A Feasibility Study on Using the Blockchain to Build a Credit Register for Individuals Who Do Not Have Access to Traditional Credit Scores

Bryony Ortlepp

A dissertation submitted to the Faculty of Commerce, University of Cape Town, in partial fulfilment of the requirements for the degree of Master of Philosophy.

February 10, 2019

MPhil in Financial Technology,
University of Cape Town.

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Philosophy to the University of Cape Town. It has not before been submitted for any degree or examination.

Signed by candidate

Bryony Ortlepp

February 10, 2019

Abstract

In South Africa and many other countries, credit registers and credit scores are used to determine how much credit a person can get access to, as well as the interest rate which they will be charged. In addition to this, some companies (such as insurance companies and rental agencies), use this data as part of the process to vet potential clients before allowing them to sign a contract. Part of the problem with this approach is that only certain records are stored on these credit registers. This excludes a large number of individuals, specifically those who are unbanked, those who have not got access to credit from formal institutions or those who do not own property and therefore pay their landlord for utilities.

The purpose of this research is to determine the feasibility of using blockchain to store payment histories from small businesses to give their clients access to a credit record. The case study for this research will look specifically at a business which offers insurance to individuals living in informal settlements. This could be extended to many other businesses who work within informal settlements which allow cash payments on a regular basis for services offered. Shops in informal marketplaces which allow people to take products on credit and only pay later could also be included. By storing these transactions on the blockchain, individuals who would not usually have access to a credit history will have access to records of transactions that they have made and will be able to use these to show their ability and willingness to meet their financial obligations.

This paper provides insight into existing credit registers and the process followed to build an informal credit register on the blockchain. The research covers an investigation into the feasibility of the project and it was found that this could is feasible and could add a lot of value, especially to those who do not have a credit history. There are many considerations, such as speed, security and costs which need to be taken into account, but these are outweighed by the benefits of the blockchain.

Acknowledgements

I would like to thank my brother, Kerren Ortlepp, for all his assistance with this paper and for all of the late night chats and lessons. I would also like to thank Anton and Jenni Ortlepp, Topsy Mokoena and Kyle Lasarow for supporting me in this journey. Thank you to all of my family in Cape Town who made me feel so at home - Karen, Simon, Kirsty and Hayden Weaver, Brina, Dave and Chloe Snyders and Barbara and Roger Barker. Thank you to all of my very special friends in Cape Town who made my year such a wonderful adventure - I will miss you all! Thank you also to my friends in Johannesburg who came to visit and who kept in contact. Thank you also to my teammates in Trading Risk Analytics - it is great to be back working with you again.

I would like to say a huge thank you to my classmates in Project X (Ryan Jacobson, Kungela Mzuku, Aidan Fourie and Kuselo Ntsaluba) for sharing their blockchain knowledge with me, as well as my other classmates for making the year as enjoyable as it was.

Thank you to Sabine Bertram and Allan Davids for all of their help over the last few months and to Co-Pierre Georg for convening our course, helping me come up with my topic and supervising this paper.

Last but not least, I would like to thank Lumkani for their initial assistance in coming up with an idea for this paper.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since its inception in 2008, blockchain has rapidly grown in popularity. While the idea behind Bitcoin, the first blockchain, was to create a decentralised, electronic payment system, subsequent blockchains, such as Ethereum have different capabilities. Ethereum allows for decentralised applications (dApps) to be run on the network through the use of smart contracts. The use of dApps has grown substantially over the years and since it went live in 2015, over 2000 dApps have been built on it. In December 2018 alone, 105 new DApps were released on the platform State of the DApps (2019). With the increasing popularity, many companies and individuals have thought of new applications of this technology, the benefits of which reach far beyond cryptocurrencies. The blockchain is decentralised, transparent and immutable and these key features are likely to make it invaluable in solving some of the challenges faced in the world today.

Some examples of blockchain solutions which are being investigated and, in some cases, implemented already are described below.

Storing medical data: Medrec provides users with the ability to access all of their Electronic Health Records, across different doctors, clinics and regions. The platform does not store the actual data on the blockchain, but rather a pointer to each record with details about the ownership, permitted viewers and the integrity of the data, making it easy for the patient to find all records relating to themselves and share that information, should they wish to. In addition to this, MedRec is able to aggregate and anonymise date, so that in some instances, medical researchers can access data to use in their studies, without compromising the patients' confidentiality Ekblaw *et al.* (2016); MedRec (2019).

Preventing piracy of electronic media: Custos provides their clients with a way to digitally and invisibly watermark media prior to distributing it. In this way, if con-

tent is leaked, it is easy to identify where the file originated from and who leaked it. The watermark is unique to each copy initially distributed and since the original receivers of media will often be competition judges and industry insiders, their reputation will be damaged if they are found to be responsible for leaked content. Custos utilises the blockchain by giving rewards in bitcoin to "bounty hunters". These are people who have installed an application which looks for pirated media and checks for digital watermarks. The user who finds newly leaked content first, gets a reward and details of the leaked file are sent to Custo and then onto their client. The rewards are small enough that the initial receivers of the content are not incentivised to risk their reputation by leaking the file, but large enough that it is worthwhile for the bounty hunters to leave the application running and collect their rewards CustoTech (2018).

Supply chain management for shipping companies: TradeLens is a joint venture of IBM and Maersk in an effort to accelerate the use of technology such as blockchain, improve the efficiency of the shipping industry and lead the way in the digital transformation of the industry. All supply chain entities, including ocean carriers, ports, customs officials and more are allowed to join the TradeLens ecosystem. TradeLens publishes real-time milestones so that all entities involved in a shipment can determine the status of it. The platform also allows for document sharing, which reduces the need for manual paperwork, which is inefficient and difficult to keep track of. The platform can be customised in terms of which entities can see what, based on business needs TradeLens (2018).

Credit scores are calculated and used differently throughout the world, but in many countries, including South Africa, they are used to determine the creditworthiness of borrowers. Prior to accepting a credit application, credit providers check the applicant's credit score as it gives an indication about whether they are likely to be good debtors over time. Once an application is approved, the better the applicants' credit score, the more favourable the interest rates are likely to be Patnick (2017a,b). In addition to applications for credit, credit checks are often done on tenants when they apply to rent a property, especially if the property is rented out by estate agents who offer screening services to landlords Rawson Property Group (2019); Pam Golding Properties (2019). These credit scores are usually provided by credit bureaus. Credit bureaus get data from companies and banks which contain details about how the applicant has managed their debt previously and whether they have missed payments or are blacklisted. This data is used to compile a report on the individual and calculate a credit score. The score range differs depending

on the bureau and the score ranges for a specific bureau are easily found online. The credit bureau charges a fee to compile the report and send it to the requestor, with the applicants permission. The score takes into account the applicants financial history including loans and store credit, judgments made, bankruptcies and sometimes utility bills, as the bills are paid after using the utilities Patnick (2017c)

One of the downsides of using this data is that it is difficult for an individual to get credit without a credit score, but to get a credit score, they need access to credit. Many unbanked individuals or those who have not financed a car or home, are unable to prove that they are trustworthy or able to manage their finances, as traditional credit scores often focus on data from banks and financial service providers. Unlike many traditional credit scoring systems, this paper will explore the use of payment histories from smaller businesses which predominantly serve informal communities to build a decentralised credit register on the blockchain.

Chapter 2

A Review of Current Credit Reporting Institutions

We begin by differentiating between the two main types of credit reporting institutions, namely credit bureaus and credit registries.

Credit bureaus are usually private companies catering to commercial lenders, whereas credit registries are usually public entities, often managed by central banks or bank supervisors, catering to policymakers and regulators. Credit bureaus collect data from retailers, banks and microfinance companies, amongst many others to allow lenders to assess the creditworthiness of individuals. The goal is to collect very detailed information on individual clients and credit bureaus will, therefore, include loans and obligations for smaller amounts than what credit registries would focus on. Credit bureaus will often offer services in addition to the credit history, such as credit scores The World Bank Group (2018a).

Credit registries, on the other hand, focus on collecting information for policymaking and regulatory purposes. In countries where a national credit registry exists, loans above a certain amount, issued by regulated financial institutions need to be registered. Regulators rely on this information to understand the credit risks faced by the credit system and mitigate it through macroprudential regulation and oversight The World Bank Group (2018b).

Financial institutions rely on credit reports provided by credit bureaus to screen loan applicants as well as to monitor the behaviour of their existing loan holders. The borrower usually knows more about their creditworthiness than the lender does, so lenders rely on credit bureaus to bridge the gap. Credit reporting allows individuals to build up a credit history and, assuming a positive history, use that to access credit. Accessible credit bureau data has allowed many households to get

access to credit products which historically would have been turned down as too risky because it offers a form of reputational collateral. In many cases, borrowers do not have access to physical assets which they can place as collateral on loans, but should they default on payments, they know that their credit history would be tarnished, making it difficult or impossible to access credit at reasonable rates in future. It is therefore in the borrowers best interest to keep up with their financial obligations, as defaults or late payments can affect their credit score for years. Experian, one of the "Big Three" credit-reporting agencies which is present in 37 countries reports on late payments for 7 years from the date of the missed payment Experian (2018b).

The purpose of this paper is to create a credit record or informal register, accessible by the individuals themselves as well any lender or institution with whom they wish to share their data. The informal register could be used in conjunction with credit bureau data, where available, for lenders to assess the creditworthiness of clients who ordinarily would not have access to a strong credit record Staten (2003).

2.1 Credit Reporting Around the World

Djankov, McLiesh and Shleifer analysed the results of a study done on private credit in 129 countries in 2003. At the time, 71 of the countries in the sample had public credit registries, and 55 countries had private bureaus. Half of the bureaus in the sample were owned or affiliated with one of 3 big international firms, namely Experian, Equifax, and TransUnion Djankov *et al.* (2005). Countries have differing approaches to credit scoring, and details are discussed below about various countries and their credit scoring systems in comparison to the United States of America Curley (2018). In the USA, an individuals credit score can determine whether they will get approved for credit and, if approved, the interest rate they get on a home loan or vehicle financing. The credit score can also affect the type of credit card the individual gets approved for. When individuals apply to rent apartments or even to calculate car insurance premiums, their credit report is requested and used in the screening process so a poor rating could impact many different aspects of their life.Avery *et al.* (2004)

In the United Kingdom and Canada, the credit systems are similar to the USA and credit scores are based on payment history, the age of accounts and credit utilisation. The three main agencies in the UK are Equifax, Experian, and Callcredit and credit scores from Experian take into account whether the client has added

their name to the electoral register to vote Experian (2018a). The main agencies in Canada are Equifax and TransUnion DebtCanada (2018).
Japan does not have an official scoring system. Each financial institution uses its own methods to determine the creditworthiness of a borrower and they do not share the information externally The Economist (2008).
The Netherlands has the Credit Registration Office (BKR) which was founded in 1965. Legally, lenders must register all loans with the BKR and they are also able to use the system to screen applicants. The register contains a register of unpaid debts which are kept for up to 5 years Stichting Bureau Krediet Registratie (n.d.). Similarly, in Spain, the National Association of Financial Credit Establishments keeps a record of defaulters called the ASNEF File. This record stores bad debts for up to six years, unless the debt is repaid, and individuals on the list will find it very difficult to get access to credit Banco Bilbao Vizcaya Argentaria (n.d.). Spain also has a central credit register which tracks loans and other risks that financial institutions have with each client Banco De Espana (n.d.).
In France, financial institutions have proprietary databases of negative and positive history, as credit bureaus are illegal in France. To get a home loan an individual requires three months of bank statements, proof of income and marital status and at least a 15% deposit. If all of these conditions are met, they will likely be considered creditworthy.
The main credit agency in Germany is SCHUFA. SCHUFA is the only credit bureau which offers positive credit information in addition to details about open accounts and unpaid debts, fines and bills. When an individual applies for a lease, a bank account or a phone line, the company will check their SCHUFA record to determine their creditworthiness. Negative records can affect an individuals score for up to three years after the debt or fine has been repaid. SCHUFA (n.d.)
Recently, Australia introduced a new comprehensive credit reporting system. Previously, credit providers could disclose credit information on a voluntary basis, but the new bill makes this reporting mandatory. Credit reports and scores will now include positive financial history such as loan and credit card repayments, in addition to the data which used historically. Previously, credit records only took negative information such as court judgments against the individual, bankruptcies, defaults, credit enquiries and credit infringements into account. The requirement that credit records include positive data gives individuals a chance to redeem themselves from a poor credit history by showing that they have repaid the debt and made consistent repayments on other accounts. Australian Government (2018) In South Africa, the four largest credit bureaus are Experian, TransUnion, Compuscan and XDS. Each bureau compiles credit reports on individuals based on data collected mostly

from lenders. By law, individuals are allowed to access their credit report free of charge from each institution once a year. Credit checks are done on an individual if they want to sign a lease or apply for a loan and the cost of accessing the report from the bureau often falls on the individual Clearscore (2017).

While there are many differences in the use of credit registers and the calculation of scores in different countries, a number of them include both positive and negative data. One could conclude that those who do not already have access to credit or do not own property often find it difficult to build up a good credit score. A study in 2015 found that in America alone, around 26 million people did not have credit scores and were considered credit invisible. An estimated 3 billion people around the world did not have access to a credit card and hence were unlikely to have access to a credit score Brevoort *et al.* (2015); Jesse Leimgruber and Backus (2018). Individuals who need to borrow money without a good credit score are often forced to pay significantly higher interest rates than those with a good score. These loans may come from banks or microlenders who will charge a higher rate than for their less risky clients or from predatory lenders like loan sharks, who often provide credit illegally at exorbitant rates. Predatory lenders in South Africa charge an average interest rate of 30% interest per month, with some being known to charge up to 50% per month Siyongwana (2004). While it is imperative that lenders protect themselves against high-risk individuals and are compensated sufficiently for risk of loss, it is also essential that individuals have an opportunity to prove that their risk profile is reasonable. Many of the current systems exclude individuals because they do not make use of institutions and businesses who traditionally supply credit information to the credit bureaus and thus lenders are often not provided with enough information to make a fair decision.

2.2 Building a Credit Register

In his paper, Asif Mian describes four steps in designing a credit registry system, based on World Bank recommendations. These steps cover data collection, data validation, data dissemination, and data usage. Data Collection: As discussed in the previous section, some registries record both positive and negative information and, in some cases, delete the negative information after a certain period. Details of the individual such as name and location are needed to build a traditional credit registry in order to track whether there is a higher concentration of risk in certain areas or sectors. Both positive and negative data should be collected from lenders in the financial sector. Positive credit data should include total credit issued, credit

outstanding, loan maturity and collateral, where applicable. Negative information includes legal judgments against the individual, missed and late payments and the number of defaults on the account over different periods.

Data validation: Once collected, the data needs to be validated to ensure that any errors are picked up and rectified. Random audits should be done to ensure that the quality of the data remains high. Audits also ensure that the transparency and reliability of banks financial data is improved.

Data dissemination: The credit registry must have clear rules on the distribution of data and who can access it. It is crucial that those who contribute to the data are certain that the data will only be used for legitimate purposes, as the data provided is very sensitive. However, in order for the registry to be extensive, it needs to include contributions of data from various entities. These entities are often incentivised to contribute to the registry by allowing them to access the full data set, so precautions need to be taken to ensure that they will not misuse the data.

Data usage: Credit registries are traditionally used by regulators to be aware of where risk is concentrated in the system, by the banking sector for risk-management purposes, as well as by policymakers to make more informed choices. Mian (2012); The World Bank Group (2011)

2.3 Decentralised Credit Registers

With the steady rise in popularity of the blockchain, thousands of apps have been built on different platforms, and naturally, there are some which also aim to build credit registers, as the shortfalls of the existing, centralised ones are a frustration to many. The main critique of the existing credit registers is that individuals do not own their data and often have to pay to access their credit history. In some cases, the history is inaccurate, and the individual has to deal with the ramifications of the errors and is responsible for fixing them. Three credit registers which currently or will utilise the blockchain to try to improve on the traditional credit reporting process are described below.

Bloom: Bloom offers an Ethereum and IPFS based credit register. IPFS is a file storage system, described in detail in section 3.2.2. Bloom aims to address the many problems that affect current credit records, such as:

- An individual has to build up a new credit history if he moves countries, as his credit history is not transferred across borders.
- When assessing creditworthiness, current credit systems rely on an individual's historical data which is unavailable if he does not have any existing credit.
- Lenders who do not have access to reliable credit information, especially in under-developed financial markets, are unlikely to give credit, making it even more difficult for borrowers to get access to credit.
- That identity theft being a real risk since there is so much personal data stored by credit bureaus. Recently, hackers stole the data of an estimated 143 million Americans in the Equifax hack.
- Lastly, they wish to make the credit scoring system more competitive as only a few big players in the credit reporting industry are responsible for scoring a large portion of the worlds population, at present.

Blooms protocol relies on three parts which are Identity Attestation, Credit Registry, and Credit Scoring. Bloom will create global identities for users, which contain only information that the user wishes to share. These IDs are linked to current and historical debt obligations, stored on the Credit Registry. A credit score is calculated based on various pieces of information. Bloom aims to allow external parties to vouch for the accuracy of an individuals details. These external parties could include friends and family of the individual or organisations such as credit bureaus who already have information about the legal standing and creditworthiness of the individual Bloom (2017).

Pave: Pave was founded in 2012 and allows credit to be provided through their platform, in addition to building a credit register. The credit register platform that they created takes into account traditional credit data, as well as allows users to upload their data such as utility bills, proof of rental payments and other documentation which traditional credit reports would not ordinarily include. Further to this, social data from Twitter and Facebook can also be uploaded. In order for an individual to update their data on the platform, they need to get a Global Access Token which is issued for free. If a user wants to read data, they need to purchase one of the tokens and Pave keeps 1% of money transacted through the system Pave Inc (2017).

Polish Credit Office: The Polish Credit Office has decided to partner with Billon to utilise blockchain technology to store sensitive customer data. The Credit Office currently tracks more than 140 million credit histories of over a million businesses and 24 million individuals. The system will store banking records, insurance claims, loan agreements, and phone bills on the blockchain and will be fully compliant with the European Unions General Protection Regulation (GDPR) Prisco (n.d.); Antonovici (2018).

