Feasibility study of using blockchain to improve transparency and trust in the charity industry

Julika Pahl

PHLJUL003

Submitted to the Department of Commerce at the University of Cape Town in fulfilment of the academic requirements for a Master of Philosophy degree in Financial Technology.

April 14, 2021

Abstract

In 2012, the UN Secretary stated that corruption prevented 30 percent of all development assistance from reaching its destination (UNSG, 2012). This thesis discusses the importance of trust and transparency in the charity sector, and how technology, specifically blockchain, could address these two factors. This paper aims to demonstrate this by developing a minimum viable product on the Ethereum blockchain, called the LoveEconomy, for a local South African non-profit organization, the Secret Love Project. The LoveEconomy is designed as a circular economy, whereby local businesses and users of the platform benefit from each other, whilst also supporting the charity, which takes care of homeless people in Cape Town. Blockchain has many features that could potentially transform charitable giving and aid distribution by enhancing transparency, reducing costs through disintermediation, and enabling new mechanisms for monitoring and tracking charities' impact. Trust and transparency are closely linked in the charity industry, as transparency about the distribution of the funds and the end impact are critical for the trust of the public (Populus and Charity Commission For England & Wales, 2018).

Declaration

I, Julika Pahl, hereby declare that the work on which this dissertation is based is my original work (except where acknowledgments indicate otherwise) and that:

- I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
- I have used the required Harvard convention for citation and referencing. Each contribution to and quotation in this assignment from the work(s) of other people has been attributed, and has been cited and referenced.
- I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
- I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.
- Neither the whole work nor any part of this dissertation has been, is being or is to be submitted for another degree in this or any other University. I empower the University of Cape Town to reproduce for the purpose of research either the whole or any portion of the contents in any manner whatsoever.

Julika Pahl

Signed by candidate

April 14, 2021

Acknowledgements

I would like to thank my mother, Sabine Pahl, for encouraging me to do this course and helping me along the way and my father, Peter Pahl, for his constant support. I would also like to thank my classmates for all their support and for sharing their knowledge with me. Thank you to Nicholas Salonen for getting me through the final stages of writing this thesis and all his support.

Thank you to my tutors, Sabine Bertram and Allan Davids, for all their help and support and to Co-Pierre Georg, my supervisor, for helping me develop this idea and turn it into what it is now. I greatly appreciate all the advice and direction you gave me along the way.

Finally, I would like to thank Michael Elion and the Secret Love Project for allowing me to work on their idea of the LoveEconomy and for their time and knowledge.

Contents

List of Figures

Chapter 1

Introduction

South Africa is a country with a wide disparity in socio-economic status, with critical challenges needing to be addressed such as inequality, poverty and unemployment (CAF Southern Africa, 2019). One way in which these challenges manifest are in the high rates of homelessness in South Africa, estimated to be approximately 200,000 people (Rule-Groenewald, Timol, Khalema and Desmond, 2015). Western Cape Government (2019) estimates there to be 4,862 homeless people in the greater Cape Town area and about 700 of those people living in the central business district. Within this climate, there is evidence that there is a willingness on the part of individuals in the South African community to donate towards charitable causes and that these non-profits make a significant contribution to the social and economic well-being of the most vulnerable in South Africa. In fact, among South African's, the most popular cause to donate money toward is helping the poor, contributing 55% of all donations (CAF Southern Africa, 2019).

Corruption is a universal problem, especially in developing countries, such as South Africa (Manyaka and Nkuna, 2014). High rates of corruption reduces the public's willingness to support charities (Populus and Charity Commission For England & Wales, 2018). To give some perspective, in 2012, the UN-Secretary General stated that corruption prevented 30 percent of all development assistance from reaching its destination (UNSG, 2012). Trust and transparency are therefore becoming an increasingly important aspect for charitable performance, and there is a need for the charity sector to increase these amongst its donors. This can be done by making use of technology and the internet (Burt and Taylor, 2011). Amongst these technologies is blockchain, which demonstrates the potential to increase transparency, build trust and enhance efficiency within the charity sector (Kshetri, 2017). Most blockchain projects in the charity industry are still in early stages, and so it is difficult to find long standuing successful cases (Shin, Kang and Bae, 2020). Some of these projects

are covered in more detail in the next chapter.

In this thesis, I develop a blockchain based prototype to increase the transparency of charitable donations. This platform is also meant to provide incentives for charitable giving, motivated by evidence on the nature of donations. Key to this, a study done by Holmes, Miller and Lerner (2002) found that people's willingness to help a charitable organization is greater when the act is represented by an economic transaction than when it is presented as an act of charity. The findings of Holmes, Miller and Lerner (2002) encouraged the development of a system where individuals receive access to a platform offering discounted deals at local restaurants and businesses in return for charitable donations to the homeless.

The inspiration for this thesis comes from a South African non-profit organization called the Secret Love Project, one of the registered non-profit organizations who aim to help the homeless people in Cape Town reintegrate into society. They do this by providing about 20 000 free heart sticker packs per month to the homeless, which they can sell for R20 each, keeping 100% of the sales (Elion, 2019). This amounts to R400 000 a month (R4.8 million per year) being earned by our homeless community. The founder, Michael Elion, came up with the idea of the LoveEconomy, which will be implemented in this thesis. The people who sell the stickers will be referred to as LoveEconomy Ambassadors (LEA) for this study. [1]

The idea of the LoveEconomy is to further extend the Secret Love Project by incorporating it into an ecosystem that will allow for the project to eventually become self-funded. This idea requires local businesses to register on the LoveEconomy platform, on which they provide discounted deals for users. Users of the platform will be people who wish to support the homeless, by purchasing the stickers sold by the homeless, or anyone else who would want access to the deals provided on the LoveEconomy platform (and so indirectly support the homeless and local businesses). A deal is something that a business can offer the users of the platform, such as discounts to their products, or two-for-one deals to their services. Users can gain access to the platform by purchasing the special sticker packs from the LEA's who sell them, which will provide them with a code to use to register on the platform. Users will then have access to the deals provided by the businesses. Users will also have the option to register themselves to the LoveEconomy platform for a fee instead of purchasing a sticker pack. The registration fee for both businesses and users will then be directly used to produce more stickers, generating a self-sustaining circular economy. For this thesis, a prototype of a blockchain-based LoveEconomy was developed. This was done to investigate the feasibility of using blockchain to create

[1] The numbers regarding the Secret Love Project were taken from their website on 5 April 2021

a more appealing, transparent and trustworthy platform for donors in Cape Town. Blockchain is a distributed database of records, or shared ledger of transactions or digital events (Crosby, Pattanayak, Verma and Kalyanaraman, 2016).

The prototype developed for this study consists of a back-end using the Ethereum blockchain. Ethereum is a public blockchain, which means that any peer can join the network at any time (Wüst and Gervais, 2018). It also allows developers to build and deploy decentralized applications, using smart contracts. Smart contracts are programs, which reside within decentralized blockchains, and act as contracts between two parties (Sayeed, Marco-Gisbert and Caira, 2020). They are terms of agreements written in code, and are self-executing. For this prototype, three smart contracts were developed. The first is the LoveEconomy smart contract which allows new businesses and users to register on the platform. The second is the Local Business smart contract which is deployed each time a new business joins the platform, this smart contract will be owned by the businesses and allows them to add new deals to their LoveEconomy profile. The third and final smart contract created is the Deals Token contract, which is deployed each time a new deal is added by a business. The front-end of the prototype consists of three webpages, allowing for participants of the LoveEconomy platform to interact with the Ethereum blockchain.

This study demonstrates the possibility of using blockchain to create a platform for the public and businesses to come together and support the homeless people of Cape Town. The platform is focused towards increasing transparency of a local charity (the Secret Love Project) in Cape Town, by creating trackable transactions and a sustainable economy.

The study proceeds as follows: chapter 2 discusses the current charity field. This includes the meaning of trust in the industry, how transparency within the industry is currently being dealt with, its cost and how technology is being used to overcome the problem of transparency and trust. The chapter will conclude by briefly describing blockchain and how it is currently being implemented. In chapter 3 the LoveEconomy is described, as well as which blockchain could be suitable for the experimental phase and the challenges that may be faced when implementing blockchain solutions in the charity sector. Chapter 4 describes the platform and the components it consists of. Chapter 5 discusses the benefits, findings and limitations of the blockchain-based LoveEconomy solution and finally, chapter 6 will give the conclusion of the thesis.

Chapter 2

Literature Review

2.1 What trust means in the Charity Sector

A charity is a non-profit organisation that exists to provide public benefits, and can also be described as an intermediary between donors (who provide resources) and beneficiaries (ultimate recipients of the donors gifts) (Hyndman and McDonnell, 2009). In this study it will also refer to any non-profit organisations working on humanitarian aid and social impact projects. The three key external stakeholders of a charity can be categorized into: donors, beneficiaries and regulators.

A key component of non-profit organizations is trust. "Trust matters. Observed behaviour shapes the perceptions of 'trustworthiness', which in turn affects the support for an organisation or sector and impacts its success in achieving its end goals" (Furneaux and Wymer, 2015; Populus and Charity Commission For England & Wales, 2018; Sargeant and Lee, 2002). The Charity Commission for England and Wales carried out a study on the public's trust and confidence in charities together with Populus (Populus and Charity Commission For England & Wales, 2018). This study includes what trust means in this sector, how it relates to its success and what charities can do to exemplify trustworthy behaviours (Populus and Charity Commission For England & Wales, 2018). The research suggests that when charities are able to show that most of the donations they receive from donors directly reach the end causes and have quantifiable positive results, both trust and self-reported inclination to donate increases (Cordery and Deguchi, 2018; Furneaux and Wymer, 2015; Hyndman, 2017; Populus and Charity Commission For England & Wales, 2018). This means that transparency on the distribution of funds and the end impact are critical to the trust of the public in charities. Some reasons for the decrease in trust are: a view that too much is spent on salaries and 'administration costs', and a lack of understanding about how the charity is run and managed, which again

points towards a lack of transparency within the organizations. In another study done for the Charity Commission, they found that 96% of the surveyed members of the public agreed that charities should provide the public with information on how they spend their money (Hyndman and McConville, 2016).

The Charity Commission found that the public's trust and confidence in charities was knocked down in 2016 and 2018 from a high in 2014. Those of the public who say that their trust in charities has decreased commonly cite news stories relating to charities in recent months (Hyndman, 2017; Populus and Charity Commission For England & Wales, 2018; Sargeant and Lee, 2004). These news articles reinforce the more general suspicion that a significant portion of the donations do not reach the beneficiaries (Populus and Charity Commission For England & Wales, 2018). Some cases where non-profit organisations abused their position and miss-used funds are mentioned below:

The Reynolds cancer charities

In May 2015, James T. Reynolds was charged by the Federal Trade Commission (FTC) with a civil suit for deceiving donors and miss-using over $187 million of donor funds (Lee and Johnson, 2017). It was found that the organizations founded by James, (which included the Cancer Fund of America, the Cancer Support Services, Children's Cancer Fund of America and the Breast Cancer Association) spent less than 3% of their revenue on programs related to their stated mission and instead spent large proportions on fundraising, salaries, bonuses and lavish expenses (Lee and Johnson, 2017).

The American Red Cross: Haiti relief

The American Red Cross came under scrutiny for their charity work done during the 2010 Haiti earthquake relief. Initially they claimed that 91% of the raised funds, which was half a billion US dollars, was used to assist Haiti, however further investigations found this not to be true. In 2016 the NPR (an independent, non-profit media organization in America) stated that the American Red Cross spent a quarter of the donated money, about $125 million on their own internal expenses, which is far more than the organization had previously disclosed (Sullivan and Elliot, 2016). They also stated that the American Red Cross released incomplete information about its Haiti programs to the public and that their ambitious plan to build housing resulted in only 6 permanent houses. There was a lack of transparency, where often the organization itself did not know how much money was spent on each project in Haiti (Sullivan and Elliot, 2016). The NPR also stated that Red Cross did not provide a list of specific programs it ran, how much they cost and

what the expenses were (Sullivan and Elliot, 2015).

ONE Campaign

In South Africa, the ONE Campaign, an anti-poverty charity, was in the media in 2018 for investigations into tax evasion (Ratcliffe, 2018).

Michelle McLean Children's Trust

In Namibia, the Michelle McLean Children's Trust was involved with a lawsuit between the trustees of the fund and the ex-director Danie Botes (Menges, 2016). The trustees sued Botes for 7.7 million Namibian dollars, for withdrawing 6.7 million dollars of which he was not entitled to from the organisations account and 1 million through a transaction cost (Menges, 2016). Botes received 30% of the trusts income as a salary, as from October 2010. In the report it stated that the trust intended for Botes to receive that proportion from the net income and not the gross income. People commented that they would rather spend money directly on children, than give money to an organization that pockets half of the money for themselves (Menges, 2016).

2.2 Transparency and its cost

Accountability and transparency are important factors for charities to maintain the trust and financial support of the public (Sinclair, 2010). Transparency can be described as providing stakeholders and the public with appropriate data and information to allow them to make an informed decision, which in the non-profit sector includes (Horton, 2015):

- Providing relevant information in an unbiased context (not just providing arbitrary numbers and data)
- Demonstrating social impact
- Making the information easy to find and understand.

In addition to this, the transparency and efficiency with which charities spend the funds they are entrusted with is becoming increasingly important to the public and donors (Hyndman and McConville, 2016). This is important if donors are unsure about how the charities are likely to spend their donations, perhaps due to the lack of available information, and therefore turning possible donors away (Hyndman and McConville, 2016). Additionally, where charities are not transparent, inefficiencies and poor performance may be more likely (Hyndman and McConville,

2016). A study done on 14,000 non-profit industries in America associated greater transparency with stronger governance, better performance and more professional staff members (Harris and Neely, 2018). However, there are also costs associated with transparency. There is a cost of disclosure and collecting information, risk of disclosure to hostile parties and concerns that the information would not be interpreted in the correct way (Hyndman and McConville, 2016). It may also change the behaviour of the managers in a negative way, causing them to make decisions that improve their reported ratios and measures, rather than focusing on being an effective organization (Hyndman and McConville, 2016).

Collecting the relevant information in a cost-effective way can be challenging. The costs are driven by the need to establish processes and systems that capture accurate and objective information, both for ongoing internal management and external reporting (McConville, 2017; Shin, Kang and Bae, 2020). This is a bigger issue for smaller charities, as they tend to have fewer available resources, outdated and underdeveloped information systems, and may be more reliant on volunteer services (McConville, 2017).

There are different forms of providing information for transparency to the public, amongst which are (Bothwell, 2001):

- legal reporting obligations (government legislation and regulation, and independent regulators)
- recently formed charity watchdogs
- independent charity monitoring organizations (voluntary certification programs)
- voluntary-provided public reports and information (commonly given on websites).

Charity watchdogs, voluntary-provided reports and websites have recently emerged to address the perceived transparency problems in charities (Szper and Prakash, 2011). Charities use these accreditation systems and voluntary programs to signal their trustworthiness to donors (Szper and Prakash, 2011). Additionally, voluntary transparent impact reporting can also help charities meet the expectations of stakeholders, focus on and achieve their charitable mission and meet legal reporting obligations (McConville, 2017)

2.2.1 Legal reporting obligations

"Having a regulator that deals swiftly and effectively with abuse when it occurs and promotes compliance by trustees with their legal obligations is at the heart

of public trust and confidence in charities" (Charity Commission, 2008). Amongst legal reporting obligations, there is government regulation and legislation, taxation agencies and independent regulators, such as the Charity Commission for England & Wales (Sinclair, 2010).

Government Regulation and Legislation

The amount of power government regulators have depends on the jurisdiction, but most commonly charities must produce annual reports, as well as further information if the regulator deems necessary (Hyndman and McDonnell, 2009). Government regulation has the potential to: increase public confidence in charities, which will possibly result in more public support, improve management in the sector and reduce likelihood of scandal (Hyndman and McDonnell, 2009).

Due to the additional cost of compliance to regulation, government regulators tend to take a light-handed approach to small- and medium-sized charities (Cordery, 2013). It was found that due to this light-handed approach, the information the charities provide is likely to be flawed (Cordery, 2013). The reports to regulators often contain a high rate of errors, which confirms the lack of expertise in financial reporting found in some charities (Cordery, 2013). The financial statements are reports that can only contribute towards the accountability and transparency of charities by being transparent and understandable, which they currently are not (Sinclair, 2010).

Countries are becoming increasingly aware of the need to combat financial fraud, money laundering and terrorist financing, causing them to step up supervision and control in their non-profit sectors and creating tighter legislative frameworks and greater regulation (Piper, Partner, Farrer and LLP, 2019).

Government Regulation and Legislation: South Africa

In 2018 there were about 100 000 registered non-profit organisations and 50 000 unregistered non-profit organisations in South Africa (Fodor, Radebe and Werksmans Attorneys, 2018). These organizations comprise of:

- Non-profit companies, registered under the Companies Act
- Trusts, registered in accordance with the Trust Property Control Act 1988
- Various community-based voluntary associations, which can be registered under the Non-Profit Organisations Act, but it is not obligatory.

