CSE 598 Project 1 Report

Title: Hyperledger Fabric Private Blockchain and Smart Contracts

Student Name: Harshith Chittajallu

ASU ID: 1218707243

Abstract:
In this project, the primary goal was to complete the codebase in JavaScript for setting up the Hyperledger Fabric blockchain framework and deploy the smart contracts that manage patient record assets on the Hyperledger framework. The code had to complemented with multiple functions which create, update and allow reading/ accessing patient records. This was implemented in a step by step manner consisting of 2 phases. In phase 1 we deal with functions that deal with patient input data and set up the chaincode then in phase 2 we make the functions which generate certain indexes that helps CouchDB to read/ access the data or query it. When deployed, the smart contract must be able to write patient records on the blockchain and execute various queries based on attributes of patient records.

Keywords:

Hyperledger, Blockchain, Smart Contract, CouchDB, Chaincode, Query, Index

Introduction:

The last couple of years have brought increase in popularity of blockchain technology with numerous projects being implemented by private and public entities. It is however an emerging technology and quickly evolving technology too. By creating a decentralized system, it removes the indulgence of central servers and provides peer-to-peer interaction. It can create a fully transparent and open to all database, which could bring transparency to the governance and elections. Traditional blockchain networks can't support private transactions and confidential contracts that are of utmost importance for businesses. Hyperledger Fabric was designed in response to this as a modular, scalable and secure foundation for offering industrial blockchain solutions.[1]

Smart contracts are lines of code that are stored on a blockchain and automatically execute when predetermined terms and conditions are met. At the most basic level, they are programs that run as they have been set up to run by the people who developed them. The benefits of smart contracts are most apparent in business collaborations, in which they are typically used to enforce some type of agreement so that all participants can be certain of the outcome without an intermediary's involvement.[2] In this report, we discuss about how we implemented a chaincode on a Hyperledger fabric blockchain framework whose primary task is to manage patient record assets which involves input, CouchDB enabled querying and updating patient records.

Terminology:

Blockchain:

A system in which a record of transactions made in bitcoin or another cryptocurrency are maintained across several computers that are linked in a peer-to-peer network. [3]

Hyper Ledger Fabric:

Hyperledger Fabric is a modular blockchain framework that acts as a foundation for developing blockchain-based products, solutions, and applications using plug-and-play components that are aimed for use within private enterprises. [4]

Smart Contract:

A smart contract is a self-executing contract with the terms of the agreement between buyer and seller being directly written into lines of code.[5]

CouchDB:

Apache CouchDB (CouchDB) is an open source NoSQL document database that collects and stores data in JSON-based document formats.[6]

Chaincode:

Chaincode is a piece of code that is written in one of the supported languages such as Go or Java. It is installed and instantiated through an SDK or CLI onto a network of Hyperledger Fabric peer nodes, enabling interaction with that network's shared ledger. [7]

Goal Description:

In this project, the goal is to complete the given codebase in step by step tasks which upon completion, when run, will be able to deploy a smart contract that can manage patient records. This mostly deals with constructing functions in JavaScript which are used for managing the input data and queries.

The process of constructing these functions is as follows:

Phase I: Deals with creating a patient record input function and another function on updating it.

1. Complete the createPatientRecord() function in the PatientRecordContract class
2. Complete the getPatientByKey function (patientrecordcontract.js)
3. Complete the getter and setter methods for lastCheckupDate field of the PatientRecord. (patientrecord.js)
4. Complete the updateCheckupDate function (patientrecordcontract.js)

Phase II: Deals with creating indexes which enables CouchDB to perform faster queries, creating query-based functions and setting up error transactions

1. physically build the index files on the top of the attributes that are defined in the PatientRecord class
2. Complete queryByGender function (patientrecordcontract.js)
3. Complete querybyBlood_Type function (patientrecordcontract.js)
4. Complete querybyBlood_Type_Dual function (patientrecordcontract.js).
5. Complete the unknownTransaction function

Description of proposed solution:

We created various function statements as per our goals for each phase. There were certain preliminary steps which we did per phase which helped complete the remaining functions as well.

Phase I:

1. We initially complete the function which creates the patient record and then call a method which basically adds the newly created patient record to the list.

2. Called a method named getPRecord() from another class called patientrecordlist(), which allows us to read the record based off its key. Then we added another statement to return this record to the user.

