",
        "nodemon": "^2.0.7"
    },
```

Figure 16. Software requirements image 1

```
  "devDependencies": {
        "chai": "^4.2.0",
        "eslint": "^5.9.0",
        "mocha": "^5.2.0",
        "nyc": "^14.1.1",
        "sinon": "^7.1.1",
        "sinon-chai": "^3.3.0"
    },
    "nyc": {
        "exclude": [
            "coverage/**",
            "test/**"
        ],
        "reporter": [
            "text-summary",
            "html"
        ],
        "all": true,
        "check-coverage": true,
        "statements": 100,
        "branches": 100,
        "functions": 100,
        "lines": 100
    }
}
```

Figure 17. Software Requirements image 2

4.2. HARDWARE DETAILS

- Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12 or Windows 7 or higher
- RAM: 4GB or higher
- CPU: Intel Core i3-9100F 9th Gen Desktop Processor 4 Core Up to 4.2 GHz LGA1151 300 Series 65W or higher/ AMD Ryzen 5 2600 Desktop Processor 6 Cores up to 3.9GHz 19MB Cache AM4 Socket or higher
- System Architecture: 64-bit x86, 32-bit x86 with windows or Linux
- Graphics: GTX 980 or 980Ms or higher
- SSD: M.2 PCIe or regular PCIe SSD with at least 256GB of storage, though 512GB is best for performance.

4.3. TESTING

USING POSTMAN TO TEST API'S:

Postman is a collaboration platform for API development. Postman's features simplify each step of building an API and streamline collaboration so you can create better APIs—faster.

Below we are sending a Post request using Postman to check if Our API endpoint works fine.

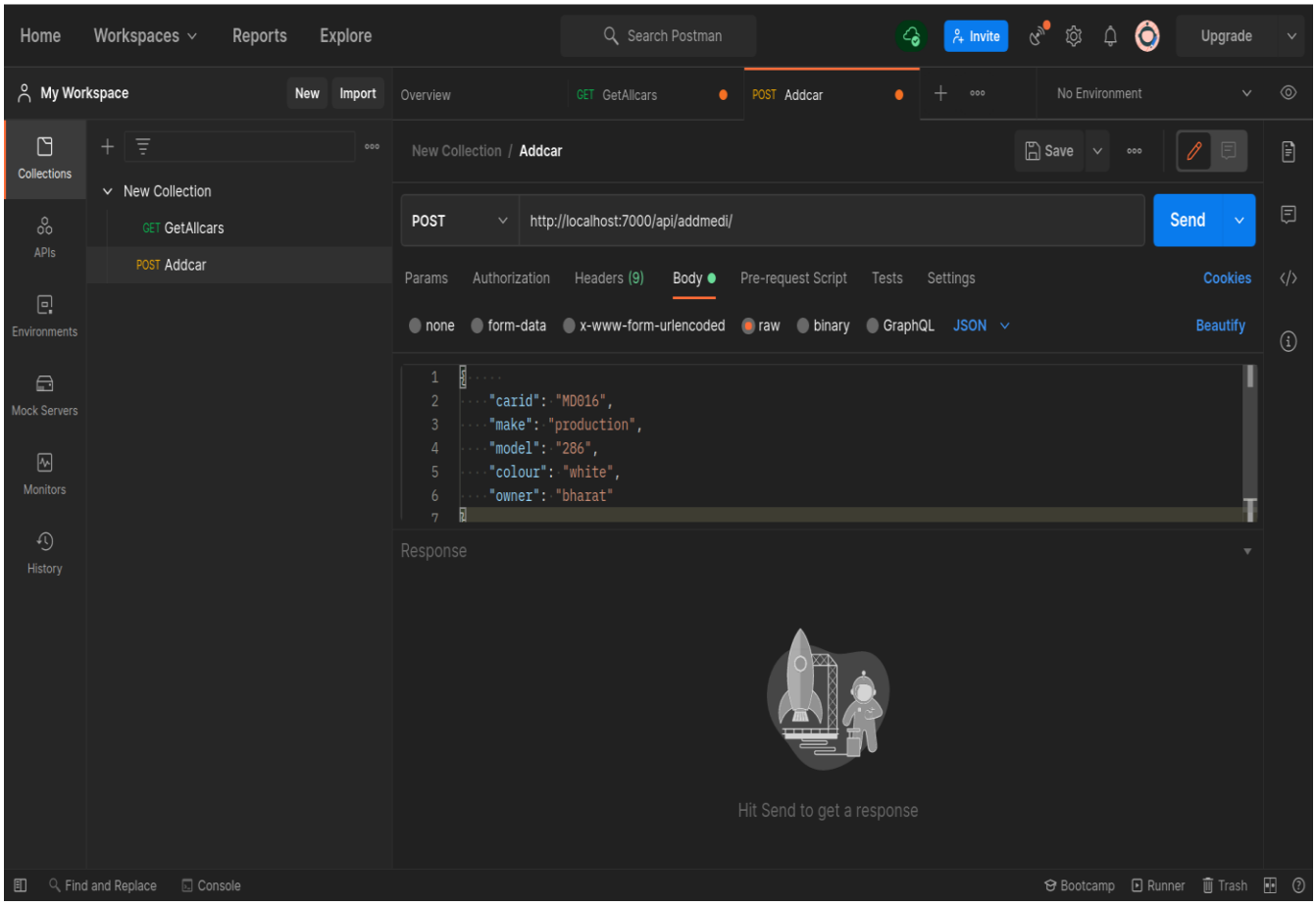


Figure 18. Testing the Api's through the postman application

CHAPTER – 5

CONCLUSION AND FUTIRE SCOPE

5.1. CONCLUSION

Hyperledger Fabric is the most dynamic of the Hyperledger projects. The community is growing steadily and the innovation produced with each subsequent release out-paces all other enterprise blockchain platforms.