Any form of non-profit organisation can register under the Non-Profit Organisation Act. This act operates on a voluntary basis in South Africa, and aims to create an

enabling environment for non-Profit organisations and setting and maintaining adequate standards of governance, accountability and transparency (Honey, n.d.). The Act provides a registration facility for non-profit organisations, provided a certain minimum establishment and reporting requirements are met. This may be a challenge for many of the smaller organizations, as mentioned above, who do not have the proper resources and funds to meet these requirements. The act provides some benefits to those that are a part of it. These smaller organizations, which are unable to meet these requirements, miss out on the provided benefits, such as the Minister prescribing allowance and benefits to the registered organisations. Additionally it is anticipated that to receive any government benefits or money from the government an organization needs to be registered under the Non-Profit Organisation Act (Honey, n.d.).

Taxation agencies

Many taxation administrators have the statutory responsibility to ensure that tax relief and tax benefits are appropriately claimed by the charities (Sinclair, 2010).

Independent regulators

An example of an independent regulators are charity commissions. The Charity Commission For England & Wales register and regulate charities in England and Wales (Charity Commission, 2008). Their aim is to provide the best possible regulation to increase the charities' effectiveness, and public trust and confidence. Most charities in England and Wales must register with the Charity Commission, with only a few exceptions (Charity Commission, 2008). The Charity Commission stated in a report from 2007/2008 that there are about 19,000 registered charities and that those with a gross annual income above £10,000 must provide annual information and accounts to the Commission. The Commission states that it ensures that the charities comply with any legal regulation and investigate any indication of miss-management or miss-conduct (Charity Commission, 2008). However, the charity commission's have also been faced by some doubts. In 2013, the Charity Commission For England and Wales was investigated by the Parliament's Public Accounts Committee (PAC), to see whether they are fit for their purpose (Rittelmeyer, 2014). This was due to the Cup Trust scandal. In the Cup Trust scandal, the charity in question was actually a tax-avoidance scheme, and despite many worrying indicators, the Charity Commission continued to certify their status (Rittelmeyer, 2014). PAC also found that "the Charity Commission's approach to regulation and enforcement lacks rigour". These investigations were done after it came to their attention that the Charity Commission's fraud detection rate was extremely low (Rittelmeyer,

2014).

2.2.2 Voluntary monitoring

Charity watchdogs

Charity watchdogs have only started emerging in recent years, they use the charities financial information and make it relevant for donors (Szper and Prakash, 2011). Compliance to charity watchdogs is completely voluntary (Rittelmeyer, 2014). One of the reasons for their establishment was due to donors finding it difficult to assess and interpret the legal information charities provide, specifically on how they are deploying resources (Szper and Prakash, 2011). Another reason for their emergence was due to the wide-spread coverage of non-profit scandals. Charity watchdogs aim to make the information available in a more convenient way for the donors, either for free or at a fee. This is in hopes that donors reward those charities that devote efficient resources to service delivery and penalise those that are less careful about controlling overheads. Another benefit of charity watchdogs is the diversification of ranking systems used by them, as they try to set themselves apart from the others (Rittelmeyer, 2014).

Charity watchdogs typically provide information, reviews and ratings of charities, to help donors make better decisions (Szper and Prakash, 2011). Some watchdogs allow charities to display their approval seals for free, while others expect the charities to pay a yearly fee based on their yearly revenue (Miragliotta, 2014). These include Guide Star, who are now global, Charity Navigator and Better Business Bureau's Wise Giving Alliance who report on some national and regional charities in the USA (Cordery, 2013). There are however some limitations to these rating systems. One such limitation is that they base their ratings on financial information and organizational criteria, rather than programmatic content, success or impact, which may arguably be more important criteria for assessing charities (Cordery, 2013). Charity Navigator does state on the home page of their website that they are planning on rating charities based on the reporting of their results in the near future (Charity Navigator, 2020).

There is contrasting research on the influence these ratings have on charitable donations, some claim that they are successful at improving charitable giving, while another study found that donor giving is not associated with the ratings (Szper and Prakash, 2011). However, even if they do not affect the donations, they may preclude the misuse of funding that would have occurred in their absence.

Voluntary certification programs

In addition to charity watchdogs, there are voluntary certification programs. However, there are additional costs involved with these programs, as there is more preparation that needs to be done, which take time, resources and effort (Bothwell, 2001). These programs also charge an annual fee to the charities, depending on the size of the charity.

Voluntary-provided public reports and information

Voluntary transparency of charities is when they voluntarily publish annual reports, grant guidelines, brochures, newsletters and websites to inform the public, potential and actual donors and grant-seekers (Bothwell, 2001). An example of this is Charity: water. Charity: water uses their website to deliver detailed project reports created by delivery partners and by the charities small team of staff on the ground, as well as performance data and financial accounts (Burt and Taylor, 2011). Another example are many newly created blockchain based non-profits, who are also advocating increased information and reports available on their websites, for example AidChain. As mentioned previously, they provide a platform for all users and charities to able to publish regular updates. However, in some cases this can also be flawed in the sense that charities may choose to disclose only selected, insufficient or misleading information.

2.2.3 Room for improvement

Under the current reporting regime charities are more concerned about legitimizing their organization rather than proving ethically driven accounts of their efficiency and impact (Hyndman and McConville, 2016). Additionally, it requires extra resources, time and costs even without being completely certain that the reports are without errors.

A recent study found that the following areas of transparency need development (McConville, 2017):

- Improving transparency on impact, including impact on society, by the largest charities
- Extending transparency on impact into smaller charities in a way that is useful for stakeholders and charities, at an acceptable cost
- Developing cost-effective ways of capturing and reporting information on impact

- Integrating external reporting with internal reporting as a basis for maximising impact.

Developing digital solutions could assist charities with impact reporting, as well as provide live information, specifically for smaller non-profit organizations.

2.3 Technology and Charities

An increased use of technology can support and create new reporting transparencies, which brings the opportunity to build trust in a sector for which it is crucial (Burt and Taylor, 2011; Shin, Kang and Bae, 2020). However, Akingbola, Rogers and Baluch (2019) stated that technology has been mostly absent in discussions on the efficiency and effectiveness of non-profit organizations. Green (2018) wrote a report on the level of technology in the non-profit industry. They did this by asking certain charities to answer an in-depth survey on their use and management of digital technology, their priorities, pain points and plans for the future (Green, 2018).

The survey highlights some key indicators of success for those already adopting digital technology and making the best use of information technology (Green, 2018). Green (2018); Limburg, Knowles, McCulloch and Spira (2017) found that those charities who make use of digital technology gain many benefits over the less-digitally inclined charities, including more confidence in future planning and ability to meet strategic goals, increased donations, increased productivity and efficiency, as well as better protection from risk. A further report found that those charities who are able to use new technologies successfully were able to transform their internal organizational arrangements and simultaneously improve the experience of service users, members, volunteers and other stakeholders (Burt and Taylor, 2011).

In the survey by Green (2018), 58% of the charities, do not have a defined digital strategy, while 92% of those charities which do have a digital strategy in place expect to increase their measurable impact. In this context, a digital strategy is defined as an organisation's plan to maximise the benefits of data assets and digital technologies for existing processes (Green, 2018). The most significant barrier mentioned by those charities with no defined digital strategy, was the lack of time and money. It is difficult to measure the return on investment when it comes to information technology, which makes it more difficult to prove its value to senior management (Green, 2018).

Some examples of where technology helped charities increase donations and improve internal organisation in the past are mentioned below:

SMS technology: In 2010 the American Red Cross charity was able to raise $32

million in $10 donations for aid and support to Haiti, after the earthquake they experienced (Krach, 2017).

Crowdfunding platforms: These platforms harness the power of the crowd, soliciting thousands or even millions of smaller donations, instead of depending on large gifts (Krach, 2017).

Fundraising and impact awareness via social media: Not only is it a good way of communicating with supporters and donors, it also provides an platform for fundraising (Krach, 2017). For example, in 2014, the ALS (Amyotrophic lateral sclerosis) Association started the Ice Bucket Challenge, where people posted videos of themselves dumping a bucket of ice water over their head to promote awareness for ALS, also known as Lou Gehrig's disease. The videos challenged others to do the same and/or donate to the ALS non-profit organization. There were more than 2.4 million tagged videos on Facebook and the ALS Association received $100 million in donations in the month of August 2014 (Krach, 2017). Charities can also use social media to communicate with their donor base and advertise their work to the broader public. This also allows for increased transparency and accountability, by showing social media participants the outcome of the charities' work on the platforms.

Cashless donation options: This has been particularly important for the homeless people on the streets due to our increasingly cashless society (Gilliland, 2019). The London mayor has launched a new campaign in partnership with TAP London non-profit organization. This involves 35 cashless donation points set up in London to allow donations to be made to the 22 charities that make up the London Homeless Charity Group. Reports have shown that Londoners have tapped to donate 2,581 times, raising £7,743, in the first two weeks of the launch of the project (Gilliland, 2019).

Enhancing performance through online volunteer management: The St John Ambulance Foundation in the United Kingdom, which started in the 19th century, is an example of an old established organisation reorganizing itself as a modern, agile service provider (Burt and Taylor, 2011). They provide emergency first aid service for all kinds of emergencies. These services are provided by volunteers, who donate up to 4 million hours a year (Burt and Taylor, 2011). Holistically, this is a highly complex national-local infrastructure and with most services being 'time-critical', it is immensely challenging in the demands placed on effective volunteer deployment (Burt and Taylor, 2011). The development and implementation of a Duty Information Planning system (DIP) helped the organization become more efficient and effective. Using the DIP system in combination with a SMS text messaging system, backed-up by emails, pagers, and telephone calls, in the event of

an emergency the organization can immediately see which staff and volunteers are available, who they are and whether they are appropriately qualified for that specific emergency (Burt and Taylor, 2011). This resulted in significant cost and mission-based benefits, due to reduced administrative burden, higher volunteer response rates and enhancing response times in emergency situations.

Technology has also already been implemented specifically to increase transparency and trust:

Reporting clarity, transparency and donor accountability: In November 2015, Charity: water started deploying sensors and cloud computing technology to provide real-time data on water system performance (Charity: water, 2020). They use these sensors to measure the flow of clean water and inform them of how projects are doing at any given time. Charity: water also uses their website to deliver detailed project reports created by the project delivery partners and by the charities small team of staff on the ground, as well as performance data and financial accounts (Burt and Taylor, 2011). In addition to this, their data is available to the public and have open-sourced all of their designs.

Evidence gathering and project reporting: CHF (Cooperative Housing Foundation) international, which is involved in international social development, uses a custom built online system for real-time worldwide program monitoring and reporting (Burt and Taylor, 2011). Their vision is to have a dynamic system with the ability to respond to rapid changes in the conditions in which their projects are being delivered, enable data capture and analysis. CHF international's digital team is also working on being able to generate and sustain high quality dialog with supporters and potential donors through social media networks (Burt and Taylor, 2011).

Using blockchain to improve transparency: In addition to the above mentioned technological advancements to increase transparency in the charity sector, blockchain is considered to have the potential to cause major social transformation (Kshetri, 2017). Blockchain is a technology that demonstrates the potential to help promote and increase transparency, build trust and enhance efficiency in transactions (Kshetri, 2017). Blockchain can be viewed as a data structure which makes it possible to create tamper-proof digital ledger transactions in real-time and share them, so that anyone can add to the ledger securely. It is extremely difficult to change or remove data blocks recorded on the ledger, making it secure (Kshetri, 2017). The unique combination of features from blockchains makes the technology appealing in its application for social impact projects. Blockchain provides the charity and aid distribution sector with the potential to reduce corruption and fraud

(Kshetri, 2017). It can also be used to empower donors, by ensuring and proving that their donations reach the intended recipients (Kshetri, 2017).

The AidCoin whitepaper lists the following types of transparency which can be designed for charitable organizations using blockchain (AidCoin, 2017):

- Tracking the actual use of the funds, from the time that the money is donated up until the conversion into local fiat currencies.
- Verification of the identity of the recipients and reporting on their use of the funds.
- Tracking administrative costs of non-profits.
- Ensuring money reserved for specific projects actually reaches those projects.
- Comparing of the effectiveness of money spent on one charity compared to another.
- Tracking the investment policies of charities.
- Reduction in transactions costs of money transferred through cryptocurrency rather than financial intermediaries.

There are both start-ups and long-established charities working towards transforming the humanitarian industry using blockchain.

2.4 Current blockchain solutions in the non-profit industry

Most humanitarian blockchain solutions, social-impact solutions and digital charities use blockchain in a similar way to come up with solutions. Currently these include, but are not limited to, financial inclusion, land titling, remittances, transparency of donations, reducing fraud, tracking of support to beneficiaries from multiple sources, cross-border transfers, micro-insurance and transforming governance systems (Coppi and Fast, 2019). Most of these applications constitute of back-end blockchain applications that support humanitarian programs, which relates to financial and internal process, increased transparency, accountability and efficiency, rather than front-end applications that directly touch the end user (Coppi and Fast, 2019). A recent study found that blockchain is most often used to facilitate payments and verify records, and its most popular benefit is being able to reduce risk and fraud, while increasing efficiency (Galen, Brand, Boucherle, Davis, Do, El-Baz, Kimura, Wharton and Lee, 2018; Shin, Kang and Bae, 2020).

Below are different forms of blockchain solutions which are currently being worked on. These are categorised into impact-based solutions, crypto-based solutions and token-based solutions. Impact-based blockchain solutions relate to projects which use blockchain for improving accountability and measuring and verifying social impact investments (Coppi and Fast, 2019). Crypto-based solutions are those solutions where donations are made via cryptocurrencies directly, while token-based solutions make use of cryptocurrencies indirectly with tokens. Both the crypto- and token-based solutions are developed to increase the transparency and trust of donors, as they allow the donors to track each payment made from the organization and also decrease transaction fees (Saleh, Avdoshin and Dzhonov, 2019). The donors can see these transactions as blockchain is a distributed ledger and the transactions exist on all devises connected to the network. Using crypto- or token-based solutions should also simplify the work needed for documentation and reports (Saleh, Avdoshin and Dzhonov, 2019).

2.4.1 Impact-based solutions

One of blockchain's qualities is that it can be implemented with smart contracts. A smart contract is a computerised transaction protocol that executes the terms of a contract (Zheng, Xie, Dai, Chen and Wang, 2018). This use-case is currently being tested and implemented as a solution in the charity industry. One can use smart contracts to freeze funds or donations, and release them to the charity once their work has been verified to have an impact according to the contracts, and cancels the payment if it does not make the impact. An example of a non-profit organization working on implementing this blockchain function is Alice. Alice is a social funding platform built on the Ethereum blockchain (Mazet and Wojciechowski, 2017). They help organizations run transparently and use result-based financing to incentivise charities by releasing funds only when specific goals are achieved. This is done through smart-contract based incentives, to ensure that the impact is independently verified and accessible to everyone. Validators are assigned to each project, which can either be humans, machines or a combination of both. Alice launched a pilot project with St Mungo's, a homelessness charity based in London (Mazet, 2018). In this pilot, 15 men and women who have been homeless for a long time received personal support from St Mungo's who needed to complete 4 steps to achieve the final goal and being granted permanent housing. For this project, the first goal was to place each of the 15 individuals into temporary homes, and the second step was to help them stay in these homes for 3 months. This allows the performance of charities to go public, and be fully auditable. Giveth is another company with a similar result-based structure on the Ethereum network, where funds are only

released once certain milestones are reached (Decoodt, 2017). However, this is still in the development phase.

Another organization that is focusing on impact-based solutions is IXO. IXO ran a pilot project in South Africa called Amply. Amply enabled early childhood development (ECD) centres to record and verify pre-school attendance claims, that are then exchanged for subsidies from the government of South Africa (TechBullion, 2018). This brings greater transparency and accountability to the funding process. The system gives every child a globally portable digital identity that proves who they are, records their educational history and so enables them to receive the funding they deserve (TechBullion, 2018). The privacy of the digital identity can be controlled by the parents. The attendance data is captured every morning by ECD teachers and submitted as an impact claim with its own digital signature, which is stored on the blockchain and verified by an evaluator (Verhulst, Young and Zambrano, 2018). The claim can be authenticated with a mathematical proof using the metadata of the data capture, which includes the time, date, location, etc. and acts as an "error checking" method. This system is meant to replace an unverifiable paper-based system, where centres have to submit lengthy attendance reports in order to receive appropriate subsidies which, from the government's perspective, was expensive and time consuming to audit. Since the pilot began in 2016, more than 61,000 digital attendance records have been created across 85 ECD centres in South Africa (TechBullion, 2018). Unfortunately Amply is no longer active as it was not able to keep up with IXO advancements with no one to carry the project forward.

2.4.2 Crypto-based solutions

There is also a small but growing number of charitable organisations that have either created their own cryptocurrency or have accepted and implemented it as a method of donating.

One example is a project called Usizo, launched by the South African bitcoin start up Bankymoon (Kshetri, 2017). This is a crowdfunding platform designed to facilitate energy payments between donors and selected South African schools (Higgins, 2016). Blockchain-enabled smart meters are positioned at the school buildings, making it possible for anyone in the world to make payments directly to the meter, using a crypto-currency of their choice (Bankymoon, 2020). There is no middle man that is responsible for distributing the funds that are donated and they are able to track the electricity being consumed by the schools and calculate the power their donation can buy.

The Norwegian Red Cross and Innovation Norway conducted a project to explore

how blockchain could add transparency and accountability to open-loop cash transfer programs (Coppi and Fast, 2019). Together with Red Rose, a technology provider for cash transfers in the humanitarian sector, they conducted the Open Loop Payments Pilot Project. The objective was to explore the risk and benefits of using blockchain technology for humanitarian cash transfer programming (Coppi and Fast, 2019).

