Cardiff School of Computer Science and Informatics

CM3202 One Semester Individual Project 40 Credits

Final Report

Creating a Data Tracking Application, utilising Blockchain Technology

Author:
Jonathan Jones

Supervisor:
Dr George Theodorakopoulos

Moderator:
Dr Angelika Kimmig

May 6, 2018

"A river cuts through rock, not because of its power, but because of its persistence"

-Jim Watkins

Acknowledgements

Firstly, I would like to thank my supervisor, Dr George Theodorakopoulos, for his invaluable advice and guidance throughout the project. His keen insight and meticulous methodology has been instrumental in bringing it to fruition.

Secondly, I would like to thank my moderator Dr Angelika Kimmig for the feedback she has provided and Pete Davison for supporting the development of the application and answering the many questions I have asked about the Hyperledger Fabric Framework.

Lastly, I thank my friends and family, all of whom have challenged and supported me to become the person I am today. A high standard of principles has been instilled which will last a lifetime.

Abstract

The General Data Protection Regulation (GDPR) is new legislation that intends to give control of data back to the user. This project aims to create an application that will complement the GDPR by providing an auditable record trail of an individual's personal data. A decentralized network is created utilizing blockchain technology to store the immutable trail and an application is developed allowing an individual to identify companies who are using their data.

Contents

1 Introduction **2**
- 1.1 Introducing the Problem and Project Motivation 2
- 1.2 Project Scope 3
- 1.3 Intended Stakeholders 3
- 1.4 Approach 3
- 1.5 Assumptions 4

2 Background **5**
- 2.1 Overview 5
- 2.2 Current data tracking/storing applications 5
- 2.3 Blockchain 6
  - 2.3.1 Cryptocurrency 6
  - 2.3.2 Proof-of-Work 7
  - 2.3.3 Permission and Permissionless 7
- 2.4 Blockchain Frameworks 7
- 2.5 Hyperledger Fabric 10
  - 2.5.1 Peers and Smart Contracts 10
  - 2.5.2 Channels 10
  - 2.5.3 Orderers 11
  - 2.5.4 Organisations 11
  - 2.5.5 Digital Certificates 12
  - 2.5.6 Certificate Authority 12
  - 2.5.7 Applications 12

3 Specification and Design **15**
- 3.1 Overview 15
- 3.2 Justification 15
- 3.3 Requirements 16
- 3.4 Overview of the Application 16
- 3.5 Blockchain Structure 17
- 3.6 Chaincode/Smart Contract 18
- 3.7 Channel 19
- 3.8 Example Scenario 19
- 3.9 Graphical User Interface Design 19
  - 3.9.1 Login Screen 19
  - 3.9.2 Register Page 19
  - 3.9.3 Main Page 19
  - 3.9.4 Insert Page 20

4 Implementation **23**
4.1 Overview . . . 23
4.2 Setting Up the Network for Development . . . 23
4.3 Interacting with the network . . . 23
4.4 Developing Chaincode . . . 24
4.5 Developing Node SDK . . . 25
4.6 Building REST API . . . 26
4.7 Building the Graphical User Interface . . . 26
4.7.1 Log In screen . . . 26
4.7.2 Register Page . . . 26
4.7.3 Main Page . . . 27
4.7.4 Insert a block Page . . . 27

5 Results and Evaluation **29**

6 Future Work **30**

7 Reflection on Learning **31**

References **32**

List of Figures

1 The Agile Methodology Cycle. [2] . . . 4
2 The Blockchain Data Structure. . . . 6
3 An Example of a Simple Peer Network. [1] . . . 11
4 A Simple Channel Example. [1] . . . 12
5 A Simple Organisation Example.[1] . . . 13
6 Querying and Inserting a New Block. [1] . . . 14
7 An Overview of the Data Tracking Application Network. . . . 17
8 The Data Tracking Blockchain. . . . 18
9 The Contents of an Example Block. . . . 18
10 The Login Screen. . . . 20
11 The Register Page. . . . 21
12 The Main Page. . . . 21
13 The Insert Page. . . . 22
14 Code Snippet for importing the shim Interface. . . . 24
15 Code Snippet of the Query Function within the Chaincode. . . . 24
16 Code Snippet for Querying the Blockchain using the SDK. . . . 25
17 Code Snippet for Inserting a New Block using the SDK. . . . 25
18 The Login Web Page. . . . 26

19 The Register Web Page. 27
20 The Main Web Page. 27
21 The Insert Web Page. 28

Table of Acronyms

API	Application Programming Interface
BYFN	Build Your First Network
CA	Certificate Authority
EU	European Union
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IOT	Internet of Things
OS	Operating System
REST	Representation State Transfer
SDK	Software Development Kit
USB	Universal Serial Bus

1 Introduction

1.1 Introducing the Problem and Project Motivation

In June 2016 the United Nations declared that internet access was a human right [5]. Connecting over 7 Billion people to the internet will exponentially increase the quantity of data generated. Not only are the number of people who have access to the internet increasing, the amount of devices per individual are becoming greater, either directly e.g. buying a smartwatch and linking it to your phone, or indirectly e.g. linking the fridge to the internet at home. Any device with a microprocessor has the potential to become part of the Internet of Things (IOT). These devices are generating vast amounts of useful data that companies can analyze, to create and develop new products. A very topical example of this, is how American Facebook users, that liked certain posts were targeted by a British Company called Cambridge Analytica. The company used specific adverts targeting individuals with the purpose of persuading them to vote for a particular candidate during the United States presidential election. The European Union(EU) have recognized this abuse of power of personal data and have introduced the General Data Protection Regulation, which will become effective from the 25th May 2018. In summary it aims to enhance the control an EU citizen has over their personal data. Important changes to the current legislation include [4]:

1. Right to Access - allow subjects to obtain confirmation that their personal data is being processed. Furthermore the controller has to provide a copy of the personal data in an electronic format.

2. Breach of Notification - need to notify users within 72 hours of first becoming aware of a data breach.

3. Right to be forgotten - entitles the data subject to have the data controller erase his/her personal data, cease further dissemination of the data, and potentially have third parties halt processing of the data.

There still persists an enormous task of tracking this data. To record it manually would prove inefficient and waste invaluable time. Thus, this report proposes a method of tracking such data automatically and delivers an application to visually display, a record of companies that have interacted with an individual's data.

1.2 Project Scope

Taking the aforestated into account, the aspiration of this project is to develop a proof of concept and demonstrate a functional application that is able to track a user's data. From the initial report, two objectives were created:

1. Create a decentralised network.
2. Develop the capability to communicate with a node within the network.

If these could be achieved, the project would be considered a success. The timeframe to achieve these goals was sufficient and careful planning of time enabled the production of this report. Nevertheless, since the concepts discussed within the later chapters had been completely alien to the author, the allocated time to learn the various components e.g. creating a system using a framework that was unfamiliar, was difficult to estimate. Therefore, the two objectives would be a priority and any additional time would be used to increase secondary functionality e.g. developing a graphical user interface.

