*HyperLedger Blockchain Implementation*

Nigel Singh  
*Ryerson University  
CPS 831*Toronto, Canada  
[nigel.singh@ryerson.ca](mailto:nigel.singh@ryerson.ca)

***Abstract –* This paper describes a simple hyperledger fabric blockchain implementation, with the use of 2 peer networks, and 2 organizational networks exchanging information in a private blockchain. The blockchain uses chaincode, written in the Go language, to create secure transactions between entities. Channels are used as the main communication method between entities. The information being passed between entities is the year a student graduated, institution that the student graduated from, and the degree that a student graduated with.**

# Introduction

This document is the final report for the accompanying blockchain hyperledger fabric project in the undergraduate course CPS 831: Software Engineering II. The hyperledger fabric tutorial was used to create the network where the peers send and receive information.

The hyperledger fabric tutorial outlined the ways in which we could: set up the environment variable with a path to the binaries [1]; create entities within the network consisting of 2 organizations and 2 peer organizations, and a genesis block; a channel of communication amongst the peers [2]. The network was executed using a series of bash commands.

The chaincode was written in Go, and was altered from the original IBM implementation of the chaincode to the purposes that I saw fit for my blockchain assignment. I decided to create an implementation of the blockchain that tracked the year, degree and institution that a student graduated from.

# Objective

At first, I wanted to create a blockchain implementation related to the medical field. After consultation, I realized that this was beyond the scope of my blockchain knowledge and experience. If I pursued this project, I would be faced with the accompanying security issues which I believe requires a deeper understanding of blockchain implementation. Then, I decided to create a secure blockchain amongst educational facilities where the administrators could securely access student records. I believe the practicality of this implementation could be applied to real institutions. They would no longer have to fear security breaches in the system, because all changes would be seen through the ledger.

# Chain Code

The chaincode was written in the Go programming language, and contains various functions which will be listed below.

The *initLedger* function creates a new structure named *certifs* and populates the ledger with the following entries (Fig. 1). It then proceeds to run through all the certificates and assign the key values to each record, identifying them by ‘PERSON[1-999]’ respectively.

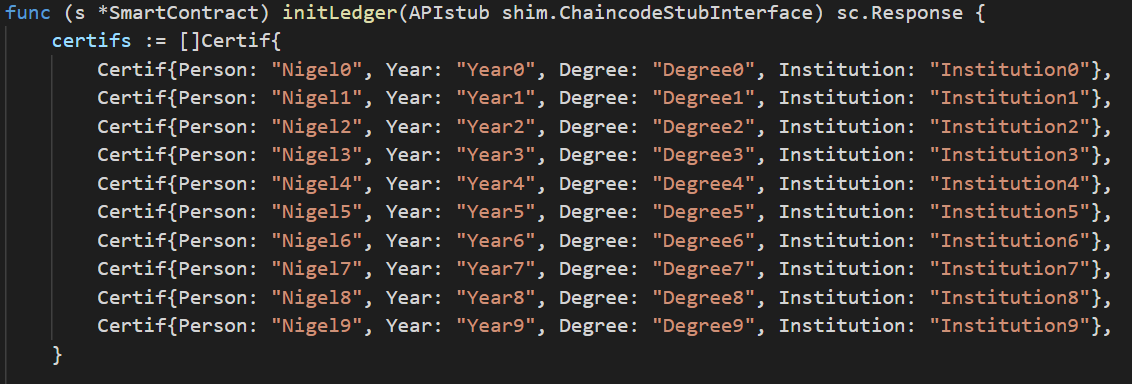


Fig. 1 initial database of students to be written to the ledger.

The createCar function gives the organization the ability to add a new record to the ledger (Fig. 2). This functionality is executed by altering the code and running the invoke.js file with the appropriate arguments.