Helperbit is an Italian start-up which emerged from the increased awareness of inefficiencies in the management of funds for humanitarian emergencies, such as earthquakes. The start-up developed two solutions on the Bitcoin blockchain: a peer-to-peer insurance service for pre-disaster phase service and a peer-to-peer donations system that allows donors to control their donations Coppi and Fast (2019). To enable decentralization, Helperbit developed a non-custodial, multi-signature bitcoin wallet. This increases the security of the wallet, protecting it from external attacks and internal mistakes, such as losing the passphrase Coppi and Fast (2019).

A well-known charity, UNICEF, has recently announced that they welcome funding in both Bitcoin and Ethereum. Binance Charity is another example of a non-profit organization dedicated to achieve global sustainable development using a blockchain donation system (Binance Charity Foundation, 2019). Binance Charity was initiated by Binance, one of the world's largest cryptocurrency exchanges. They currently accept Binance Coin, Ether, Bitcoin and fiat donations. Binance Charity's strategy is to enable beneficiaries to directly receive the funds donated, which means opening cryptocurrency accounts for each of them. The reason for this is to restrict the abuse of funds by third parties and empowering the beneficiaries to decide on the use of their funds by themselves, instead of passively receiving goods (Binance Charity Foundation, 2019).

2.4.3 Token-based solutions

AidChain has created a platform which allows fundraisers to accept donations in all the major cryptocurrencies by embedding the AidPay widget (AidCoin, 2017). It allows charities to manage the cryptocurrencies in a single wallet, by converting the cryptocurrencies into AidCoin, an ERC20 token. ERC tokens are blockchain-based assets, with similar functionality to bitcoin and ether. These tokens are hosted on the Ethereum blockchain and they can hold value and be sent and received (Di Angelo and Salzer, 2020). These donations are recorded on the Ethereum blockchain, the platform also allows charities to let donors know how the funds raised will be spent. AidChain was conceived and developed by CharityStars, a well-established charity fundraising company (AidCoin, 2017). CharityStars aim to develop an ecosystem of charities, donors, celebrity donors and gala events. They hope that this will

facilitate immediate use of their token. Donors would be able to search, discover and review charities, causes and projects on a single platform, while charities will be able to publish regular updates, marketing campaigns, request feedback and share volunteering opportunities (AidCoin, 2017). AidChain aims to also implement AidGift, which is the ERC721 non-fungible token, to increase the accuracy of their Donation Tracking System.

Clean Water Coin Initiative is a non-profit organization which raises funds for "Charity: water", who provide clean water to those who have no access to it. They do this by partnering with local experts and community members for each new project to find the most suitable and sustainable water solution (Charity: water, 2020). Clean Water Coin Initiative let's the community actively participate in distributing clean water by donating all funds in the Clean Water Coin Initiative's 'Charity Wallet' going to Charity: water (Clean Water Coin, 2014). Clean Water Coin Initiative built a function into their coin network so that anyone who mines or even uses the Clean Water Coin are contributing to the charity. Every transaction will donate 0.1% to the Charity Wallet, alternatively donors can purchase Clean Water Coins from the Charity Wallet.

From the above examples, one can see that blockchain has the potential to add value to, and improve the non-profit sector. Many of these solutions are based on the lack of trust and focused on increasing transparency and accountability in the non-profit sector. It can be used as a form of secure data storage, specifically in a verifiable claim format and digitization of records. However in a study done by Galen et al. (2018), a Stanford team listed 193 organisations, initiatives and projects using blockchain for social impact. The team's analysis demonstrated that although there is a rapid uptake of blockchains in the social impact sector, there is no actual impact yet, as most initiatives are still in early stages.

2.5 What is Blockchain?

Blockchain is a distributed database of records, or a shared ledger of all transactions or digital events that have been executed and shared among participating parties (Crosby et al., 2016). Distributed ledger means that identical copies of the entire record of transactions are available to all participants of the network, and so each network member has access to an up-to-date and validated instance of the ledger (Lapointe and Fishbane, 2019). This becomes important as most transactions are done online, where we rely on a third party to keep our digital assets secure and private, and these third parties can be hacked, manipulated and compromised (Crosby et al., 2016). Bitcoin is the most widely known example that is intrinsically linked to

the blockchain technology. The bitcoin blockchain was set up to be a decentralised, electronic payment system, however subsequent blockchains since then have many more applications beyond this.

There are two main types of blockchains:

Public blockchains: These are also known as permissionless blockchains and they allow any peer to join the network at any time (Wüst and Gervais, 2018). Bitcoin and Ethereum are well known examples of public blockchains. Public blockchains allow for anyone to read or write to the network, with no central entity managing the membership.

Permissioned blockchains: These blockchains only authorize a limited set of readers and writers on the network. With permissioned blockchains a central entity decides and attributes the right to individual peers to participate in the operations of the blockchain (Wüst and Gervais, 2018). These types of blockchains are used between trusted members that need to trade confidential information.

Blockchain provides many useful properties. Public verifiability allows nodes or peers on the network to verify the correctness of the state of the system (Zheng et al., 2018). In a distributed ledger, each state transaction is confirmed by verifiers, which can be a restricted set of participants. This makes it almost impossible to change or remove data blocks recorded on the ledger and it reduces or even eliminates integrity violations, making blockchains immutable. The transactions are stored forever and are traceable, making it fully auditable. The processes of accepting data onto a blockchain is known as the consensus protocol. This also allows for a persistency property of blockchain, where true transactions are validated and invalid transactions would not be admitted by miners (Zheng et al., 2018). Some more detail on each of the above mentioned properties are mentioned below:

Immutability: The immutability of information on the blockchain ensures that any data or information on the blockchain is protected from unauthorized modifications. No transactions can be changed or removed. As mentioned above, this property is closely linked to public verifiability. It would also allow organizations to store data safely and securely (Wüst and Gervais, 2018).

Transparency: Transparency of the data and the process of updating the state of the ledger is a requirement for public verifiability. However, the amount of information that is transparent to an observer can differ (Wüst and Gervais, 2018). This is a key feature of permissionless blockchains, as it makes it possible for anyone to audit the transactions taking place. Copies of the entire record of transactions are available to all participants of the network. Both transparency and immutability

allow blockchains to be used to combat and reduce fraud and corruption.

Privacy: This property has an inherent tension with transparency. However, a system can still provide significant privacy guarantees, while making the process of state transition transparent (Wüst and Gervais, 2018). This implementation would depend on the use of the blockchain.

Global and decentralized: Most blockchain networks have high levels of decentralization, which means that it does not rely on a centralized government or other institution. This allows for 'middle men' to be excluded from a process. It can also lead to reduced expenses since blockchain has the potential to simplify process and reducing overall costs for organizations, by requiring fewer intermediaries.

Blockchain is a trustless environment, which means that it is a challenge to reach consensus on the content of a distributed ledger (Zheng et al., 2018). Proof of Work is one of such consensus process, and it is currently used on the Bitcoin and Ethereum blockchain. Each block is verified through mining before its information is stored. It is verified using algorithms which attach a unique hash to each block based on the information stored in the block. The complexity in this lies with finding a specific hash corresponding to the information stored on the block (Zheng et al., 2018). Proof of Work requires a complicated computational process of authentication, as it requires a large group of distributed users to continuously vary the hashes of transactions. The process thus also requires large amounts of energy and can therefore be damaging to the environment (Zheng et al., 2018). Proof of Stake is an energy-saving alternative to Proof of Work. The Proof of Stake concept requires users to repeatedly prove ownership in their own share in the underlying currency. The idea behind this is that people with more currencies are less likely to attack the system. There are other consensus mechanisms, but currently Proof of Work and Proof of Stake are the most widely used.

There are many financial and non-financial applications of blockchain in areas that traditionally relied on a third trusted entity. Some example of blockchain solutions which are being investigated, considered, and in some cases, already implemented are mentioned below:

Blockchain based digital identities: The immutability and verifiability of blockchain systems enable the creation of permanent and portable digital identities (Lapointe and Fishbane, 2019). This is of particular interest for those who have no official documents, for example, refugees (Bradshaw, 2018). Accenture, IBM and the United Nations collaborated in building such a blockchain. Their solution uses biometric systems, such as figure prints, iris scans and other data to create a unique identity (Accenture, 2020).

Supply management chain: TradeLens is a blockchain enabled digital shipping platform, jointly developed by Maersk and IBM (Maersk, 2019). TradeLens was launched to help modernize the world's supply chain ecosystem. Many of the processes involved in transporting and trading of goods are costly, paper-based and manual, often leading to unreliable information exchanges. TradeLens enables participants to digitally connect and share information across the shipping supply chain ecosystem.

Smart contracts: Smart contracts are computer programs that can automatically execute terms of a contract (Crosby et al., 2016). Smart contracts have many use cases and can be used to eliminate third party transactions, as well as automate the system. This can then lead to lower costs and provide a greater degree of security (Mohanta, Panda and Jena, 2018).

Within this realm of financial and non-financial applications, blockchain also has a range of particular social-good applications:

Expanding access to services: Blockchain allows for more financial inclusivity by allowing people with no formal identity credentials or credit histories to build a secure digital identity (Lapointe and Fishbane, 2019). One example of such a company is BanQu's economic-identity blockchain, which aggregates personal identifiers, such as financial transaction histories, property records and education records, so that people can create a portable and vetted personal history, which reduces the risk for lenders and allows them formal financial services (Lapointe and Fishbane, 2019).

Healthcare applications: MedRec is a project started by MIT Media Lab and Beth Israel Deaconess Medical Center (Angraal, Krumholz and Schulz, 2017). This system offers a decentralized approach of managing permission, authorization and data sharing between healthcare systems. The goal is to give patients agency over their medical data and knowledge of who can access their healthcare data (Angraal, Krumholz and Schulz, 2017). The permissions, data storage locations and audit logs are maintained on the blockchain and all healthcare information remains in EHR (electronic health records) systems and requires additional software components to enable true inter-operability.

Recording public transactions: Blockchain based land registries are being piloted in countries all around the world (Lapointe and Fishbane, 2019). For example, the non-profit foundations Landesa and the Cadasta Foundation are testing blockchain applications to record and transfer property titles.

Preventing human trafficking: Some Governments and organizations are look-

ing into finding ways to use blockchain to combat and prevent human trafficking (Lapointe and Fishbane, 2019). The United Nations, together with the World Identity Networks, launched a pilot in Moldova of a blockchain-based digital identity system for undocumented children.

Summary

Trust is the foundation on which non-profit and voluntary organizations are built (Sargeant and Lee, 2002). Due to the current lack of trust in the industry, transparency is becoming increasingly important. One form of providing transparency is to present the public with data and information on how this funds are used. There are different forms of providing the relevant information to the public, such as legal reporting obligations, charity watchdogs, independent charity monitoring organizations and voluntary-provided public reports and information. However, collecting and presenting this information in a cost-effective way can be challenging, especially for smaller non-profit organizations. Additionally, these reporting regimes are more concerned with legitimizing their organization, instead of providing ethically driven accounts of their efficiency and impact. Technology, specifically blockchain, can be used to improve both of these issues in reporting. Blockchain's unique combination of features, makes the technology appealing to charity applications.

The benefits of blockchain are especially needed in poor countries, like South Africa, which lack good institutions that ensure strict enforcement of property rights, the ability to deal with corrupt practices effectively and equal opportunities to all members of society (Kshetri, 2017).

Chapter 3

The Secret Love Project

The Secret Love Project is a registered charity and non-profit organization in South Africa, dedicated to helping the homeless in Cape Town. They aim to help homeless people reintegrate into society, by providing them with a way to make their own money, become self-sufficient and enjoy acceptance by the community. The Secret Love Project provides 20,000 free heart sticker packs every month to the homeless for them to sell at R20 each, of which they keep 100% of the sales. This ends up being R400,000 going directly into the hands of the sellers. Currently there are 200 registered sellers, with demand increasing every week (Elion, 2019). Multiple success stories are reported on the Secret Love Project website (Elion, 2019). The project started in Cape Town, but is planning on expanding to other cities and has recently started in Port Elizabeth. Currently it is managed and funded almost exclusively by the founder Michael Elion. The experimental phase of this paper is based on an idea by Michael Elion, the LoveEconomy, to further expand this project and incorporate it into an ecosystem that will allow for the project to become self-funded. The aim of the experimental phase is to put some of the benefits given by blockchain towards helping the homeless people of Cape Town, and will be explained in more detail in chapter 4.

3.1 The LoveEconomy

The LoveEconomy is designed to help the homeless and simultaneously support the local businesses in Cape Town. Any local business that is interested in being a part of The Secret Love Project can join the LoveEconomy at a fixed fee. Interested users will be able to register on the platform at either a fixed fee or by using a discount code found on the Secret Love Project's sticker packs. Businesses can then add deals to the platform. For example, if one of these local businesses was the Labia

theatre, they might add a two-for-one movie ticket deal for users to purchase on the platform. Once a user is a part of this platform, they have access to these discounts and two-for-one deals offered by the local businesses, which could unlock some of the best experiences in Cape Town (Elion, 2019). For the business this platform could boost their customer base and their sales, act as an advertisement platform, as well as being part of a social project. The LoveEconomy is designed to work as an incentive to give to the homeless, while receiving something in return: "When you buy stickers you get rewards to support local businesses, who in turn support The Secret Love Project, who support the homeless" (Elion, 2019).

With the introduction of the LoveEconomy, the Secret Love Project would not only distribute the standard Secret Love Project stickers already being sold, but also additional special stickers which can be bought at a slightly higher price and contain the discount code to access the LoveEconomy. Since the Secret Love Project is currently being financed and run almost solely by its founder, Michael Elion, it needs more external support and funding for it to grow and be sustainable.

The platform will be built using blockchain as its main ledger. All transactions will be recorded on the blockchain. This includes the registration of the local businesses, the registration of users, which is split into users registering by paying a fee or the discount code, the number of discount codes used, the deals created by the businesses and the deals traded and used by the users. Once the user is registered on the platform, he/she can choose a deal they would like to redeem, purchase the deal on the platform and receive a token to use at the local business. These transactions are an indication of the impact on local businesses and sticker packs sold by the LEAs. The final step of this prototype would be to send the fees, paid by the businesses and users, directly to the company responsible for creating the sticker packs. This would ensure the fees are not used for anything other than the creation of the stickers, which will be distributed to the homeless.

People are more willing to give to charitable organisations when they are demonstrated as economic exchanges, which means that they are more likely to give when they get something in return (Holmes, Miller and Lerner, 2002). The LoveEconomy is based on this concept. It is a platform which aims to motivate people to give to the homeless, in exchange for being part of a platform on which they can have access to local deals. This means that, not only do they give back and allow the homeless an opportunity to make money, but they get something in return. Similarly to the users, the businesses help the Secret Love Project to fund the manufacturing of the heart stickers, while increasing their customer base and possibly their sales.

As briefly mentioned before, the LoveEconomy is based on a circular economy that

is designed to support itself, by each transaction encouraging and supplementing the next. The manufacturing company creates heart stickers for the Secret Love Project, which would include the special sticker packs with the discount code for the LoveEconomy. The LEAs sell the sticker packs to people who want to support the Secret Love Project and/or be a part of the LoveEconomy. This helps the LEAs make their own money to eventually become self-sufficient and reintegrated into society. Potential users then use the discount code to register on the LoveEconomy platform. Businesses interested in being on the LoveEconomy can register by paying a registration fee and add deals once they are signed up. Being on the platform can boost the business sales and expand their customer base. Users can also choose to register on the platform by paying the user fee instead of using the discount code. Although not yet implemented, the business fees and user fees paid to the LoveEconomy would go directly to the manufacturing company to fund the production of new stickers to be sold. Figure 3.1 illustrates this circular economy. This will help the creator of the Secret Love Project to keep the charity going and eventually expand to further cities in South Africa.

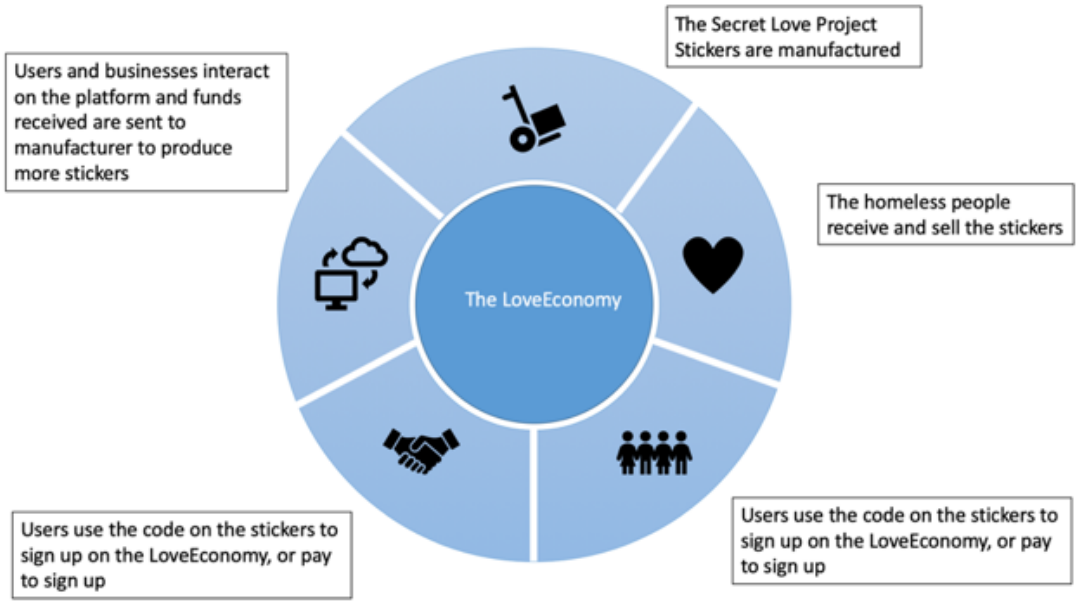

Figure 3.1: A schematic diagram illustrating the LoveEconomy's functionality as a circular economy.