1.3 Intended Stakeholders

The GDPR affects any company that holds or processes data of EU citizens. Failure to comply with the regulations can result in a penalty of up to €10 million, or 2% of the worldwide annual revenue of the prior financial year, whichever is higher [6]. The stakeholders will be companies/organizations throughout the world that processes or stores EU citizens data, EU governments and any EU citizen that wishes to track their data using this application.

1.4 Approach

The applications development followed an agile methodology. This development life cycle has four core principles; adaptive planning, early delivery, continuous improvement and being able to respond to change quickly and easily [2]. Figure 1 shows the different stages:

This project focuses on the early stages of the lifecycle i.e. Planning, Design and Development, because of time restrictions. At the start of each week, a list of tasks was assigned i.e. detailed in the initial report. These needed to be completed within the timeframe, to ensure the progress of the project. At the end of the week the tasks were reviewed, to modify plans appropriately i.e. re-plan to finish tasks that had not been completed, and to learn from mistakes to increase productivity.

Figure 1: The Agile Methodology Cycle. [2]

1.5 Assumptions

In the scope of this project, data/personal data is defined as information that involves several of the following terms: First Name, Last Name, Email Address and Age. The list is minimal to reduce the complexity and variants that may appear in a large scale system. There is an assumption that companies will want to use and participate within the network. If in reality this becomes unrealistic the target audience would become a state of the EU e.g. the United Kingdom's (UK) government (whilst they are still in the EU), would ideally issue legislation enforcing all companies/organizations to use the application. Lastly, data that has been generated before registering for the application will not be tracked/recorded. The timeframe dictated that only the essential features of the application be developed, this did not include tracking data previous to using the application.

2 Background

2.1 Overview

The following section provides a brief summary of; data tracking applications that are currently available, blockchain technology and frameworks that aid the creation of decentralised systems. In addition, the frameworks have been compared and an overview produced to identify the most suitable for this project.

2.2 Current data tracking/storing applications

Currently, it is unintuitive to find the data that a company has stored about its users. Take Google for example, if a user has a Gmail account and frequently browses YouTube, they would be able to view their history by navigating to their account settings. They would not be able to see the exact location they were at and the device on which they viewed a particular video i.e. Geolocation tracking. Google keeps this information private and stores it, to process later.

A British company called Databox has attempted to address this problem by keeping all the users data in a single place i.e. a box of data. Traditionally the user would give the company a copy of all their data. Companies that want to run analysis on users' data, would download the databox application and provide the code to process the data. The benefit of this is that companies do not receive a copy of the data. Although this is an interesting method, there seems to be some limitations. Firstly, Databox may not store geolocational data - in which particular companies may want to process. Secondly, the data is stored in a centralized repository. An attacker targeting an individual would need only to gain access to this repository, to obtain all of their information. Finally, there is no detailed explanation of what happens to the data generated via IOT devices. For example, would Databox collect this data directly or would the user need to insert it somehow.

Furthermore, there appears to be a lack of companies attempting to track user data. A potential reason for this is because they have no obligation to share this information and selling it is more lucrative. The GDPR does not become effective until 25th May 2018, hence companies are more concerned with becoming compliant with the new legislation and thus there has been little development, creating applications that aim to incorporate these laws to benefit EU citizens.

2.3 Blockchain

Blockchain technology was first discussed in a white paper, published in 2008 about the digital cryptocurrency Bitcoin [3]. The author suggested a method to record all transactions on the network i.e. a "chain of blocks" which was more commonly referred to as the "blockchain". To fully comprehend how a blockchain works and its practical relevance, the following section will explain how the underlying Bitcoin technology works. The blockchain is embedded within the foundations of Bitcoin and understanding the structure of the cryptocurrency will help to explain the design decisions in chapter 3.0.

2.3.1 Cryptocurrency

Bitcoin has three distinct layers:

1. **Bitcoin Blockchain**: Underlying decentralized ledger.
2. **Bitcoin Protocol and Client**: Software programs that conduct transactions.
3. **Cryptocurrency**: Bitcoin (BTC).

The Bitcoin network consists of several nodes (network components) that share a ledger i.e. a record of all transactions that have occurred on the network. Each node contains a copy of this ledger which is stored as a blockchain. The blockchain is a linked list data structure built from hash pointers:

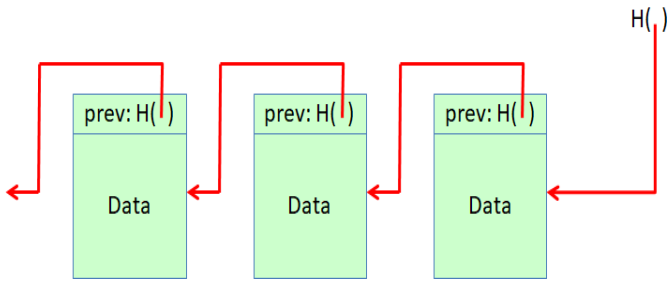


Figure 2: The Blockchain Data Structure.

Each block contains various information about the transaction e.g. the block size and the hash of the previous block. The genesis block i.e. the first block in the chain, does not point to any block because there are no previous blocks to hash upon. This method produces an immutable history of all transactions that have occurred on the network. An attacker attempting to

insert a malicious block in the chain, would need to create a block such that it has the same hash as the original block, (violating the security properties of a cryptographic hash function) or create the hash of all subsequent blocks. To add a new block onto the blockchain, each node needs to agree which block goes first because several nodes are attempting to insert their block onto the blockchain first. This is called consensus and the Bitcoin network uses the Proof-of-Work method.

2.3.2 Proof-of-Work

Miners (Nodes) in the network, compete to solve hash puzzles of the block they want to insert onto the blockchain. The first miner that solves the hash puzzle appends their block to the chain. That miner receives a number of Bitcoins as a reward. The process restarts and miners compete to solve the next hash puzzle. The disadvantage of this method, is that it is highly CPU/GPU intensive, meaning that the computers solving the hash puzzle are using a vast amount of electricity to find a solution.

2.3.3 Permission and Permissionless

Bitcoin is known as a public permissionless network because any node is allowed to join the network. Therefore, methods for example Proof-of-Work, need to be in place to prevent malicious attacks. The opposite of this concept is called private permissioned networks i.e. only authorized nodes can join the network, this creates the opportunity to use less CPU/GPU intensive consensus algorithms.