The blockchain could potentially solve the issues faced by existing credit registers, revolutionising the way in which credit history is collected, stored and utilised. Individuals will be able to own their data, and interactions with lenders will be traceable and transparent. These features will be discussed further in the next chapter.

2.4 A New source of Data in the South African Context

As discussed in previous sections, there are a large number of people who do not have a credit history and hence find themselves unable to get access to credit when they need it. Most credit bureaus get data from banks, which do not have data on unbanked individuals. Bureaus also sometimes get information from the municipality in the form of utility bills which can serve as proxies for ability and willingness to pay, because customers are billed on a regular basis, for utilities which they have already used Tobias Baer and Schiff (2013).

In South Africa millions of people live in informal settlements, many of which do not have easy access to banking facilities or credit, resulting in them being unable to build up a credit history. Further to this, individuals often do not receive utility bills, either because they do not own a property or the utilities are not available in the settlement on a post-paid basis. If an individual needs to borrow funds and manages to access a bank or a lender, he is likely to either be turned away as there is no credit record available or be charged exorbitant interest rates to compensate the lender for the additional risk taken on. While the individuals living in informal settlements may not have access to formal credit or get utility bills which they pay diligently, they may have regular financial obligations to smaller businesses which cater to their needs. Potentially, this data could be used as a proxy for the individual's creditworthiness.

Lumkani is an example of a business which provides services to customers who live in informal settlements in the form of fire insurance and, recently, funeral

cover Lumkani (2018). Importantly, Lumkani understands the communities that they work within, and they are aware that not all of their clients have bank accounts, although they still want to get insurance for a monthly premium. Lumkani has opted to hire agents who live in the communities they serve and collect the monthly premium from individuals on a monthly basis. These clients can pay with cash or cards, and the payment is recorded on a tablet with the details being sent to a server and stored. The fire insurance is thus still accessible to individuals who do not have a bank account or smartphone. These customers may not get billed for utilities or have a home loan, credit card or another traditional form of credit which they are paying back. However, if they pay Lumkani and other similar providers on time every month over an extended period, it shows that they are reliable and that they are able and willing to make payments on a regular basis.

Although this paper focuses on the concept that Lumkani's data could be used to build a credit register, there are conceivably many more businesses with similar payment histories which could be included on the platform. The intention is to have many small businesses who service informal settlements adding data to the credit registry. Over time, as the registry grows, it could prove to be invaluable to those who wish to lend to this sector at more fair rates than those charged by payday loan providers and loan sharks. This registry is likely to mostly include relatively small, frequent payments which individuals have made so the intention is not for the registry to be used to prove that an individual can afford something as substantial as a home loan. It could, however, supplement information, such as payslips to show that the customer has proven to be a good payer in the past and can manage obligations which they have made.

There may also be businesses in the community who want to subsidise specific projects for example, Lumkani may want to assist some of their clients in further reducing the risk of fire by giving them money to build a brick home. Rather than donating the full amount, they could use the registry to determine which of their clients would be most likely to pay the money back, based on their credit history, if half of it were given as a loan with low interest rates. Once the loan portion has been repaid, Lumkani could give it to another deserving customer. In this way, they can assist the communities that they are so passionate about and, potentially, in the long run, save money in insurance payouts because those areas in the settlements that have fire detectors and brick homes will be even less susceptible to fires.

Further to this, an extension could be that the money could be transferred directly

to trusted service providers, rather than the individual themselves. For example, if Lumkani wanted to loan money to someone to build a brick home, it would be important that the person or business who does the building knows how to build a structurally sound brick home with the right materials. Lumkani could agree to transfer the money directly to one of the reputable suppliers within the community to ensure that the money is well spent.

It is worth noting here that legally, only registered credit providers can extend credit in South Africa. The businesses wishing to loan money to their customers would probably have to register or partner with a registered credit provider, but the legal side of lending money is out of scope for this paper.

Chapter 3

A Case for Blockchain

With many great technologies out there, some older and more established than others, there initially seemed to be many options for platforms to use to build a credit register with payment records from multiple businesses who predominantly service informal settlements. This project requires a platform that allows multiple users to write to it, but also allows many other users to view their data and permit others to view their data. Further to this, it is imperative that the information cannot be tampered with retrospectively and that it is possible to audit the data and identify which business uploaded it. The blockchain stands out as a potential candidate, as it contains all of these features.

The blockchain is a shared ledger and each network participant has access to an up to date, validated instance of the ledger. IBM and Maersk have researched using the blockchain in supply chain management as network participants which include the supplier, logistics company, port authority, and others, can have equal visibility of all activities. Each participant can also upload documents or details onto the blockchain. Currently, copious amounts of documentation are required for shipments, as the goods move across borders and there are multiple participants responsible for the successful delivery of goods. Moving this process onto the blockchain could increase the efficiency of it as well as add a level of transparency which has, until now, been lacking. This is just one practical example of how the blockchain is set to revolutionise processes where multiple participants are involved at different stages IBM Blockchain (2017); Lal and Johnson (2018).

The blockchain is immutable, meaning that data cannot be changed once it has been written to it. It is decentralised, so no one authority owns the data. It is fully auditable as the transactions are stored on it forever and are traceable. It also uses cryptography to ensure that it is secure. There are different types of blockchains, discussed below. Some of the defining characteristics of a blockchain are whether

it is public or private and also which nodes or peers on the network are allowed to decide on which data gets accepted onto the blockchain. This is known as the consensus process. A consensus process is needed in order to determine which blocks should be added to the blockchain next. Since there is no central node in a blockchain, multiple nodes on the network need to agree that a block contains valid transactions. Proof of Work is one of the algorithms which requires nodes to compete to solve a computationally expensive mathematical problem. Once a node has a solution and proposes the new block to the network, the other nodes on the network confirm that it is correct and the new block gets added to each of their blockchains. Alternatively, some blockchains use Proof of Stake which requires that those validating blocks own an amount of currency on the network, rather than having to perform computations. This is more energy efficient than Proof of Work and assumes that nodes which hold currency on the network are less likely to attack it. There are other consensus mechanisms which are used, but Proof of Work and Proof of Stake are the most common currently. Zheng *et al.* (2017)

Permissionless or public blockchains, such as Bitcoin and Ethereum, allow for any peer to join the network at any time. The peers are allowed to read or write to the network. Permissioned and partially permissioned blockchains only allow specific users to see data and only users who are certified are allowed to join the consensus process to validate transactions. Since there is no limitation on which nodes can join a public blockchain, malicious nodes could be present, but, with effective consensus mechanisms in place, it should be nearly impossible for them to take over the network. Permissioned blockchains assume that the other nodes are known and that they are trusted. Zheng *et al.* (2016)

3.1 Determining the Type of Blockchain: The Wüst and Gervais Model

The Wüst and Gervais model Wüst and Gervais (2018) provides a method of determining whether or not blockchain can and should be used for a specific application, as well as whether it should be public or permissioned. The process begins by determining whether we want to store data and whether there will be multiple writers of the data. In our case, the answer is yes to both of these questions. We want to store payment history, and we want multiple businesses to be able to add this data to the database. Further to this, we want all individuals with payment histories loaded on the blockchain to be able to access their information, as well as give permission for other specified users to see their payment histories.

The next consideration is whether an always online Trusted Third Party (TTP) could be used instead. A TTP is an entity which is trusted by two parties who are interacting with each other, to facilitate their interaction. Since both parties trust the TTP, they do not need to trust each other. An example of a TTP is Thawte Thawte (n.d.). In building a credit register, we want the different businesses who own the data to store it on the blockchain and be responsible for it. Including a TTP just adds a middleman and an extra step to the process.

The last consideration is around the writers of the data. If all the writers are known, then blockchain is not necessary. In our case, although the businesses will have been vetted before being added to the platform, the customers on the platform are not all known. The customers are allowed to write to the blockchain to permit other users to view their profiles. Further to this, they must be allowed to load documents or details in the case of a dispute. Therefore, not all writers are known or trusted which leads to the conclusion that a permissionless blockchain is the best approach. Data on the blockchain is still encrypted, and through the use of smart contracts, it is still possible for data to only be visible by specific nodes.

3.2 Other Applications of Interest

3.2.1 uPort

uPort provides tools and protocols to make it easier to create decentralised applications in a user-friendly manner UPort (2018). uPort allows the user to create a self-sovereign identity, which is a digital identity owned and managed by the user. Other users or systems can only access information such as name or ID number if the user permits it. UPort allows a user to store all of their identifying information on the platform so that they do not have to send all of that data to other platforms. Developers of new apps can integrate with uPort to get the credentials that they require, with the users permission. uPort will share the minimum amount of data so, for example, if an app only needs to know that the user is over 18 years old, uPort can confirm that the user is above that age, without having to provide their date of birth or exact age. The users data is not stored on the apps that they use, and they are in control of who can access the information through uPort.

From the clients perspective, if a website or application is integrated with uPort, they merely have to give uPort permission to provide the website with the information it is requesting. The client scans a QR code in the support mobile application

and agrees to share specific information with the website requesting it. It is easier and quicker than standard onboarding processes where clients are required to populate a form with the required information manually. From a developers perspective, accessing and verifying user details is easy as they do not have to write code to take in all the information. In addition to this, once initially set up in a project, uPort can call Ethereum smart contracts within the code and automatically request the user to approve it.

3.2.2 IPFS

IPFS stands for Inter-Planetary File Services and is a peer-to-peer protocol which allows for distributed storage of files - forever. It offers many features such as peer-to-peer content delivery, deduplication of documents and decentralised archiving of files. By providing users with a hash once they have uploaded files, it ensures that they can quickly retrieve their files or share them with others. In order to access the file, the user pastes a URL into their browser followed by the hash provided when the file was uploaded. IPFS works well with blockchain as one can upload files onto IPFS and store the hash on the blockchain. Both the transaction on the blockchain and the IPFS hash are immutable, and the transaction is timestamped, so although the data is stored elsewhere, it will still be available and auditable. Keeping in mind the bandwidth needed by blockchains and the various transaction fees charged by many of them, IPFS could be instrumental in bringing down the costs associated with using the blockchain for decentralised applications IPFS (2018).

3.2.3 Infura

Infura offers tools which provide access to Ethereum and IPFS securely and reliably, for free. Further to this, it promises to ensure that a developer's decentralised application on Ethereum will be scalable. Infura makes accessing the Ethereum network and IPFS much quicker, and developers do not need to run an Ethereum or IPFS node to gain access to these platforms Infura (2018).

3.3 Which Blockchain

Bitcoin and Ethereum are currently the most well-known blockchains, but there are many different blockchains which could be used for this project Bartoletti and Pompianu (2017).

Bitcoin was the first decentralised cryptocurrency and the blocks store records of currency transactions which are made on the platform. Bitcoin features a scripting language, but it is non-Turing complete and is limited to basic operations. Further to this, many of the nodes on the Bitcoin network only process standard transactions which do not contain these operations. Bitcoin uses the Proof of Work consensus mechanism to verify blocks which is very computationally expensive Nakamoto (2009).

Ethereum also relies on the public blockchain and uses the Proof of Work consensus mechanism currently. However, it is likely that Casper, a Proof of Stake protocol, set to be released in mid-2019, will allow Ethereum to scale more efficiently. Ethereum allows users to create smart contracts which can be coded to do many things, including storing money and sending or receiving Ether, the currency on the Ethereum blockchain, to other users or contracts. The smart contracts can be written in Solidity and then compiled into the bytecode language required. There are numerous tutorials, demos, and testing frameworks online to assist developers in coding these smart contracts. Transaction fees are paid in the form of gas and are calculated based on the different operations within each function Buterin (2014).

Quorum is a distributed ledger protocol based on Ethereum and built by JP Morgan. It offers a permissioned blockchain with the promise of high speed and high throughput JP Morgan (2016).

Stellar is a payment transfer protocol whose primary focus and capability are cross-border payments. It does have the capability to be a development platform for basic smart contracts, mostly linked to payment transfers. Stellar has its own consensus mechanism called the Steller Consensus Protocol (SCP) and does not have a specific language for smart contracts to be coded in. Stellar is both faster than Ethereum, and charges lower transaction fees Stellar (2018); Mazieres (2016).

Hyperledger Fabric is a permissioned blockchain solution designed for businesses. Various consensus rules are supported, depending on the size of the network. Hyperledger Fabric recognises the fact that even when looking at one transaction, not every participant in the transaction needs to or should know every detail. It is designed such that each nodes ledger is only updated with the transactions which apply to that node Hyperledger (2018).

As discussed earlier in this paper, a public blockchain is best suited to this project,

which excludes some of the blockchains mentioned above. While each of the remaining blockchains has something useful to offer, Ethereum was chosen for the experimental phase of this project. Ethereum is currently one of the more established platforms and has a large community of developers, relative to the other blockchains. There are thousands of decentralised apps of varying quality on the Ethereum blockchain and many resources and products available to assist in building on the network. One of the most useful resources in testing decentralised applications is Truffle Suite. It provides a testing framework and a program called Ganache which creates a personal blockchain for testing purposes Truffle (2018). Infura which was discussed earlier in this chapter is built for use with Ethereum, making it easy to scale the project at a later stage. There are also multiple future extensions to the project which Ethereums smart contracts would cater for. An example of one of these extensions is discussed in 2.4, where businesses could potentially loan money to customers through the platform for a specific project, such as building a brick home. These businesses could transfer funds in Ether directly to the builder that the customer chooses from a list of reputable builders, ensuring that the job is done correctly and the customer still has the freedom to choose the builder they like.

Chapter 4

Experimental Methodology

In theory, we should be able to build a credit registry on the blockchain which could store credit data from multiple businesses, allowing them all to write to the registry at the same time and allowing the customers to own their data and share it when they want to.

There were multiple tutorials online, and the coursework in this degree gave us a basic introduction into coding in Solidity as well as brief introductions to JavaScript and HTML. There is not much literature on Solidity yet, and there are not many established best practices to follow, but there is a large community of developers who are willing to give guidance when asked.

The smart contracts were coded in Solidity and tested using the Truffle Suite. Truffle allows tests to be written in JavaScript and uses the Mocha testing framework and the assert module from Chai to ensure that the code runs as expected. Ganache was used to run a personal blockchain to use for testing, and MetaMask was used to interact with the accounts in the browser.

In order to see whether it would be feasible to use the blockchain to build a registry of payment histories, the aim was to build a minimum viable product to demonstrate how it would work. The platform created is called CreditRegister.

As with building any program, there were numerous considerations and design decisions to be made. Often the most straightforward approach was taken since it was a minimum viable product and, should this solution go into production at a later stage, some changes will be required. Functionality was the top priority for the minimum viable product as this directly links to the feasibility of the project. The front-end display is a simple browser application where the aesthetics were not the focus, as the look and feel of CreditRegister can be improved at a later stage

without impacting the outcome of this paper.

One of the big decisions taken at the start, as discussed in 3.3, was that Ethereum is the blockchain of choice, at least for the prototype. Based on the Wüst and Gervais model, a public blockchain was chosen, as we do not know all of the customers who will be added to the platform.

4.1 Front-end Overview

CreditRegister is a browser application which will link to the blockchain. The web-page contains four portals, described below. There are checks done in the back-end, and the functionality of each portal is only accessible to specific wallets.

4.1.1 Master Portal

Only CreditRegister can use the functionality on the Master Portal, where businesses are added to the platform. As an example, CreditRegister could load Lumkani on the platform by adding the business name and wallet address, thereby deploying a new business contract onto the blockchain. The business wallet address is automatically set as the owner of the deployed contract. Thus, CreditRegister can add businesses but is unable to perform any functions on behalf of them. Refer to Figure 4.1 for further details.

4.1.2 Business Portal

Only the owner of a Business contract (i.e. business wallet) can use the functionality on the Business Portal. Businesses can add customers to the portal, as well as upload invoices and receipts for the customer which in turn updates the customers balance.

The business contract stores a mapping of the customer wallet to customer balance. Initially, the balance is loaded as zero. The same customer can be added to multiple business contracts as it is likely that they will interact with different businesses on the platform. The only customer information stored on the business contract is the wallet address and balance, so it was unnecessary to create a customer contract for the prototype. Refer to Figure 4.2 for further details.

Once a business has loaded a customer, it can upload invoices and receipts which will contain the details of the transaction. The documents are loaded and stored on IPFS, and only the IPFS hash is stored on the business contract, which reduces the amount of data stored on the blockchain. Refer to Figure 4.3 for further details.

The customer balance stored on the mapping is updated when a new document is loaded. The latest balance is stored and always accessible by looking it up on the mapping table. The customer balance shows how much the business owes to the customer, so an invoice decreases the balance while a receipt increases it.

4.1.3 Customer Portal

The Customer Portal displays all documents linked to the current wallet, ensuring a customer can only see documents and details linked to their wallet.

The Customer Portal includes the customers credit score, document details and the functionality for the customer to permit external parties to view their history for a specific amount of time. Refer to Figure 4.4 and Figure 4.5 for further details.

Currently, the score is the sum of the negative balances across all of the businesses. A negative balance implies that the customer owes the business money, so the credit score shows how much money the customer owes in total to businesses on the platform. The overall score will be negative or zero, and the closer to zero the score is, the better.

The calculation excludes positive balances which are amounts where businesses owe the customer money, as, from a credit perspective, showing offset between independent businesses does not provide useful information. For example, a customer could pay their utility bills three months in advance and have a balance of R1,000 with the municipality. However, they may be overdue on two accounts of R500 each from other businesses, so the balances for the two businesses would be negative R500. If all three amounts are taken into account, the score would be zero, implying a perfect score. However, the score needs to reflect that the customer is overdue on two accounts, which is the case if the score calculation excludes the positive balance of R1,000, leaving a score of negative R1,000.

The current credit score calculation is very simplistic and aggregates the payment history available. Should this solution go into production, other calculations could be considered, such as giving higher weightings to more recent transactions and

overdue accounts. This calculation was out of scope when building the minimum viable product, and the prototype shows that it is indeed possible to access all of the balances and use them to generate a score.