3.2 Determining the type of Blockchain

To determine whether blockchain and subsequently which type of blockchain is suitable for the LoveEconomy two decision making models were used. The Wüst and Gervais (2018) model critically analyses whether blockchain, and also which blockchain, is appropriate for certain applications. They have created a structured methodology which will be used to argue whether blockchain should be used and is applicable to non-profit organizations, specifically the LoveEconomy. The Ethical Design Framework tool designed by Lapointe and Fishbane (2019) will be used to extend the Wüst and Gervias model to further include non-profit/social challenges. This model raises key considerations that need to be taken into account when considering whether blockchain is a viable solution in the non-profit sector.

Wüst and Gervais (2018), state that it only makes sense to use a blockchain, whether it is permissionless or permissioned, when multiple mutually mistrusting entities want to interact and change the state of a system, and are not willing to agree on a trusted online third party. If this is the case, there are four additional steps in the Wüst and Gervias model, which can guide users towards making the appropriate decision: whether or not to use blockchain. If the first three steps are answered with a "yes", blockchain is an appropriate solution, while the last step determines whether this solution should be a permissioned or permissionless blockchain.

The first step is to determine whether data needs to be stored, as blockchain is a form of database, and the second step is to determine whether there are multiple writers who contribute to this database (Wüst and Gervais, 2018). In this case a writer refers to someone who:

- has access to the blockchain/database
- is able to change the state of the blockchain
- and is a consensus participant (Wüst and Gervais, 2018).

To both of these steps, the answer is yes when referring to the LoveEconomy. Data needs to be stored and there are multiple writers to the state of the database, which include the registered businesses and users. In a more general non-profit scenario, the writers are public donors, the charities themselves and any other participating stakeholder depending on the organization. The third step is whether a trusted third party (TTP) is available to verify state transactions. Having a TTP would add another middleman and additional costs, which would include needing to trust that third party. As mentioned in chapter 2, non-profit organisation want to minimise cost and possible fraud. The final step is whether all the writers are known, in both the LoveEconomy and in general non-profit organisations, not all writers are known,

for example the public donors. In conclusion to the above answers, permissionless blockchains are an appropriate approach. We will extend the Wüst and Gervais (2018) with the Ethical Design Frameworks by Lapointe and Fishbane (2019). We will consider the Ethical Design Framework specifically for the LoveEconomy.

The Ethical Design Frameworks extends the Wüst and Gervias model by asking the following additional questions (Lapointe and Fishbane, 2019):

- Can a consistent set of rules help achieve the outcome?
- Will the governing rules be consistent over time?
- Is transparency of the transactions an important feature?
- Is an immutable, auditable record of transactions important?
- Are transactions dependent or interrelated?
- Can the distributed infrastructure reduce the risk of censorship and attack?

A fundamental characteristic of blockchain is having a set of rules, by which all transactions are governed. In the case of the LoveEconomy, there are a consistent set of rules that can be set up to ensure the LoveEconomy operates as planned, and these should not change over time. It is important to ensure that sound human governance is driving the technology, and that the rules set in the system are fair and as expected to all parties (Lapointe and Fishbane, 2019). Additionally, transparency of transactions and an immutable, auditable record of transactions is the most important feature of our design, as we found earlier that trust is one of the major concerns of stakeholders of non-profit organizations. So, to the first three questions above, the answer is yes. Transactions in the LoveEconomy are interrelated, as it is designed to incorporate blockchain into an ecosystem as mentioned in section 3.1. Finally, in this context, the risk of censorship refers to being nontransparent about the ways the funds in a non-profit are used and also not being able to conceal the way the funds are used. The risk of an attack refers to something like a cyber attack (i.e. data breach) or an inside party misusing the funds of the charity. The LoveEconomy structure reduces the risk of censorship by directly using the funds received for the manufacturing of new stickers. This lowers the potential misuse of funds by the charity. Since trust is one of the major concerns of stakeholders of non-profit organizations, we can conclude that a blockchain, and more specifically public blockchains, can be a suitable solution.

There are multiple public blockchains that can be used for such a project. Bitcoin is the first open blockchain, which then initiated further development in the area (Wüst and Gervais, 2018). Bitcoin is a peer-to-peer network which records the

public history of transactions of electronic payments. Zerocash and Ethereum are blockchains built on the techniques used by Bitcoin and are extended to address issues faced by the Bitcoin blockchain.

Zerocash is a privacy-preserving version of Bitcoin. It extends Bitcoins protocol and software by including payment transactions that do not contain any public information about the payments origin, destination or amount and demonstrates the correctness of transactions using zero-knowledge proofs (Zerocash, 2020).

Ethereum blockchain is a permissionless blockchain with more expressive smart contracts. This allows developers to build and deploy decentralized applications (Ethereum, 2020). There are a broad range of building applications on Ethereum, as it allows smart contracts, which enable the Ethereum Virtual Machine, to execute on the blockchain. The most popular languages for writing smart contracts are Solidity and Viper, which are then compiled into the bytecode language required for the computer on which it runs. Examples of existing applications are: cryptocurrency markets, financial applications, decentralized markets and games (Ethereum, 2020). The Ethereum's cryptocurrency is ether, which is used for the execution and creation of smart contracts. Additionally, there are advanced resources available, which allow developers to test their smart contracts and decentralized applications before deploying them in the real world.

Although many other blockchains could be suitable for this experimental project, the Ethereum blockchain will be used. As mentioned before, Ethereum is a permissionless blockchain, and so well suited for the LoveEconomy. Additionally, it is an open source, globally decentralized computing structure, which executes programs called smart contracts. These smart contracts enable developers to build decentralized applications, with already built-in economic functions (Antonopoulos and Wood, 2018). Ethereum community is also one of the largest and most active in the world. It has extensive guides, resources and tools for developers building on its blockchain. There are many videos and channels available, for example Dapp University and Consensys Academy, as well as Ethereum's own user guides and web based learning experiences, such as CryptoZombies, to assist developers with their decentralized applications (dApps). There are also useful resources available to test the smart contracts and decentralized applications, for example Remix and more importantly the Truffle suite. This is important as once a smart contract is deployed, it cannot be altered. Truffle is a testing framework which, alongside Ganache, allows developers to test their smart contract and dApps in a real life setting (Truffle Blockchain Group, 2019). Ganache allows developers to create a personal blockchain.

Ethereum has its own built-in currency, which is called ether. Ether is used as a liquidity layer to allow exchange between various types of digital assets, and for paying transaction fees (Chinchilla, 2019). Each transaction on the Ethereum blockchain contains the recipient of the message, a signature identifying the sender, the amount of ether to be transferred between receiver and sender, an optional data field, a startgas value and a gasprice value. The transaction fee is equal to the startgas times the gasprice. The transaction fee is paid to the miners for mining blocks of transactions and for securing the Ethereum blockchain. The fee system is there to prevent malicious behaviour, infinite loops, distributed service denial attacks or any other computational wastage in code (Chinchilla, 2019). All fees are counted in gas and settled in ether. [1]

3.3 Challenges of adopting Blockchain in the humanitarian sector

Blockchain is a promising breakthrough with great potential for the construction of future applications, specifically for a wide spectrum of humanitarian initiatives. However, there are many challenges and potential consequences, specifically ethical, that may arise from the practical applications of blockchain in this sector (Lapointe and Fishbane, 2019). The problems that enterprises or organizations are trying to solve may be very different from those faced by vulnerable populations. This is due to a widespread of information asymmetry and uneven power dynamics (Coppi and Fast, 2019). Some of these challenges and consequences are mentioned below:

3.3.1 Transparency and immutability

Transparency is one of blockchains most significant benefits, but there are potential problems that may come with it. Although increased transparency often leads to increased trust, it could be undermined through a lack of meaningful consent and engagement in the project design from end-users (Coppi and Fast, 2019). This is where it becomes important to incorporate humanitarian principles into the digital products. Data protection and privacy are important factors to consider, as blockchain is referred to being an open structure by design (Zwitter and Boisse-Despiaux, 2018). This is particularly relevant when it comes to identity registries and other blockchain databases in which personal identifiable information is held. The transparency of personally identifiable information could put people at risk of exploitation, while transparency of a person's ethnic or religious background, sexual

[1] Gas is the name for the fee that comes with each Ethereum transaction. This is explained in more detail in Section 5.4

orientation or other identifiers could put a person at risk of persecution (Lapointe and Fishbane, 2019). The immutability of blockchain removes the ability to be forgotten, for example political refugees, crime witnesses or domestic abuse survivors would not have the choice to completely change their identity (Lapointe and Fishbane, 2019). In our experimental phase, no personal identifiers of the homeless people will be used on the LoveEconomy platform, only how much money has been distributed to them through the use of the LoveEconomy platform.

3.3.2 Decentralization

Another significant benefit of blockchain is decentralization and no longer having to trust a third party, which has many benefits but also comes at a cost. One trade off to consider between decentralization and using a trusted third party is the performance in terms of latency and throughput, which is usually much better in centralised systems. This is because blockchains add additional complexity through their consensus mechanisms (Wüst and Gervais, 2018). Before implementing a blockchain solution, one needs to consider the scalability needs of the project.

3.3.3 Costs and Resources

Non-profit organizations typically face additional challenges from blockchain implementation due to internal processes and lack of understanding amongst the organisations personnel (Coppi and Fast, 2019). Sometimes there exists difficulties between organisations and their partners, for example with issues such as intellectual property and project governance. Both new projects and blockchain implementations in existing organisations may face external and internal challenges which result in delays and additional costs, which can undermine efforts to scale up projects and make them sustainable (Coppi and Fast, 2019).

Another factor to remember is that blockchain always requires servers and computers to process transactions. This can have a major impact on blockchain projects and its scalability in countries where the internet is frequently shutdown (Zwitter and Boisse-Despiaux, 2018). However, those that do not need to worry about the connection, need to consider the consumption of significant environmental energy.

3.3.4 Legal and Regulatory

Developing blockchain solutions are, in part, held back by the absence of a legislative framework. For the non-profit and humanitarian sector this also means that the technology needs to comply with international humanitarian law and human rights law (Zwitter and Boisse-Despiaux, 2018). Examples of these are the principle of

distinction and issues around non-discrimination. In addition to these laws, the technology needs to comply with legal norms and professional codes of conduct (Zwitter and Boisse-Despiaux, 2018). There is a gap in the legal and regulatory space, which could lead to unintended consequences. Resources need to be pooled together for the development of knowledge in this area, to get clear and reliable guidelines.

3.3.5 Public adoption

There is significant risk in the widespread adoption of a blockchain solution by the public (Crosby et al., 2016). The first risk that it involves is change, and although change is constant, there is a resistance to it. One of the important factors needed for mass adoption is usability and simplicity. Currently there is a lack of education and awareness of blockchain, as well as a lack of easy-to-use applications which lead to potential users avoiding having to deal with the unknown (Kshetri, 2017). The LoveEconomy is based in Cape Town, South Africa, and blockchain adoption by the population may be the main concern for the project. In blockchain, individuals need to remember and store their private keys in a safe place. If this key is lost there is no way of retrieving it. People may also be reluctant to pay and trade in ether, as the LoveEconomy is built on the Ethereum blockchain.

3.3.6 Existing literature

In addition to the above, Coppi and Fast (2019) found some issues that arise both in the literature of blockchain applications and also in current blockchain projects in the humanitarian sector. In the current literature there is a lack of clarity about which blockchain applications are specifically humanitarian (Coppi and Fast, 2019). Additionally, there is a disconnect between the hype and the evidence of current blockchain solutions in this field. Coppi and Fast (2019) state that this may be due to the failure to reuse and build upon existing technology, and most pilot projects involving external partners being centred around something new, instead of building on previous projects. Many aid organisations have published contents aimed to present their efforts in adopting blockchain technology, but many of these are informal, vague and sometimes contradictory statements about blockchain, which could contribute to a sense of mistrust regarding the capacity to deliver (Coppi and Fast, 2019). Similarly, new organisations developed specifically as blockchain solutions for humanitarian aid fail to explain exactly what they do and how they do it on their websites and with no proper feedback on their pilot projects. Only recently have projects, such as Disperse, started publishing learnings and analysis relating their work (Coppi and Fast, 2019).

Chapter 4

Experimental Methodology

The LoveEconomy is a browser application linked to Ethereum blockchain back-end functions. To create the LoveEconomy platform HTML, CSS, Javascript and Solidity are used. The smart contracts are written in Solidity, which is a relatively new coding language influenced by C++, Python and Javascript, and used on the Ethereum blockchain. One needs to ensure that the latest version of the code is used, as breaking changes as well as new features and bug changes are introduced regularly. The Truffle Suite was used for the developing and testing the smart contracts and has built-in smart contract compilation, linking and deployment (Truffle Blockchain Group, 2019). Truffle uses the Mocha testing framework and Chai for assertions to provide a solid framework, with which JavaScript tests can be written to ensure that the smart contracts work as expected (Truffle Blockchain Group, 2019). This is important, as once a contract is deployed on the Ethereum Main Net, it cannot be changed or adjusted. Ganache and MetaMask were also used, and played an important role in the development of the prototype. Ganache is a personal blockchain for Ethereum which can be used to deploy and test smart contracts. MetaMask is used to interact with the personal blockchain and allows for interaction with the accounts in the browser. In order to interact with the smart contracts and the Ethereum blockchain environment using HTTP, web3.js can be used. It is the Ethereum Javascript API and has a collection of modules containing specific functionality for the Ethereum ecosystem. For users to interact with the LoveEconomy platform, they would need to download MetaMask and create an account. This would allow them to send and receive ether.

To view the solidity and JavaScript code developed for the minimum viable product, refer to appendix B. A short video on how the prototype currently works can be viewed by following the link below:

`https://drive.google.com/open?id=1b5Tc8Z9ypUvrtj5ZHyFIZDu_FseLetIT`.

4.1 Front-End Overview

The front-end of the LoveEconomy was built using the Bootstrap 5 development network. The minimum viable product platform has three different browser pages. The first is LoveEconomy portal, which is the main site for both the businesses and users to register on the platform. The second portal is the local_deals portal where businesses can add new deals to the platform. On this site businesses first have to a pay their access fee, before being able to add a new deal. In the local_deals portal both businesses and users can also view the current activity on the platform, by looking at the deals available and how many of them have already been sold. Finally, there is the Users portal where users can view and purchase these deals. At the top of each of the portals there is a navigation bar which can be used to access each of the other portals. The navigation bar is also linked to the business and user sign-up modals, which can be used. Under the navigation bar, both the current MetaMask account and the current MetaMask balance of the account in use are given, in each of the three portals. For the prototype, this is mainly for testing purposes, to ensure the correct account is being used for the different applications.

4.1.1 The LoveEconomy portal

The LoveEconomy portal is the home page of the browser application. On this home portal, the LoveEconomy can add new businesses to the platform by clicking the "Add Business" button. Only the LoveEconomy wallet address can add new businesses at this stage. This is to ensure only approved local businesses with adequate deals to offer for the users are added. When a new business is added, using the business name and wallet address, a new business contract is deployed. This means that the LoveEconomy can add new businesses but are unable to perform any further functions for them. When a new business contract is deployed, the business wallet address is immediately set as the owner of the deployed contract.

Users also register on the platform through the LoveEconomy portal, by selecting either the "Sign up with discount code" button, or the "Sign up with ether" button. When a user registers using the discount code, the number of stickers sold would be incremented, as well as the number of users. This is displayed on the portal. When

a user chooses to sign up by paying ether, only the user count is incremented on the platform. At this stage users can only make payments in ether, using MetaMask. Additional fiat options could be implemented in further development stages of the prototype.

The portal displays the number of businesses that have already signed up on the platform, the number of deals sold by the local businesses through the platform, the number of users signed up on the platform and how many of these users used the discount code on a sticker pack. The LoveEconomy can also view details of all the businesses on the platform or search for a specific business using the business address. This would show the LoveEconomy the business name, business wallet address, the business contract address and whether that business is currently active. For privacy reasons, only the LoveEconomy has the option to view these details. Refer to figure 6.1 in the Appendix A for a visual representation of the above descriptions.

4.1.2 Local deals portal

The local deals portal can be reached from the LoveEconomy portal by clicking on "Current Deals" in the navigation bar at the top of the page. This portal does not only show the current MetaMask details, but also the business contract address linked to the current business address logged in. If the logged in account has no business contract, the "Current contracts" field will be left blank.

In this portal, businesses can pay their activation fee and subsequently add new deals to the platform. To pay the business activation fee, the business only has to add the business contract address under the "Pay activation fee" button. Only the business address is able to call this function. If any other address attempts to do this, it will give an error. When the payment is successful it will give a confirmation message below the input box and if the business contract has already paid their activation fee, it will return an alert below the input box. This means that businesses do not need to worry about paying twice, as the platform would not allow this, due to a functionality included in the business smart contracts.

New deals are added using the following inputs: deal name, deal details, expiration date in months and the price in ether. The deal details refer to whether the deal is a two-for-one special, half price special or any other form of discount. The expiration date needs to be given in the number of months, which is then added to the current time in the back-end. However this functionality is not fully implemented yet, as working with time and expiration dates can be very tricky in solidity and more time would be needed for further development. Once a new deal is uploaded, a new token contract is deployed by the business contract.