2.4 Blockchain Frameworks

To build a blockchain application there are three available options, the first is to use a cryptocurrency e.g. Bitcoin, Ethereum, etc. The second option is to use a framework that allows the creation of blockchain applications without the cryptocurrency layer. Thirdly, create a blockchain framework from scratch specific to the application. The first option benefits from greater maturity (especially if Bitcoin was used) and has a large developer base, meaning there would be greater support for development. The drawbacks include, initially installing and setting up the wallet, which may cost valuable time which could be spent on more beneficial avenues. The use of cryptocurrencies within the project could limit its potential, firstly it would require the developer to buy Bitcoin (or alternative tokens) to be able to test

it successfully and secondly it would incorporate the application to use cryptocurrency which would hypothetically slow audience adoption if the project was to go live. The benefits of the second option are; it does not have any cryptocurrency i.e. the developer is able to build the application without the limits of the cryptocurrency. In addition, it allows the creation of private permissioned networks, providing greater control of the system and eliminating the high cost that is associated with CPU/GPU intensive consensus algorithms such as the Proof-of-Work method. The disadvantages may include; little support for development as there may not be vast amounts of programming tutorials/examples. Possible software "bugs" i.e. the frameworks are still under development and susceptible to programming issues. The third option would provide little benefit, building an application specific for the problem might implement improvements on current frameworks, but the project would be too ambiguous even a simplified version would turn the project into "building a blockchain framework" rather than a "data tracking application". In conclusion, it was decided the second option was the best method to build the application. Research was conducted to find frameworks that aided the creation of blockchain applications. Several frameworks were found, but they all used similar terminology and promised the same functionality. Each framework would require time to set up the software, read the documentation and experiment programming. From discovering this, I decided to compare the frameworks by Maturity, Developer base, Documentation and Endsorment. These properties help to identify, the size of the development community, including how much help can be expected and additionally any resources that companies are contributing. I then experimented with the best two frameworks after choosing from the following; Openchain, Multichain, BigChainDB, Hyperledger Fabric, Monax and Credits. Although there are many more, it was decided to concentrate on the most popular six frameworks. Below is a table summarizing the comparison:

Framework	Maturity	Developer Base	Documentation	Endorsement
Openchain	2015	Small	Highly Detailed	N/A
Multichain	2015	Medium	Moderately Detailed	Accenture, Adriot CadenaTech, and several other companies.
BigchainDB	2016	Small	Moderately Detailed	Microsoft, Toyota and several other companies.
Hyperledger Fabric	2016	Large	Highly Detailed	IBM, Intel, and several other companies.
Monax	2014	Small	Moderately Detailed	Quadrio, Ifin Sistemi and two other companies.
Credits	2013	Small	Detailed	N/A

The two frameworks which would be best suited to this project are; Hyperledger Fabric and Multichain. The Hyperledger Fabric framework, was the more difficult to install because there were issues setting up Docker; a software prerequisite (detailed in the implementation section). Once installed, an example network was provided and there was a vast amount of documentation, to explain the different features that were available. Multichain offered similar examples but were not as intuitive. Multichain provided a single node, meaning that the user would need to create additional network components manually. Although, this could be automated by developing a script to initialize the network, it would still involve wasting extra time that could be allocated to developing a prototype. Furthermore, Hyperledger tutorials were far superior than its competitor, they provided more detail and had a greater selection of possible features. It was clear that Hyperledger Fabric was the best framework to use. In addition, during a discussion with PhD student, Pete Davidson (his research focuses on blockchain technologies at Cardiff University), we talked about the advantages/disadvantages of the various frameworks and concluded that Hyperledger Fabric was the best. He mentioned that he had used it for over a year and agreed to provide

guidance with any issues encountered. In conclusion the Hyperledger Fabric framework was the best software to use for this project, it had the most examples out of the compared frameworks and is endorsed by large technological companies including the Linux foundation i.e. one of the largest open source organizations in the world. The current version of the most stable release of Hyperledger Fabric is 1.0, this is the version used throughout this project.

2.5 Hyperledger Fabric

Hyperledger Fabric is one of several projects developed by a consortium of companies known as the Hyperledger Foundation. Its aim is to develop tools and frameworks to assist with building decentralized applications. The Hyperledger projects are managed by the Linux Foundation and include partners such as IBM and Intel. The main projects available are; Fabric, Sawtooth, Iroha, Burrow and Indy. They each aim to provide different blockchain functionality e.g. Sawtooth incorporates the use of RFID tags i.e. embedded tracking devices to link items to a global decentralized IOT. Fabric is a general framework, that provides the foundations of a decentralized application and allows the developer to easily build upon it. In addition, it is also very good for prototyping i.e. once you understand the tools, it should be relatively fast to build a viable proof-of-concept. The following section aims to describe an overview of the framework, necessary to understand the design decisions in the next chapter.

2.5.1 Peers and Smart Contracts

A fabric network is composed of several different types of nodes. The most important is called a Peer node because it stores an instance of the ledger and chaincode. Note: Peers are able to store several different ledgers/chaincode instances. The only way to access the ledger is by invoking methods detailed in the chaincode. This may be referred to interchangeably, as a smart contract. The chaincode determines the functionality an entity may have on the blockchain. Figure 3, shows an example of a simple network.

2.5.2 Channels

Channels are a mechanism for peers to communicate and transact privately i.e. Peers that share the same instance of a ledger are part of the same channel. Again, a peer may be a part of several different channels, because it contains an instance of several different ledgers. Conceptually, it can be

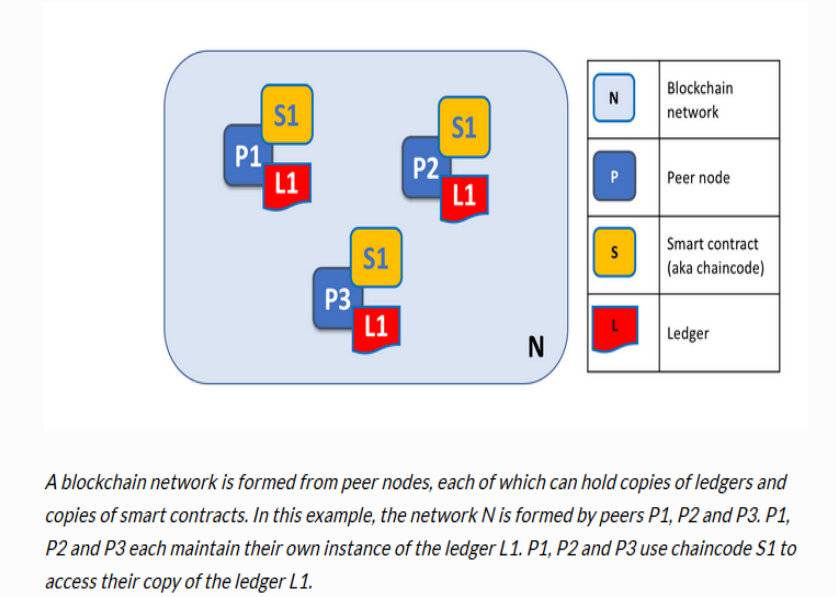


Figure 3: An Example of a Simple Peer Network. [1]

thought as a group of friends. A person (the Peer) may have different social groups e.g. work friends and school friends, correlating to two different channels. Figure 4, shows an example of a simple channel network.

2.5.3 Orderers

To ensure consensus throughout the instances of the ledgers, Fabric uses Orderer nodes. An orderer gathers all the proposed transactions from the peers and groups them together to form a block, these blocks are then sent to all peer nodes to be verified i.e. the Peer checks each transaction to ensure it is valid. Then the block is added to the blockchain in each Peers' local ledger.

