**Title**: **Hyperledger Fabric Private Blockchain and Smart Contracts**

**Student Name:** Kenji Mah

**ASU ID:** 1219012444

**Abstract:**

For this project we will be implementing the logic that goes into making a simple application that manages patient record digital assets utilizing blockchain technology. We will be using Hyperledger Fabric blockchain environment, an open source enterprise-grade permissioned distributed ledger technology platform that supports smart contracts which is where we will be implementing most of the logic. We will also be utilizing Hyperfabric’s API to handle the blockchain itself, so all the logic behind aspects of a blockchain are already handled.

**Keywords:**

Smart contracts, Ledger, State, Transaction context, Transaction handlers

**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.

**Terminology:**

Smart contracts:

A smart contract, or what Fabric calls “chaincode”, functions as a trusted distributed application that gains its security/trust from the blockchain and the underlying consensus among the peers. It is the business logic of a blockchain application. [[1]](#footnote-1)

Ledger:

A ledger at a high level is a record of all the transactions throughout the blockchain’s history and a world state that holds a cache of the current values of these states. It is important to note that these records are immutable

State:

A state in terms of this project is a Patient’s record. On an abstract level, states can be any type of object that is kept track of in the blockchain.

Transaction context:

A transaction context performs two functions. Firstly, it allows a developer to define and maintain user variables across transaction invocations within a smart contract. Secondly, it provides access to a wide range of Fabric APIs that allow smart contract developers to perform operations relating to detailed transaction processing.[[2]](#footnote-2)

Transaction handlers:

Transaction handlers allow smart contract developers to define common processing at key points during the interaction between an application and a smart contract.[[3]](#footnote-3)

**Goal Description**:

**Phase I:** **Understand the Hyperledger Fabric environment**

1. The first goal we need to achieve is a solid understanding of the environment that we are working in.
2. Set up proper workspace needed to create a solution.

**Phase II: Implement the code and logic**

1. Complete the functions of the code that creates and maintains the patient records
2. Complete the functions of the code that allows for querying of the patient records

**Description of proposed solution:**

A project template for the code is given to us so we must learn about the input parameters for the functions in the code, the environment that it is settled in, and the output of the program. After identifying how everything integrates with each other, we can then fill in the logic for the functions.

Phase I: Understand the Hyperledger Fabric environment

A quick overview of how Hyperledger works from an application developer’s perspective is that there are mainly just smart contracts and the ledger. There are of course many other terminologies and aspects of the blockchain, however we are only concerned with these two main terms for the scope of this project. Smart contracts in Hyperledger Fabric are authored in many general-purpose programming languages. In this case we are going to use Node.js to develop our application. The three main files that will help us understand the problem or are part of the code we need to complete are as follows: patientrecord.js, patientrecordcontract.js, and patientrecordlist.js.

The state file acts as a formatter for the data that will be transacted upon in the chaincode. In the patientrecordcontract file, the Patient Record Contract and the Patient Record Context are defined to extend the contract class and context class respectively. Once the contract is created in the blockchain, it will then create a transaction context which creates a patient record list for keeping track of the patient records for the rest of the chaincode’s lifetime. For further clarification a patient record is an extension of a state and a patient record list is an extension of a state list.

In order to prepare our environment, we could potentially create our own blockchain network of nodes locally and use that environment for debugging, however I judged the projects difficulty and believed that the auto grader with limited feedback output would suffice.

Phase II: Implement the code and logic

The code was sectioned into eight tasks and a quick description of how each are solved will follow.

Task 0 **createPatientRecord**: This task was simple and straight forward, we needed to utilize the transaction context that was passed into the function as a parameter and call the addPRecord from the patientRecordList class. The record added was the state of a PatientRecord with the given parameters.

Task 1 **getPatientByKey** : Again, for this task we will use the transaction context and call the getPRecord method from the PatientRecord list.

Task 2 **lastCheckupDate field for PatientRecord** : For this task we must create setter and getter methods for a new variable that we want to keep track of for each state which is called lastCheckupDate. This was simple because we are just initializing a new variable for the state and was straightforward considering that the syntax for previous variables were already defined.

Task 3 **updateCheckupDate**: For this method we need to get the record from the transaction context, and then perform updates on the state that was returned. Finally, we need to use the same transaction context and perform an update transaction by passing the updated patient record.

Task 4 - 6 **queryByGender, querybyBlood\_Type, querybyBlood\_Type\_Dual**: Before we can query anything from CouchDB, we must first setup an index for more efficient queries. We do this by creating a JSON file in the couchdb’s indexes folder. Because the logic for querying given a transaction context is already written for us, all we really need to do is create the specific query string for gender. The only difficult part for this task is understanding the format that the sting needs to be in which can be found in Hyperledger’s documentation about CouchDB. [[4]](#footnote-4)

Task 7 **unknownTransaction**: This function is one of the many optional transaction handlers’ functions that receive control before or after every transaction in a smart contract is invoked. This function handles an attempt of a transactions that is not in the contract which for our case will return the error “Function name missing”.

**Issues Faced and Methods used to resolve them:**

Most of the problems that came with this project mainly came from debugging using the auto grader. Small discrepancies from the assignment instructions and the comments in the code was a determining factor for whether functions passed the testing scripts. Because the output of the auto grader was vague, misleading instructions could throw off an entire function’s correctness. One specific case of this was in task 7 where the auto grader dismissed the instruction’s message because of the incorrect letter case for the words “name” and “missing”.

Another slight setback was knowing when to use “await”. This was new to me because I have never coded in Javascript or Node.js.

**Conclusions**:

In conclusion I learned about how the Hyperledger environment works and how to implement basic code for a simple application in the environment. The usefulness of a blockchain became apparent to me after looking at all the examples that Hyperledger provided and I truly believe that this technology can revolutionize many industries.

**Bibliography**:

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

<https://hyperledger-fabric.readthedocs.io/en/latest/developapps/transactioncontext.html#clientidentity>

<https://hyperledger-fabric.readthedocs.io/en/release-1.4/couchdb_tutorial.html>

<https://hyperledger-fabric.readthedocs.io/en/latest/developapps/transactionhandler.html>

1. https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html [↑](#footnote-ref-1)
2. https://hyperledger-fabric.readthedocs.io/en/latest/developapps/transactioncontext.html#clientidentity [↑](#footnote-ref-2)
3. https://hyperledger-fabric.readthedocs.io/en/latest/developapps/transactionhandler.html [↑](#footnote-ref-3)
4. https://hyperledger-fabric.readthedocs.io/en/release-1.4/couchdb\_tutorial.html [↑](#footnote-ref-4)