The customer can view all of the document details which have been loaded for their wallet by various businesses and can use the IPFS hashes to access the actual documents. If this were to be implemented into production, there would be a Dispute button added for each document, where the client would be able to raise any issues they have with businesses that have invoiced them. The dispute functionality was not implemented in the minimum viable product but would work in a similar way to normal document processing, with either the customer or the business owner being able to add details to the dispute table. An example of a dispute document might be proof of payment if the customer has one or a letter stating why the invoice should be excluded from their record. The business will then see the dispute and rectify the situation.

4.1.4 Viewer Portal

The Viewer Portal allows external users to view payment histories and credit scores, subject to having permission from the customer. The viewer gets permitted by the customer until a specified date, so a check is done to ensure that the correct account is logged in and it has permission to view. The view currently displays the same history as what the customer would see, excluding the IPFS hash of the document. It would also include details of disputes so that the viewer is aware if anything has been loaded incorrectly by the businesses and can make an informed decision about the credit history. Refer to Figure 4.6 for further details.

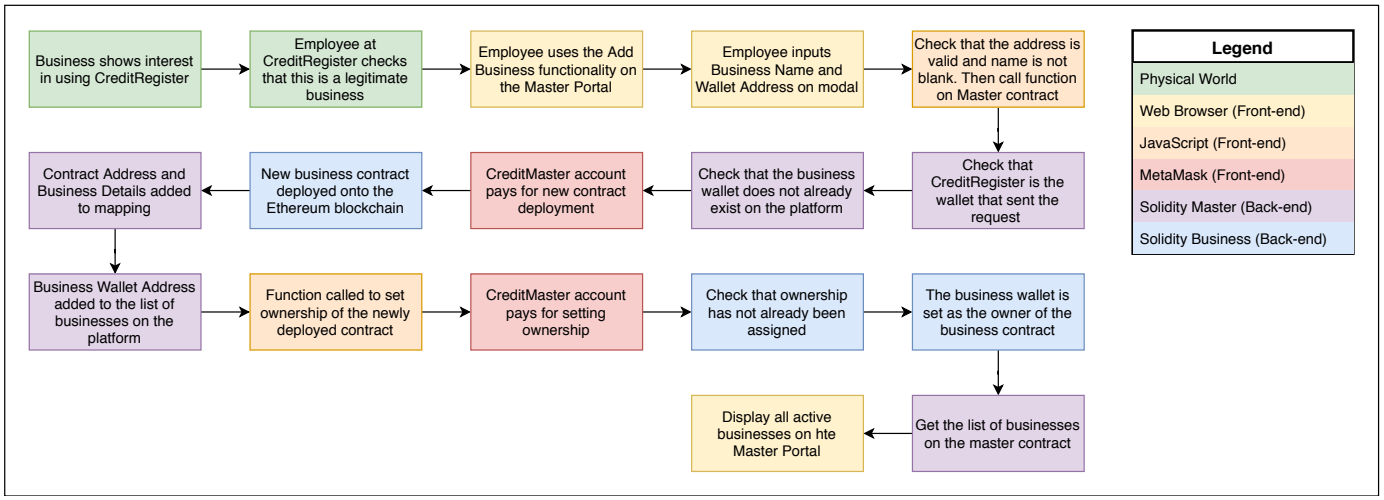

Fig. 4.1: Flow Diagram of Adding a Business to CreditRegister

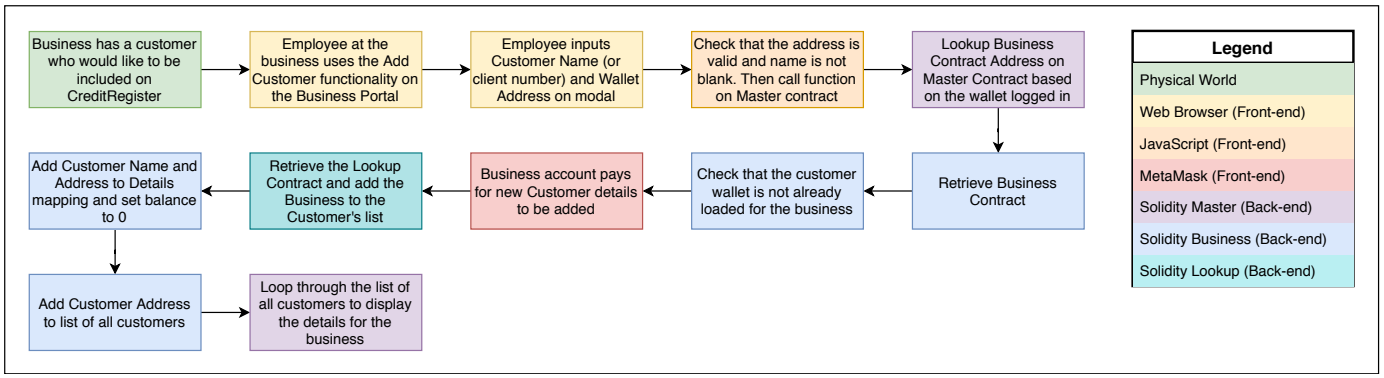

Fig. 4.2: Flow Diagram of Adding a Customer to CreditRegister

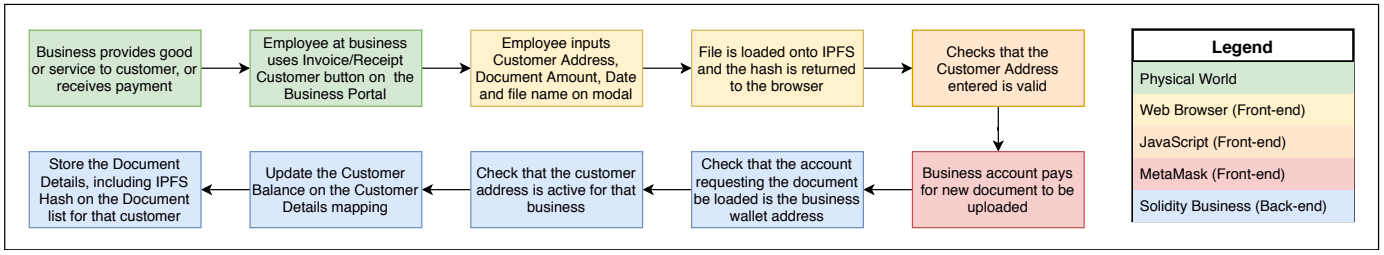

Fig. 4.3: Flow Diagram of Adding a Document to a Customer Record

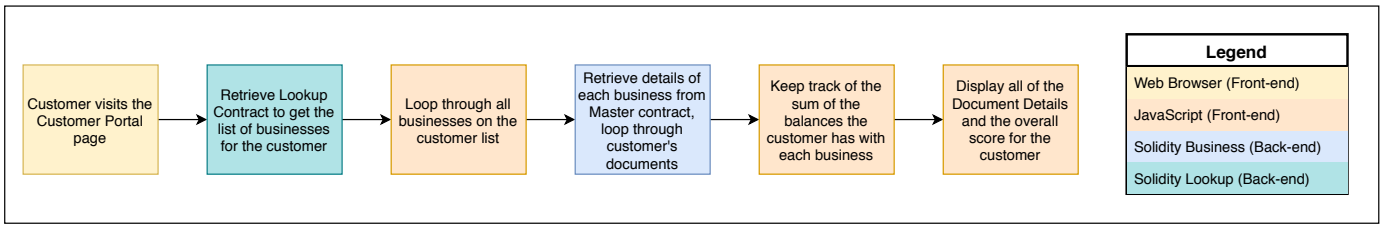

Fig. 4.4: Flow Diagram of a Customer Viewing Their Profile

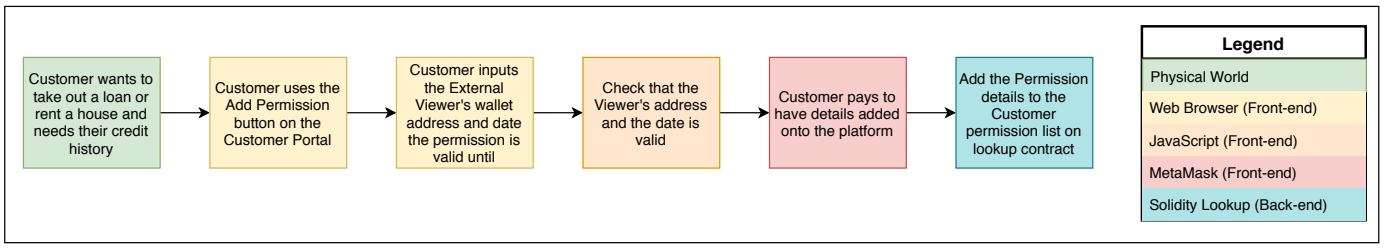

Fig. 4.5: Flow Diagram of a Customer Giving an External Viewer Permission to See Their Score

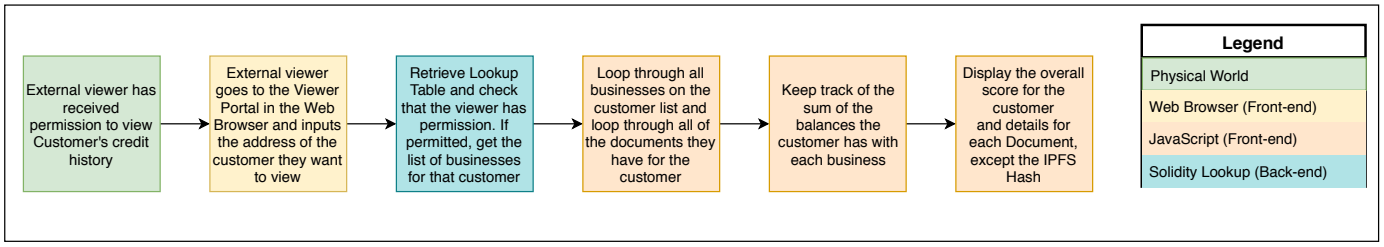

Fig. 4.6: Flow Diagram of an External Viewer Viewing a Customer's Profile

4.2 Back-end Technical Overview

The following section documents the technical details of the implementation of CreditRegister on the Ethereum blockchain. The platform uses three types of smart contracts, namely a Master Contract, a Lookup Contract, and multiple Business Contracts.

4.2.1 Master Contract

The Master Contract is the first contract deployed and is used to manage the platform and deploy Business contracts.

owner: The Master Contract owner is CreditRegisters wallet.

allBusinesses: An array of all of the business addresses loaded on the platform. This is a private array so external accounts cannot view or add to the array, even if they attempt to build functions to do so maliciously. Functions on the Master Contract which can update or retrieve information from this array only work if the request comes from the owner of the contract (i.e. CreditRegister).

businessWalletAddressToDetails: A mapping of the business addresses to their details on the platform including Name, Contract Address and Active flag. Similar to allBusinesses, this is private, and only CreditRegister can add mappings.

addBusiness: A function which takes in the Business Name and Business Wallet Address and checks whether the wallet sending the request is the Master Contract Owner (i.e. CreditRegister). If the request came from the correct account, a new Business Contract is deployed. The Business Wallet Address is added to allBusinesses and the details of the deployed Business Contract are added to businessWalletAddressToDetails.

getBusinessDetails: A public view function which takes in the Business Wallet Address and returns the details stored on the businessWalletAddressToDetails.

getAllBusinesses: A public view function which returns allBusinesses.

setActiveFlag: A function which allows CreditRegister to activate or deactivate a business on the platform. If a business were to close down or be found to be maliciously uploading incorrect information, it would be deactivated and would no longer have the ability to add clients or documents to the platform.

4.2.2 Lookup Contract

The Lookup Contract stores details of the customers on the platform and is deployed after the Master Contract. This contract allows customers to interact with multiple businesses and be able to view details which were uploaded by all business contracts. Furthermore, it allows customers to give external viewers permission to view their full history for a limited time.

customerAddressToBusinessList: A mapping of the Customer Wallet Address to the Business Wallet Addresses to which they are linked.

customerAddressToPermissionList: A mapping of the Customer Wallet Address to a mapping of permitted addresses to the date their permission lapses.

getCustomerBusinessList: A public function which takes in a Customer Wallet Address and returns the list of businesses they are linked to in customerAddressToBusinessList.

addBusinessToCustomerList: A public function which adds a Business Wallet Address to the customers mapping on customerAddressToBusinessList.

addPermissionToCustomerList: A public function which allows a wallet to give another wallet permission to view their history. Importantly, the mapping is updated for the wallet which sent the request, so only the customer can grant permission to an external viewer.

checkPermission: A public function which checks that a user requesting to view information has permission and that it has not lapsed, based on the current date.

4.2.3 Business Contract

The Business Contract has two sections, namely Business and Customer. The Business section is used while setting up businesses, while the Customer Section contains the functionality for the business to interact with customers. The Business Contract also has Master and Lookup Interfaces to allow for it to interact with the CreditRegister platform.

Business Section

owner: Each Business Contract is owned by its business wallet address.

creator: Business contracts are currently only created by CreditRegister, but it is necessary to store this information for in case this changes in future.

businessName: A string with the actual name of the business, used for display purposes and to keep track of the businesses on the platform. assigned: This is initially set to false and is changed to true when ownership of the contract is set. Once this indicator has been changed to true, the ownership cannot be changed.

allCustomers: An array of the addresses of all customers that the business has interacted with or added to their profile.

setOwnership: A function allowing ownership of the contract to be set as long as the assigned indicator is false. Once ownership is set, the assigned indicator is changed to true.

Customer Section

customerAddressToDetails: A mapping of customer wallet address to their details which include name, balance and active indicator.

customerAddressToDocuments: A mapping of customer wallet address to details of all documents that the business has processed on their behalf. The document details include ipfsHash of the actual document, the amount and the date. If the amount is less than zero, the document is an invoice, and if it is greater than zero, it is a receipt.

addCustomer: A function allowing a business to add a customer to their contract. Only the business owner can add customers, and the customer cannot already exist on the business contract. The customer can be new to the platform or already be loaded on other business contracts. The business wallet address is added to the customerAddressToBusinessList on the relevant Lookup Contract. Furthermore, the customerAddressToDetails mapping is updated, and the customer address is added to the businesses allCustomers array.

getAllCustomers: A public function which returns the allCustomers array for the business.

getCustomerDocumentsLength: A public function which returns the length of the documents array for a specified customer address, mostly used for looping purposes when displaying details.

getCustomerDocument: A function which returns the document details at a given index.

processDocument: A function which takes in the customer wallet address, amount where negative implies an invoice and positive implies a receipt, the IPFS hash of the actual document and the document date. The function adjusts the customer balance accordingly and adds the Document Details to the customerAddressToDocuments mapping.

4.3 Additional Applications Used

IPFS was successfully implemented in the prototype, allowing for document details to be stored on the IPFS network with only the IPFS hash being stored on the blockchain. This significantly reduces the amount of data stored on the blockchain and hence will save gas costs. IPFS was run using Infura as this is the easiest way to use IPFS without having to run a node. Infura also makes this a scalable solution, as they handle the infrastructure required. uPort was not implemented in the prototype, as there was recently an update done which was not well documented or backward compatible. A decision was taken to exclude it from the prototype with the intention of adding it to CreditRegister if it goes into production, once uPort can guarantee stable updates with backward compatibility.

Chapter 5

Analysis

5.1 Platform Owner

The prototype was built based on the idea that a company called CreditRegister would deploy the initial contract and would be responsible for adding businesses to the platform once they have been vetted. Although the platform will exist even if CreditRegister closes down or loses access to their wallet, new businesses will not be able to be added to the platform. Further to this, having a central party like CreditRegister means that the application is not fully decentralised.

Creating a mechanism to allow other vetted businesses to add businesses to the platform was out of scope for this project. However, it is likely that if this idea were to be pursued, in order for it to be fully decentralised, some logic would need to be put in place to ensure that CreditRegister is not solely responsible for maintaining the business list.

Logic could be built to say that if there are less than a certain number of businesses on the platform, then CreditRegister is the only one which can add businesses to the platform after being vetted. However, once that threshold is reached, businesses can create a business contract which will only be deployed when the majority of the other businesses approve the request. Potentially a business score could be built based on ratings of the business, the number of disputes raised against it and length of time it has been on the platform. Businesses with a score that is too low could be deactivated, which could reduce the risk of malicious businesses being accepted onto the platform with no way of removing them.

5.2 Functionality

The minimum viable product contains the functionality required to create and build a credit register on the blockchain. The platform was able to have businesses loaded onto it, which could in turn load customers and upload documents. The customers can be linked to various different businesses and can easily view all of the documents that businesses have loaded for them. Further to this, the customer can give other wallets permission to view their history thereby ensuring that they have full control over who can view their data and for how long.

Additional functionality would be added if CreditRegister were to go live into production, but for the purposes of this paper, the prototype is sufficient to show what is possible on the blockchain.

5.3 Speed

When running the prototype on a personal Ethereum network using Ganache, each transaction takes at least a few seconds process. This is likely to be even slower on the Ethereum Main Net, and it is worth noting that at the time of writing this paper, the average block time according to https://ethstats.net/ was 15.39 seconds, so there is a relatively long wait before data gets added to the blockchain. If the purpose of the prototype were to process the transaction, then speed would be an important factor. However, the intention is to keep a record of the transactions after they have happened, as the business will collect the payments in cash or by card and then upload the invoice and receipt onto the CreditRegister platform.

If we were to take this solution into production, the code would need to be optimised which would likely make it run faster, but speed is not a major concern when loading document details onto the platform. Accessing and viewing the data quickly is more important, but since this does not involve writing anything to the blockchain, this functionality is already sufficiently fast in the prototype.

5.4 Security/Privacy

As discussed, CreditRegister uses the public blockchain, as opposed to a permissioned one. However, we can still control who can access the relevant functions, including view functions. Data on the blockchain is encrypted, so even though it is public, not all of the information is viewable unless a user is signed in with a wallet

which has permission to view it. Personal data other than a name is not stored on the blockchain, and the name is just an identifier for the business to keep track of their customers. This means that they could load a customer number instead of a name if they wanted to. In that way, all of the data is anonymous because it is all linked to a wallet address rather than an ID number or full name.