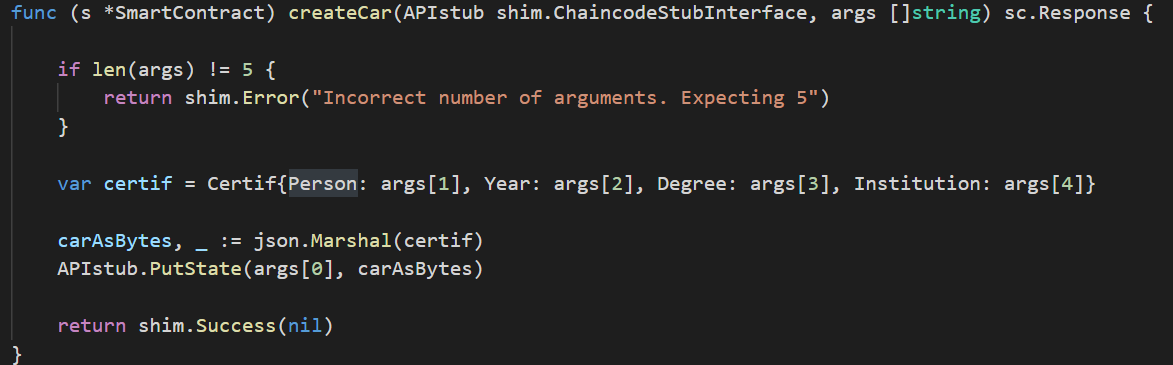


Fig. 2 function which allows users to generate new car records.

The queryAllCars function gives the peers the ability to query all the records in the database. The records are then output on the terminal where the list of names are displayed, with the respective record pairing (Fig. 3).

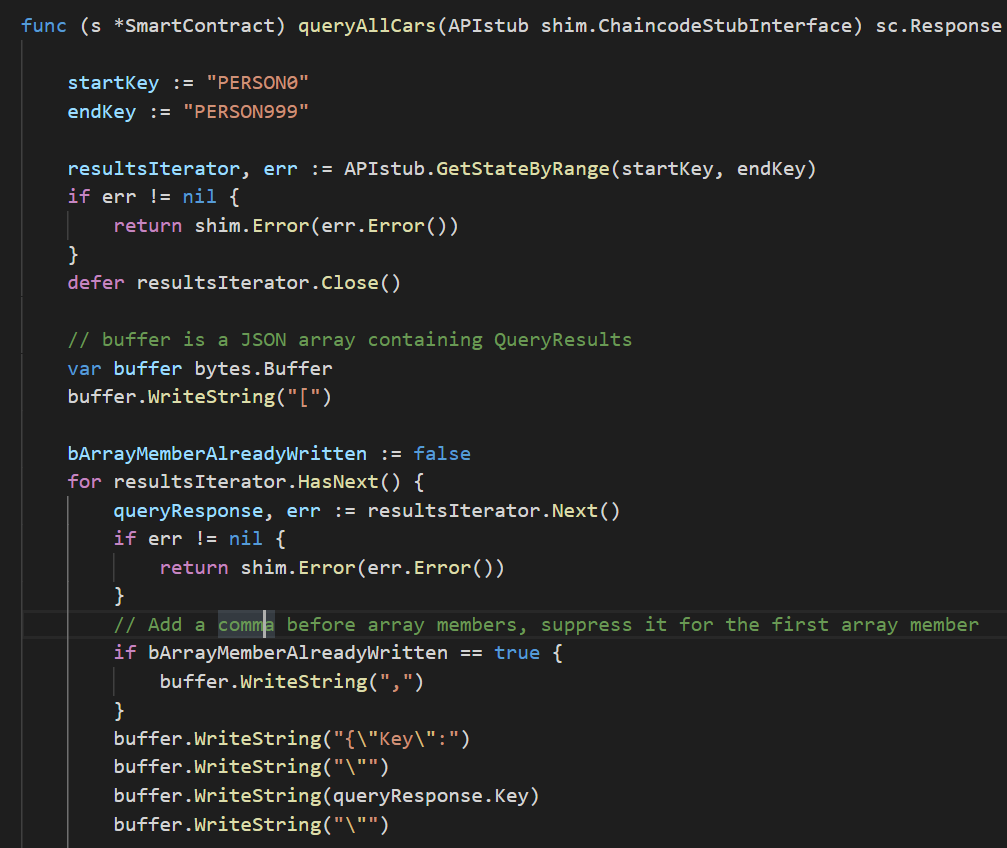


Fig. 3

The changeCertifInstitution provides the organization with the functionality to alter the ledger. By calling this function an organization can select the person in the ledger that they wish, and change **only** their institution name to the new desired name (Fig. 4). These changes are then submitted to the ledger.