On this portal the users and businesses can view all the activity on the platform. When clicking the "View all activity" button the business name, the deal name and the number of tokens sold by that point for each deal is shown in the table below the button. Refer to figure 6.2 in the Appendix for a visual representation of the platform.

4.1.3 Users portal

On the users portal, users can view all the current deals being offered by clicking the "View all deals" button, which displays the business name, all the details of the deals active on the platform and an option to purchase them. The user has the option to click on the "Buy" button in the Buy Deal row. The price of the deal shown both in ether and in the current converted Rand amount. The exchange rate information is taken from the CryptoCompare website using their API. When a user purchases a new deal, they receive a token from that deal's smart contract, which they can then use at the business to redeem the deal when they wish.

The Users portal can be reached by clicking on "Users" on the navigation bar. The users can also redeem their tokens on this portal, however this functionality has not yet been included on the front-end of the platform due to time constraints and can be added at a later stage in the development of the prototype. For a visual representation of the users portal, refer to figure 6.3 in the Appendix.

4.2 Back-End Overview

The back-end overview will describe the technical details implemented on the Ethereum blockchain for the LoveEconomy. There are three different smart contracts that needed to be developed: The LoveEconomy contract, the Local Business contract and the Deals Token contract. Individual business and deals contracts are deployed for two main reasons:

1. If all the deals and their tokens were sold and traded in each of the LocalBusiness's smart contracts, it may blow up the business contracts. This means that it is more vulnerable to technical flaws and malicious attacks, which could lead to large losses (Sayeed, Marco-Gisbert and Caira, 2020). Additionally, by deploying the DealsToken contracts, it makes the dApp more flexible to changes. These contracts hold the general information of the deals, stored in state variables and deals-customer information in arrays. The contracts also contain the functionality for users to trade and use these tokens.

2. Enable easier upgradability. If errors in the code of the DealsToken contracts

are found, they can be fixed and the contract addresses can be switched out, from the old to the new one.

4.2.1 The LoveEconomy contract

The LoveEconomy is the only smart contract which is deployed at the initial deployment. It has three constructor variables which need to be set before deployment: the user discount code, the user fee and the business fee. The user discount count will be a complicated combination of numbers and letters provided at the back of the sticker packs. The discount code is hashed using the SHA-3 algorithm which stands for Secure Hash Algorithm 3, and is also known as Keccak algorithm. Hashing is a cryptographic method of converting any kind of data into a string of characters (Rothrie, 2018). This provides security for the discount code through encryption, to prevent people being able to see the discount code on the platform and using it to register. There is no way to "reverse engineer" the hash function by analysing its values. Setting the state variable to private in the smart contract would not secure the discount code. The user fee, will be in ether and is for those users who would like to have access to the platform by paying a fee instead of purchasing a sticker pack. A constructor function is declared with the constructor key word and is executed upon the contract creation. It allows developers to run contract initializing code.

AllUsers: Is an array of all user addresses on the platform.
AllBusinesses: Is an array of all business addresses on the platform.
owner: The address of the owner of the LoveEconomy, which will be the LoveEconomy's address and the address which deploys this contract initially.
discountCodeHashed: A variable containing the hashed discount code.
userFee: A variable giving the user fee, which users need to pay to sign up to the platform.
businessFee: A variable giving the fee, which businesses need to pay to access the platform and add new deals.
userCount: A variable which tracks the number of users which registered to the platform.
businessCount: A variable which tracks the number of businesses registered to the platform.
discountCodeCount: A variable which tracks the number of discount code used by users to register. This also represents the number of "special" stickers sold by the LEAs.
businessAddresstoDetails: Is a mapping from the business addresses to the business details held in a struct called local_business, which are: the business name, the business contract address deployed when the business was added and bool repre-

senting whether or not the business is currently active.

userAddresstoDetails: Is a mapping from the user address to their details held in a struct called user. These details include users name, a bool representing whether the user used a discount code to register and a second bool, representing whether the user is currently active or not.

addBusiness: This function adds new businesses to the platform. It takes in the business name and the business wallet address as its arguments. The function checks whether the business address is not already an active participant on the platform and whether the address calling the function is that of the owner. If these two requirements are met, a new business is added and a new business contract deployed. This deployed contract takes the business name and wallet address as constructors and sets the business wallet address as the owner of the new contract in the constructor function. The business address is then pushed into the AllBusinesses array, the details of the new business are added to the businessAddresstoDetails mapping and the businessCount is incremented by one.

addDiscountUser: This function adds new users to the platform who want to register using the discount code. The function takes the user name, the user wallet address and a bool for whether or not they are using a discount code. It checks that the user wanting to register is not already an active user on the platform, and that the discount code provided by the user is the correct code. Once both of the these requirements are met both the discountCodeCount and the userCount are incremented by one. The user address is then pushed into the AllUsers array and the new users details are added to the userAddresstoDetails mapping.

addUser: This function adds new users to the platform who want to register by paying the user fee. The addUser function works in a similar way to the addDiscountUser function. However, this function is a payable function, which means that it can receive ether, and only takes the user name and wallet address as input variables. The function checks that the user registering is not already an active user of the platform and whether the fee being paid is at least equal to the userFee variable mentioned above. Once the requirements are met, userCount is incremented, the user address is pushed into the AllUsers array and the new users details are added to the userAddresstoDetails mapping. This function is separate from the above function as it needs to be a payable function and by separating them, they are easier to use and implement on the front end. However this can be made more efficient at later stages of development.

setActiveFlag: The LoveEconomy can change the active status of a business once they no longer want to be a part of the platform or no longer want to offer any deals. It can also re-activate any deactivated business accounts. The function checks that the address calling this function is that of the owner and takes the business wallet

address and a bool as to whether activate or deactivate the business as input variables. This functionality is not fully implemented at this stage.
updateUserFee: The LoveEconomy can choose to change the user fee. The function checks that the address calling this function is that of the owner and takes the updated fee as an input variable. It then updates the userFee variable with the new fee.
updateBusinessFee: The LoveEconomy can also choose to change the business fee. The function checks that the address calling this function is that of the owner and takes the updated business fee as an input variable. It then updates the businessFee variable with the new fee.
isActive: This function takes a business wallet address as an input and returns the active bool of the business from the businessAddresstoDetails mapping. This function is used in the LocalBusiness smart contracts.
Fallback function: At the end of the contract there is an empty payable function, which is required for the contract to be able to receive ether sent to it by another contract. For businesses to activate their account, they need to pay the business fee to the LoveEconomy contract.

4.2.2 The LocalBusiness contract

These smart contracts are deployed each time a new business is added to the platform. When the contract is deployed, the business wallet address is set as the owner of the contract. The contract allows businesses to create new deals. Before being able to add new deals the business contract needs to pay the business fee to the LoveEconomy contract by calling the activateContract function. Once this fee has been paid, the business contract is free to add new deals to the platform. Each time the business adds a new deal, a DealsToken contact is deployed.

owner: The owner of the contract, which is automatically set to the business wallet address.
paid: This is a bool variable, which is initially set as false when the contract is created. Once the business is added and the activateContract function can be called, the bool is set to true. This is so that each business pays a fee to be a part of the LoveEconomy, before they add a deal to the platform.
LoveEconomyAddress: The address of the LoveEconomy smart contract.
businessName: The name of the business, which created this contract.
tokenContracts: An array of all the token contract addresses created by this business. Each new deal created by the business deploys a new DealsToken smart contract. The contract addresses are used as the deals identification (id's).
tokenAddresstoDeal: Is a mapping from the token contract address of the deal

to the details of the deal held in the Deal struct. This contains the businesses name, the business contract address, the deals token contract address, the deals name, details, price, expiration date and an active bool.

activateContract: This function allows businesses to pay their activation fee. It checks that the caller of the function is the owner of the contract, that the amount of ether sent to the function is more than or equal to the businessFee variable and that the paid variable mentioned above is still set as false. Once these requirements are met, the paid variable of the contract is set to true and the message value sent by the caller of the function is sent to the LoveEconomy contract address.

addDeal: This function adds a new deal for the business and takes the deals name, details, expiry date and price as inputs. It checks that the address calling the function is that of the owner, that the business is currently active, which is done by calling the isActive function from the LoveEconomy smart contract and that the paid variable is set to true. Once these requirements are met, a new deal is added to the platform and a DealsToken smart contract is deployed, with the deal's name and price as constructor variables. The new token address is pushed into the tokenContracts array and the deal's information is added to the tokenAddresstoDeal mapping. Once the contract is deployed, the constructor function in the DealsToken contract sets the business contract address variable as the address which deployed the contract and the token contract which is the deployed contracts' address and also the identification of that deal.

getDealCount: This is a function that returns the number of deals created by this particular business, by calling the length of the tokenContract array.

setDealActiveFlag: The business can choose to deactivate a deal. Only the owner of the contract can call this function.

4.2.3 The DealsToken contract

This contract tracks the tokens of the available deals purchased by the users. The contract inherits from the ERC20 smart contracts in the OpenZeppelin library and uses some of the functionality of the fungible ERC20 tokens. The ERC20 token is a fungible token, which means that each token is exactly equal to any other token of that contract and no token has special rights or behaviours associated with them. It was decided to use the functionality of fungible ERC20 tokens, rather than non-fungible ERC721, as every purchased deal is identical. Each deal purchased from the same DealsToken smart contract is interchangeable. Non-fungible tokens are unique and cannot be replaced by any other non-fungible token 0xcert (2018).

businessContractAddress: The business contract address which created the new deal and deployed the token contract.

tokenContractAddress: The address of that particular contract.
dealName: The name of the deal created.
dealPrice: The price of the deal created. Although the platform offers deals, the users still have to pay the rest of the fee that the deal is worth.
dealsBoughtCount: A counter to track the number of deals purchased. The TotalSupply function of the ERC20 token cannot be used in this case, as when a user redeems a deal they have purchased, the business contract acknowledges the deal by burning the token. So the dealsBoughtCount counts the overall tokens purchase from that deal, even if they have been used and burnt already.
pruchaseDeal: This is a payable function which allows users to purchase the deal of this contract. Initially this function checked whether the account calling this function is an active user of the platform, by calling a function from the parent contract which checks whether the user is active, however this functionality had to be excluded in the prototype, due to gas limits. The function checks that the amount of ether sent to the contract is at least equal to the price of the deal. Once the requirement is met, the dealsBoughtCount variable is incremented by one and the _mint function of the ERC20 contract is called to mint one token to the address calling the function. The _mint function takes two input variables, the address to which the token should be minted to and the amount of tokens to mint to that contract. The _mint function checks that it is not the 0x0 address calling the function, if this is approved, the _totalSupply of the token is updated, and so is the balance of the user calling the function.
sellToken: This is also a payable function, and allows users to sell their tokens to other users. It takes the recipient address and the amount of tokens to be sent as inputs. It checks that the address calling this function is not the 0x0 or the token contract address itself and that the recipient of the token is also an active user on the platform. If these requirements are met, the caller of the function is set to the sender and the _transfer function from the ERC20 token contract is called. The _transfer function checks that both the sender and recipient of the token are not the 0x0 address. It then adds the token amount to the balance of the recipient and subtracts the token amount form the balance of the sender.
useDeal: The useDeal function takes the account address of the user wanting to redeem their token as an input variable. It checks that the caller of the function is the business wallet address. The token of the user is then burnt, by calling the _burn function from the ERC20 token contract. This function deletes the token from the balance of the user and from the total supply of the tokens. The burn function checks that the address calling the function is not the 0x0 address.

Truffle assertion tests were written and passed for each of the functions discussed above.

Chapter 5

Discussion

5.1 Current functionality towards transparency

The minimum viable product created for this study contains the functionality needed to build the LoveEconomy on the blockchain. The functionalities included in the prototype are:

1. Allowing the Secret Love Project to register businesses
2. User registration to the platform
3. A secure count of:
    - discount codes used as an estimate of stickers sold
    - registered businesses
    - registered users
    - deals sold per business
4. Allowing businesses to add new deals to the platform
5. Allowing the users to buy, use or trade these deals amongst each other

The platform allows the public to view and interpret the impact which the LoveEconomy has on the homeless people in Cape Town and the local businesses. Since people are giving money directly to the homeless by purchasing the stickers and the registration fees would be directly transferred to the sticker manufacturing company, there is a transparent flow of the funds and how the charity operates. This transparency would hopefully lead to more participation in the Secret Love Project and higher levels of trust within the public for the charity.

The activity that can be observed on the website and on the Ethereum blockchain confirms that the charity is active and providing the services that it claims to. This already lowers the chances of a blockchain based charity being discovered to be some sort of scheme, like the Cup Trust scandal mentioned in section 2.2.1. By looking at which business are active over a long period of time and add new deals regularly, one can observe that the LoveEconomy is supported and works as a functioning circular economy. However, naturally it will take time before the LoveEconomy becomes a fully self-supported economy. By looking at the number of discount codes used by users to register to the platform, the public can observe that stickers are still being sold and used. The displayed number of deals indicate the activity of the users on the platform and the impact the platform has on local businesses.

There are additional functionalities which can be added to the platform, if it were to go into further production.

5.2 Security

As discussed in section 3.3, storing personal information of already vulnerable people could lead to further problems. The personal information, or any information, about the LEAs is purposefully excluded on the LoveEconomy platform. The goal of the Secret Love Project is to reintegrate the LEAs back into society and once they are reintegrated, they may wish to leave that part of their life behind, and not be able to be tracked back to it. Additionally, the LoveEconomy and the Secret Love Project do not control what the homeless do with the money that they make through them. This is a way of empowering the beneficiaries to decide on the use of their funds, similarly to how Binance Charity distribute funds as mentioned previously in section 2.4.2. The Secret Love Project does provide help to those homeless people who seek help when addicted to drugs, for example, however that is a separate part of the organization and could be included into the LoveEconomy platform at later stages of development. This can be done in a transparent way by being able to pay rehab centres directly with ether paid to the LoveEconomy by businesses and users.

5.3 Limitations and future work

One of the current limitations in the design of the prototype is that the discount code is not completely secure. Although it is secure from being seen on the blockchain by being hashed, under the current design people can give the code to their friends or even post it online. This is because currently the discount code is given to the LoveEconomy smart contract as a constructor variable, which subsequently means

the discount code is fixed and the same for all users. This was only done for testing purposes and if the prototype would go into further development, each code given on a sticker pack would be unique and expire once it is used. An example of a tool that could be used for this is voucherify.

Voucherify is a platform which offers an API for developers to build personalized coupon, discount, referral and loyalty campaigns (rspective, 2019). It allows developers to customize discount codes using a flexible coupon API, enabling them to create fixed-code coupons or generate millions of random, once-off codes. These codes can be given in text or in QR/barcode format. The LoveEconomy would be interested in generating many random, once-off codes, which could be printed on the sticker packs in either text or as QR codes. To use the API on the blockchain directly, one can make use of a blockchain oracle. A blockchain oracle is a third-party information source, which has the sole function of supplying data to blockchains. It is a way for blockchains to communicate outside of their network. Smart contracts can evaluate incoming data from an oracle and initiate a flow of execution depending on the code in the smart contract. Currently the LoveEconomy is dependent on the integrity of the people who want to be a part of the platform.

Another functionality which has not been fully implemented on the front-end of the platform is the redemption of the deal tokens once they have been purchased by the user. One way of implementing this is by creating a unique QR code for the users token, which can be scanned by the business once the user redeems the token. The useDeal function in the DealsToken smart contract would be executed and the token belonging to that user will be burnt and removed from the total supply. Additionally, a platform for users to sell and buy tokens from each other would need to be set up. This functionality has not been implemented at this stage of the prototype.

The LoveEconomy is designed such that businesses (and some users) register by paying the LoveEconomy some amount in ether. If the prototype would go into further development, these businesses and users would get a certain amount of time, for example 6 months, before having to reactivate their accounts. Additionally, deals have expiry dates, which put a time limit on their availability. These functionalities have not been fully implemented.

For the LoveEconomy to become a circular economy, the sticker manufacturing company would need be on board with the LoveEconomy and possibly be willing to accept ether as a form of payment. However, this has not been implemented at this stage of the prototype, and would include the cooperation of an outside firm, which may be a whole new challenge within itself.

5.3.1 Platform owner

One of blockchains many features is that it can be made to be a fully decentralized platform. This means that the platform would not depend on central governance or any middle-men to conduct the operations. Having a decentralized platform can increase trust and lower dependencies on third parties, as well as reduce transaction costs. However, the LoveEconomy prototype is not fully decentralised, as the address which deploys the LoveEconomy smart contract, which will belong to the Secret Love Project, is responsible for adding new businesses once they have been vetted and approved. This vetting process can be improved and made more transparent in later development stages by, for example, using a verification system that verifies businesses according to certain documentations they can provide, such as business licenses using IPFS. IPFS stands for Inter Planetary File System and it is a peer-to-peer distributed file sharing system (Protocol Labs, 2020). Alternatively, a different logic could be set up, which would make the vetting of the businesses more decentralized. This would be to ensure that the decision of which businesses are allowed on the platform is not solely dependent on the Secret Love Project. This is to avoid any biases or possible fraud, which could occur if the Secret Love Project prevents certain businesses from being on the platform due to personal reasons.