2.5.4 Organisations

The network is composed of components belonging to several organizations. For it to be truly decentralized each organization will contribute nodes e.g. organization 1 may be contributing 4 Peer nodes, and 2 Orderers to the network, while organization 2 may be contributing 3 Peer nodes and 1 Order node. Figure 5 shows an example of four organizations within the network.

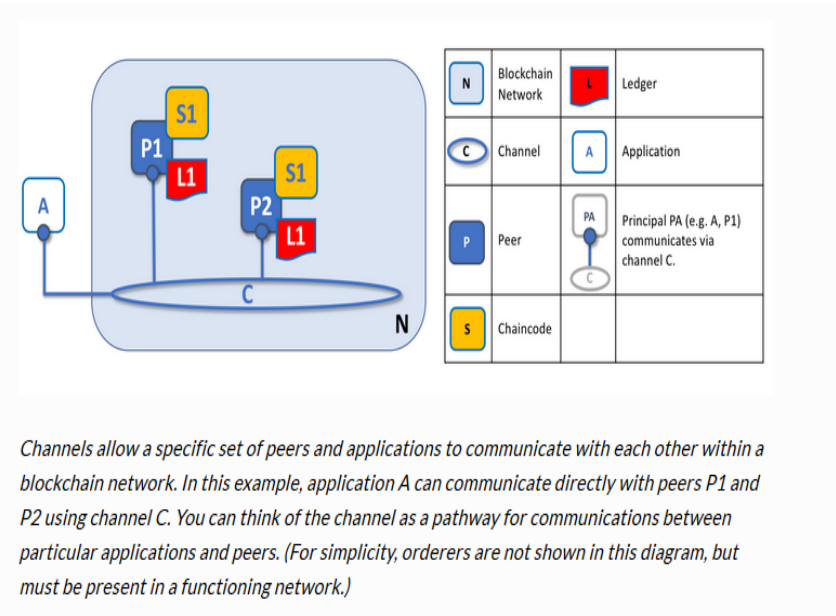


Figure 4: A Simple Channel Example. [1]

2.5.5 Digital Certificates

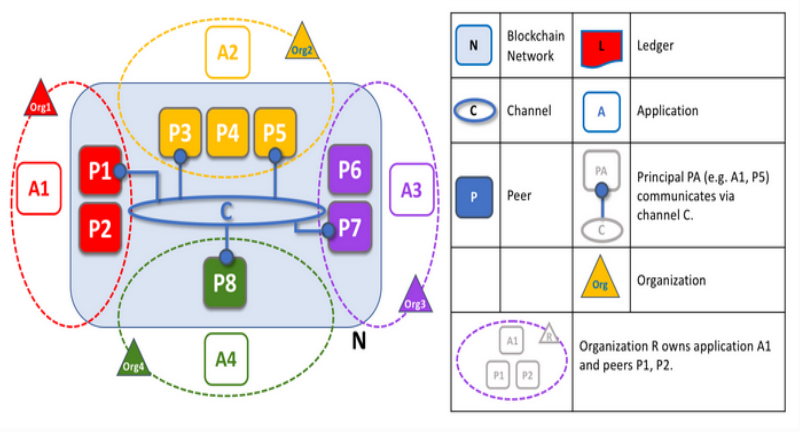
A digital certificate binds the identity of individual/organization to a public key. This acts a confirmation that a person is who they claim to be and allow secure encryption/decryption to take place.

2.5.6 Certificate Authority

A Certificate Authority (CA) is an organization that stores and validates digital certificates. For example, if you wanted to find out the digital certificate of a bank you would go to a trusted CA and ask them for the banks certificate. Allowing the secure transfer of information. Within the network each Peer needs a certificate to insert blocks onto the blockchain.

2.5.7 Applications

An application is an entity that interacts with a Peer node to access the blockchain network i.e. the ledger. The application is able to invoke the methods within the chaincode. Within the scope of this project, the following two functions are used; querying the ledger and inserting a new block onto the blockchain. Figure 6 shows an overview of the interactions.



Peers in a blockchain network with multiple organizations. The blockchain network is built up from the peers owned and contributed by the different organizations. In this example, we see four organizations contributing eight peers to form a network. The channel C connects five of these peers in the network N – P1, P3, P5, P7 and P8. The other peers owned by these organizations have not been joined to this channel, but are typically joined to at least one other channel. Applications that have been developed by a particular organization will connect to their own organization's peers as well as those of different organizations. Again, for simplicity, an orderer node is not shown in this diagram.

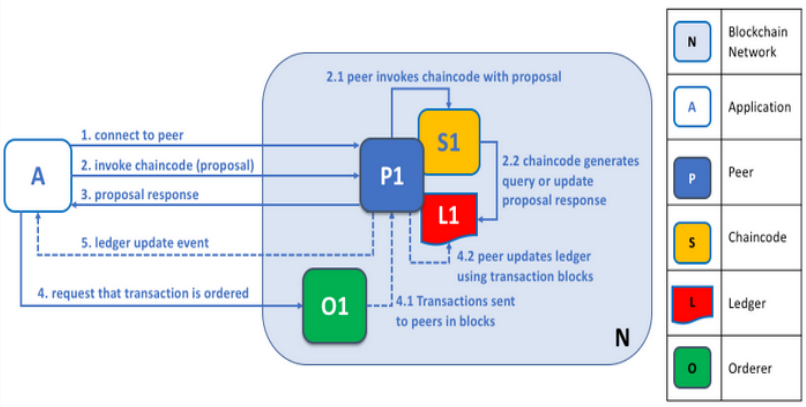
Figure 5: A Simple Organisation Example.[1]

Query

1. SDK connects to a peer node.
2. Access and apply the chaincode/smart contract to query the local ledger.
3. Receive the response.

Inserting Block

1. Repeat steps 1-3 of querying.
2. The blocks are then sent to an orderer.
3. If the transaction is valid, update all instances of the ledger.



Peers, in conjunction with orderers, ensure that the ledger is kept up-to-date on every peer. In this example application A connects to P1 and invokes chaincode S1 to query or update the ledger L1. P1 invokes S1 to generate a proposal response that contains a query result or a proposed ledger update. Application A receives the proposal response, and for queries the process is now complete. For updates, A builds a transaction from the all the responses, which it sends it to O1 for ordering. O1 collects transactions from across the network into blocks, and distributes these to all peers, including P1. P1 validates the transaction before applying to L1. Once L1 is updated, P1 generates an event, received by A, to signify completion.

Figure 6: Querying and Inserting a New Block. [1]

3 Specification and Design

3.1 Overview

The following section describes the architectural details of the system and includes justification of the design choices. The aim of this prototype is to create a record of the companies that have stored/collected an individual's data. This is achieved by storing the information on the blockchain. Each user has a dedicated channel, a new block is added to their ledger each time a company collects data on that individual. The blockchain is shared between the companies that store the data, to ensure no central company has complete control.