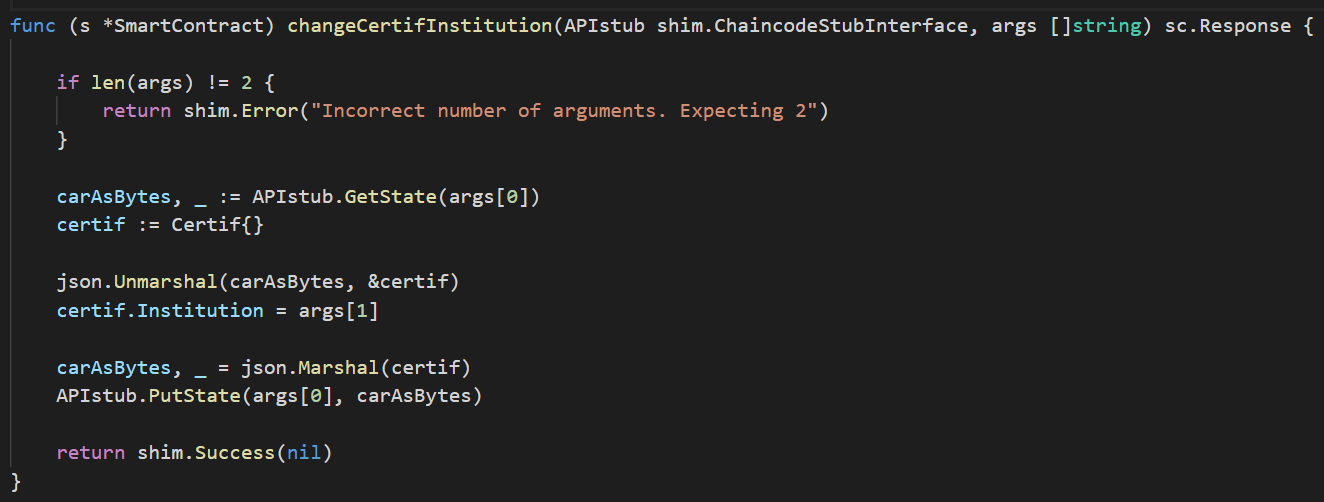


Fig. 4

The following functions: changeCertifYear (Fig. 5), changeCertifDegree (Fig. 6), and changeCertifPerson (Fig. 7) are all related to the changeCertifInsitituion. They all make changes to a specific person in the ledger, and alter the year, degree, or person ID. Once an organization calls this function, the changes will be submitted to the ledger.

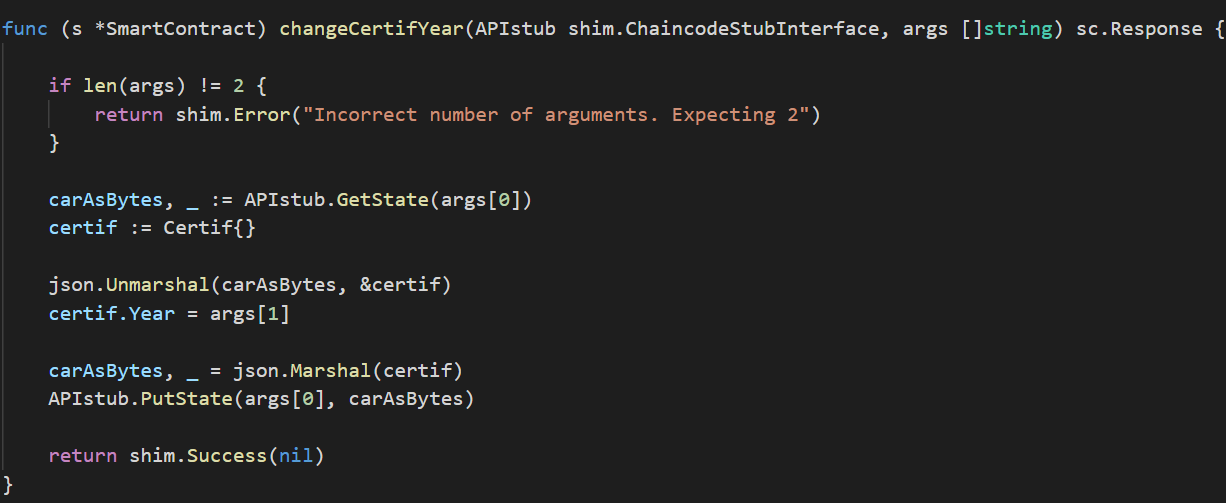
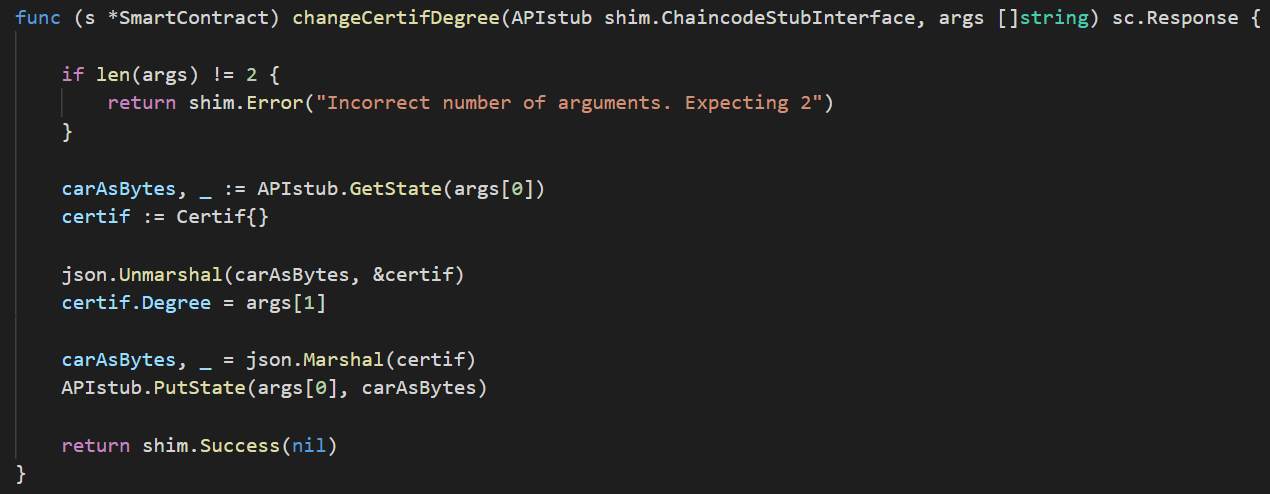