5.3.2 Time

The current design of the prototype is purely on a personal Ethereum blockchain, Ganache, where each transaction takes a few seconds to be executed. This is considerably slower than the transaction speeds of existing legacy transaction processing systems. Additionally, it is slow when calling information from the blockchain to view on the front-end. These transactions are likely to be even slower on the Ethereum Main Net. The average block time, at the time of writing this paper, is 13.48 seconds according to the the ethstats.net website. This means that there is a lag between when the transaction is made and when it is added to the blockchain. Currently the prototype is primarily designed to help the homeless people within Cape Town, this means that at this stage of development the lag time is not a major concern. Compared to Bitcoin's block time, which has an average block time of just under 10 minutes at the time of writing this paper, according to data.bitcoinity.org/, Ethereum is fairly fast.

However, viewing the information, retrieved from the blockchain, on the front-end should be improved. There is a lag between clicking the button to retrieve the information, until it is displayed. Currently it is slow, because to call the relevant data, such as all the current deals available, one needs to loop through each of the

business contracts and retrieve information from those contracts. This could be improved by adding a SQL database from which the information is called, and then verified by the blockchain.

If this product were to go into further development, the code would need to be optimized further, which could increase the transaction speed and make the code more efficient.

5.4 Cost

Each transaction on the Ethereum blockchain is associated with a fee, known as gas. The users of the blockchain are charged gas for each computational step taken to execute certain operations. Usually, a computational step on the Ethereum blockchain costs 1 gas, however, some operations cost more because either they are computationally more expensive or they increase the amount of data that must be stored as part of the state (Chinchilla, 2019). No fees are charged for viewing data on the blockchain. The fee is there to incentivise miners to mine blocks onto the blockchain, as well as prevent any malicious activity (Chinchilla, 2019).

Each sender must specify a gas limit before submitting it to the network, which is the maximum amount of gas the sender is willing to pay for the transaction. Miners stop executing when the gas runs out and everything reverts back to its original state before the transaction (Chinchilla, 2019). If there is gas left once the transaction is done, it gets returned to the sender. However, miners can only include transactions that are less then or add up to the block gas limit, and so are more likely to include those transactions which do not inflate that overall gas. It is wiser for senders to send an amount of gas that is only slightly higher than what is needed for the transaction (Chinchilla, 2019).

The total transaction fee is equal to the gas price multiplied by the gas used. Gwei is the unit of ether which is typically used to denominate gas prices. One gwei is equal to 0.000000001 ether. Currently the standard gas price is 2 gwei. There is also a gas price for faster transactions and slower transactions, which are currently 8 and 1 gwei respectively (Concourse Open Community, 2020). One can choose a price, depending on the urgency of the transaction, as transactions with higher prices get processed faster than those lower prices. If one does not care how long the transaction will take, one would choose the cheapest gas price, which is also safe. A safe price is one that will be successful, even if it is slower. To give an example, the deployment of the LoveEconomy smart contract uses about 6 609 731 gas, which at a price of 2 gwei, has the total cost of 0.013219427 ether. The gas used is given from

a private blockchain and is likely to be different on the Main Ethereum network.

These costs need to be factored into the fees of the businesses. For example, the Secret Love Project could end up charging slightly higher registration fees to cover the cost for using the blockchain. Businesses and users will also need to be made aware of these fees. However, the fees are minimal and could be reduced further by optimizing the code.

Chapter 6

Conclusion

This LoveEconomy prototype was developed as a foundation for further improvement and despite being in its early stages of development, this minimum viable product illustrates that blockchain is a feasible solution for improving transparency in the charity industry. The immutability characteristic of blockchain, as described in section 2.5, ensures that the data made available to the public cannot be modified and is therefore secure and true.

The concept of this prototype is to motivate people and businesses to help each other, by also uplifting those who are less fortunate. Designing the LoveEconomy as a circular economy allows local businesses, users and LEAs to continuously support each other.

For this to become an active platform for the Secret Love Project charity, this minimum viable product developed for this study would need to be further developed and improved to incorporate the limitations suggested in section 5.3.

This initial objective was to create a platform for the Secret Love Project, which would contribute to increasing the trust of donors in the charity. This has been partly met, as all transactions on the platform would be transparent for anyone to see. The main shortcoming is in the final step, where any funds given to the Secret Love Project on that platform are transparently and automatically being used to manufacture more stickers, to keep the project going.

References

0xcert. 2018. "Fungible vs non-fungible tokens on the blockchain." https://medium.com/0xcert/fungible-vs-non-fungible-tokens-on-the-blockchain-ab4b12e0181a.

Accenture. 2020. "Blockchain for digital identity." https://www.accenture.com/za-en/services/blockchain/digital-identity.

AidCoin. 2017. "AidCoin (white paper)." https://www.aidcoin.co/assets/documents/whitepaper.pdf.

Akingbola, K, Rogers, SE and Baluch, A. 2019. Nature of Change in Nonprofit Organizations. In *Change Management in Nonprofit Organizations*. Springer pp. 37–72.

Angraal, S, Krumholz, HM and Schulz, WL. 2017. "Blockchain technology: applications in health care." *Circulation: Cardiovascular Quality and Outcomes* 10(9):e003800.

Antonopoulos, Andreas M and Wood, Gavin. 2018. *Mastering ethereum: building smart contracts and dapps*. O'reilly Media.

Bankymoon. 2020. "USIZO: Crowd funding utilities for needy schools." http://bankymoon.co.za/social-projects/.

Binance Charity Foundation. 2019. "Binance Charity: About Us." https://www.binance.charity/about.

Bothwell, RO. 2001. "Trends in self-regulation and transparency of nonprofits in the US." *International Journal of Not-for-Profit Law* 2(3):604–622.

Bradshaw, DJ. 2018. Technology Disruption and Blockchain: Understanding Level of Awareness and the Potential Societal Impact. Master's thesis Dublin, National College of Ireland.

Burt, E and Taylor, J. 2011. Charities' use of the internet: Current activities and future opportunities. Technical report nominet trust.

CAF Southern Africa. 2019. South Africa Giving 2019: An overview of charitable giving in South Africa. Technical report Charities Aid Foundation Southern Africa.

Charity Commission. 2008. Charities back on track: Themes and lessons from

the Charity Commission's compliance work 2007-08. Technical report London: Charity Commission.

Charity Navigator. 2020. "Charity Navigator: Your Guide to Intelligent Giving." https://www.charitynavigator.org/.

Charity: water. 2020. "Charity: water." https://www.charitywater.org/.

Chinchilla, C. 2019. "A Next-Generation Smart Contract and Decentralized Application Platform (white paper)." https://github.com/ethereum/wiki/wiki/white-paper.

Clean Water Coin. 2014. "Clean Water Coin." https://www.cleanwatercoin.org/.

Concourse Open Community. 2020. "ETH Gas Station." https://ethgasstation.info/.

Coppi, G and Fast, L. 2019. Blockchain and distributed ledger technologies in the humanitarian sector. Technical report HPG Commissioned Report.

Cordery, C. 2013. "Regulating small and medium charities: Does it improve transparency and accountability?" *Voluntas: International Journal of Voluntary and Nonprofit Organizations* 24(3):831–851.

Cordery, Carolyn and Deguchi, Masayuki. 2018. "Charity registration and reporting: a cross-jurisdictional and theoretical analysis of regulatory impact." *Public Management Review* 20(9):1332–1352.

Crosby, M, Pattanayak, P, Verma, S and Kalyanaraman, V. 2016. "Blockchain technology: beyond bitcoin." *Applied Innovation Review* 2(6-10).

Decoodt, C. 2017. "An overview of the Giveth Donation Application." https://medium.com/giveth/what-is-the-future-of-giving-d50446b0a0e4.

Di Angelo, Monika and Salzer, Gernot. 2020. Tokens, types, and standards: identification and utilization in Ethereum. In *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE pp. 1–10.

Elion, M. 2019. "Secret Love Project." https://www.secretloveproject.co.za/.

Ethereum. 2020. "Ethereum." https://ethereum.org/what-is-ethereum/.

Fodor, S, Radebe, T and Werksmans Attorneys. 2018. "Foreword to the Charity Global Guide." https://uk.practicallaw.thomsonreuters.com/9-632-4485?transitionType=Default&contextData=(sc.Default).

Furneaux, Craig and Wymer, Walter. 2015. "Public trust in Australian charities: Accounting for cause and effect." *Third Sector Review* 21(2):99–127.

Galen, D, Brand, N, Boucherle, L, Davis, R, Do, N, El-Baz, B, Kimura, I, Wharton, K and Lee, J. 2018. "Blockchain for social impact: moving beyond the hype." *Center for Social Innovation, RippleWorks* .

URL: *https://www.gsb.stanford.edu/sites/gsb/files/publication-pdf/study-blockchain-impact-moving-beyond-hype_0*

.pdf

Gilliland, N. 2019. "How digital technology is impacting the charity sector." https://econsultancy.com/digital-technology-impacting-charity-sector/.

Green, C. 2018. No charity left behind: the need for a digital third sector. Technical report techtrust.

Harris, EE and Neely, D. 2018. "Determinants and consequences of nonprofit transparency." *Journal of Accounting, Auditing & Finance* p. 0148558X18814134.

Higgins, S. 2016. "How Bitcoin Brought Electricity to a South African School." *Coindesk, blog* .

Holmes, JG, Miller, DT and Lerner, MJ. 2002. "Committing altruism under the cloak of self-interest: The exchange fiction." *Journal of experimental social psychology* 38(2):144–151.

Honey, M. n.d. "Guide to the Nonprofit Organisations (NPO) Act." https://www.etu.org.za/toolbox/docs/building/guide.html.

Horton, R. 2015. Transparency and Trust in the Charity Sector. Technical report The Centre for Social Innovation at Cambridge Judge Business School, University of Cambridge.

Hyndman, N and McConville, D. 2016. "Transparency in Reporting on Charities' Efficiency: A Framework for Analysis." *Nonprofit and Voluntary Sector Quarterly* 45(4):844–865.

Hyndman, N and McDonnell, P. 2009. "Governance and charities: An exploration of key themes and the development of a research agenda." *Financial Accountability & Management* 25(1):5–31.

Hyndman, Noel. 2017. "The charity sector—changing times, changing challenges." https://doi.org/10.1080/09540962.2017.1281608.

Krach, K. 2017. "8 of the Top Ways Technology Has Influenced Philanthropy." https://medium.com/@KeithKrach/8-of-the-top-ways-technology-has-influenced-philanthropy-9a30c41f87bb.

Kshetri, N. 2017. "Will blockchain emerge as a tool to break the poverty chain in the Global South?" *Third World Quarterly* 38(8):1710–1732.

Lapointe, C and Fishbane, L. 2019. "The blockchain ethical design framework." *Innovations: Technology, Governance, Globalization* 12(3-4):50–71.

Lee, W and Johnson, G. 2017. Nonprofit Ethical Case Study: Reynolds' fraudulent cancer charities. Technical report University of San Francisco, School of Management.

Limburg, Diana, Knowles, Cathy, McCulloch, Maureen and Spira, Laura. 2017. "Integrated performance management using information technology: a study of UK charities." *Public Money & Management* 37(3):181–188.

Maersk. 2019. "TradeLens blockchain-enabled digital shipping platform continues expansion with addition of major ocean carriers Hapag-Lloyd and Ocean Network Express." https://www.maersk.com/news/articles/2019/07/02/hapag-lloyd-and-ocean-network-express-join-tradelens.

Manyaka, RK and Nkuna, NW. 2014. "The phenomenon of corruption in the South African public sector: challenges and opportunities." *Mediterranean Journal of Social Sciences* 5(27 P3):1572.

Mazet, R. 2018. "Alice: introducing impact investing." https://medium.com/alice-si/alice-introducing-impact-investing-1d040f4cb427.

Mazet, R and Wojciechowski, J. 2017. "Alice (white paper)." https://github.com/alice-si/whitepaper/blob/master/Alice%20white%20paper%20-%20FV%200.9.pdf.

McConville, D. 2017. "New development: Transparent impact reporting in charity annual reports—benefits, challenges and areas for development." *Public Money & Management* 37(3):211–216.

Menges, W. 2016. "McLean trust in court over millions." https://www.namibian.com.na/148824/archive-read/McLean-trust-in-court-over-millions.

Miragliotta, P. 2014. "Everything You Need To Know About the 4 Major Charity Watchdogs." https://bloomerang.co/blog/everything-you-need-to-know-about-the-4-major-charity-watchdogs/.

Mohanta, BK, Panda, SS and Jena, D. 2018. An overview of smart contract and use cases in blockchain technology. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE pp. 1–4.

Piper, A-M, Partner, Farrer and LLP, Co. 2019. "Foreword to the Charity Global Guide." https://uk.practicallaw.thomsonreuters.com/w-008-9958?navId=15C8EEF5DB47F2C0FDC2E2ACBC0EFC1D&comp=pluk&transitionType=Default&contextData=%28sc.Default%29.

Populus and Charity Commission For England & Wales. 2018. Trust in Charities, 2018: How the public views charities, what this means for the sector, and how trust can be increased. Technical report Populus, England.

Protocol Labs. 2020. "IPFS powers the Distributed Web." https://ipfs.io/.

Ratcliffe, R. 2018. "Bono's anti-poverty campaign faces claims of harassment." https://www.theguardian.com/global-development/2018/mar/10/bonos-anti-poverty-campaign-faces-claims-of-harassment.

Rittelmeyer, H. 2014. *Independent Charities, Independent Regulators: The future of not-for-profit regulation*. Centre for Independent Studies St Leonards.

Rothrie, S. 2018. "How cryptographic algorithms and hashing keep blockchain secure." https://jaxenter.com/cryptographic-hashing-secure

-blockchain-149464.html.

rspective. 2019. "Personalized trackable discount coupons." https://www.voucherify.io/discount-coupons.

Rule-Groenewald, Candice, Timol, F, Khalema, E and Desmond, C. 2015. "More than just a roof: Unpacking homelessness." *Human Sciences Resource Center Review* .

Saleh, H, Avdoshin, S and Dzhonov, A. 2019. Platform for Tracking Donations of Charitable Foundations Based on Blockchain Technology. In *2019 Actual Problems of Systems and Software Engineering (APSSE)*. IEEE pp. 182–187.

Sargeant, Adrian and Lee, Stephen. 2002. "Individual and contextual antecedents of donor trust in the voluntary sector." *Journal of Marketing management* 18(7-8):779–802.

Sargeant, Adrian and Lee, Stephen. 2004. "Donor trust and relationship commitment in the UK charity sector: The impact on behavior." *Nonprofit and Voluntary Sector Quarterly* 33(2):185–202.

Sayeed, Sarwar, Marco-Gisbert, Hector and Caira, Tom. 2020. "Smart contract: Attacks and protections." *IEEE Access* 8:24416–24427.

Shin, Eun-Jung, Kang, Hyoung-Goo and Bae, Kyounghun. 2020. "A study on the sustainable development of NPOs with blockchain technology." *Sustainability* 12(15):6158.

Sinclair, RMS. 2010. Understandability and transparency of the financial statements of charities PhD thesis Auckland University of Technology.

Sullivan, L and Elliot, J. 2015. "How the Red Cross Raised Half a Billion Dollars for Haiti and Built Six Homes." https://www.propublica.org/article/how-the-red-cross-raised-half-a-billion-dollars-for-haiti-and-built-6-homes.

Sullivan, L and Elliot, J. 2016. "Report: Red Cross Spent 25 Percent Of Haiti Donations On Internal Expenses." https://www.npr.org/2016/06/16/482020436/senators-report-finds-fundamental-concerns-about-red-cross-finances.

Szper, R and Prakash, A. 2011. "Charity watchdogs and the limits of information-based regulation." *VOLUNTAS: International Journal of Voluntary and Nonprofit Organizations* 22(1):112–141.

TechBullion, PR. 2018. "UNICEF's First Blockchain Investment Amply Ensures South African Early Childhood Development Centres Get the Funding They Need." https://techbullion.com/unicefs-first-blockchain-investment-amply-ensures-south-african-early-childhood-development-centres-get-the-funding-they-need/.

Truffle Blockchain Group. 2019. "SWEET TOOLS FOR SMART CONTRACTS."

https://www.trufflesuite.com/.

UNSG. 2012. "Secretary-General's closing remarks at High-Level Panel on Accountability, Transparency and Sustainable Development." https://www.un.org/sg/en/content/sg/statement/2012-07-09/secretary-generals-closing-remarks-high-level-panel-accountabilityl.

Verhulst, S, Young, A and Zambrano, R. 2018. "Amply - Blockchain and Secure Identity for Early Childhood Development (ECD).".

Western Cape Government. 2019. "Homelessness." https://www.westerncape.gov.za/general-publication/homelessness-0.

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

Zerocash. 2020. "Zerocash." http://zerocash-project.org/.

Zheng, Z, Xie, S, Dai, H-N, Chen, X and Wang, H. 2018. "Blockchain challenges and opportunities: A survey." *International Journal of Web and Grid Services* 14(4):352–375.

Zwitter, A and Boisse-Despiaux, M. 2018. "Blockchain for humanitarian action and development aid." *Journal of International Humanitarian Action* 3(1):1–7.