Another privacy concern is that businesses may upload invoices with private data on them which the customers might not want people to see. The IPFS hash is excluded from the viewer portal so that the viewer is not able to easily find the relevant file. With the vast number of IPFS hashes out there, it is highly unlikely that anyone will find a specific file without knowing the hash. However, if this were to go into production, it would be essential to keep the document information secure. This could be achieved by adding in an encryption step to the invoice and receipt upload processes. Currently, in the prototype, the file is just sent to IPFS as is, but the business uploading the file could first encrypt the file using the customers public key and then upload the encrypted version. Only the customers private key will be able to decrypt the file. This is called public key encryption and could be implemented on the platform using a program called GPG.

5.5 Costs

An important but sometimes overlooked aspect of using the blockchain is the actual cost of deploying and interacting with contracts. These costs and how they are calculated are different depending on which blockchain is used. The costs ensure that there is an incentive for nodes to mine blocks to add to the blockchain. Ethereum charges users gas fees when they interact with smart contracts. These fees are paid to the miners to compensate them for their computing power. The miners choose which transactions to include in their blocks and will usually choose the transactions where they receive the most money for the least work. In this way, an increase in the price the user is willing to pay for a transaction results in a higher likelihood that the transaction will be processed quickly. Likewise, paying a lower gas fee may result in the transaction taking much longer to be included on the blockchain.

The units of gas needed for a given transaction are calculated based on the number of operations that are performed. These operations include deploying contracts, writing data to the blockchain, mathematical and logical operators, amongst others and there is a spreadsheet online which gives the gas costs for each operation. Viewing data does not cost any gas, as no changes are made to the data or the

blockchain.

Gas costs are quoted in Ether and in order for a user to perform a transaction, they need to cover the costs. This means that real money is used for these fees, as the user would need to either mine or purchase Ether for the transaction to go through.

Gas costs are paid to miners to compensate for their computing power and also ensure that the network remains stable by preventing denial of service attacks. A denial of service attack involves attackers flooding the system with illegitimate requests, making the system unavailable to those that want to use it. Since the attacker would need to pay a fee for each transaction, it would become too expensive to attack the system for an extended period.

A certain amount of gas is charged for each operation, other than viewing data. When performing a transaction, a user needs to specify a gas limit, which is the number of units of gas they are prepared to pay for the transaction. The gas limit can be calculated by looking at the operations required for the transaction to take place. For example, it costs three units to add or subtract two numbers, five units to multiply or divide two numbers and a starting price of 32,000 to create a new contract.

The user also needs to specify the gas price that they are willing to pay. This is the cost per unit of gas and, as mentioned earlier, the higher the gas price, the quicker the transaction is likely to be picked up by a miner and included on the blockchain. The current recommended gas price can be checked online to ensure that the amount the user has specified is reasonable. The current recommended gas prices based on acceptable transaction times are included in Table 5.1.

For the gas cost calculations, the Standard price was used, as speed is not the most important factor, but waiting up to 30 minutes for a transaction to be written to the block is not optimal. The Standard price is converted into South African Rands (ZAR) in Table 5.2, as this paper is based on South African businesses and individuals. The total transaction fee is the gas limit multiplied by the gas price. Wood (2014)

In order to calculate the cost of deploying and using this contract on Ethereum, it was necessary to deploy the contract to a test network because Ganache, which was used for testing the code did not give accurate gas cost estimates since it is just an emulator. There are a number of Ethereum test networks, but the Rop-

Speed	Gas Price (gwei)
SafeLow (<30mins)	2.7
Standard (<5mins)	3
Fast (<2mins)	7

Tab. 5.1: Gas Prices as at 23 January 2019 from ETH GasStation (2019)

ETH/USD	117.45
Gwei/ETH	0.000000001
Gas Price - Standard (Gwei)	3
USD/ZAR	13.85

Tab. 5.2: Gas Prices converted to ZAR on 23 January 2019 using exchange rates from x Rates (2019) and CoinMarketCap (2019)

sten network uses the same consensus mechanism (proof of work) as the Ethereum Mainnet and is considered to be the most similar network to the Mainnet. The Ethereum Mainnet is the main network where actual transactions are processed, and real Ether is used. The test networks work very similarly for developers to test new dApps, but the Ether used on these networks do not have monetary value.

The contract was deployed onto the Ropsten test network using the online version of the Remix IDE. The Remix IDE allows interactions with the blockchain, and the details of the costs incurred can be seen in Metamask as well as on Etherscan for Ropsten.

The costs linked to different aspects of the solution are summarized in Table 5.3. The costs of deploying the Master and Lookup contracts are once-off costs and are affordable. Similarly, the costs of adding customers or documents are reasonable. However, these costs are incurred every time a business adds a new customer or invoice, and this could become expensive, considering that all transactions require the upload of an invoice or receipt. Some businesses may be sceptical about using CreditRegister, as the customers loaded on the platform are the ones that benefit not necessarily the businesses yet the businesses are responsible for the costs.

However, these costs are not prohibitive and could probably be reduced by further optimising the code and by investigating whether it would be worth giving up some of the flexibility offered by Ethereum to move onto a cheaper blockchain.

Alternatively, there may be some organisations which can recognise the value that this solution can add to the unbanked, and may be willing to fund the costs of using it.

Operation	Gas Used (Units)	USD Cost	ZAR Cost	Paid By
Deploy Master	2 603 108	0.92	12.70	CreditRegister
Deploy Lookup	563 654	0.20	2.75	CreditRegister
Add Business	1 468 777	0.52	7.17	CreditRegister
Set Ownership	88 122	0.03	0.43	CreditRegister
Add Customer	154 903	0.05	0.76	Business
Add Invoice	172 786	0.06	0.84	Business

Tab. 5.3: Transaction costs calculated using Gas Used from Remix (2019) and rates from Table 5.1 and Table 5.2

Chapter 6

Conclusion

In order to look into the feasibility of building a credit register on the blockchain, it was important to research how credit histories are used around the world and what is included in them. Often, they contain limited information, as the credit bureaus that compile the credit reports and calculate credit scores use data from banks and financial institutions, which unbanked individuals are often excluded from. Credit is often hard to come by without a strong credit history, but a credit history is difficult to build without access to credit. This paper proposes building a credit register using data from businesses that offer services mostly to the unbanked and could show that certain individuals have a good track record of regular payments.

Blockchain technology can be used to store all of this information as it allows for different businesses to contribute data and ensures that the data cannot be tampered with once it has been uploaded. Since the blockchain is a decentralised ledger, there is not a dependency on a central database to be online. Ethereum was chosen as the blockchain of choice for this project, primarily because it is one of the more established blockchains and because it has smart contracts. Using smart contracts, businesses can be added to the platform who can, in turn, add customers and upload documents. A credit score is calculated based on how much money the customer owes the businesses they interact with and that credit score is only accessible to the customer and users whom the customer has explicitly permitted. This gives individuals the ability to own their data and to be able to provide proof that they have been able to keep up with monthly payments to businesses on the platform.

A prototype for CreditRegister was built, which allows businesses to be added onto the platform through a web browser. Each business is able to add customers and upload invoices and receipts for each customer. This ensures that the customer is aware of what the business is charging them for and that they can check that the

business has correctly receipted them when they have made a payment. The invoices and receipts are stored on IPFS. Only the business and the customer know the IPFS hash and, as discussed in 5.4, the document should be encrypted, so any confidential information within the document will not be shared with anyone else. There is a credit score assigned to each customer based on their outstanding balance at each business. Since multiple businesses can be loaded on the platform, a customer can build up a credit record which takes into account their interactions with various businesses. The details are stored on the blockchain, which means that no one can tamper with the amounts and it is easy to see which business has loaded which documents. The customer is able to view their credit score as well as all of the documents which have contributed to the score. They are also able to give permission to certain wallet addresses to view their credit score and a breakdown of the invoices and receipts which have been loaded for the customer. The actual documents are not visible to the viewer though, as these may contain confidential information. The customer owns their data because they can specify how long each wallet is permitted to view their information for and, without that permission, one cannot view the information. The viewer gets the benefit of being able to see how long the customer usually takes to make a payment, as well as whether there are any outstanding payments. This is a good indication of the creditworthiness of the customer.

Through using additional products such as Infura and IPFS, this could be a scalable solution, and the data stored on the blockchain is kept to a minimum. Although CreditRegister does not charge fees, there are transaction fees on the blockchain and, despite minimising the data stored on the blockchain, it could become expensive for businesses to upload documents on a regular basis if they have a large customer base. However, these costs can likely be reduced by further optimising the code or by investigating using a different, cheaper blockchain.

In conclusion, building an informal credit register on the blockchain, using credit histories from businesses which serve the informal sector is feasible. This solution could address the shortfalls of current credit registers and have a very positive impact, specifically to those living in informal settlements who do not have a credit history but who are obligated to make payments on a regular basis. Over time, the payment history loaded onto the platform could be invaluable and could give some individuals access to affordable credit, rather than paying the exorbitant fees they are sometimes charged, due to lack of credit history.

Bibliography

Antonovici, A. (2018). Polish credit office picks billon for customer data blockchain. [Online; accessed 23-December-2018].
URL: *https://cryptovest.com/news/polish-credit-office-picks-billon-for-customer-data-blockchain/*

Australian Government (2018). Mandatory comprehensive credit reporting.
URL: *https://ris.pmc.gov.au/2018/07/16/mandatory-comprehensive-credit-reporting*

Avery, R. B., Calem, P. S. and Canner, G. B. (2004). Credit report accuracy and access to credit, *Fed. Res. Bull.* **90**: 297.

Banco Bilbao Vizcaya Argentaria (n.d.). What is the ASNEF and how does it influence the granting of a loan. [Online; accessed 30-January-2019].
URL: *https://www.bbva.es/eng/general/finanzas-vistazo/prestamos/prestamos-con-asnef/index.jsp*

Banco De Espana (n.d.). Central credit register.
URL: *https://www.bde.es/bde/en/secciones/servicios/Particulares_y_e/Central_de_Infor/Central_de_Info_04db72d6c1fd821.html*

Bartoletti, M. and Pompianu, L. (2017). An empirical analysis of smart contracts: platforms, applications, and design patterns, *CoRR* **abs/1703.06322**.
URL: *http://arxiv.org/abs/1703.06322*

Bloom (2017). What is Bloom? - An Introduction to the Bloom Protocol. [Online; accessed 21-December-2018].
URL: *https://www.youtube.com/watch?v=ntNKcBlw9FU*

Brevoort, K. P., Grimm, P. and Kambara, M. (2015). Data point: Credit invisibles, *Washington, DC: The Consumer Financial Protection Bureau* . [Online; accessed 23-December-2018].
URL: *http://files. consumerfinance. gov/f/201505_cfpb_data-point-credit-invisibles.pdf*

Buterin, V. (2014). Ethereum White Paper: A next-generation smart contract and decentralized application platform.
URL: *https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf*

Clearscore (2017). A Guide to the Credit Bureaus in South Africa. [Online; accessed 22-December-2018].
URL: *https://www.clearscore.co.za/credit-score/what-are-credit-bureaus*

CoinMarketCap (2019). Coinmarketcap: Historical Data for Ethereum. [Online; accessed 24-January-2019].
URL: *https://coinmarketcap.com/currencies/ethereum/historical-data/?start=20190123&end=20190123*

Curley, C. (2018). Many countries don't use credit scores like the US here's how they determine your worth. [Online; accessed 21-December-2018].
URL: *https://www.businessinsider.com/credit-score-around-the-world-2018-8?IR=T*

CustoTech (2018). Watermarking is not enough. [Online; accessed 30-January-2019].
URL: *https://custostech.com/wp-content/uploads/2018/10/Custos-WP-Watermarking.pdf*

DebtCanada (2018). Credit rating 101. [Online; accessed 30-January-2019].
URL: *https://www.debtcanada.ca/library/credit-rating-101*

Djankov, S., McLiesh, C. and Shleifer, A. (2005). Private credit in 129 countries, *Working Paper 11078*, National Bureau of Economic Research.
URL: *http://www.nber.org/papers/w11078*

Ekblaw, A., Azaria, A., Halamka, J. D. and Lippman, A. (2016). A case study for blockchain in healthcare:medrec prototype for electronic health records and medical research data, *Proceedings of IEEE open & big data conference*, Vol. 13, p. 13.

ETH GasStation (2019). ETH Gas Station. [Online; accessed 23-January-2019].
URL: *https://ethgasstation.info/*

Experian (2018a). The electoral roll and your credit score. [Online; accessed 30-January-2019].
URL: *https://www.experian.co.uk/consumer/guides/electoral-roll.html*

Experian (2018b). How long late payments stay on credit report. [Online; accessed 30-January-2019].
URL: *https://www.experian.com/blogs/ask-experian/how-long-past-due-remains/*

Hyperledger (2018). Hyperledger Fabric. [Online; accessed 21-December-2018].
URL: *https://www.hyperledger.org/projects/fabric/*

IBM Blockchain (2017). IBM and Maersk demo: Cross-border supply chain solution on blockchain. [Online; accessed 21-December-2018].
URL: *https://www.youtube.com/watch?v=tdhpYQCWnCwl*

Infura (2018). Infura - Your Access to the Ethereum Network. [Online; accessed 21-December-2018].
URL: *https://infura.io/*

IPFS (2018). IPFS is the distributed web. [Online; accessed 21-December-2018].
URL: *https://ipfs.io/*

Jesse Leimgruber, A. M. and Backus, J. (2018). Bloom Protocol - Decentralized credit scoring powered by Ethereum and IPFS. [Online; accessed 23-December-2018].
URL: *https://hellobloom.io/whitepaper.pdf*

JP Morgan (2016). Quorum whitepaper.
URL: *https://github.com/jpmorganchase/quorumdocs/blob/master/Quorum% 20Whitepaper% 20v0*

Lal, R. and Johnson, S. (2018). Maersk: Betting on blockchain, *Harvard Business School* **518-089**.

Lumkani (2018). Lumkani - Protecting Against Fires. [Online; accessed 23-December-2018].
URL: *https://lumkani.com/*

Mazieres, D. (2016). The Stellar Consensus Protocol: A federated model for internet-level consensus. [Online; accessed 23-December-2018].
URL: *https://www.stellar.org/papers/stellar-consensus-protocol.pdf*

MedRec (2019). MedRec technical documentation. [Online; accessed 30-January-2019].
URL: *https://medrec.media.mit.edu/technical/*

Mian, A. (2012). The case for a credit registry, *in* M. Brunnermeier and A. Krishnamurthy (eds), *Risk Topography: Systemic Risk and Macro Modeling*, University of Chicago Press, pp. 163–172.
URL: *http://www.nber.org/chapters/c12553*

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. Whitepaper, 2009.

Pam Golding Properties (2019). Landlord and tenant services. [Online; accessed 30-January-2019].
URL: *https://www.pamgolding.co.za/about-us/services/residential-letting*

Patnick, H. (2017a). A guide to the credit bureaus in south africa. [Online; accessed 30-January-2019].
URL: *https://www.clearscore.com/za/credit-score/what-are-credit-bureaus*

Patnick, H. (2017b). How to get yourself the best rate on a loan. [Online; accessed 30-January-2019].
URL: *https://www.clearscore.com/za/loans/compare-best-loan-rates*

Patnick, H. (2017c). What is a credit score? [Online; accessed 30-January-2019].
URL: *https://www.finder.com/how-countries-score-credit*

Pave Inc (2017). Pave - A decentralized credit bureau. [Online; accessed 23-December-2018].
URL: *https://gcp.pave.com/Pave_Whitepaper.pdf*

Prisco, G. (n.d.). Polish credit office to deploy blockchain solution for credit histories.
URL: *https://bitcoinmagazine.com/articles/polish-credit-office-deploy-blockchain-solution-credit-histories/*

Rawson Property Group (2019). Our screening process. [Online; accessed 30-January-2019].
URL: *https://www.rawson.co.za/landlords#screening*

Remix (2019). Remix IDE. [Online; accessed 23-January-2019].
URL: *https://remix.ethereum.org/#optimize=false&version=soljson-v0.4.23+commit.124ca40d.js*

SCHUFA (n.d.). For your business with private customers (B2C).
URL: *https://www.schufa.de/en/corporate-customers/credit-rating/*

Siyongwana, P. Q. (2004). Informal moneylenders in the Limpopo, Gauteng and Eastern Cape provinces of South Africa, *Development Southern Africa* **21**(5): 851–866.
URL: *https://www.tandfonline.com/doi/abs/10.1080/0376835042000325741*

State of the DApps (2019). New dapps per month. [Online; accessed 30-January-2019].
URL: *https://www.stateofthedapps.com/stats*

Staten, M. (2003). The value of comprehensive credit reports: Lessons from the US experience, *Credit reporting systems and the international economy* **8**: 273–310.

Stellar (2018). Stellar Network Overview. [Online; accessed 21-December-2018].
URL: *https://www.stellar.org/developers/guides/get-started/index.html*

Stichting Bureau Krediet Registratie (n.d.). Bkr general. [Online; accessed 31-January-2019].
URL: *https://www.bkr.nl/veelgestelde-vragen/*

Thawte (n.d.). About Thawte.
URL: *https://www.thawte.com/about/*

The Economist (2008). Japan's moneylenders offer banks a lesson in risk management. [Online; accessed 31-January-2019].
URL: *https://www.economist.com/finance-and-economics/2008/05/22/lenders-of-first-resort*

The World Bank Group (2011). General principles for credit reporting. [Online; accessed 22-December-2018].
URL: *http://documents.worldbank.org/curated/en/662161468147557554/General-principles-for-credit-reporting*

The World Bank Group (2018a). Credit bureau - definition and comparison to credit registries. [Online; accessed 22-December-2018].