Fig. 5

Fig. 6

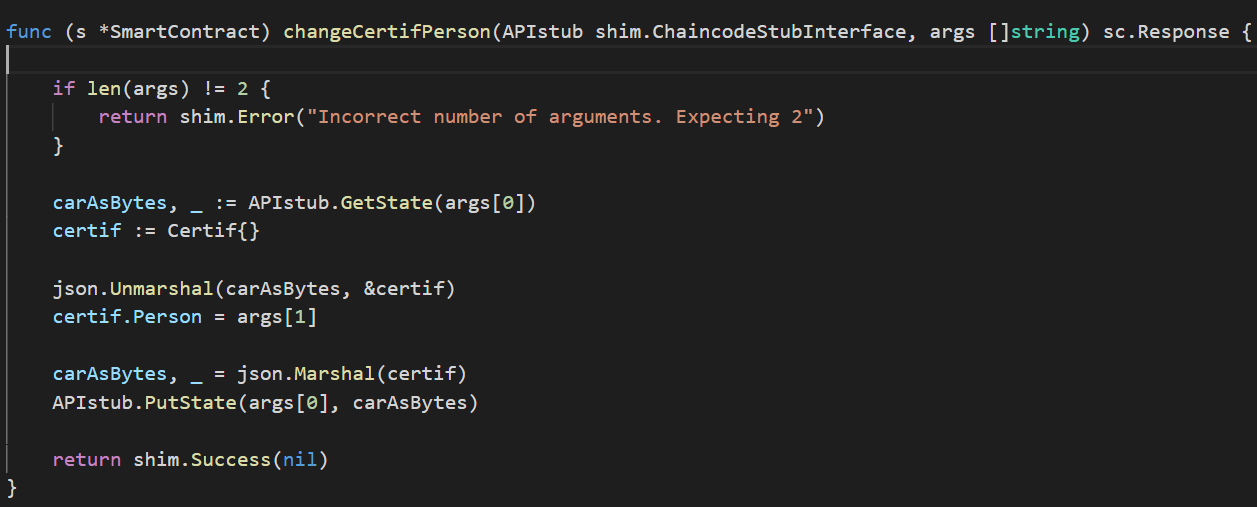


Fig. 7

# Other Functions

In order to display the results of the ledger the function file query.js must be altered to the input parameters that meet your specifications (Fig. 8).

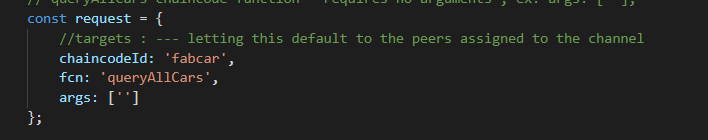


Fig. 8

In order to alter the ledger, you must alter the invoke.js file which sends the proposal to the endorsing peers (Fig. 9).