Appendix A

Front-End Overview

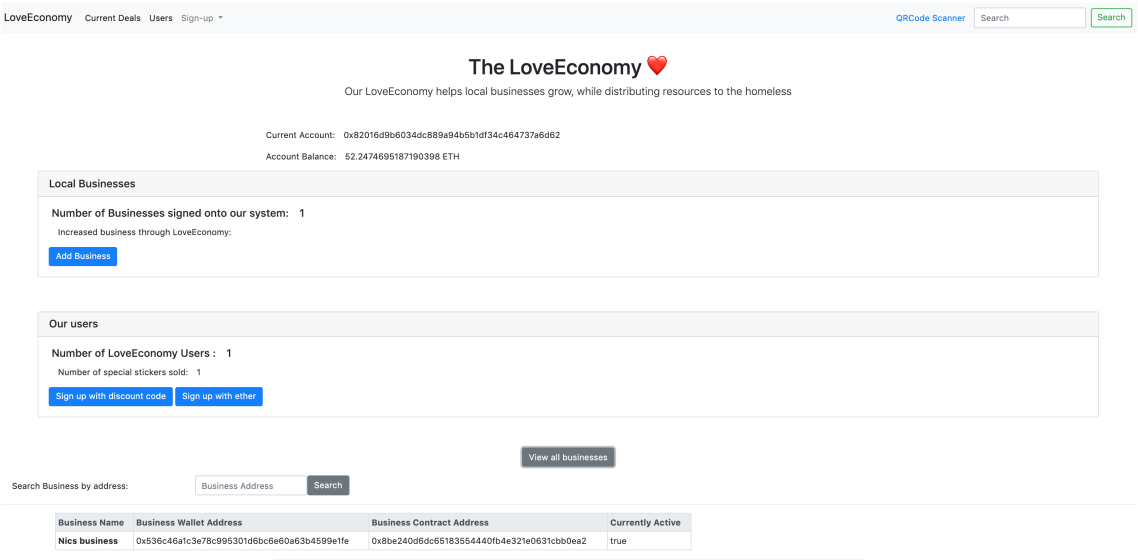

Figure 6.1: LoveEconomy portal

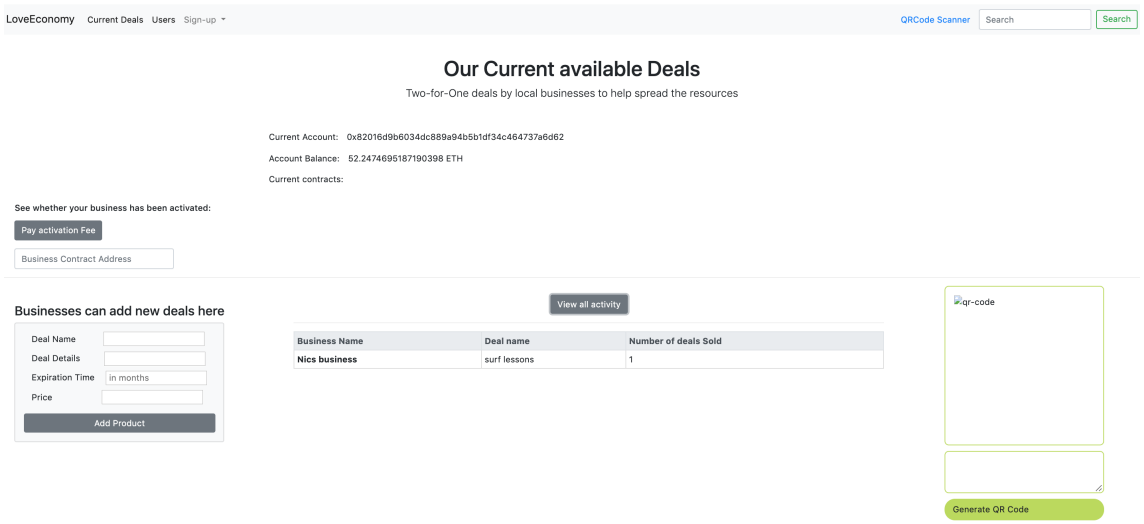

Figure 6.2: Business portal

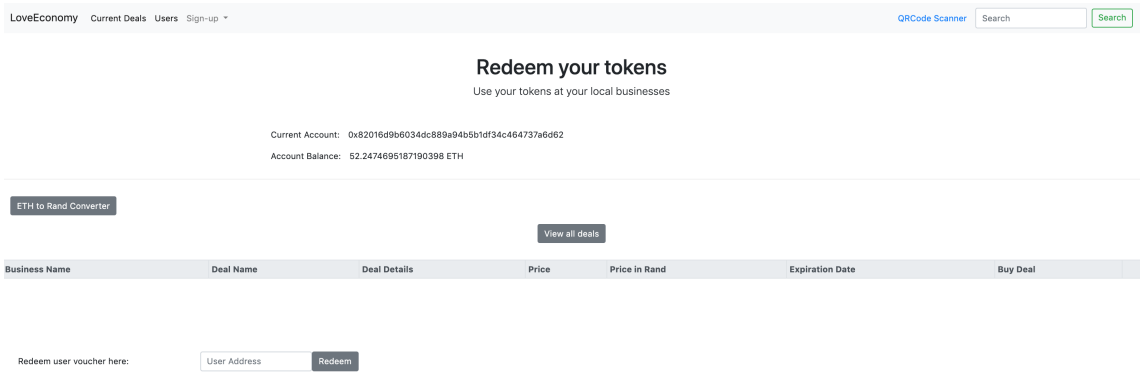

Figure 6.3: Users portal

Appendix B

For access to the latest code, refer to my GitHub account `https://github.com/julipahl/LoveEconomy_2`.

B.1 Solidity code

B.1.1 LoveEconomy smart contract

```
1  pragma solidity >=0.4.21 <0.6.2;
2
3  import "./LocalBusiness.sol";
4
5  contract LoveEconomy {
6
7   // object to store local businesses that sign up for the Love
       Economy
8
9      struct local_business {
10         string businessName;
11         address businessContractAddress;
12         bool active;
13     }
14
15 // object to store Love Economy user details
16
17     struct user {
18         string userName;
19         bool discountCodeUsed; // whether the user signed in using
               a discount code
20         bool active;
21     }
22
23     address[] public AllUsers; // array holding all user addresses
24
```

```
25      address[] public AllBusinesses; // array holding all business
            addresses
26
27      address public owner; // this will be the LoveEconomy who is
            the owner of all the contracts
28      bytes32 private discountCodeHashed; // this will be the
            discount code at the time of contract deployment
29      uint8 public userFee; // the fee will represent Wei, where the
            actual value received needs to be userFee * 10**18
30      uint8 public businessFee; // the fee that is required to be
            paid by the business to access the platform
31
32      uint24 public userCount; // number of users on the system
33      uint24 public businessCount; // number of local businesses
34
35      uint24 public discountCodeCount; // number of discount codes
            used
36
37      mapping(address => local_business) public
            businessAddresstoDetails; // id based lookup for businesses
38      mapping(address => user) public userAddresstoDetails; // id
            based look up for users
39
40      constructor(string memory _discountCode, uint8 _userFee, uint8
            _businessFee) public {
41          owner = msg.sender;
42          discountCodeHashed = keccak256(abi.encodePacked(
                _discountCode));
43          userFee = _userFee;
44          businessFee = _businessFee;
45      }
46
47      // events
48      event businessAdded (
49          string _businessName,
50          address _businessAddress,
51          address _businessContractAddress,
52          bool _active
53      );
54
55
56      // function to add a new local business to the LoveEconomy
57      // make the function private, as we only want this contract to
            be able to add new businesses
58
59      function addBusiness(address _businessAddress, string memory
            _businessName) public {
```

```
60          require(msg.sender == owner, "only the LoveEconomy can add
                new businesses"); // can only register once viable deal
                is aggreed upon
61          require(businessAddresstoDetails[_businessAddress].active
                != true, "business already exists");
62
63          // Deploy new business contract
64          LocalBusiness newBusinessContract = new LocalBusiness(
                _businessAddress, _businessName);
65          // Store the new contract address
66          address businessContractAddress = address(
                newBusinessContract);
67          AllBusinesses.push(_businessAddress);
68
69          businessCount++;
70          // update the business details and add to the array
71          businessAddresstoDetails[_businessAddress] = local_business
                (_businessName, businessContractAddress, true);
72          emit businessAdded(_businessName, _businessAddress,
                businessContractAddress, true);
73      }
74
75      // new users signing up to platform
76      // discount code is a bool: whether one was used or not
77      function addDiscountCodeUser(address _userAddress, string
            memory _userName, bool _discountCodeUsed, bytes32
            _discountCode) public {
78          require(userAddresstoDetails[_userAddress].active != true,
                "This customer is already registered");
79          require(_discountCode == discountCodeHashed, "Not the
                correct Code");
80          // check that the correct dicount code was added
81
82          discountCodeCount++;
83          userCount ++;
84          userAddresstoDetails[_userAddress] = user(_userName,
                _discountCodeUsed, true);
85          AllUsers.push(_userAddress);
86      }
87
88  // new user sign up when they do not have a discount code and
        rather want to pay, need the function to be payable
89  // unsure as could anyone just use someone elses address and then
        ether will be subtracted from that address?
90      function addUser(address _userAddress, string memory _userName)
            public payable {
```

```
91          require(userAddresstoDetails[_userAddress].active != true,
                "This customer is already registered");
92          require(msg.value >= userFee * 10**18, "Not enough ether");
                // check that the correct dicount code was added
93
94          userCount ++;
95          userAddresstoDetails[_userAddress] = user(_userName, false,
                true);
96          AllUsers.push(_userAddress);
97      }
98
99      // Allow LoveEconomy to activate or deactivate businesses on
            the platform
100         // function setActiveFlag(address _businessWalletAddress,
                bool _active) public {
101         //     require(msg.sender == owner, "Only LoveEconomy can
                change the active flag of a business.");
102         //     businessAddresstoDetails[_businessWalletAddress].
                active = _active;
103         // }
104
105     // function to update the signup value, this can only be called
            by the LoveEconomy contract owner
106     function updateUserFee(uint8 _newFee) public {
107         require(msg.sender == owner, "only the LoveEconomy can
                change the fee charged to new users");
108         userFee = _newFee;
109     }
110
111     // function to update the signup value for businesses,
112     // this can only be called by the LoveEconomy contract owner
113     function updateBusinessFee(uint8 _newFee) public {
114         require(msg.sender == owner, "only the LoveEconomy can
                change the fee charged to new businesses");
115         businessFee = _newFee;
116     }
117
118     // function to call informations and run tests
119     function getAllBusinesses() public view returns(address[]
            memory){
120         return AllBusinesses;
121     }
122
123     function getAllUsers() public view returns(address[] memory){
124         return AllUsers;
125     }
126
```

```
127     function getBusinessDetails(address _businessWalletAddress)
            public view
128         returns (string memory businessName, address
                businessContractAddress, bool active) {
129 
130             return(
131             businessAddresstoDetails[_businessWalletAddress].
                    businessName,
132             businessAddresstoDetails[_businessWalletAddress].
                    businessContractAddress,
133             businessAddresstoDetails[_businessWalletAddress].active
134             );
135 
136     }
137 
138     function isActive(address _businessWalletAddress) public view
            returns(bool active) {
139         return(businessAddresstoDetails[_businessWalletAddress].
                active);
140     }
141 
142     // function to recieve ether from the business contracts
143     // fallback function
144     function() external payable {
145 
146     }
147 }
```

B.1.2 LocalBusiness smart contract

```
1  pragma solidity >=0.4.21 <0.6.2;
2
3  import "./LoveEconomy.sol";
4  import "./DealsToken.sol";
5
6  contract LocalBusiness {
7
8  address public owner;
9  address payable public LoveEconomyAddress;
10 bool public paid = false;
11
12 LoveEconomy loveEconomy;
13
14 string public businessName;
15
16 event DealCreated(
17     string businessName,
18     address businessAddress, // which business contract the deal
           belongs to
19     address tokenContractAddress,
20     string dealName,
21     string dealDetails,
22     uint price,
23     uint expiryDate,
24     bool active
25   );
26
27 // supply will not be included as once the business puts up a deal,
       anyone on the platform should be able to use it, hence no limit
       should be set
28 struct Deal {
29     string businessName;
30     address businessContractAddress;
31     address tokenContractAddress; // could be used as the id
32     string dealName; // can be two for one or % discount
33     string dealDetails;
34     uint price;
35     uint expiryDate;
36     bool active;
37 }
38
39 // array holding all the deals
40
41 address[] public tokenContracts;
```

```
42 mapping (address => Deal) public tokenAddressToDeal;
43
44 // set owner of newly created business contract to the business
       address that created the contract
45 constructor(address _businessAddress, string memory _businessName)
       public {
46     LoveEconomyAddress = msg.sender; // address of the LoveEconomy
           contract
47     owner = _businessAddress;
48     businessName = _businessName;
49     loveEconomy = LoveEconomy(LoveEconomyAddress);
50 }
51
52 function activateContract() public payable {
53     require(msg.sender == owner, "Only the business contract can
           all this function");
54     require(msg.value >= loveEconomy.businessFee(), "the value must
           be bigger or equal to the business fee");
55     require(paid == false, "business has already paid the fee");
56
57     paid = true;
58     address(LoveEconomyAddress).transfer(msg.value);
59 }
60
61
62 function addDeal(string memory _dealName, string memory
       _dealDetails, uint _expiryDate, uint8 _price) public payable
       returns(address) {
63     require(msg.sender == owner, "Only the business contract owner
           can create new deals");
64     require(loveEconomy.isActive(msg.sender) == true, "the business
           must be active currently");
65     // the business that is creating this deal must be active
66     //require(_price > 0, "price needs to be higher than zero"); //
           price needs to be more than 0
67     require(paid == true, "the business can only add a new deal
           once the businessfee has been paid");
68
69      // Deploy new token contract
70     DealsToken newTokenContract = new DealsToken(_dealName, _price)
           ;
71     // Store the new contract address
72     address tokenContractAddress = address(newTokenContract);
73
74     tokenContracts.push(tokenContractAddress);
75
76     tokenAddressToDeal[tokenContractAddress] = Deal(businessName,
```

```
                                                    address(this),
                                                    tokenContractAddress
                                                        ,
                                                    _dealName,
                                                    _dealDetails,
                                                    _price,
                                                    _expiryDate,
                                                    true);

    emit DealCreated(businessName, msg.sender, tokenContractAddress
        , _dealName, _dealDetails,
                        _price, _expiryDate, true);
    return(tokenContractAddress);
}

// function to return the number of deals created by the business
function getDealCount() public view returns(uint){
        return tokenContracts.length;
}

// set a deal to no longer be active
function setDealActiveFlag(address _tokenAddress, bool _active)
    public view {
        tokenAddressToDeal[_tokenAddress].active == _active;
}

function getBusinessAddress() public view returns(address) {
    return owner;
}

// return the array of deals
function getAllDeals() public view returns(address[] memory){
    return tokenContracts;
}

function getDealDetails(address _tokenAddress) public view
        returns(string memory, address, string memory, string
            memory, uint,
                uint) {

            return(
            tokenAddressToDeal[_tokenAddress].businessName,
            tokenAddressToDeal[_tokenAddress].tokenContractAddress,
            tokenAddressToDeal[_tokenAddress].dealName,
            tokenAddressToDeal[_tokenAddress].dealDetails,
            tokenAddressToDeal[_tokenAddress].price,
            tokenAddressToDeal[_tokenAddress].expiryDate
```

```
120                 );
121 
122             }
123 
124 
125 }
```

B.1.3 DealsToken smart contract

```
pragma solidity >=0.4.21 <0.6.2;

import "../node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "./LocalBusiness.sol";

contract DealsToken is ERC20 {

    address public businessContractAddress;

    address public tokenContractAddress;
    string public dealName;
    uint8 public dealPrice;

    uint24 public dealsBoughtCount;

    LocalBusiness businessContract;

     constructor(string memory _dealName, uint8 _price) public {
         businessContractAddress = msg.sender;
         // since the business contract will deploy the token contract once a deal is created, the msg.sender will be the business contract
         tokenContractAddress = address(this);
         businessContract = LocalBusiness(businessContractAddress);
         dealName = _dealName;
         dealPrice = _price;
     }

     // function to create a new and unique non-fungible token each time a user buys a deal on the platform
     // want to be able to map the token id to specific deal
     function purchaseDeal() public payable {
         require(msg.value >= dealPrice, "price paid should be atleast the price of the deal");
         //require(businessContract.userActiveStatus(msg.sender) == true, "User can only purchase deals if active");

         address _to = msg.sender; // the person calling this function will be the address who should receive the token
         dealsBoughtCount++;
         _mint(_to, 1);
     }
```

```
37
38      // need to be able to approve and transfer token id's (if a
            user wants to sell their deal to another user)
39      // not sure how to make the recipient pay?
40      function sellToken(uint _amount, address _recipient) public
            payable {
41          require(msg.sender != address(0) && msg.sender !=
                tokenContractAddress, "sender cannot be the zero address
                 or the contract itself");
42          //require(businessContract.userActiveStatus(_recipient) ==
                true, "User can only recieve deals if active");
43
44          address _sender = msg.sender;
45          _transfer(_sender, _recipient, _amount);
46      }
47
48      // function to burn the token if redeemed at the business (
            possibly done by scanning a QR code?)
49      function useDeal(address _account) public {
50          require(msg.sender == businessContract.getBusinessAddress()
                ,
51              "only the business can burn the token when it accepts
                    the token as payment");
52
53          // burned amount will be 1 token, each time a token owner
                uses it
54          _burn(_account, 1);
55      }
56
57      // functions to extract information
58      function _dealPrice() public view returns(uint) {
59          return dealPrice;
60      }
61
62      function _businessContractAddress() public view returns(address
            ) {
63          return (businessContractAddress);
64      }
65
66        // returns the account balance of another account with
              address _sender
67      function _balanceOf(address _owner) external view returns (
            uint256){
68          return balanceOf(_owner);
69      }
70
71      // return the number of deals bought count
```

```
72      function dealsBought() public view returns(uint) {
73          return dealsBoughtCount;
74      }
75
76  }
```