3. Then we created helper functions called setlastCheckupDate and getLastCheckupDate in a class named patientrecord() for assigning when was the last physical exam and return the date of the last checkup if entered.

4. In the updateCheckupDate function, we used a method named getPRecord() from class patientRecordList to read a record by key. Then, used the helper function setlastCheckupDate to input the last checkup date after which, called a method named updatePRecord to update the previous lastCheckupDate entry with the new one.

Phase II:

1. Created the blood_typeIndex.json file in the indexes folder with the same structure as genderIndex.json, but referencing another attribute. These Indexes enable the CouchDB to perform faster searches on all records given a certain query string.

2. Completing the queryByGender function:
    - Constructed the JSON CouchDB selector query-string object (queryString) that uses the genderIndex file in the queryByGender() function.
    - This was done by initially creating a query String JSON object to query using the genderIndex file modified in.
    - To make sure the query will actually use the index that we have created we used the use_index attribute inside the queryString.
    - Then we finally passed the built query String to the queryWithQueryString() function. This function will return a list of records that correspond to the gender that is passed as the input.
    - We then added another statement to return this list from the queryByGender function

3. Completing the querybyBlood_Type function:
   This is similar to step 2 except here we used the attribute for blood_type.

4. Completing the querybyBlood_Type_Dual function:
    - This function takes the transaction context and 2 blood types as input. So we constructed a JSON CouchDB selector query-String object that uses 2 blood_type indexes.

- Then we modified the queryJSON.selector to choose one of the two available bloodtypes after which we proceeded with the same steps as in step 3.

5. Completing the unknownTransaction function:
    - Implemented a certain default function that will, instead, execute every time a function is invoked that does not exist in the smart contract. This default function is called unknownTransaction and it receives the transaction context only.
    - Completed the function to return a string message [“Function Name Missing”]

Issues Faced and Methods used to resolve them:

Some of the issues faced while execution were as follows:

“Insertion of a patient digital asset (1) has failed”
“Insertion of a patient digital asset (2) has failed”
“Update of a patient digital asset (1) has failed”

Most likely the errors above were caused by a syntax error in the base code of the createPatientRecord function and updateCheckupDate function. These were promptly fixed as they were simple.

“unknown-function-call”

Calling an unknown function was not handled as specified in the assignment due to contradictions from references and hence had to be redone in a far more simplistic manner with only one statement, however this time it worked perfectly.

Related Works (if any):

https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html

https://hyperledger-fabric.readthedocs.io/en/release-1.4/chaincode.html

https://hyperledger-fabric.readthedocs.io/en/latest/developapps/smartcontract.html

https://github.com/hyperledger/fabric-samples/tree/master/commercial-paper/organization/magnetocorp/contract

https://hyperledger-fabric.readthedocs.io/en/release-2.2/developapps/transactioncontext.html

https://docs.couchdb.org/en/stable/api/database/find.html#selector-syntax

https://hyperledger.github.io/fabric-chaincode-node/master/api/fabric-contract-api.Contract.htm%20l

https://hyperledger-fabric.readthedocs.io/en/release-1.4/tutorials.html

Conclusion:

From this project, I have learnt how to implement a Hyperledger fabric smart contract for managing patient assets. This required learning to code in JavaScript and learning how to use CouchDB selector to query indexes. Now that I know the intricacies that go into coding a smart contact, I will be able to develop better frameworks or upgrade existing ones in a better way. This project has been an extremely important steppingstone into the world of the applications of blockchain engineering and I am quite glad that I can take part in learning more about this secure technology.

Bibliography:

[1] https://www.investopedia.com/terms/h/hyperledger-fabric.asp

[2] https://www.ibm.com/blogs/blockchain/2018/07/what-are-smart-contracts-on-blockchain/

[3] https://techterms.com/definition/blockchain

[4] https://www.investopedia.com/terms/h/hyperledger-fabric.asp#:~:text=Hyperledger%20Fabric%20is%20a%20modular,for%20use%20within%20private%20enterprises.

[5] https://www.investopedia.com/terms/s/smart-contracts.asp

[6] https://hyperledger-fabric.readthedocs.io/en/release-1.4/couchdb_tutorial.html

[7]http://fabrictestdocs.readthedocs.io/en/latest/chaincode.html#:~:text=Chaincode%20is%20a%20piece%20of,with%20that%20network's%20shared%20ledger.