3.2 Justification

Blockchain technology has been used because it provides several advantages over centralized databases. Firstly, a blockchain is distributed among several nodes i.e. there are several instances of the record. Therefore, an attacker would need to edit the majority of nodes to control the blockchain - the greater nodes the network has, the more secure it becomes. Within a centralized database there is a single point of failure - compromising it would take less effort than attacking multiple nodes. Secondly, access to the data can be controlled via channels i.e. companies can only access a user's channel if they have been approved. Finally, several companies can view all the data on a channel and there is not a central company that can control it e.g. charging other companies to access the data. There are several disadvantages of using blockchains solutions. Firstly, the high costs that are associated with consensus algorithms e.g. Proof of work. Secondly, the anonymity of participants could encourage nefarious activities within the network. Finally, each block increases the size the blockchain, as the ledger becomes larger, it becomes more difficult to store all the data. In addition, this data is replicated several times, one instance for each ledger. This is an unnecessary precaution against a central point of failure system i.e. it might be easier in a central repository to copy the data for example to three external databases. Note: the first two points do not apply to private permissioned blockchain networks. In conclusion, there are many advantages and disadvantages of using public/private blockchain networks. This technology is still in its infancy and this project attempts to identify new and exciting scenarios where it can be used effectively.

3.3 Requirements

The initial report defined two high-level requirements essential for the application to be operational, they are as follows:

1. Create a decentralised network.
2. Develop the capability to communicate with a node within the network.

These objectives correspond to the following low-level requirements, specifying the architecture of the application as follows:

1. Allow a user to view a timeline of the companies that have collected their data.
2. Allow a company to write "blocks" to a user's timeline specifying that they have either collected this users information or have exchanged the data with a third party.
3. Allow companies/organisations to be added to the network while it is still active.
4. Develop a GUI for the user to interact with the network via an android application.

The completion of these requirements will determine the outcome of the project. The project can be considered successful if the two high level requirements and 75% of the low-level requirements have been achieved.

3.4 Overview of the Application

The entire system is split into two components the first being a decentralized network that stores the blockchain and smart contracts. The second part is an application to interact with the network. For the scope of this project a simplified network is created, consisting of two organizations, each with two peers and a certificate authority:

The NodeJS Software Development Kit (SDK) connects to a peer in the network and is able to call methods in the chaincode. A Representational State Transfer (REST) Application Programming Interface (API) is used to allow a web application to interact with the NodeJS functions. Allowing the use of POST and GET commands to interact with the blockchain network and visually display the information on a Hypertext Markup Language (HTML) webpage.

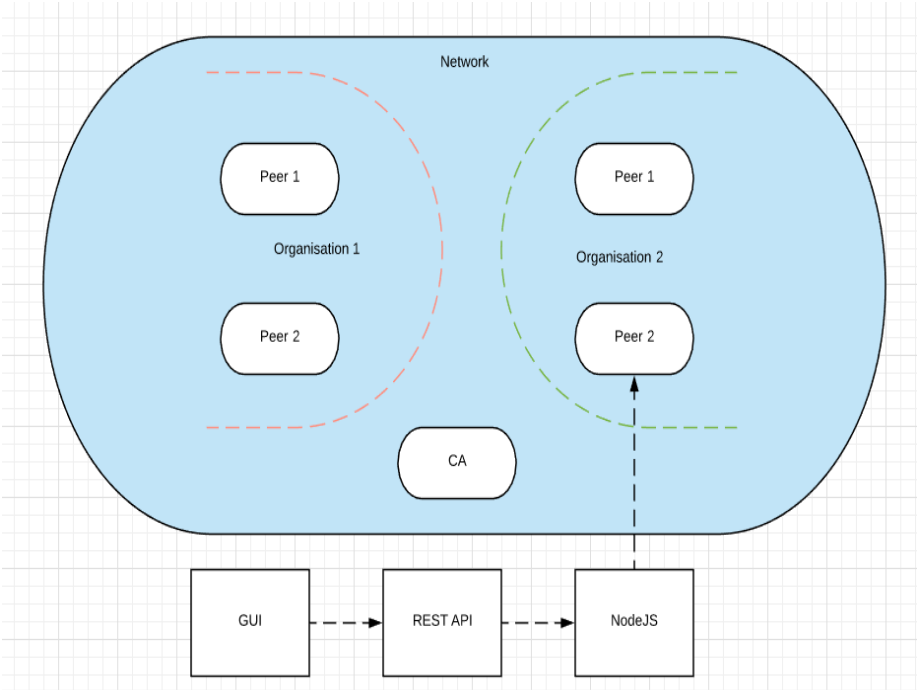


Figure 7: An Overview of the Data Tracking Application Network.

3.5 Blockchain Structure

Each block in the blockchain contains the following information; the date, the name of the organization that has stored the data, a list of the data that was collected and the source of the data i.e. did they collect it from the user or obtain it from another company, helps identify if a company has sold their data to a third party:

To clarify the system does not store the actual data itself, only a record of what data was collected e.g. if User1 gave their First Name "Bob", age "23" and their email address "fakeaddress@gmail.com" to Twitter, the block would appear as the following:

Recording the actual data is beyond the scope of this project and is mentioned in the chapter 6. Future Work. It was not included within the project because a method would need to be created to restrict the level of access to data i.e. ensuring companies had limited access to parts of users data. For example, all of a users data is on the blockchain, a company called Org1 has collected their name and email address, they now have access to the blockchain. Org1 then examines the blockchain and gathers more information about the user without their knowledge. Although, there are security techniques that could solve this problem, to implement them would

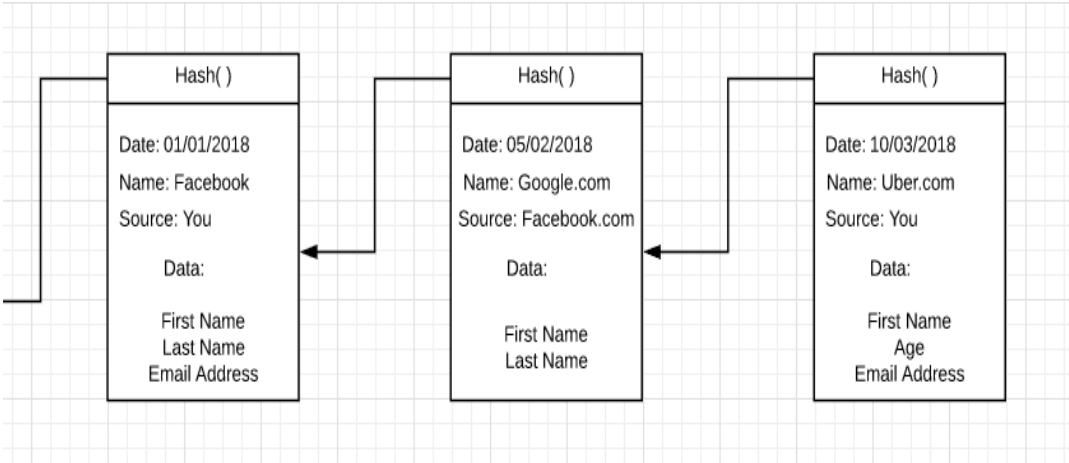


Figure 8: The Data Tracking Blockchain.