URL: *http://www.worldbank.org/en/publication/gfdr/background/key-terms-explained#3*

The World Bank Group (2018b). Credit registry - definition and comparison to credit bureaus. [Online; accessed 22-December-2018].
URL: *http://www.worldbank.org/en/publication/gfdr/background/key-terms-explained#4*

Tobias Baer, T. G. and Schiff, R. (2013). New credit-risk models for the unbanked, *McKinsey & Company* . [Online; accessed 23-December-2018].
URL: *https://www.mckinsey.com/business-functions/risk/our-insights/new-credit-risk-models-for-the-unbanked#0*

TradeLens (2018). TradeLens solution brief. [Online; accessed 30-January-2019].
URL: *https://s3.us-south.cloud-object-storage.appdomain.cloud/tradelens-bucket/wp-content/uploads/2018/08/09185955/TradeLens-Solution-Brief_Edition-One.pdf*

Truffle (2018). Truffle Suite. [Online; accessed 21-December-2018].
URL: *https://truffleframework.com/*

UPort (2018). Helping you build user centric apps on blockchains. [Online; accessed 21-December-2018].
URL: *https://developer.uport.me/*

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger, *Ethereum project yellow paper* **151**: 1–32. [Online; accessed 21-January-2019].
URL: *https://ethereum.github.io/yellowpaper/paper.pdf*

Wüst, K. and Gervais, A. (2018). Do you need a blockchain?, *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, IEEE, pp. 45–54.

x Rates (2019). x-Rates: Lookup Historic Exchange Rates. [Online; accessed 24-January-2019].
URL: *https://www.x-rates.com/historical/?from=ZAR&amount=1&date=2019-01-23*

Zheng, Z., Xie, S., Dai, H.-N. and Wang, H. (2016). Blockchain challenges and opportunities: A survey, *Work Pap.–2016* .

Zheng, Z., Xie, S., Dai, H.-N. and Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends, *Big Data (BigData Congress), 2017 IEEE International Congress on*, IEEE, pp. 557–564.

Appendix A

Using the CreditRegister Prototype

Before running the CreditRegister Prototype, it is important to have at least four different wallets with Ether in to be able to see the functionality from all of the user groups. You can deploy to the Test Network, in which case Ether can be requested from a faucet or the contracts can be deployed locally and an emulator like Ganache can be used. The four wallets will represent CreditRegister, a business, a customer and a viewer respectively.

The following sections describe how to run the prototype in a Chrome web browser using Ganache, with a brief description of how to install each program necessary to run the prototype. Further details on the installation and usage of each one can be found on the Fintech and Cryptocurrencies Wiki at https://github.com/cogeorg/teaching/wiki/Fintech-and-Cryptocurrencies.

Ganache: Download Ganache from http://truffleframework.com/ganache/ and install it. Make sure that you open it before starting CreditRegister.

MetaMask: Install the MetaMask extension in your browser. When it has been added, it will guide you through the setup. On the first screen, click "Continue" and, when asked to create a password, choose the "Import with seed phrase" option below the "Create" button. Copy the 12-word phrase from the mnemonic section at the top of the Accounts screen in Ganache. Paste the phrase into the "Wallet Seed" block and create a password for the account. Read and accept the Terms of Use, Privacy Warning and Phishing Warning.

Now, in order to link the account to your Ganache accounts, click on the block that says "Main Ethereum Network" and on the list of networks, choose "Custom RPC". In the "New Network" section, paste in http://127.0.0.1:7545, click "Save and close the settings window. Now click on the colourful circle and click "Create Account" four times, naming them Master/CreditRegister, Business, Customer and Viewer.

CreditRegister: Clone the GitHub repo with the CreditRegister code from https://github.com/bryonySA/Thesis and open it in Visual Studio Code.
Migrate the contracts by running `truffle migrate --network ganache --reset --`

`compile -all`.
Test the contracts by running `truffle test --network ganache`.
When they have all passed successfully, run `npm start`. This should build the program and open it in a web browser.

Check that the details of the account you are logged in with are displayed on the home page and make a copy of the Master and Lookup Contract Addresses, should you want to reload them at a later stage.
Navigate to the Master Portal and add the first business to the system by clicking the "Add Business" button. Note that you must be logged in with the first account on the Ganache list of accounts, as this account is the owner of the Master Contract. Use the second address on the list as the Business Address and make up a name. When you click the "Submit" button, Metamask will pop up asking you to pay for the contract creation and then again for setting the ownership. Confirm both of these transactions and a new Business contract should be deployed.

In Metamask, switch to the Business account and navigate to the Business Portal. Here you can load a new customer using the third account on the Ganache list of accounts.
Metamask will pop up again asking for payment to write to the blockchain.
Confirm the transaction and then use the "Invoice Customer" button to upload an invoice for the customer you just created. The customer details and new balance should be displayed.

Switch to the Customer account in Metamask and navigate to the Customer Portal. Here, you can view details of all of the documents which have been loaded against a particular client. In addition to this, you can give a viewer permission to also view the document details.
Use the "Add Permission" button to give the fourth Ganache account permission to view the details.

Now switch to the Viewer account in Metamask and go to the Viewer Portal.
Click the "View Documents" button and enter the address of the Customer account. Since permission has been given, the document details should load, and the viewer can see whether the customer is up to date with due payments. The customer score should also be displayed which will show the sum of their unpaid accounts.

Appendix B

Code

For access to the latest code for CreditRegister, please see my GitHub Repository at: https://github.com/bryonySA/Thesis.

B.1 Solidity Code

B.1.1 Master Contract

```
pragma solidity ^0.4.23;

import "./Business.sol";

contract Master {

    /*
        DESCRIPTION:
        The Master contract will be owned by CreditRegister. This is a contract factory which
        allows us to deploy new instances of the Business contract when reputable businesses
        want to be added onto the platform. These businesses will be able to record payment
        histories, so they will need to be vetted first by CreditRegister.
    */


    ////////////////
    // VARIABLES //
    ///////////////

    struct BusinessDetails{
        address businessContractAddress;
        string businessName;
        bool businessActive;
    }

    // The wallet address for the owner of the contract which is always CreditRegister
    address public owner;

```

```
29      //This mapping stores the Business Details for each business wallet
            address
30      mapping(address => BusinessDetails) private
            businessWalletAddressToDetails;
31
32      // This array stores the buisness wallet addresses of all businesses
            loaded on the platform
33      address[] private allBusinesses;
34
35
36      //////////////
37      // EVENTS //
38      ////////////
39
40      event businessAdded(address _businessWalletAddress, string
            _businessName, address _contractAddress);
41
42
43      ////////////////
44      // FUNCTIONS //
45      //////////////
46
47      // Set the owner of the contract do the wallet address that deployed
            it
48      constructor() public{
49          owner = msg.sender;
50      }
51
52
53      // Add business to the platform by deploying a new business contract
            as long as  CreditRegister sends the request and the business does
54      // and the business does not already exist on the platform
55      function addBusiness(address _businessWalletAddress, string
            _businessName) public {
56          require(msg.sender == owner, "Only CreditRegister can add
                businesses.");
57          require(
58              businessWalletAddressToDetails[_businessWalletAddress].
                    businessActive != true,
59              "Business wallet already exists."
60          );
61
62          // Deploy new business contract
63          Business newBusiness = new Business();
64
65          // Store the new contract address
66          address businessContractAddress = address(newBusiness);
67
68          // Update the Business Details on the mapping and add the
                business wallet address to the list of businesses on the
                platform
69          businessWalletAddressToDetails[_businessWalletAddress] =
                BusinessDetails(businessContractAddress, _businessName,true);
70          allBusinesses.push(_businessWalletAddress);
71
72          // Emit the event to confirm that the business has been added
```

```
73          emit businessAdded(_businessWalletAddress, _businessName,
                businessContractAddress);
74      }
75
76
77      function getBusinessDetails(address _businessWalletAddress) public
            view
78          returns (
79          address businessContractAddress,
80          string businessName,
81          bool businessActive
82          )
83      {
84          return (
85              businessWalletAddressToDetails[_businessWalletAddress].
                    businessContractAddress,
86              businessWalletAddressToDetails[_businessWalletAddress].
                    businessName,
87              businessWalletAddressToDetails[_businessWalletAddress].
                    businessActive
88          );
89      }
90
91
92      function getAllBusinesses() public view returns(address[]){
93          return allBusinesses;
94      }
95
96      // Allow CreditRegister to activate or deactivate businesses on the
            platform
97      function setActiveFlag(address _businessWalletAddress, bool
            _businessActive) public {
98          require(msg.sender == owner, "Only CreditRegister can change the
                active flag of a business.");
99          businessWalletAddressToDetails[_businessWalletAddress].
                businessActive = _businessActive;
100     }
101 }
```

B.1.2 Business Contract

```
1  pragma solidity ^0.4.23;
2
3  import "./Master.sol";
4  import "./Lookup.sol";
5
6  // Interface to allow the Business Contract to use Master Contract
       functionality
7  contract MasterInterface {
8      function getBusinessDetails(address _businessWalletAddress)
9          public
10         view
11         returns (
12         address businessContractAddress,
13         string businessName,
14         bool businessActive
15         );
16 }
17
18
19 // Interface to allow the Business Contract to use Lookup Contract
       functionality
20 contract LookupInterface {
21     function checkCustomerExists(address _customerAddress) public view
           returns(bool);
22     function getCustomerBusinessList(address _customerAddress) public
           view returns (address[]);
23     function addBusinessToCustomerList(address _customerAddress,address
           _businessWalletAddress) public;
24 }
25
26
27 contract Business {
28      /*
29         DESCRIPTION:
30         The Business Contract has two sections - Business and Customer -
               and provides the main functionality for the
31         credit registry. It allows businesses to interact with customers
               as well as allows for customers to interact with
32         the platform.
33     */
34
35     ////////////////////////////////////
36     ///// BUSINESS SECTION /////////////
37     ////////////////////////////////////
38
39     ////////////////
40     // VARIABLES //
41     ///////////////
42
43     address public owner; //This will be the businesses wallet
44     address public creator; //This will be the CreditRegister wallet
45     string public businessName;
46     bool public assigned = false; // Changes to true when ownership
           assigned
```

```
47      address[] allCustomers; // Array with addresses of all customers the
            business has interacted with
48
49      //////////////
50      // EVENTS //
51      ////////////
52
53      event ownershipSet(address indexed _businessWalletAddress, string
            _businessName, address _contractAddress);
54
55      ////////////////
56      // FUNCTIONS //
57      ///////////////
58
59      constructor() public {
60          creator = msg.sender;
61      }
62
63      // Function allowing ownership of a new contract to be set. This can
            only be done once
64      function setOwnership(address _businessWalletAddress, string
            _businessName) public{
65          require(assigned == false, "Contract has already been assigned
                ownership");
66          owner = _businessWalletAddress;
67          businessName = _businessName;
68
69          // Change assigned to true so that ownership cannot be changed
70          assigned = true;
71          emit ownershipSet(_businessWalletAddress, _businessName, this);
72      }
73
74
75      ////////////////////////////////////
76      ///// CUSTOMER SECTION //////////////
77      ////////////////////////////////////
78
79      ////////////////
80      // VARIABLES //
81      ///////////////
82
83      struct CustomerDetails{
84          string customerName;
85          int customerBalance;
86          bool customerActive;
87      }
88
89      struct DocumentDetails{
90          string ipfsHash;
91          int amount; // Note that if amt < 0 then invoice, if > 0 receipt
92          string dueDate;
93      }
94
95      // Mapping of customer wallet address to their details including
            balance
96      mapping(address => CustomerDetails) customerAddressToDetails;
```

```
97      // Mapping of customer wallet address to all documents the business
            has processed for them
98      mapping(address => DocumentDetails[]) customerAddressToDocuments;
99
100     //////////////
101     // EVENTS //
102     ///////////
103
104     event customerAdded(string _customerName, address _customerAddress,
            address _businessWalletAddress);
105     event documentProcessed(address _customerAddress, int
            _customerBalance);
106
107     ////////////////
108     // FUNCTIONS //
109     ///////////////
110
111     // Function allowing a customer to be added to the platform. It needs
             to be linked to the correct Master and Lookup contracts,
112     // so these contract addresses are taken in as inputs
113     function addCustomer(
114         address _customerAddress,
115         string _customerName,
116         address _lookupContractAddress,
117         address _masterContractAddress
118     )
119         public
120     {
121         // Only the business wallet can add customers to their list
122         require(owner == msg.sender, "Only the business owner can add
                customers");
123         // Customer must not already be loaded on the businesses list of
                clients
124         require(
125             customerAddressToDetails[_customerAddress].customerActive !=
                    true,
126             "This customer is already active. Please use amend function"
127         );
128         // Customer must have a balance = 0 if they are loaded and
                inactive
129         require(
130             customerAddressToDetails[_customerAddress].customerBalance ==
                     0,
131             "This customer is inactive but has a non-zero balance. Please
                     use amend function"
132         );
133
134         //It doesn't matter if the customer already exists on the
                platform - either create a new array or add to existing array
135         LookupInterface lookupContract = LookupInterface(
                _lookupContractAddress);
136         lookupContract.addBusinessToCustomerList(_customerAddress, msg.
                sender);
137
138         //The opening balance must be 0 as the business needs to upload
                an invoice or receipt before changing the balance
```

```
139         customerAddressToDetails[_customerAddress] = CustomerDetails(
                _customerName, 0,true);
140         allCustomers.push(_customerAddress);
141 
142         emit customerAdded(_customerName, _customerAddress, msg.sender);
143     }
144 
145     function getAllCustomers() public view returns(address[]){
146         return allCustomers;
147     }
148 
149     function getCustomerDetails(address _customerAddress)
150         public
151         view
152         returns (
153             string customerName,
154             int customerBalance,
155             bool customerActive
156         )
157     {
158         return (
159             customerAddressToDetails[_customerAddress].customerName,
160             customerAddressToDetails[_customerAddress].customerBalance,
161             customerAddressToDetails[_customerAddress].customerActive
162         );
163     }
164 
165     // Return the length of the customers document list (mainly for
            looping purposes)
166     function getCustomerDocumentsLength(address _customerAddress)
167         public
168         view
169         returns (uint)
170     {
171         return customerAddressToDocuments[_customerAddress].length;
172     }
173 
174     // Return a document from the customer document list
175     function getCustomerDocument(address _customerAddress, uint index)
176         public
177         view
178         returns (
179             string ipfsHash,
180             int amount,
181             string dueDate
182         )
183     {
184         DocumentDetails memory thisDocument = customerAddressToDocuments[
                _customerAddress][index];
185         return (
186             thisDocument.ipfsHash,
187             thisDocument.amount,
188             thisDocument.dueDate
189         );
190     }
191 
```

```
192     // Process invoices or receipts for the customer to update their
            balance
193     function processDocument(
194         address _customerAddress,
195         int _amount,
196         string _ipfsHash,
197         string _dueDate
198     )
199         public
200     {
201         //NB!!! Negative number = INVOICE
202         require(owner == msg.sender, "Only the business owner can process
                documents");
203         require(customerAddressToDetails[_customerAddress].customerActive
                == true, "This customer must be active");
204
205         // Update the customer balance
206         CustomerDetails storage thisCustomer = customerAddressToDetails[
                _customerAddress];
207         thisCustomer.customerBalance = thisCustomer.customerBalance +
                _amount;
208
209         // Add the Document Details to the customer list
210         DocumentDetails memory document = DocumentDetails(_ipfsHash,
                _amount, _dueDate);
211         customerAddressToDocuments[_customerAddress].push(document);
212
213         emit documentProcessed(_customerAddress, thisCustomer.
                customerBalance);
214     }
215 }
```

B.1.3 Lookup Contract

```
pragma solidity ^0.4.23;

import "./Business.sol";

contract Lookup {

    /*
        DESCRIPTION:
        The Lookup contract is used to keep track of all of the customers on the CreditRegister platform
        and the businesses that they interact with. This was necessary as the platform needed to allow
        for the same customer to interact with multiple businesses on the platform.
        Further to this, it stores details of external viewers who are permitted to view a specific customer's
        history and score.
    */


    ////////////////
    // VARIABLES //
    ///////////////

    // Mapping of the cusotmer wallet address to an array of business wallets they interact with
    mapping(address => address[]) public customerAddressToBusinessList;

    // Mapping of the customer address to a mapping of permitted viewers and the date they have permission until
    mapping(address => mapping(address => uint256)) public customerAddressToPermissionList;


    //////////////
    // EVENTS //
    ///////////

    event permissionAdded(address _customerAddress, address _businessWalletAddress, uint256 _viewableUntil);

    ////////////////
    // FUNCTIONS //
    ///////////////

    // Function which returns the list of business addresses that a customer wallet is linked to
    function getCustomerBusinessList(address _customerAddress) public view returns (address[]) {
        return customerAddressToBusinessList[_customerAddress];
    }


    // Function which adds a business address to the list of businesses
```

```
        the customer is linked to
45      function addBusinessToCustomerList(address _customerAddress, address
            _businessWalletAddress) public {
46          customerAddressToBusinessList[_customerAddress].push(
                _businessWalletAddress);
47      }
48
49
50      // Function which allows a customer to add a permitted viewer to
            their list. The mapping for msg.sender is updated
51      // so view permission is only given for the account that sent the
            request
52      function addPermissionToCustomerList(
53          address _viewerWalletAddress,
54          uint256 _viewableUntil
55      )
56          public
57      {
58          customerAddressToPermissionList[msg.sender][_viewerWalletAddress]
                = _viewableUntil;
59          emit permissionAdded(msg.sender, _viewerWalletAddress,
                customerAddressToPermissionList[msg.sender][
                _viewerWalletAddress]);
60      }
61
62
63      // Functon that checks that the viewer requesting access has
            permission
64      function checkPermission(address _customerAddress, address
            _viewerAddress) public view returns (bool){
65          return (customerAddressToPermissionList[_customerAddress][
                _viewerAddress]) > now;
66
67      }
68  }
```