By harnessing the capabilities of Hyperledger, we are developing a decentralized application that aims to set up clear communication in all phases of the production of medicines. So that medicine containers can be tracked in case of false deliveries. A Hyperledger Fabric acts as a peer-to-peer network. Complete transparency is maintained as the information shared among all the participants is immutable. One of the most significant benefits of using the Hyperledger fabric in the healthcare supply chain is the authenticity it gives the stakeholders to assess the transactions which are present in the ledger. In this project, we examined Blockchain applications and the benefits they provide to healthcare supply machine management.

It is accurate and transparent to the core, which effectively saves time, cost and effort, sparing the hassle of ongoing management. The technology magnifies the collaboration among participants and the users with the help of the distributed ledger technology. They can come together and provide deep insights, and do research. One of the critical issues that the pharma and the medical industry is facing is the leakage of crucial data used for malicious means. Another means of info that more often is critical is the accurate drug data which contains information about its dosage, the vital raw materials used in its manufacturing and the most important of all, the formula for developing it. This information is easily accessible to people having access to the database. To overcome all these problems in the project, the fabric uses advanced cryptographic aspects using hashing, accompanied by a digital signature and foolproof data security stabilizing trust issues and stability.

Streamlined, quality, smooth data sharing and distribution across the crucial network participants and healthcare providers help us create cost-effective and sophisticated treatments and cures for various diseases. This solution processes the documents in a step-by-step manner exclusively in the digital format. This increases the security and eases the process of billing and other procedures, removing the need for multiple intermediaries and validations that act on behalf of other entities.

5.2. FUTURE SCOPE

In the future, we would like to extend this project in many possible ways. The first one is using Amazons' AWS or Microsoft Azure as a database instead of putting the data from the transactions in the local server or simple database engines like CouchDB, MySQL or Oracle. Now, using these services would bring a

lot more scalability and a lot more flexibility in using the ledger. The concept of pay as you go pricing comes into the picture, and we can have flexible plans. We can just set up the database and not even worry about it, and we will be making use of the power of cloud computing as these services are run on the cloud. As a result, we do not even have to worry about hosting the server on our local and deploy it wherever we want across the globe.

The other enhancement or experiment that we would like to make is using Kubernetes as the microservice engine instead of docker. Both of them scale at the same level, but the service organisation takes place as pods. As a result, it will be easy to containerize the application and deploy the image of the same on the cloud making the project much more robust and dynamic. Moreover, AWS and Kubernetes provide a great combination of services.

We have implemented the front end using simple web development resources like HTML, CSS and ExpressJS for the API engine. We want to change this implementation by using Django for implementing the front end, which would give us the flexibility to use automation tools as it is written in python. By using the Django rest framework, we can implement the entire API server. As a result, we can have a single code base in Django, which would take care of the front end and the API calls necessary for the GET and POST requests.

Also, we want to make use of much stronger cryptographic algorithms in order to encrypt the data which is exchanged during the transactions instead of making use of the inbuilt blockchain hashing mechanism. This would increase the security prospect of the applications.

By making use of machine learning algorithms, we want to develop a codebase that would help in tracking the location of the medicine while it is out for transport with pin point accuracy.

6. REFERENCES

[1] https://hyperledger-fabric.readthedocs.io/en/release-2.2/blockchain.html

[2]https://101blockchains.com/hyperledger-use-cases/#:~:text=One%20of%20the%20most%20obvious%20and%20prominent%20use,can%20take%20full%20control%20of%20the%20supply%20chain.

[3]https://medium.com/swlh/hyperledger-chapter-6-hyperledger-fabric-components-technical-context-767985f605dd#:~:text=Roles%20within%20a%20Hyperledger%20Fabric%20Network%201%20Clients,and%20delivers%20the%20blocks%20to%20the%20committing%20peers.

[4]https://www.cilans.net/developed-pharma-supply-chain-with-amazon-qldb-migration-from-hyperledger/

[5] https://docs.docker.com/

[6] https://learning.postman.com/docs/getting-started/introduction/

[7] https://flask.palletsprojects.com/en/2.0.x/

[8] https://jinja.palletsprojects.com/en/3.0.x/

[9] https://expressjs.com/en/5x/api.html

[10]https://theblockbox.io/blog/blockchain-technology-in-healthcare-in-2021/#:~:text=What%20is%20Blockchain%20in%20Healthcare,secure%20interoperability%20between%20healthcare%20organizations.