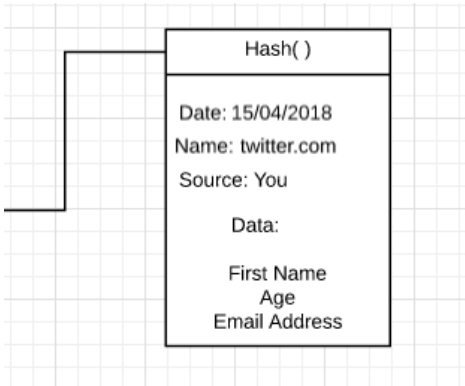


Figure 9: The Contents of an Example Block.

increase the time needed for development which is limited.

3.6 Chaincode/Smart Contract

The chaincode dictates the functionality that is available to the user. There are three functions that need to be developed. The first is the ability to query a single block i.e. the input parameters are a block id and UserID. The function returns the data inside that block. The second, is the ability to query all blocks i.e. parameter is the UserID and it returns the data of all the blocks in the ledger. Finally, insert a block onto the blockchain i.e. the input parameters are; the UserID, the Date, the Name, the Source and the Data. The function inserts the block onto the blockchain and returns a confirmation message.

3.7 Channel

Each new user would have their own channel i.e. a new ledger containing only their data record. The advantage of this is that it is easily auditable, if the system was scaled to find a users data you would go to their channel opposed to a single blockchain with everyone's data, which could be difficult to filter as the number of blocks increase.

3.8 Example Scenario

An EU citizen would register to use the application, they will then be given a unique identifier (ID). Where ever they go on the internet, when they register to a website, they will also provide the identifier. The company who owns the website would then insert a block onto the blockchain detailing the data that they have collected.

3.9 Graphical User Interface Design

The interface design is an important part of the project, but emphasis will be focused on providing the functionality of the application rather than creating an aesthetically pleasing interface. The following are the screens in which a user may access. Note there is not a register screen for an organization because the functionality to add organizations to the network will not have been developed yet.

3.9.1 Login Screen

The login screen is the first screen a user will view. A regular user will be directed to the main page. While an organization (represented by a single user) that collects/stores data would be taken to the insertion page. It also allows a user to register for the application. This can be seen in Figure 10.

3.9.2 Register Page

The register page will ask for an email address that will be used as the username, a password and to re-type the password. This can be seen in Figure 11.

3.9.3 Main Page

The main page shows the users unique ID and a list of all the data that has been collected about them in chronological order. They can then inspect

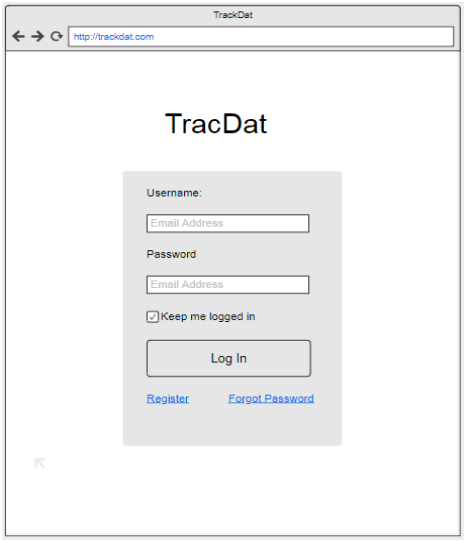


Figure 10: The Login Screen.

each block.This can be seen in Figure 12.

3.9.4 Insert Page

This page allows a company to insert new blocks onto the blockchain. This can be seen in Figure 13.

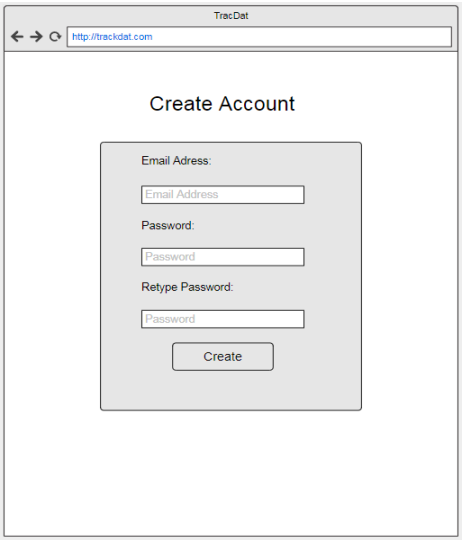


Figure 11: The Register Page.

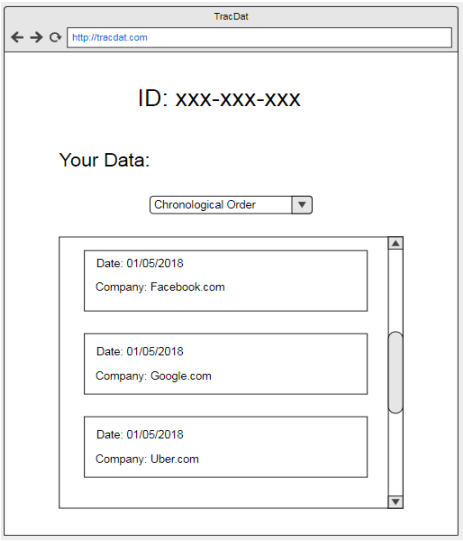


Figure 12: The Main Page.

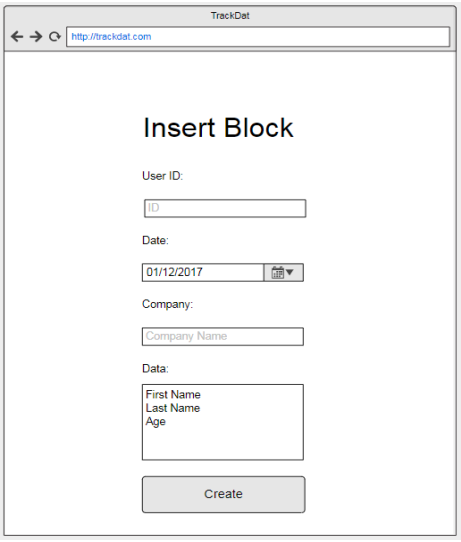


Figure 13: The Insert Page.

4 Implementation

4.1 Overview

The development of the system, followed an incremental approach. The following chapter describes in detail; setting up the network for development, creating the chaincode and block data structure, building the NodeJS functions that interact with Peers, finally constructing the REST API and integrating it with the GUI.