B.2 JavaScript Code

```
var Web3 = require('web3');
var TruffleContract = require('truffle-contract');
var buffer = require('buffer/').Buffer;
var IPFS = require('ipfs-http-client');

App = {
     web3Provider: null,
     contracts: {},
     account: 0x0,
     loading: false,
     web3: null,
     //existingMasterContractAddress: 0x0,
     existingMasterContractAddress: '0
         xe6cd3a3ee9aefdb7dea9a2a0087ffbc3efb58536',
     //existingLookupContractAddress: 0x0,
     existingLookupContractAddress: '0
         x85ada02a4ff81691e39f5d77b926a81cc4d9a0af',
     masterAddress: 0x0,
     lookupAddress: 0x0,


     //NB: Make sure you've run npm install web3
     init: function () {
          return App.initWeb3();
     },

     initWeb3: function () {
          // initialize web3
          if (web3) {
               // reuse the provider of the web3 object injected by
                   MetaMask
               App.web3Provider = web3.currentProvider;
               this.web3 = web3;
          } else {
               // either create a new provider, here connecting to
                   Ganache
               App.web3Provider = new Web3.providers.HttpProvider('http
                   ://127.0.0.1:7545')
               // instantiate a new web3 object
               this.web3 = new Web3(App.web3Provider);
               // or handle the case that the user does not have MetaMask
                    by showing her a message asking her to install
                   Metamask
          }
          return App.initContract();
     },

     initContract: function () {
          $.getJSON('Master.json', function (MasterArtifact) {

               // get the contract artifact file and use it to
                   instantiate a truffle contract abstraction
               App.contracts.Master = TruffleContract(MasterArtifact);

```

```
47                // set the provider for our contract
48                App.contracts.Master.setProvider(App.web3Provider);
49
50                //update account info
51                App.displayAccountInfo();
52           });
53
54           $.getJSON('Business.json', function (BusinessArtifact) {
55                App.contracts.Business = TruffleContract(BusinessArtifact)
                       ;
56                App.contracts.Business.setProvider(App.web3Provider);
57           });
58
59           $.getJSON('Lookup.json', function (LookupArtifact) {
60                App.contracts.Lookup = TruffleContract(LookupArtifact);
61                App.contracts.Lookup.setProvider(App.web3Provider);
62           });
63      },
64
65      returnMaster: function () {
66           // Check whether we are using a new or existing instance of
                  master contract
67           // This is dependent on whether the existing address declared
                  at the beginning of this function is 0x0 or not
68           if (App.existingMasterContractAddress == 0x0) {
69                // get the most recently deployed instance of the master
                       contract
70                return App.contracts.Master.deployed();
71           } else {
72                // get the existing master contract from its address
73                return App.contracts.Master.at(App.
                       existingMasterContractAddress);
74           };
75      },
76
77      returnLookup: function () {
78           // Check whether we are using a new or existing instance of
                  lookup contract
79           // This is dependent on whether the existing address declared
                  at the beginning of this function is 0x0 or not
80           if (App.existingLookupContractAddress == 0x0) {
81                // get the most recently deployed instance of the master
                       contract
82                return App.contracts.Lookup.deployed();
83           } else {
84                // get the existing master contract from its address
85                return App.contracts.Lookup.at(App.
                       existingLookupContractAddress);
86           };
87      },
88
89
90      displayAccountInfo: function () {
91           if (!this.web3) {
92                this.initWeb3();
93           }
```

```
94          var web3 = this.web3;
95          // get current account information
96          web3.eth.getCoinbase(function (err, account) {
97               // if there is no error
98               if (err === null) {
99                    //set the App object's account variable
100                   App.account = account;
101                   // insert the account address in the p-tag with id='
                          account'
102                   $("#account").text(account);
103                   // retrieve the balance corresponding to that account
104                   web3.eth.getBalance(account, function (err, balance)
                          {
105                        // if there is no error
106                        if (err === null) {
107                             // insert the balance in the p-tag with id
                                  ='accountBalance'
108                             $("#accountBalance").text(web3.fromWei(
                                  balance, "ether") + " ETH");
109                        }
110                   });
111              }
112         });
113
114    },
115
116
117    displayContractInfo: function () {
118         App.returnMaster().then(async function (instance) {
119              masterAddress = instance.address;
120              $("#masterContract").text(masterAddress);
121         });
122         App.returnLookup().then(async function (instance) {
123              lookupAddress = instance.address;
124              $("#lookupContract").text(lookupAddress);
125         });
126    },
127
128
129    addBusiness: function () {
130         console.log('Add Business button clicked');
131         // get information from the modal
132         var _businessName = $('#BusinessName').val();
133         var _businessWalletAddress = $('#BusinessWalletAddress').val();
134
135         // if the name or valid address was not provided
136         if ((_businessName.trim() == '') || (web3.isAddress(
                _businessWalletAddress) != true)) {
137              // we cannot add a business
138              console.log('Cannot load business because name or address
                     is invalid')
139              return false;
140         };
141
142         console.log('Adding business (' + _businessName + ') - Please
                check Metamask');
```

```
143           App.returnMaster().then(function (instance) {
144                // call the addBusiness function,
145                // passing the business name and the business wallet
                       address
146                return instance.addBusiness(_businessWalletAddress,
                       _businessName, {
147                  from: App.account,
148                  gas: 5000000
149                });
150           }).then(function (receipt) {
151                console.log(receipt.logs[0].args._businessName + ' added')
                       ;
152                businessContractAddress = receipt.logs[0].args.
                       _contractAddress;
153                console.log('Business contract address: ' +
                       businessContractAddress);
154                return businessContractAddress;
155           }).then(function (contractAddress) {
156                App.setOwnership(contractAddress);
157                return App.displayActiveBusinesses();
158                // log the error if there is one
159           }).catch(function (error) {
160                console.log(error);
161           });
162      },
163
164      //This displays business details in the console for testing purposes
165      displayBusinessConsole: function (contractAddress) {
166           var contractAddress = contractAddress || $('#
                  BusinessContractAddress').val();
167           return App.contracts.Business.at(contractAddress).then(function
                       (instance) {
168                console.log('displaying ' + contractAddress);
169                console.log('business contract address: ' + instance.
                       address);
170                return instance.owner();
171           }).then(function (owner) {
172                console.log('business owner: ' + owner);
173                return App.contracts.Business.at(contractAddress)
174           }).then(function (instance) {
175                return instance.creator();
176           }).then(function (creator) {
177                console.log('business creator: ' + creator);
178           });
179      },
180
181      //This displays all business details linked to the master account if
            the master account is signed in
182      displayActiveBusinesses: function () {
183           // avoid reentry
184           if (App.loading) {
185                return;
186           };
187           App.loading = true;
188
189           // refresh account info
```

```
190           App.displayAccountInfo();
191
192           var businessRow = $('#businessRow');
193           businessRow.empty();
194
195           //define placeholder for contract
196           var masterInstance;
197
198           // check if the account is the master wallet
199
200           App.returnMaster().then(function (instance) {
201                masterInstance = instance;
202                return instance.owner();
203           }).then(function (owner) {
204                if (owner == App.account) {
205                     console.log("Displaying active businesses");
206                     masterInstance.getAllBusinesses().then(function (
                            businessAddresses) {
207                          console.log("Array length " + businessAddresses.
                                length);
208                          businessAddresses.forEach(businessWalletAddress
                                => {
209                               console.log(businessWalletAddress);
210                               return masterInstance.getBusinessDetails(
                                      businessWalletAddress).then(function (
                                      businessDetails) {
211                                    if (businessDetails[2] == true) {
212                                         App.displayBusiness(
213                                              businessDetails[1],
214                                              businessWalletAddress,
215                                              businessDetails[0]
216                                         );
217                                    }
218                               });
219                          });
220                     });
221
222                     App.loading = false;
223
224                } else {
225                     console.log("Only Master Account can display
                            businesses not " + App.Account);
226                }
227           });
228
229
230
231      },
232
233
234      displayBusiness: function (name, address, contract) {
235           var businessRow = $('#businessRow');
236           var businessTemplate = $('#businessTemplate');
237           businessTemplate.find('.business-name').text(name);
238           businessTemplate.find('.business-wallet').text(address);
239           businessTemplate.find('.business-contract').text(contract);
```

```
240
241             //add this business to the placeholder
242             businessRow.append(businessTemplate.html());
243
244         },
245
246
247         setOwnership: function (contractAddress) {
248             var _businessName = $('#BusinessName').val();
249             var _businessWalletAddress = $('#BusinessWalletAddress').val();
250
251             return App.contracts.Business.at(contractAddress).then(function
                    (instance) {
252                 console.log('Setting ownership - Please check Metamask');
253                 return instance.setOwnership(_businessWalletAddress,
                        _businessName, {
254                     from: App.account,
255                     gas: 5000000
256                 });
257             }).then(function (receipt) {
258                 if (receipt.logs[0].event == "ownershipSet") {
259                     console.log('Business contract event' + receipt.logs
                            [0].args._contractAddress);
260                     console.log('Ownership Set event(' + receipt.logs[0].
                            args._businessName + ')');
261                     console.log('Business address event: ' + receipt.logs
                            [0].args._businessWalletAddress);
262                     $('#BusinessName').val('');
263                     $('#BusinessWalletAddress').val('');
264                 } else {
265                     console.log("Wrong event: " + receipt.logs[0].event);
266                 };
267             }).catch(function (error) {
268                 console.log(error);
269             });
270         },
271
272
273         addCustomer: function () {
274             console.log('Add Customer button clicked');
275             // get information from the modal
276             var _customerName = $('#CustomerName').val();
277             var _customerAddress = $('#CustomerAddress').val();
278             var _businessContractAddress;
279
280             // if the name or valid address was not provided
281             if ((_customerName.trim() == '') || (web3.isAddress(
                    _customerAddress) != true)) {
282                 // we cannot add a business
283                 console.log('Cannot load customer because name or address
                        is invalid')
284                 return false;
285             };
286
287             //get business contract address from their wallet = App.account
                    stored on the master contract
```

```
288             App.returnMaster().then(function (masterInstance) {
289                 console.log(App.account);
290                 // Gets the name and contract address of the business
                        linked to account (Contract, Name, Active)
291                 return masterInstance.getBusinessDetails(App.account);
292             }).then(function (businessDetails) {
293                 if (businessDetails[2] != true) {
294                     console.log('This is not an active business. Customer
                            cannot be loaded')
295                 } else
296                     _businessContractAddress = businessDetails[0];
297                 console.log(_businessContractAddress);
298                 // get the instance of the business contract
299                 return App.contracts.Business.at(_businessContractAddress)
                        ;
300             }).then(function (businessInstance) {
301                 console.log('Adding customer (' + _customerName + ') -
                        Please check Metamask');
302                 // call the addCustomer function,
303                 // passing the business name and the business wallet
                        address
304                 return businessInstance.addCustomer(_customerAddress,
                        _customerName, lookupAddress, masterAddress, {
305                     from: App.account,
306                     gas: 5000000
307                 });
308             }).then(function (receipt) {
309                 console.log(receipt.logs[0].args._customerName + ' added')
                        ;
310                 console.log(receipt.logs[0].args._customerAddress + '
                        added');
311                 console.log(receipt.logs[0].args._businessWalletAddress +
                        ' added');
312                 App.displayActiveCustomers();
313                 // log the error if there is one
314             }).catch(function (error) {
315                 console.log(error);
316             });
317         },
318
319         //This displays customer details in the console for testing purposes
320         displayCustomerConsole: function (customerAddress) {
321             var customerAddress = customerAddress || $('#
                    displayCustomerAddress').val();
322             var businessInstance;
323
324             App.returnMaster().then(function (instance) {
325                 console.log(App.account);
326                 return instance.getBusinessDetails(App.account);
327             }).then(function (businessDetails) {
328                 businessContractAddress = businessDetails[0];
329                 console.log(businessContractAddress)
330                 return App.contracts.Business.at(businessContractAddress);
331             }).then(function (instance) {
332                 businessInstance = instance;
333                 console.log('displaying ' + businessContractAddress);
```

```
334                 console.log('business contract address: ' + instance.
                        address);
335                 return businessInstance.getCustomerDetails(customerAddress
                        );
336           }).then(function (customerDetails) {
337                 console.log('customer address', customerAddress);
338                 console.log('customer name', customerDetails[0]);
339                 console.log('customer balance', customerDetails[1]);
340                 console.log('customer active', customerDetails[2]);
341           });
342     },
343
344     //This displays all customers linked to the business account if the
          master account is signed in
345     displayActiveCustomers: function () {
346           // avoid reentry
347           if (App.loading) {
348                 return;
349           };
350           App.loading = true;
351
352           // refresh account info
353           App.displayAccountInfo();
354
355           var customerRow = $('#customerRow');
356           customerRow.empty();
357
358           //define placeholder for contract
359           var businessContractAddress;
360           var businessInstance;
361
362           // check if the account is the master wallet
363
364           App.returnMaster().then(function (instance) {
365                 console.log(App.account);
366                 return instance.getBusinessDetails(App.account);
367           }).then(function (businessDetails) {
368                 businessContractAddress = businessDetails[0];
369                 console.log(businessContractAddress)
370                 return App.contracts.Business.at(businessContractAddress);
371           }).then(function (instance) {
372                 businessInstance = instance;
373                 console.log(businessInstance);
374                 return instance.getAllCustomers();
375           }).then(function (customerAddresses) {
376                 console.log("Array length " + customerAddresses.length);
377                 customerAddresses.forEach(customerWalletAddress => {
378                       console.log(customerWalletAddress);
379                       return businessInstance.getCustomerDetails(
                              customerWalletAddress).then(function (
                              customerDetails) {
380                             console.log(customerDetails);
381                             if (customerDetails[2] == true) {
382                                   App.displayCustomer(
383                                         customerDetails[0],
384                                         customerWalletAddress,
```

```
385                                     customerDetails[1]
386                                 );
387                             };
388                         });
389                     });
390                 });
391     },
392
393
394     displayCustomer: function (name, address, balance) {
395         var customerRow = $('#customerRow');
396         var customerTemplate = $('#customerTemplate');
397         customerTemplate.find('.customer-name').text(name);
398         customerTemplate.find('.customer-wallet').text(address);
399         customerTemplate.find('.customer-balance').text(balance);
400
401         //add this business to the placeholder
402         customerRow.append(customerTemplate.html());
403
404     },
405
406
407     processDocument: function (ipfsHash, docType) {
408         if (docType == "invoice") {
409             var _amount = -1 * $('#invoiceAmount').val();
410             var _customerAddress = $('#invoiceCustomerAddress').val();
411             var _dueDate = $('#invoiceDueDate').val();
412             console.log('Invoicing customer: ', ipfsHash);
413         } else {
414             var _amount = $('#receiptAmount').val();
415             var _customerAddress = $('#receiptCustomerAddress').val();
416             var _dueDate = $('#receiptDueDate').val();
417             console.log('Receipting customer: ', ipfsHash);
418         }
419
420         if (web3.isAddress(_customerAddress) != true) {
421             // we cannot add a business
422             console.log('Cannot process document because address is
                    invalid');
423             return false;
424         };
425
426         //get business contract address from their wallet = App.account
                stored on the master contract
427         App.returnMaster().then(function (masterInstance) {
428             console.log(App.account);
429             // Gets the name and contract address of the business
                    linked to account (Contract, Name, Active)
430             return masterInstance.getBusinessDetails(App.account);
431         }).then(function (businessDetails) {
432             if (businessDetails[2] != true) {
433                 console.log('This is not an active business. Customer
                        cannot be invoiced or receipted')
434             } else
435                 _businessContractAddress = businessDetails[0];
436             console.log(_businessContractAddress);
```

```
437                 // get the instance of the business contract
438                 return App.contracts.Business.at(_businessContractAddress)
                        ;
439           }).then(function (businessInstance) {
440                 console.log('Processing document (' + _customerAddress + '
                        ) - Please check Metamask');
441                 // call the addCustomer function,
442                 // passing the business name and the business wallet
                        address
443                 return businessInstance.processDocument(_customerAddress,
                        _amount, ipfsHash, _dueDate, {
444                   from: App.account,
445                   gas: 4612388
446                 });
447           }).then(function (receipt) {
448                 console.log(receipt.logs[0].args._customerAddress + '
                        added');
449                 console.log("New Balance: " + receipt.logs[0].args.
                        _customerBalance);
450                 console.log("IPFS Hash: " + receipt.logs[0].args._ipfsHash
                        );
451                 $('#invoiceAmount').val('');
452                 $('#invoiceCustomerAddress').val('');
453                 $('#receiptAmount').val('');
454                 $('#receiptCustomerAddress').val('');
455                 App.displayActiveCustomers();
456                 // log the error if there is one
457           }).catch(function (error) {
458                 console.log(error);
459           });
460     },
461
462     //This displays all customers linked to the business account if the
            master account is signed in
463     displayDocuments: function (_customerAddress) {
464           // avoid reentry
465           if (App.loading) {
466                 return;
467           };
468           App.loading = true;
469
470           // refresh account info
471           App.displayAccountInfo();
472           $("#customerScore").text("");
473           $("#viewerCustomerId").text("Customer Score for " +
                  _customerAddress);
474           $("#viewerCustomerScore").text("");
475
476           //define placeholder for contract
477           var businessContractAddress;
478           var total = 0;
479           var masterInstance;
480           var lookupInstance;
481           var customerAddress = _customerAddress || App.account;
482           var businessName;
483           var documentRow = $('#documentRow');
```

```
484           documentRow.empty();
485           var viewerDocumentRow = $('#viewerDocumentRow');
486           viewerDocumentRow.empty();
487
488           App.returnLookup().then(function (instance) {
489                lookupInstance = instance;
490                return App.returnMaster();
491           }).then(function (instance) {
492                masterInstance = instance;
493                count = 0;
494                return lookupInstance.getCustomerBusinessList(
                       customerAddress);
495           }).then(function (businessList) {
496                console.log("Business List length " + businessList.length)
                       ;
497                businessList.forEach(businessWalletAddress => {
498                     console.log(businessWalletAddress);
499                     masterInstance.getBusinessDetails(
                           businessWalletAddress).then(function (
                           businessDetails) {
500                       businessContractAddress = businessDetails[0];
501                       businessName = businessDetails[1];
502                       console.log(businessName);
503                       return App.generateDocumentsByBusiness(
                              businessContractAddress, businessName,
                              customerAddress).then(function(
                              totalForBusiness){
504                            if (totalForBusiness< 0) {
505                            console.log("Total for "+businessName+": "+
                                   totalForBusiness)
506                            total = total + totalForBusiness;
507                            }
508                       });
509                });
510           });
511           console.log("Overall total " + total);
512           $("#customerScore").text(total);
513           $("#viewerCustomerScore").text(total);
514      });
515      },
516
517      generateDocumentsByBusiness: function(_businessContractAddress,
             _businessName, _customerAddress){
518           var businessInstance;
519           var customerAddress = _customerAddress || App.account;
520           var totalForBusiness = 0;
521           var dueDate;
522           var ipfsHash;
523           var documentAmount;
524           var documentType;
525
526           return App.contracts.Business.at(_businessContractAddress).then
                  (function (instance) {
527           businessInstance = instance;
528           console.log(_businessName);
529           return businessInstance.getCustomerDetails(customerAddress);
```

```
530     }).then(async function (customerDetails) {
531          //Add the balance if the customer owes money
532          if (customerDetails[1] <= 0) {
533               let totalForBusiness = await totalForBusiness +
                      customerDetails[1];
534               console.log('Current total for business', totalForBusiness
                      );
535          }
536          return businessInstance.getCustomerDocumentsLength(
                 customerAddress);
537     }).then(function (documentCount) {
538          console.log("number of documents for " + _businessName +" "  +
                 documentCount)
539          for (let i = 0; i < documentCount; i++) {
540               console.log("generating " + i)
541               businessInstance.getCustomerDocument(customerAddress, i).
                      then(function (document) {
542                    ipfsHash = document[0];
543                    documentAmount =  document[1];
544                    dueDate =   document[2];
545                    if (documentAmount < 0) {
546                         documentType = "Invoice";
547                    } else {
548                         documentType = "Receipt";
549                    }
550                    console.log("displaying "+ _businessContractAddress +
                           i);
551                    if (customerAddress == App.account){
552                         App.displayDocumentToCustomer(
553                              dueDate,
554                              _businessName,
555                              documentAmount,
556                              documentType,
557                              ipfsHash
558                         );
559                    }else {
560                         App.displayDocumentToViewer(
561                              dueDate,
562                              _businessName,
563                              documentAmount,
564                              documentType
565                         );
566                    }
567               });
568          };
569          return totalForBusiness;
570     });
571     },
572
573
574     displayDocumentToCustomer: function (date, businessName,
            documentAmount, documentType, ipfsHash) {
575          var documentRow = $('#documentRow');
576          var documentTemplate = $('#documentTemplate');
577          console.log("displaying to customer portal");
578          documentTemplate.find('.document-date').text(date);
```

```
579           documentTemplate.find('.document-business').text(businessName);
580           documentTemplate.find('.document-amount').text(documentAmount);
581           documentTemplate.find('.document-type').text(documentType);
582           documentTemplate.find('.document-ipfs').text(ipfsHash);
583
584           //add this document to the placeholder
585           documentRow.append(documentTemplate.html());
586       },
587
588       displayDocumentToViewer: function (date, businessName,
              documentAmount, documentType) {
589           var viewerDocumentRow = $('#viewerDocumentRow');
590           var viewerDocumentTemplate = $('#viewerDocumentTemplate');
591           console.log("displaying to viewer portal");
592           viewerDocumentTemplate.find('.viewer-document-date').text(date)
                  ;
593           viewerDocumentTemplate.find('.viewer-document-business').text(
                  businessName);
594           viewerDocumentTemplate.find('.viewer-document-amount').text(
                  documentAmount);
595           viewerDocumentTemplate.find('.viewer-document-type').text(
                  documentType);
596
597           //add this document to the placeholder
598           viewerDocumentRow.append(viewerDocumentTemplate.html());
599       },
600
601       addPermission: function(){
602           console.log('Add Permission button clicked');
603           // get information from the modal
604           var _permissionAddress = $('#permissionAddress').val();
605           var _permissionValidity = $('#permissionValidity').val();
606
607           var _permissionValidityConverted = new Date(_permissionValidity
                  ).getTime() / 1000;
608           console.log(_permissionValidityConverted);
609           console.log(Date.now());
610           return App.returnLookup().then(function(lookupInstance){
611               lookupInstance.addPermissionToCustomerList(
                      _permissionAddress, _permissionValidityConverted, {
612                     from: App.account,
613                     gas: 5000000
614               });
615               console.log('Permission Added');
616           });
617       },
618
619       viewDocumentsAsExternal: function(){
620           var customerAddress = $('#viewerRequestedAddress').val();
621           return App.returnLookup().then(function(lookupInstance){
622               return lookupInstance.checkPermission(customerAddress, App
                      .account);
623           }).then(function(result){
624               console.log(result);
625               if(result == true){
626                   App.displayDocuments(customerAddress);
```

```
627                     console.log("Documents Displayed")
628                 } else {
629                     console.log("You do not have permission to view these
                            documents")
630                     $("#viewerCustomerId").text("You do not have
                            permission to view " + customerAddress);
631                 };
632             });
633         },
634
635         uploadInvoice: function() {
636
637             const reader = new FileReader();
638
639             // Set up the on load end function
640             reader.onloadend = function (r, ev) {
641                 const ipfs = new IPFS({host:'ipfs.infura.io', port: 5001,
                        protocol: 'https' }); // Connect to IPFS
642                 const buf = buffer.from(reader.result); // Convert data
                        into buffer
643                 ipfs.add(buf, (err, result) => { // Upload buffer to IPFS
644                     if (err) {
645                         console.error(err)
646                         return
647                     }
648                     let url = `https://ipfs.io/ipfs/${result[0].hash}`
649                     console.log(`Url --> ${url}`)
650                     App.processDocument(result[0].hash, "invoice");
651                 })
652             }
653
654             // Find the file and run the file reader on it
655             const invoice = document.getElementById("invoice");
656             setTimeout(() => reader.readAsArrayBuffer(invoice.files[0]),
                    500); // Read Provided File
657         },
658
659         uploadReceipt: function() {
660             const reader = new FileReader();
661
662             // Set up the on load end function
663             reader.onloadend = function (r, ev) {
664                 const ipfs = new IPFS({host:'ipfs.infura.io', port: 5001,
                        protocol: 'https' }); // Connect to IPFS
665                 const buf = buffer.from(reader.result); // Convert data
                        into buffer
666                 ipfs.add(buf, (err, result) => { // Upload buffer to IPFS
667                     if (err) {
668                         console.error(err)
669                         return
670                     }
671                     let url = `https://ipfs.io/ipfs/${result[0].hash}`
672                     console.log(`Url --> ${url}`)
673                     App.processDocument(result[0].hash, "receipt");
674                 })
675             }
```

```
676
677             // Find the file and run the file reader on it
678             const receipt = document.getElementById("receipt");
679             setTimeout(() => reader.readAsArrayBuffer(receipt.files[0]),
                    500); // Read Provided File
680         },
681
682 };
683
684 $(function () {
685     $(window).load(function () {
686         App.init();
687     });
688     // placeholder for current account
689     var _account;
690
691     // set the interval
692     var accountInterval = setInterval(function () {
693         // check for new account information and display it
694         App.displayAccountInfo();
695         //console.log("loading")
696         // only reload the contract info if account has changed
697         if (_account != App.account) {
698             App.displayContractInfo();
699             // update the current account
700             _account = App.account;
701         }
702     }, 10);
703 });
```