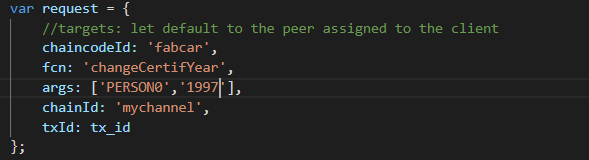


Fig. 9

# Performance Results

Provided below are the results of running the functions listed above, including:

* queryAllCars
* changeCertifInstitution
* changeCertifYear
* changeCertifDegree
* changeCertifPerson

The first function run below is queryAllCars which pulls the most recent ledger update amongst all peers (Fig. 10)

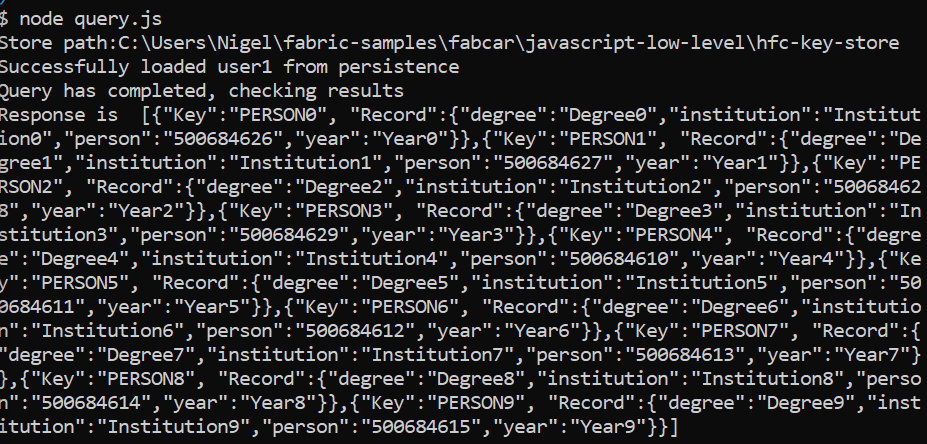


Fig. 10

When you alter the function invoke.js to the changeCertifInstitution and change the respective arguments (Fig. 11), you get a Status – 200 message, stating that the transaction was successful. This transaction has now been added to the ledger, and is hashed with the transaction id which is displayed below (Fig. 12). Once again, when you query the ledger you can see that the pair with the key ‘PERSON0’ has had their institution changed to ‘RYERSON’ (Fig. 13)

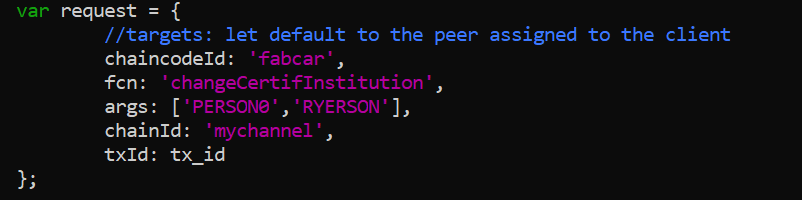


Fig. 11

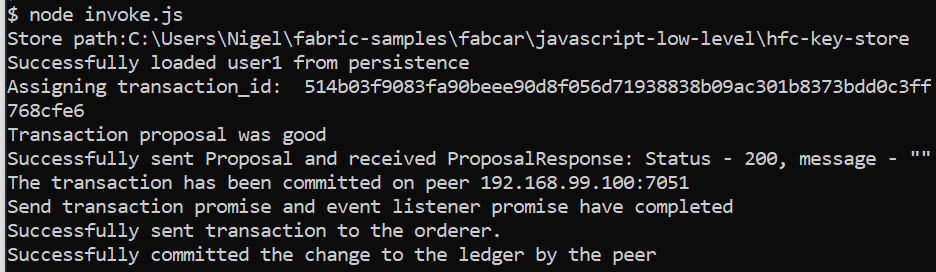


Fig. 12

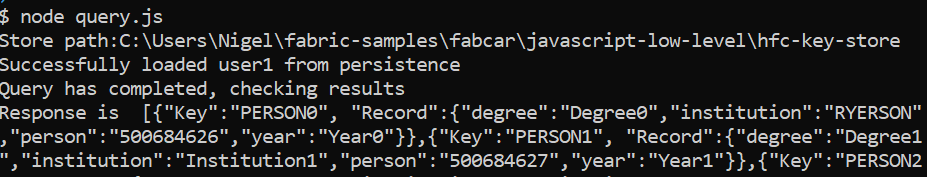


Fig. 13

Below I have shown the functionality of the remaining functions.