4.2 Setting Up the Network for Development

Hyperledger Fabric uses a program called Docker to create a virtual network on a stand alone machine. Docker uses containerization to create standalone components/packages i.e. it contains everything it needs to execute the contents; code, runtime, system tools, system libraries and settings. Installing and setting up the software, had caused initial issues. The laptop being used for development had the Windows 10 Home edition operating system(OS). Docker requires the Windows 10 Pro edition to be installed. An attempt was made to install an Ubuntu 64bit OS on a virtual machine (Docker requires the 64Bit version for all linux distributions). Unfortunately, the laptop would only allow a 32bit OS to be installed. Finally, there was success after creating a bootable universal serial bus(USB) device with the Ubuntu 64 bit OS installed.

Hyperledger has two example networks, the first is called "Building your First Network" (BYFN). Its purpose is to help users become familiar with the different components and test that everything works. It contained two organizations each with two peers. The second example had a slightly different network topology; it had a single organization with two peers and a CA. The purpose of this network was to show how the user could interact with it using the NodeJS SDK. To show the full capabilities of the data tracking application, the two networks were combined to take advantage of the different functionalities i.e. multiple organizations with a CA and accessing it via the NodeJS SDK.

4.3 Interacting with the network

There were two methods for interacting with the network, the first was via the terminal, to access a client container. This allowed the user to install and instantiate the chain code on the Peer nodes. The second option was to use the NodeJS SDK and create a script to automate the process. Throughout

the development, the second option was used because it was the faster method and changes to the script could be made effortlessly.

4.4 Developing Chaincode

Chaincode is written in GO, it defines the methods that are available to the user i.e. its a go file that contains functions in which the programmer/user is able to call to interact with the network. To create chaincode, the Hyperledger "Shim" needs to be imported into the code, this can be seen in Figure 14. Each function, requires the shim interface and any arguments as a parameter, this can be seen in Figure 15.

```
import (
        "bytes"
        "encoding/json"
        "fmt"
        "strconv"

        "github.com/hyperledger/fabric/core/chaincode/shim"
        sc "github.com/hyperledger/fabric/protos/peer"
)
```

Figure 14: Code Snippet for importing the shim Interface.

```
func (s *SmartContract) queryBlock(APIstub shim.ChaincodeStubInterface, args []string) sc.Response {

        if len(args) != 1 {
                return shim.Error("Incorrect number of arguments. Expecting 1")
        }

        blockAsBytes, _ := APIstub.GetState(args[0])
        //messageAsBytes := []byte("args[0] = " + args[0])
        //return shim.Success(messageAsBytes)
        return shim.Success(blockAsBytes)
}
```

Figure 15: Code Snippet of the Query Function within the Chaincode.

During the design stage, it was stated that three functions needed to be developed. Once the programming started, it was realized an additional function to initiate the ledger was required. The four functions were;

1. Initiate the ledger - Creates several example blocks and inserts them onto the blockchain.

2. Query a single block on the ledger - Takes the block id as a parameter and then calls a Fabric function “getState()” and returns the contents.

3. Query the entire blockchain - The code iterates through each block in the blockchain and formats it into a readable string. The results are returned to the user.

4. Insert a block onto the blockchain - Takes a list of arguments i.e. the block data and uses the Fabric function “PutState” to insert it into the blockchain.

4.5 Developing Node SDK

To interact with the network, the NodeJS enrolled an administrator. This involved generating a digital certificate that was used by the CA, to ensure the network remained secure. The next step was to enroll the user by creating its account. Note: this method was not ideal because the user had the same privileges as an administrator. To call a method within the chaincode, the NodeJS file sends a request detailing the function to be called and any parameters that need to be passed. Figure 16 shows an example for querying the blockchain.

```
var request = {

    chaincode: chainCodeID,
    fcn: queryAllBlocks,
    args: [''],


}
```

Figure 16: Code Snippet for Querying the Blockchain using the SDK.

Similarly, the insert block code can be seen in Figure 17.

```
var request = {

    chaincode: chainCodeID,
    fcn: insertBlock,
    args: ['BLOCK4','15/04/2018','Twitter.com','You','First Name'],


}
```

Figure 17: Code Snippet for Inserting a New Block using the SDK.

4.6 Building REST API

The REST API allows a user to interact with the network via a web application. The "Express" library was used to set up a local webserver."POST" and "GET" requests can be sent to the server and the API responds accordingly. For example, the user accesses the login page. They insert their username, password and click the "Log In" button. The webpage sends a "POST" request to the web server with the login details. The server checks if the user details are valid and then (assuming they are) sends a request, using the NodeJS SDK to the blockchain network, querying all blocks within the ledger. The network returns all the details to the server, which renders the HTML/CSS pages using embedded javascript and displays it to the user.

4.7 Building the Graphical User Interface

There are four webpages:

4.7.1 Log In screen

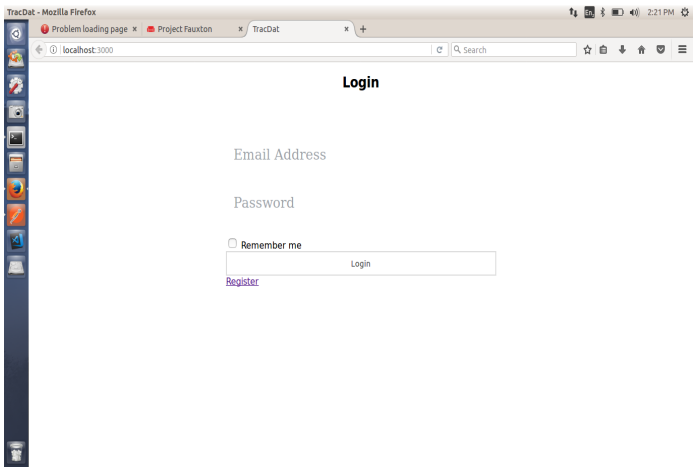


Figure 18: The Login Web Page.

4.7.2 Register Page

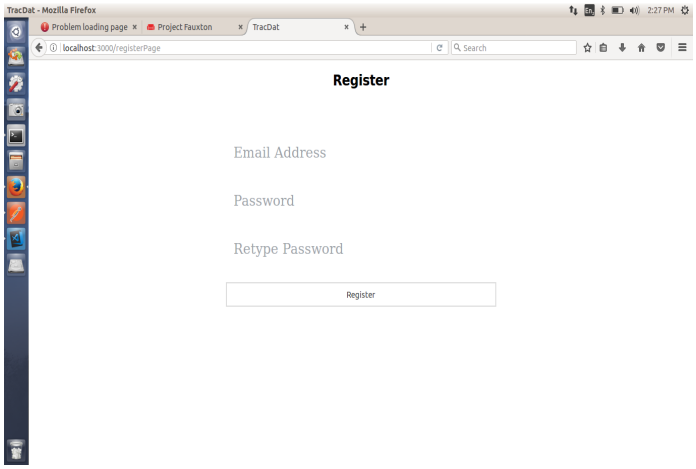


Figure 19: The Register Web Page.