B.3 HTML Code

B.3.1 CreditRegister Home Page

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other
        head content must come *after* these tags -->
    <title>CreditRegister</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
        media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file
        :// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.
          js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"><
          /script>
    <![endif]-->

    <!-- Application -->
    <link href="css/app.css" rel="stylesheet">

</head>

<body>

    <ul class="nav nav-pills">
        <li class="nav-item">
            <a class="nav-link active" href="index.html">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="masterPortal.html">Master Portal</a
                >
        </li>
        <li class="nav-item">
            <a class="nav-link" href="businessPortal.html">Business
                Portal</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="customerPortal.html">Customer
                Portal</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="viewerPortal.html">Viewer Portal</a
                >
        </li>
```

```
44      </ul>
45
46      <!-- Jumbotron with heading -->
47      <div class="container">
48          <div class="jumbotron text-center">
49              <h1>Welcome to CreditRegister</h1>
50          </div>
51          <div class="container" id="current-details">
52              <!--<div class="row">
53                  <div class="col-lg-12">
54                      <p id="message" class="welcome pull-left"></p>
55                      <p id="account2" class="welcome pull-right"></p>
56                      <p id="accountBalance2" class="welcome pull-left"></p
                            >
57                  </div>
58              </div>-->
59              <div class="row">
60                  <div class="row">
61                      <div class="col-lg-4"><strong>Your Wallet ID:</strong
                            ></div>
62                      <div class="col-lg-8" id="account"></div>
63                  </div>
64                  <div class="row">
65                          <div class="col-lg-4 font-weight-bold"><strong>
                                Your Account Balance:</strong></div>
66                          <div class="col-lg-8" id="accountBalance"></div>
67                  </div>
68                  <div class="row">
69                          <div class="col-lg-4 font-weight-bold"><strong>
                                Current Master Contract ID:</strong></div>
70                          <div class="col-lg-8" id="masterContract"></div>
71                  </div>
72                  <div class="row">
73                          <div class="col-lg-4 font-weight-bold"><strong>
                                Current Lookup Contract ID:</strong></div>
74                          <div class="col-lg-8" id="lookupContract"></div>
75                  </div>
76                  <!--<div class="row">
77                          <div class="col-lg-4 font-weight-bold"><strong>
                                IPFS Status:</strong></div>
78                          <div class="col-lg-8" id="ipfsStatus"></div>-->
79                  </div>
80              </div>
81
82
83          </div>
84      </div>
85
86      <div id="footer" class="container">
87          <nav class="navbar navbar-default navbar-fixed-bottom">
88              <div class="navbar-inner navbar-content-center text-center">
89                  <p class="text-muted" credit><a href="https://github.com/
                        bryonySA/Thesis.git">CreditRegister</a> -
90                        &copy; 2018 - <a href="https://github.com/bryonySA">
                            Bryony Ortlepp</a></p>
91              </div>
```

```
92          </nav>
93      </div>
94
95
96      <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
97      <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/
            jquery.min.js"></script>
98      <!-- Include all compiled plugins (below), or include individual
            files as needed -->
99      <script src="js/bootstrap.min.js"></script>
100     <script src="https://unpkg.com/ipfs/dist/index.min.js"></script>
101     <script src="js/main.js"></script>
102 </body>
103
104 </html>
```

B.3.2 CreditRegister Master Portal

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other
        head content must come *after* these tags -->
    <title>CreditRegister - Master Portal</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
        media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file
        :// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.
          js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"><
          /script>
    <![endif]-->

    <!-- Application -->
    <link href="css/app.css" rel="stylesheet">

</head>

<body>
    <!-- Nav Bar -->
    <ul class="nav nav-pills">
        <li class="nav-item">
            <a class="nav-link" href="index.html">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link active" href="masterPortal.html">Master
                Portal</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="businessPortal.html">Business
                Portal</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="customerPortal.html">Customer
                Portal</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="viewerPortal.html">Viewer Portal</a
                >
        </li>
    </ul>

```

```
46      <!-- Modal form to add a business -->
47      <div class="modal fade" id="addBusiness" role="dialog">
48          <div class="modal-dialog">
49
50              <!-- Modal content-->
51              <div class="modal-content">
52                  <div class="modal-header">
53                      <button type="button" class="close" data-dismiss="
                            modal">&times;</button>
54                      <h4 class="modal-title">Add a business</h4>
55                  </div>
56                  <div class="modal-body">
57
58                      <div class="row">
59                          <div class="col-lg-12">
60                              <form>
61                                  <div class="form-group">
62                                      <label for="business_name">Business
                                            name</label>
63                                      <input type="text" class="form-
                                            control" id="BusinessName"
                                            placeholder="Enter the name of the
                                             business to add.">
64                                      <label for="business_address">
                                            Business address</label>
65                                      <input type="text" class="form-
                                            control" id="BusinessWalletAddress
                                            " placeholder="Enter the wallet
                                            address of the business to add.">
66                                  </div>
67                              </form>
68                          </div>
69                      </div>
70                  </div>
71                  <div class="modal-footer">
72                      <button type="button" class="btn btn-primary btn-
                            success" data-dismiss="modal" onclick="App.
                            addBusiness(); return false;">Submit</button>
73                      <button type="button" class="btn" data-dismiss="modal
                            ">Close</button>
74                  </div>
75              </div>
76
77          </div>
78      </div>
79
80      <!-- Modal form to display a business -->
81      <div class="modal fade" id="displayBusinessConsole" role="dialog">
82          <div class="modal-dialog">
83
84              <!-- Modal content-->
85              <div class="modal-content">
86                  <div class="modal-header">
87                      <button type="button" class="close" data-dismiss="
                            modal">&times;</button>
88                      <h4 class="modal-title">Display a business</h4>
```

```
89                  </div>
90                  <div class="modal-body">
91
92                      <div class="row">
93                          <div class="col-lg-12">
94                              <form>
95                                  <div class="form-group">
96                                      <label for="business_contract">
                                            Business Contract Address</label>
97                                      <input type="text" class="form-
                                            control" id="
                                            BusinessContractAddress"
                                            placeholder="Enter the contract
                                            address of the business to display
                                            .">
98                                  </div>
99                              </form>
100                         </div>
101                     </div>
102                 </div>
103                 <div class="modal-footer">
104                     <button type="button" class="btn btn-primary btn-
                            success" data-dismiss="modal" onclick="App.
                            displayBusinessConsole(); return false;">Submit</
                            button>
105                     <button type="button" class="btn" data-dismiss="modal
                            ">Close</button>
106                 </div>
107             </div>
108
109         </div>
110     </div>
111
112
113
114     <!-- Jumbotron showing account details and Add business button -->
115     <div class="container">
116         <div class="jumbotron text-center">
117             <h1>CreditRegister</h1>
118         </div>
119         <div class="col-md-12" id="business-list">
120             <div class="row">
121                 <div class="col-lg-12">
122                     <p id="message" class="welcome pull-left"></p>
123                     <p id="account" class="welcome pull-right"></p>
124                     <p id="accountBalance" class="welcome pull-left"></p>
125                 </div>
126             </div>
127
128             <div class="row">
129                 <button class="btn btn-info btn-lg btn-create" data-
                        toggle="modal" data-target="#addBusiness">Add a
                        business</button>
130                 <button class="btn btn-info btn-lg btn-create" data-
                        toggle="modal" data-target="#displayBusinessConsole">
                        Display
```

```
132                     details in console</button>
133                 <button class="btn btn-info btn-lg btn-create" data-
                        toggle="modal" onclick="App.displayActiveBusinesses();
                         return false;">Display
134                     all businesses</button>
135             </div>
136
137
138             <div id="businessRow" class="row">
139                 <!-- BUSINESSES LOAD HERE -->
140             </div>
141         </div>
142         <div id="businessTemplate" style="display: none;">
143             <div class="row-lg-12">
144                 <div class="panel panel-default panel-article">
145                     <div class="panel-heading">
146                         <h3 class="business-name"></h3>
147                     </div>
148                     <div class="panel-body">
149                         <strong>Wallet ID</strong>:
150                         <span class="business-wallet"></span>
151                         <br />
152                         <strong>Contract ID</strong>:
153                         <span class="business-contract"></span>
154                     </div>
155                 </div>
156             </div>
157         </div>
158
159     </div>
160
161
162     <div id="footer" class="container">
163         <nav class="navbar navbar-default navbar-fixed-bottom">
164             <div class="navbar-inner navbar-content-center text-center">
165                 <p class="text-muted" credit><a href="https://github.com/
                        bryonySA/Thesis.git">CreditRegister</a> -
166                     &copy; 2018 - <a href="https://github.com/bryonySA">
                            Bryony Ortlepp</a></p>
167             </div>
168         </nav>
169     </div>
170
171
172      <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
173      <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/
            jquery.min.js"></script>
174      <!-- Include all compiled plugins (below), or include individual
            files as needed -->
175      <script src="js/bootstrap.min.js"></script>
176      <script src="https://unpkg.com/ipfs/dist/index.min.js"></script>
177      <script src="js/main.js"></script>
178 </body>
179
180 </html>
```

B.3.3 CreditRegister Business Portal

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other
        head content must come *after* these tags -->
    <title>CreditRegister</title>


    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
        media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file
        :// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.
          js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"><
          /script>
    <![endif]-->

    <!-- Application -->
    <link href="css/app.css" rel="stylesheet">

    <!--<script src="https://wzrd.in/standalone/buffer"></script>-->
    <script src="https://unpkg.com/ipfs-api@9.0.0/dist/index.js"
    integrity="sha384-5bXRcW9kyxxnSMbOoHzraqa7Z0PQWIao+
        cgeg327zit1hz5LZCEbIMx/LWKPReuB"
    crossorigin="anonymous"></script>
</head>

<body>
    <ul class="nav nav-pills">
        <li class="nav-item">
            <a class="nav-link" href="index.html">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="masterPortal.html">Master Portal</a
                >
        </li>
        <li class="nav-item">
            <a class="nav-link active" href="businessPortal.html">
                Business Portal</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="customerPortal.html">Customer
                Portal</a>
        </li>
        <li class="nav-item">
```

```
46              <a class="nav-link" href="viewerPortal.html">Viewer Portal</a
                    >
47          </li>
48      </ul>
49
50
51      <!-- Modal form to add a customer -->
52      <div class="modal fade" id="addCustomer" role="dialog">
53          <div class="modal-dialog">
54
55              <!-- Modal content-->
56              <div class="modal-content">
57                  <div class="modal-header">
58                      <button type="button" class="close" data-dismiss="
                            modal">&times;</button>
59                      <h4 class="modal-title">Add a customer</h4>
60                  </div>
61                  <div class="modal-body">
62
63                      <div class="row">
64                          <div class="col-lg-12">
65                              <form>
66                                  <div class="form-group">
67                                      <label for="customer_name">Customer
                                            name</label>
68                                      <input type="text" class="form-
                                            control" id="CustomerName"
                                            placeholder="Enter the name of the
                                             customer to add.">
69                                      <label for="customer_address">
                                            Customer address</label>
70                                      <input type="text" class="form-
                                            control" id="CustomerAddress"
                                            placeholder="Enter the wallet
                                            address of the customer to add.">
71                                  </div>
72                              </form>
73                          </div>
74                      </div>
75                  </div>
76                  <div class="modal-footer">
77                      <button type="button" class="btn btn-primary btn-
                            success" data-dismiss="modal" onclick="App.
                            addCustomer(); return false;">Submit</button>
78                      <button type="button" class="btn" data-dismiss="modal
                            ">Close</button>
79                  </div>
80              </div>
81
82          </div>
83      </div>
84
85      <!-- Modal form to display a customer in console -->
86      <div class="modal fade" id="displayCustomer" role="dialog">
87          <div class="modal-dialog">
88
```