The figure below shows the changeCertifYear function has changed the year in the key ‘Person0’, to ‘2019’.

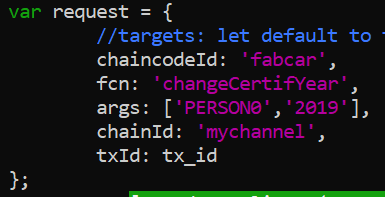


Fig. 14

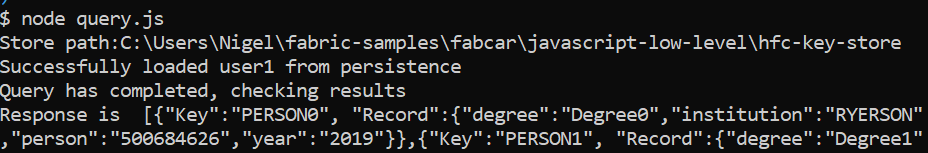


Fig. 15

The figure below shows the changeCertifDegree function has changed the degree in the key ‘Person0’ to ‘Ph. D’.

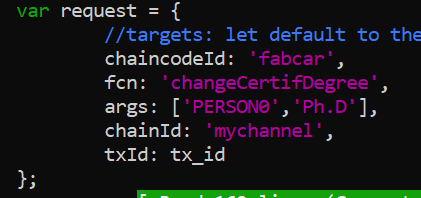


Fig. 16

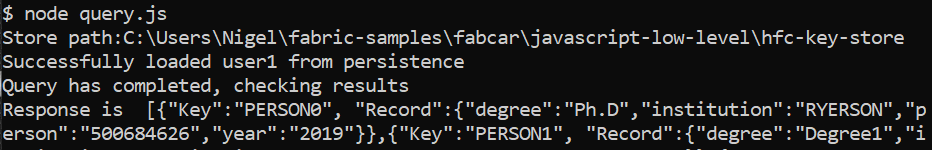


Fig. 17

The figure below shows the changeCertifPerson function has changed the year in the key ‘Person0’, to ‘Nigel Singh’.

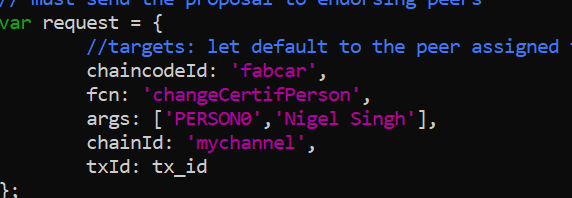


Fig. 18

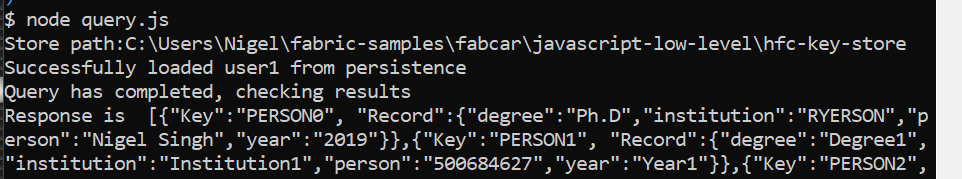


Fig. 19

# Consensus in HyperLedger

HyperLedger Fabric was confusing to me at first when trying to understand the concept of consensus with regards to my program. However, I realized that the ordering service would order the People in the ledger along with their unique ID’s. Not to mention, this is a private block chain, and the organizations have created an endorsement policy. The client in this case collects the endorsing peers of the organizations in order to consider the transaction valid.

# Conclusion

At first, the project tested my coding knowledge, as well, it tested my ability to understand the fundamentals of HyperLedger Fabric. As showcased throughout this report, I successfully created a hyperledger blockchain decentralized application with the use of the Go programming language. I still believe that I would need a deeper understanding of blockchain in order to develop a distributed ledger in the medical field, however, in the future I believe I could accomplish this task. As my first blockchain assignment, I believe that I have gained a deeper understanding for the open source HyperLedger implementation.

# References

[1] Install Samples, Binaries and Docker Images. (n.d.). Retrieved from <https://hyperledger-fabric.readthedocs.io/en/release-1.4/install.html>

[2] Building Your First Network. (n.d.). Retrieved from <https://hyperledger-fabric.readthedocs.io/en/latest/build_network.html>