4.7.3 Main Page

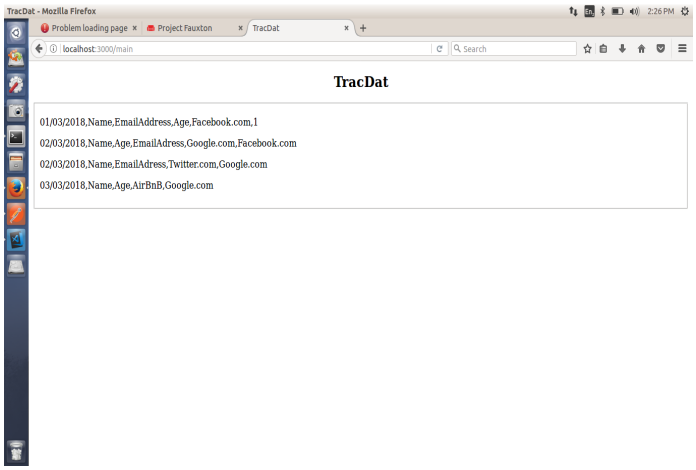


Figure 20: The Main Web Page.

4.7.4 Insert a block Page

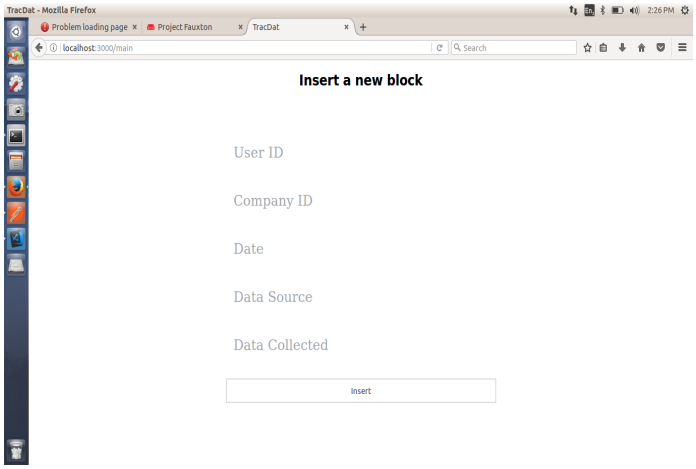


Figure 21: The Insert Web Page.

5 Results and Evaluation

This report stated the system can be considered successful if the two high-level requirements and 75% of the low-level requirements were completed. It is with great pride, I can declare the project a success. A decentralized network was created and GUI was developed to allow communication with a network node. In addition, the application allows the user to view a timeline of the companies that have collected data about them and provide the functionality to insert new blocks onto the blockchain. The ability to add new organizations to the network, while it was still active was not achieved. There was a tutorial to add this functionality using the SDK, unfortunately it did not work when tested. Organizations can be added to the network topologies manually by editing the Docker compose file. For this to work the network would need to be shut down and restarted.

The project shows, that it is possible to track user data - a task no company/organization has publicly attempted, and the blockchain data structure is a valid solution. This application has the potential to have a positive effect for a lot of people. No longer will large corporations abuse the trust that is bestowed upon them. It is understood that if the it was released, several companies may oppose it. The sale of data is a very profitable business model and broadcasting this may appear to cause damages. This project's goal is not to disrupt the capitalist structure of our society, it's to take trust out of the equation. Give the control back to the user and provide the opportunity to create new ventures to enrich everyone lives (detailed in the future work chapter).

6 Future Work

The eruption of public opinion, concerning companies that are storing personal data and applying unethical practices e.g. Cambridge Analytica, is creating momentum for the need of a system similar to that described in this project. Therefore, the development of this system will continue after the authors graduation from Cardiff University. There are several improvements that will initially be targeted. Security is a primary aim, the user and administrator's privileges need to be created and assigned accordingly using security industry practices i.e. clearly defined in the network configuration file. The GUI needs to provide enhanced user experience which it currently lacks. The data that is captured will also be increased to include various other forms e.g. Home Address, Height, Online Preferences, etc. The current functionalities will need to be increased to allow organizations to join the network while it's still active. The system will aim to incorporate other aspects of the GDPR e.g. allowing a user to view what their data is being used for and send a request for deletion to specific organizations if necessary.

A major decision that will need to be addressed is the step to go further than the functionality that this system provides by storing the actual data onto the blockchain. This is discussed briefly in the design section, to create a worldwide blockchain where companies no longer store data locally and instead store it on a decentralized repository shared between everyone. Obviously, restrictions will be in place to limit who can see what data. The benefit globally is the reduction of unnecessary resources to replicate data, which would provide an environmental relief in a massive way. Secondly, it will give individuals complete control of their data, no more data thefts or identity fraud. There will be many great challenges ahead, with some hard work and determination, this project has the potential to change the world.

7 Reflection on Learning

During the year in industry, I was exposed to a large development project with a timeframe spanning over several years. From this experience I understood that, starting the project is difficult. There is a lot to learn and the workload starts incrementing very quickly. So I knew it was important to attempt to make as much progress early on as possible. My approach involved reading the entire documentation for the Hyperledger Fabric framework several times and then attempting to work through the examples. This proved very difficult - my experience with blockchain technology was very limited at the start and the Hyperledger framework is still in its infancy. There was a lot of terminology I did not comprehend and the structure of the framework is very complex. Development is even more so, tutorials and examples are very scarce, limiting the help that could be obtained. I asked Pete for help throughout development, but I always tried to limit the amount of questions asked because I was cautious not to waste his time. I have identified this flaw in previous reports and usually get around it by asking multiple people small amounts of questions. In this situation it was difficult because there were not many people to ask. Multiple issues/errors, occurred throughout the project, I tackled them by trying various approaches and if none prevailed, the I would consult with Pete. I realized my communication needs greater development. Explaining the problem to someone who is not familiar with the exact details of your system is hard, especially when rushing and making mistakes in my language. To overcome this issue I need to slow down, think about what I am going to say beforehand and write down notes.

I really enjoyed controlling my workflow and making major decisions about the project. Beforehand, in placement there was a level of autonomy but not as much responsibility as this project. There was always that fear of not completing the project on time, but from the experience gained on placement, there is never enough time - it's better to make progress rather than making it perfect. Project management skills were really developed and I had a taste of the intricacies of a small scale project, understanding that meeting deadlines ensures the progress of the development.

In addition, a wide range of soft skills were exercised and refined, these included; problem solving, planning and responsibility. I hope to take the experience I have gained and apply it productively to my future endeavors.

References

[1] Hyperledge-fabricdocs documentation, 2018.

[2] Robert C Martin. *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002.

[3] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[4] Viviane Reding. The european data protection framework for the twenty-first century. *International Data Privacy Law*, 2(3):119–129, 2012.

[5] T Sandle. Un thinks internet access is a human right. *Business Insider*, 22(07), 2016.

[6] Paul Voigt and Axel von dem Bussche. Enforcement and fines under the gdpr. In *The EU General Data Protection Regulation (GDPR)*, pages 201–217. Springer, 2017.