```
89              <!-- Modal content-->
90              <div class="modal-content">
91                  <div class="modal-header">
92                      <button type="button" class="close" data-dismiss="
                            modal">&times;</button>
93                      <h4 class="modal-title">Add a customer</h4>
94                  </div>
95                  <div class="modal-body">
96
97                      <div class="row">
98                          <div class="col-lg-12">
99                              <form>
100                                 <div class="form-group">
101                                     <label for="customer_address">
                                            Customer address</label>
102                                     <input type="text" class="form-
                                            control" id="
                                            displayCustomerAddress"
                                            placeholder="Enter the wallet
                                            address of the customer to display
                                            .">
103                                 </div>
104                             </form>
105                         </div>
106                     </div>
107                 </div>
108                 <div class="modal-footer">
109                     <button type="button" class="btn btn-primary btn-
                            success" data-dismiss="modal" onclick="App.
                            displayCustomerConsole(); return false;">Submit</
                            button>
110                     <button type="button" class="btn" data-dismiss="modal
                            ">Close</button>
111                 </div>
112             </div>
113
114         </div>
115     </div>
116
117
118     <!-- Modal form to invoice a customer  -->
119     <div class="modal fade" id="invoiceCustomer" role="dialog">
120         <div class="modal-dialog">
121
122             <!-- Modal content-->
123             <div class="modal-content">
124                 <div class="modal-header">
125                     <button type="button" class="close" data-dismiss="
                            modal">&times;</button>
126                     <h4 class="modal-title">Invoice a customer</h4>
127                 </div>
128                 <div class="modal-body">
129
130                     <div class="row">
131                         <div class="col-lg-12">
132                             <form>
```

```
133                                 <div class="form-group">
134                                     <label for="invoice_address">Customer
                                            address</label>
135                                     <input type="text" class="form-
                                            control" id="
                                            invoiceCustomerAddress"
                                            placeholder="Enter the wallet
                                            address of the customer to invoice
                                            .">
136                                     <label for="invoice_amount">Invoice
                                            Amount</label>
137                                     <input type="text" class="form-
                                            control" id="invoiceAmount"
                                            placeholder="Enter the invoice
                                            amount.">
138                                     <label for="invoice_dueDate">Due Date
                                            </label>
139                                     <input type="date" class="form-
                                            control" id="invoiceDueDate"
                                            placeholder="Enter the invoice due
                                             date.">
140                                     <label for="invoice_upload">Please
                                            upload invoice</label>
141                                     <input type="file" name="invoice" id=
                                            "invoice">
142                                 </div>
143                             </form>
144                         </div>
145                     </div>
146                 </div>
147                 <div class="modal-footer">
148                     <button type="button" class="btn btn-primary btn-
                            success" onclick="App.uploadInvoice();">Upload
                            Invoice</button>
149                     <button type="button" class="btn" data-dismiss="modal
                            ">Close</button>
150                 </div>
151             </div>
152
153         </div>
154     </div>
155
156     <!-- Modal form to invoice a customer  -->
157     <div class="modal fade" id="receiptCustomer" role="dialog">
158         <div class="modal-dialog">
159
160             <!-- Modal content-->
161             <div class="modal-content">
162                 <div class="modal-header">
163                     <button type="button" class="close" data-dismiss="
                            modal">&times;</button>
164                     <h4 class="modal-title">Receipt a customer</h4>
165                 </div>
166                 <div class="modal-body">
167
168                     <div class="row">
```

```
169                         <div class="col-lg-12">
170                             <form>
171                                 <div class="form-group">
172                                     <label for="receipt_address">Customer
                                            address</label>
173                                     <input type="text" class="form-
                                        control" id="
                                        receiptCustomerAddress"
                                        placeholder="Enter the wallet
                                        address of the customer to receipt
                                        .">
174                                     <label for="receipt_amount">Receipt
                                        Amount</label>
175                                     <input type="text" class="form-
                                        control" id="receiptAmount"
                                        placeholder="Enter the receipt
                                        amount.">
176                                     <label for="receipt_dueDate">Payment
                                        Date</label>
177                                     <input type="date" class="form-
                                        control" id="receiptDueDate"
                                        placeholder="Enter the payment
                                        date.">
178                                     <label for="receipt_upload">Please
                                        upload receipt</label>
179                                     <input type="file" name="receipt" id=
                                        "receipt">
180                                 </div>
181                             </form>
182                         </div>
183                     </div>
184                 </div>
185                 <div class="modal-footer">
186                     <button type="button" class="btn btn-primary btn-
                            success" data-dismiss="modal" onclick="App.
                            uploadReceipt(); return false;">Upload Receipt</
                            button>
187                     <button type="button" class="btn" data-dismiss="modal
                            ">Close</button>
188                 </div>
189             </div>
190 
191         </div>
192     </div>
193 
194     <!-- Jumbotron showing account details and Add customer button -->
195     <div class="container">
196         <div class="jumbotron text-center">
197             <h1>CreditRegister</h1>
198         </div>
199         <div class="col-md-12" id="business-list">
200             <div class="row">
201                 <div class="col-lg-12">
202                     <p id="message" class="welcome pull-left"></p>
203                     <p id="account" class="welcome pull-right"></p>
204                     <p id="accountBalance" class="welcome pull-left"></p>
```

```
205                 </div>
206             </div>
207
208             <div class="row">
209                 <button class="btn btn-info btn-lg btn-create" data-
                        toggle="modal" data-target="#addCustomer">Add a
210                     customer</button>
211                 <button class="btn btn-info btn-lg btn-create" data-
                        toggle="modal" data-target="#displayCustomer">Display
212                     details in console</button>
213                 <button class="btn btn-info btn-lg btn-create" data-
                        toggle="modal" onclick="App.displayActiveCustomers();
                        return false;">Display
214                     all customers</button>
215                 <button class="btn btn-info btn-lg btn-create" data-
                        toggle="modal" data-target="#invoiceCustomer">Invoice
216                     customer</button>
217                 <button class="btn btn-info btn-lg btn-create" data-
                        toggle="modal" data-target="#receiptCustomer">Receipt
218                     customer</button>
219
220             </div>
221
222             <div id="customerRow" class="row">
223                 <!-- CUSTOMERS LOAD HERE -->
224             </div>
225         </div>
226
227         <div id="customerTemplate" style="display: none;">
228             <div class="row-lg-12">
229                 <div class="panel panel-default panel-article">
230                     <div class="panel-heading">
231                         <h3 class="customer-name"></h3>
232                     </div>
233                     <div class="panel-body">
234                         <strong>Customer Wallet</strong>:
235                         <span class="customer-wallet"></span>
236                         <br />
237                         <strong>Balance</strong>:
238                         <span class="customer-balance"></span>
239                     </div>
240                     <!--  <div class="panel-footer">
241                         <button type="button" class="btn btn-primary btn-
                                success btn-addCustomer" onclick="App.
                                addCustomer(this); return false;">Add Customer
                                </button>
242                     </div> -->
243                 </div>
244             </div>
245         </div>
246
247     </div>
248
249
250     <div id="footer" class="container">
251         <nav class="navbar navbar-default navbar-fixed-bottom">
```

```
252             <div class="navbar-inner navbar-content-center text-center">
253                 <p class="text-muted" credit><a href="https://github.com/
                        bryonySA/Thesis.git">CreditRegister</a> -
254                     &copy; 2018 - <a href="https://github.com/bryonySA">
                            Bryony Ortlepp</a></p>
255             </div>
256         </nav>
257     </div>
258
259
260     <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
261     <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/
            jquery.min.js"></script>
262     <!-- Include all compiled plugins (below), or include individual
            files as needed -->
263     <script src="js/bootstrap.min.js"></script>
264     <script src="https://unpkg.com/ipfs/dist/index.min.js"></script>
265     <script src="js/main.js"></script>
266 </body>
267
268 </html>
```

B.3.4 CreditRegister Customer Portal

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other
        head content must come *after* these tags -->
    <title>CreditRegister</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
        media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file
        :// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.
          js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"><
          /script>
    <![endif]-->

    <!-- Application -->
    <link href="css/app.css" rel="stylesheet">

</head>

<body>
    <ul class="nav nav-pills">
        <li class="nav-item">
            <a class="nav-link" href="index.html">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="masterPortal.html">Master Portal</a
                >
        </li>
        <li class="nav-item">
            <a class="nav-link" href="businessPortal.html">Business
                Portal</a>
        </li>
        <li class="nav-item">
            <a class="nav-link active" href="customerPortal.html">
                Customer Portal</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="viewerPortal.html">Viewer Portal</a
                >
        </li>
    </ul>


```

```
46
47      <!-- Jumbotron showing account details and Add business button -->
48      <div class="container">
49          <div class="jumbotron text-center">
50              <h1>CreditRegister</h1>
51          </div>
52          <div class="col-md-12" id="customer-list">
53              <div class="row">
54                  <div class="col-lg-12">
55                      <p id="message" class="welcome pull-left"></p>
56                      <p id="account" class="welcome pull-right"></p>
57                      <p id="accountBalance" class="welcome pull-left"></p>
58
59                  </div>
60              </div>
61              <div class="row">
62                  <div class="col-lg-4 font-weight-bold"><strong>Your
                        Current Score:</strong></div>
63                  <div class="col-lg-8" id="customerScore"></div>
64              </div>
65          </div>
66          <div class="row">
67              <button class="btn btn-info btn-lg btn-create" data-toggle="
                    modal" onclick="App.displayDocuments(); return false;">
                    Display
68                  Documents</button>
69              <button class="btn btn-info btn-lg btn-create" data-toggle="
                    modal" data-target=#addPermission>Give View
70                  Rights</button>
71          </div>
72
73          <div id="documentRow" class="row">
74              <!-- Invoices LOAD HERE -->
75          </div>
76      </div>
77      <div id="documentTemplate" style="display: none;">
78          <div class="row-lg-12">
79              <div class="panel panel-default panel-article">
80                  <div class="panel-heading">
81                      <h3 class="document-date"></h3>
82                  </div>
83                  <div class="panel-body">
84                      <strong>Business Name</strong>:
85                      <span class="document-business"></span>
86                      <br />
87                      <strong>Document Amount</strong>:
88                      <span class="document-amount"></span>
89                      <br />
90                      <strong>Document Type</strong>:
91                      <span class="document-type"></span>
92                      <br />
93                      <strong>IPFS Hash</strong>:
94                      <span class="document-ipfs"></span>
95                  </div>
96                  <!--  <div class="panel-footer">
97                          <button type="button" class="btn btn-primary btn-
```

```
                            success btn-addCustomer" onclick="App.
                            addCustomer(this); return false;">Add Customer
                            </button>
98                      </div> -->
99              </div>
100         </div>
101     </div>
102
103     </div>
104
105     <!-- Modal form to add permission  -->
106     <div class="modal fade" id="addPermission" role="dialog">
107         <div class="modal-dialog">
108
109             <!-- Modal content-->
110             <div class="modal-content">
111                 <div class="modal-header">
112                     <button type="button" class="close" data-dismiss="
                            modal">&times;</button>
113                     <h4 class="modal-title">Give an account view rights</
                            h4>
114                 </div>
115                 <div class="modal-body">
116
117                     <div class="row">
118                         <div class="col-lg-12">
119                             <form>
120                                 <div class="form-group">
121                                     <label for="permission_address">
                                            Viewers address</label>
122                                     <input type="text" class="form-
                                            control" id="permissionAddress"
                                            placeholder="Enter the wallet
                                            address of the viewer.">
123                                     <label for="permission_validity">End
                                            Date of View Permission</label>
124                                     <input type="date" class="form-
                                            control" id="permissionValidity"
                                            placeholder="Enter the date that
                                            the viewer has permission until.">
125                                 </div>
126                             </form>
127                         </div>
128                     </div>
129                 </div>
130                 <div class="modal-footer">
131                     <button type="button" class="btn btn-primary btn-
                            success" data-dismiss="modal" onclick="App.
                            addPermission(); return false;">Load
132                         Permission</button>
133                     <button type="button" class="btn" data-dismiss="modal
                            ">Close</button>
134                 </div>
135             </div>
136
137         </div>
```

```
138     </div>
139
140
141     <div id="footer" class="container">
142         <nav class="navbar navbar-default navbar-fixed-bottom">
143             <div class="navbar-inner navbar-content-center text-center">
144                 <p class="text-muted" credit><a href="https://github.com/
                        bryonySA/Thesis.git">CreditRegister</a> -
145                     &copy; 2018 - <a href="https://github.com/bryonySA">
                            Bryony Ortlepp</a></p>
146             </div>
147         </nav>
148     </div>
149
150
151     <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
152     <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/
            jquery.min.js"></script>
153     <!-- Include all compiled plugins (below), or include individual
            files as needed -->
154     <script src="js/bootstrap.min.js"></script>
155     <script src="https://unpkg.com/ipfs/dist/index.min.js"></script>
156     <script src="js/main.js"></script>
157 </body>
158
159 </html>
```

B.3.5 CreditRegister Viewer Portal

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other
        head content must come *after* these tags -->
    <title>CreditRegister</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
        media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file
        :// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.
          js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"><
          /script>
    <![endif]-->

    <!-- Application -->
    <link href="css/app.css" rel="stylesheet">

</head>

<body>
    <ul class="nav nav-pills">
        <li class="nav-item">
            <a class="nav-link" href="index.html">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="masterPortal.html">Master Portal</a
                >
        </li>
        <li class="nav-item">
            <a class="nav-link" href="businessPortal.html">Business
                Portal</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="customerPortal.html">Customer
                Portal</a>
        </li>
        <li class="nav-item">
            <a class="nav-link active" href="viewerPortal.html">Viewer
                Portal</a>
        </li>
    </ul>


```

```
46
47      <!-- Jumbotron showing account details and Add business button -->
48      <div class="container">
49          <div class="jumbotron text-center">
50              <h1>CreditRegister</h1>
51          </div>
52          <div class="col-md-12" id="business-list">
53              <div class="row">
54                  <div class="col-lg-12">
55                      <p id="message" class="welcome pull-left"></p>
56                      <p id="account" class="welcome pull-right"></p>
57                      <p id="accountBalance" class="welcome pull-left"></p>
58
59                  </div>
60              </div>
61
62              <div class="row">
63                      <div class="row">
64                              <div class="col-lg-4 font-weight-bold" id="
                                    viewerCustomerId"><strong></strong></div>
65                              <div class="col-lg-8" id="viewerCustomerScore
                                    "></div>
66                          </div>
67
68                  <button class="btn btn-info btn-lg btn-create" data-
                        toggle="modal" data-target=#viewDocuments>View
69                      Documents</button>
70              </div>
71
72              <div id="viewerDocumentRow" class="row">
73                  <!-- DOCUMENTS LOAD HERE -->
74              </div>
75          </div>
76          <div id="viewerDocumentTemplate" style="display: none;">
77              <div class="row-lg-12">
78                  <div class="panel panel-default panel-article">
79                      <div class="panel-heading">
80                          <h3 class="viewer-document-date"></h3>
81                      </div>
82                      <div class="panel-body">
83                          <strong>Business Name</strong>:
84                          <span class="viewer-document-business"></span>
85                          <br />
86                          <strong>Document Amount</strong>:
87                          <span class="viewer-document-amount"></span>
88                          <br />
89                          <strong>Document Type</strong>:
90                          <span class="viewer-document-type"></span>
91                      </div>
92                      <!--  <div class="panel-footer">
93                          <button type="button" class="btn btn-primary btn-
                                success btn-addCustomer" onclick="App.
                                addCustomer(this); return false;">Add Customer
                                </button>
94                      </div> -->
95                  </div>
```

```
96              </div>
97          </div>
98
99      </div>
100
101     <!-- Modal form to view documents  -->
102     <div class="modal fade" id="viewDocuments" role="dialog">
103         <div class="modal-dialog">
104
105             <!-- Modal content-->
106             <div class="modal-content">
107                 <div class="modal-header">
108                     <button type="button" class="close" data-dismiss="
                            modal">&times;</button>
109                     <h4 class="modal-title">View A Cusotmers Documents</
                            h4>
110                 </div>
111                 <div class="modal-body">
112
113                     <div class="row">
114                         <div class="col-lg-12">
115                             <form>
116                                 <div class="form-group">
117                                     <label for="viewer_requested_address"
                                            >Customers address</label>
118                                     <input type="text" class="form-
                                            control" id="
                                            viewerRequestedAddress"
                                            placeholder="Enter the wallet
                                            address of the customer you'd like
                                             to view.">
119                                 </div>
120                             </form>
121                         </div>
122                     </div>
123                 </div>
124                 <div class="modal-footer">
125                     <button type="button" class="btn btn-primary btn-
                            success" data-dismiss="modal" onclick="App.
                            viewDocumentsAsExternal(); return false;">View
                            Credit History</button>
126                     <button type="button" class="btn" data-dismiss="modal
                            ">Close</button>
127                 </div>
128             </div>
129
130         </div>
131     </div>
132
133
134     <div id="footer" class="container">
135         <nav class="navbar navbar-default navbar-fixed-bottom">
136             <div class="navbar-inner navbar-content-center text-center">
137                 <p class="text-muted" credit><a href="https://github.com/
                        bryonySA/Thesis.git">CreditRegister</a> -
                        &copy; 2018 - <a href="https://github.com/bryonySA">
138
```

```
                        Bryony Ortlepp</a></p>
139             </div>
140         </nav>
141     </div>
142
143
144     <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
145     <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/
            jquery.min.js"></script>
146     <!-- Include all compiled plugins (below), or include individual
            files as needed -->
147     <script src="js/bootstrap.min.js"></script>
148     <script src="https://unpkg.com/ipfs/dist/index.min.js"></script>
149     <script src="js/main.js"></script>
150 </body>
151
152 </html>
```