B.2 Javascript code

```
const web3_utils = require('web3-utils');
const Web3 = require("web3");

App = {
  web3Provider: null,
  contracts: {},
  account: '0x0',

  // existingLoveEconomyAddress: '0x0',
  // existingLoveEconomyAddress: '0
     x257D7d25CC9D9C753971D67bb13254f95dee280e',


  init: function() {
    return App.initWeb3();
  },

  // initial connection of client side application to local
     blockchain
//   initWeb3: function() {
//       App.web3Provider = new Web3(Web3.givenProvider || "http://
    localhost:8545");
//       App.web3Provider = web3

//     return App.initContract();
//   },

initWeb3: function() {
     if (typeof web3 !== 'undefined') {
       // If a web3 instance is already provided by Meta Mask.
       App.web3Provider = web3.currentProvider;
       web3 = new Web3(web3.currentProvider);
     } else {
       // Specify default instance if no web3 instance provided
       App.web3Provider = new Web3.providers.HttpProvider('http://
          localhost:8545');
       web3 = new Web3(App.web3Provider);
     }
     // // Specify default instance if no web3 instance provided
     // App.web3Provider = new Web3.providers.HttpProvider('http://
        localhost:7545');
     // web3 = new Web3(App.web3Provider);
     return App.initContract();
```

```
39    },
40
41   initContract: function () {
42     $.getJSON('contracts/LoveEconomy.json', function (LoveEconomy)
          {
43
44           // get the contract artifact file and use it to
                instantiate a truffle contract abstraction
45           App.contracts.LoveEconomy = TruffleContract(LoveEconomy);
46
47           // set the provider for our contract
48           App.contracts.LoveEconomy.setProvider(App.web3Provider);
49
50           //update account info
51           //App.displayAccountInfo();
52
53           return App.render();
54     });
55
56     $.getJSON('contracts/LocalBusiness.json', function (
          LocalBusiness) {
57           App.contracts.LocalBusiness = TruffleContract(
                LocalBusiness);
58           App.contracts.LocalBusiness.setProvider(App.web3Provider);
59     });
60
61     $.getJSON('contracts/DealsToken.json', function (DealsToken) {
62      App.contracts.DealsToken = TruffleContract(DealsToken);
63      App.contracts.DealsToken.setProvider(App.web3Provider);
64 });
65
66
67 },
68
69 ////////////////////
70 //index site code//
71 ////////////////////
72
73 // display some contact information
74 displayAccountInfo: function () {
75      web3.eth.getAccounts(async function(err, account) {
76        // if there is no error
77        if (err === null) {
78          //set the App object's account variable
79          let accounts = await web3.eth.accounts;
80          console.log("accounts");
81          App.account = accounts[0];
```

```
82          // insert the account address in the p-tag with id='
                account'
83          $("#account").html(App.account);
84          // retrieve the balance corresponding to that account
85          web3.eth.getBalance(App.account, function(err, balance) {
86            // if there is no error
87            if (err === null) {
88              // insert the balance in the p-tag with id='
                    accountBalance'
89              $("#accountBalance").html(web3.utils.fromWei(balance,
                    "ether") + " ETH");
90            }
91          });
92        }
93      });
94  },
95
96
97  render: function() {
98
99      // show total number of businesses
100
101     App.contracts.LoveEconomy.deployed().then(function(instance) {
102           return instance.businessCount();
103     }).then(function(result){
104           console.log(result);
105           $("#numberOfBusinesses").html(" " + result);
106     }).catch(function(err) {
107           console.error(err);
108     });
109
110     // show the number of new users that register
111     App.contracts.LoveEconomy.deployed().then(function(instance) {
112           return instance.userCount();
113     }).then(function(result){
114           console.log(result);
115           $("#numberOfUsers").html(" " + result);
116     }).catch(function(err) {
117           console.error(err);
118     });
119
120     // show number of times that the discount code was used
121     App.contracts.LoveEconomy.deployed().then(function(instance) {
122           return instance.discountCodeCount();
123     }).then(function(result){
124           console.log(result);
125           $("#discountCodesUsed").html(" " + result);
```

```
126       }).catch(function(err) {
127            console.error(err);
128       });
129
130       App.contracts.LoveEconomy.deployed().then(function(instance) {
131            return instance.getBusinessDetails(App.account).then(
                  function(result){
132                 console.log(result[1]);
133                 $("#currentBusinessContracts").html(" " + result[1])
                        ;
134            }).catch(function(err){
135                 $("#currentBusinessContracts").html("none");
136            })
137       });
138
139       web3.eth.getAccounts(async function(err, account) {
140         // if there is no error
141         if (err === null) {
142           //set the App object's account variable
143           console.log("GOT ACCOUNT");
144           console.log(account);
145           console.log(web3);
146           App.account = account[0];
147           // insert the account address in the p-tag with id='
                  account'
148           $("#account").html(App.account);
149           console.log("SET");
150           console.log(App.account);
151           // retrieve the balance corresponding to that App.account
152           web3.eth.getBalance(App.account, function(err, balance) {
153             // if there is no error
154             if (err === null) {
155               // insert the balance in the p-tag with id='
                      accountBalance'
156               $("#accountBalance").html(web3.utils.fromWei(balance,
                      "ether") + " ETH");
157             }
158           });
159         }
160         console.log(err);
161       });
162
163 },
164
165 // function to display the ether to be paid to sign upt for the
      LoveEconomy for businesses
166 businessFee: function() {
```

```
167       // show total number of businesses
168       App.contracts.LoveEconomy.deployed().then(function(instance) {
169           return instance.businessFee();
170       }).then(function(result){
171           console.log(result);
172           $("#businessFeeValue").html(result + " ether");
173       }).catch(function(err) {
174           console.error(err);
175       });
176   },
177
178 //adding a new business: only LoveEconomy address can add
      businesses
179
180 addBusiness: function () {
181   console.log('Add Business button clicked');
182   // get information from the modal
183   var _businessName = $('#BusinessName').val();
184   var _businessAddress = $('#BusinessAddress').val();
185
186       // if the name is not provided or invalid address was provided
187       if ((_businessName.trim() == '') || (web3.utils.isAddress(
          _businessAddress) != true)) {
188           // we cannot add a business
189           console.log('Cannot load business because name or address
              is invalid')
190           return false;
191       };
192
193   console.log('Adding business (' + _businessName + ') - Please
      check Metamask');
194   App.contracts.LoveEconomy.deployed().then(function(instance) {
195           // call the addBusiness function,
196           // passing the business name and the business wallet
              address
197         return  instance.addBusiness(_businessAddress,
             _businessName, {
198             from: App.account,
199             gas: 5000000
200       });
201
202
203   }).then(function (receipt) {
204       console.log(receipt.logs[0].args._businessName + ' added');
205       businessContractAddress = receipt.logs[0].args.
          _businessContractAddress;
```

```
206         console.log('Business contract address: ' +
                businessContractAddress);
207
208       // alert("You have successfully signed up on the LoveEconomy
              ")
209   }).then(function () {
210           alert("a new business has been added");
211   }).catch(function (error) {
212         console.log(error);
213   });
214
215 },
216
217
218 // add new user that uses the dicount code
219 addDiscountCodeUser: function () {
220       console.log('Add User button clicked');
221       // get information from the modal
222       var _userName = $('#discountCodeUserName').val();
223       var _userAddress = $('#discountCodeUserAddress').val();
224       var _discountBool;
225       var _discountCodeUsed;
226
227       var checkbox = $("#chkDiscountCode");
228
229       if(checkbox.is(':checked')) {
230             _discountBool = "true";
231             _discountCodeUsed = $('#txtDiscountCode').val();
232       }
233
234
235         // if the name is not provided or invalid address was
                provided
236         if ((_userName.trim() == '') || (web3.utils.isAddress(
                _userAddress) != true)) {
237             // we cannot add a business
238             console.log('Cannot load user because name or address
                    is invalid')
239             return false;
240         };
241
242
243       console.log('Adding user (' + _userName + ') - Please check
              Metamask');
244       console.log('your discount bool is ' + _discountBool);
245       console.log("Your discount code used is " + String(
              _discountCodeUsed));
```

```
246
247       // console.log("the first address is " + web3.utils.isAddress(
              _userAddress));
248       console.log(web3_utils.solioditySha3("'"+_discountCodeUsed+"'")
              );
249
250       App.contracts.LoveEconomy.deployed().then(function(instance) {
251           // call the addBusiness function,
252           // passing the business name and the business wallet
                  address
253           return instance.addDiscountCodeUser(_userAddress,
                  _userName, _discountBool,  web3_utils.solioditySha3(
                  _discountCodeUsed), {
254             from: App.account,
255             gas: 5000000
256           });
257       }).then(function () {
258           alert('You have successfully signed up on the LoveEconomy
                  ');
259       }).catch(function (error) {
260           console.log(error);
261       });
262
263     },
264
265
266 // function to display the ether to be paid to sign upt for the
      LoveEconomy
267     userFee: function() {
268       // show total number of businesses
269       App.contracts.LoveEconomy.deployed().then(function(instance) {
270           return instance.userFee();
271       }).then(function(result){
272           console.log(result);
273           $("#userFeeValue").html(result + " ether");
274       }).catch(function(err) {
275           console.error(err);
276       });
277       },
278
279 // function to add user that isn't using a discount code and has to
       pay the user fee value
280
281     addUser: function () {
282       console.log('Add User button clicked');
283       // get information from the modal
284       var _userName = $('#UserName').val();
```

```
285       var _userAddress = $('#UserAddress').val();
286       // var _userFee = $("#userFeeValue").val();
287       var addUserinstance;
288
289      // console.log('User fee is (' + _userFee + ') - Please check
             Metamask');
290
291          // if the name is not provided or invalid address was
                 provided
292          if ((_userName.trim() == '') || (web3.utils.isAddress(
                 _userAddress) != true)) {
293              // we cannot add a business
294              console.log('Cannot load user because name or address
                     is invalid')
295              return false;
296          };
297
298       console.log('Adding user (' + _userName + ') - Please check
             Metamask');
299
300       App.contracts.LoveEconomy.deployed().then(function(instance) {
301            addUserinstance = instance;
302            // call the addBusiness function,
303            // passing the business name and the business wallet
                   address
304            return addUserinstance.userFee().then(function(fee){
305            console.log('User fee is (' + fee + ') - Please check
                   Metamask')
306                addUserinstance.addUser(_userAddress, _userName, {
307                     from: _userAddress,
308                     gas: 5000000,
309                     value: fee * 10**18,
310                });
311            })
312       }).then(function () {
313            // console.log(' new user added');
314            alert("You have successfully signed up on the LoveEconomy
                   ");
315       }).catch(function (error) {
316            console.log(error);
317       });
318
319    },
320
321    // display all businesses if LoveEconomy is logged in
322 displayBusinessDetails: function() {
323
```

```
324             var LoveEconomyInstance;
325             // get current metamask logged in
326             // refresh account info
327             App.displayAccountInfo();
328             var current_address = App.account;
329             console.log("current address " + current_address)
330 
331             App.contracts.LoveEconomy.deployed().then(function(
                    instance) {
332                 LoveEconomyInstance = instance;
333                 return LoveEconomyInstance.owner();
334             }).then(function(owner){
335                 // if(owner == current_address) {
336                     console.log("will display details");
337                     LoveEconomyInstance.getAllBusinesses().then(
                            function(businessAddresses) {
338                         console.log("there are " +
                                businessAddresses.length + " businesses
                                ");
339                         businessAddresses.forEach(
                                businessWalletAddress => {
340 
341                             LoveEconomyInstance.
                                    getBusinessDetails(
                                    businessWalletAddress, {from:
                                    current_address}).then(function(
                                    businessDetails){
342                                 App.showBusiness(
343                                     businessDetails[0],
344                                     businessWalletAddress,
345                                     businessDetails[1],
346                                     businessDetails[2]
347                                 );
348                             })
349 
350                         });
351                     });
352 
353 
354                 // } else {
355                 //      console.log("Only Master Account can display
                        businesses not " + App.Account);
356                 // }
357             })
358 
359     },
360 
```

```
SearchBusinesses: function() {

    var LoveEconomyInstance;
    var businessSearch = $("#ViewBusinessAddress").val();

  // get current metamask logged in
      // refresh account info
      App.displayAccountInfo();

      App.contracts.LoveEconomy.deployed().then(function(instance)
          {
           LoveEconomyInstance = instance;
           return LoveEconomyInstance.owner();
      }).then(function(owner){
         //  if(owner == App.account) {
               console.log("will display details");
               console.log("details of " + businessSearch + "
                  will be shown")
               LoveEconomyInstance.getBusinessDetails(
                  businessSearch, {from: App.account}).then(
                  function(businessDetails){
                    App.showBusiness(
                         businessDetails[0],
                         businessSearch,
                         businessDetails[1],
                         businessDetails[2]
                    );
               });
         //      } else {
         //        console.log("Only Master Account can display
            businesses not " + App.Account);
         //   }
      })

},

showBusiness: function (name, address, contractAddress, active) {
    var businessName = name;
    var businessWallet = address;
    var businessContract = contractAddress;
    var businessActive = active;
    var businessDetails = $("#businessDetails");

    console.log(" the business name is " + name + " wallet address
        is " + address + " contract address " + contractAddress
                + " active:" + active)
    // Render candidate Result
```

```
402       var businessTemplate = "<tr><th>" + businessName + "</th><td>"
              + businessWallet + "</td><td>" + businessContract + "</td
          ><td>"
403                                 + businessActive + "</td></tr>"
404       businessDetails.append(businessTemplate);
405   },
406
407   ////////////////////////
408   //local_deals site code//
409   ////////////////////////
410
411   businessPaidStatus: function () {
412
413       $('#deals_loadingGIF').modal({ show: true });
414
415    console.log("checking if business paid status is true or false")
416       var businessInstance;
417       var businessDetails = $("#displayBusinessStatus");
418       var businessContractSearch = $("#businessActivated").val();
419
420           // get current metamask logged in
421         // refresh account info
422         App.displayAccountInfo();
423
424         App.contracts.LocalBusiness.at(businessContractSearch).then(
              function(instance) {
425                 businessInstance = instance;
426
427                 console.log("will display details");
428
429                 businessInstance.paid().then(function(paid){
430                     if(paid == true){
431                      businessDetails.html("the business has paid its
                             fee")
432                     } ;
433                 });
434               //  } else {
435               //    console.log("Only Master Account can display
                     businesses not " + App.Account);
436              //   }
437
438         })
439         $('#deals_loadingGIF').modal("hide");
440
441   },
442
443   payFee: function() {
```

```
444     console.log("paying business fee");
445
446     $('#deals_loadingGIF').modal({ show: true });
447
448     var businessInstance;
449     var businessFee;
450     var businessDetails = $("#displayBusinessStatus");
451     var businessContract = $("#businessActivated").val();
452
453     // get current metamask logged in
454       // refresh account info
455      // App.displayAccountInfo();
456
457       App.contracts.LoveEconomy.deployed().then(function(instance)
              {
458           return instance.businessFee({from: App.account});
459                }).then(function(fee) {
460                     console.log("the fee is " + fee);
461                     businessFee = fee; }).then(function() {
462
463           App.contracts.LocalBusiness.at(businessContract).then(
                 function(instance) {
464             businessInstance = instance;
465
466             return businessInstance.paid({from: App.account});
467
468             }).then(function(paid){
469                  console.log("this business variable is set to:
                        " + paid)
470
471                  if(paid == false) {
472
473                            businessInstance.activateContract({
474                                 from: App.account,
475                                 gas: 5000000,
476                                 value: businessFee * 10**18}).
                                    then(function(){
477                                 console.log("you have
                                    successfully paid 2 ether to
                                    activate your contract");
478                                 businessDetails.html("You have
                                    successfully paid, please add
                                     your first deal");
479                       });
480                       $('#deals_loadingGIF').modal("hide");
481
482
```

```
483                     } else {
484                         console.log("you have already paid the fee
                                ");
485                         businessDetails.html("this business has
                                paid");
486                         $('#deals_loadingGIF').modal("hide");
487                     }
488                 })
489
490             });
491
492
493
494 },
495
496 //0x8be240d6dc65183554440fb4e321e0631cbb0ea2
497
498 // adding new deals
499
500 addDeal: function () {
501     console.log('Add Deal button clicked');
502     // get information from the modal
503     var _dealName = $('#dealName').val();
504     var _dealDetails = $('#dealDetails').val();
505     var _expirationMonth = $('#expirationMonth').val();
506     var _price = $('#price').val();
507     var _expirationDate;
508
509     var _currentDate = new Date();
510
511     _date = new Date(_currentDate.setMonth(_currentDate.getMonth()
            + _expirationMonth)); // NOT WORKING
512     var day = _date.getDate();
513     var month = _date.getMonth();
514     var year = _date.getFullYear();
515
516     _expirationDate2 = day + "/" + month + "/" + year;
517
518     _expirationDate = _expirationMonth; // doing this for testing
            for now
519
520     var _businessContractAddress;
521
522         // if the name is not provided or invalid address was
                provided
523         if ((_dealName.trim() == '')) {
524             // we cannot add a business
```

```
525             console.log('Cannot load business because name or
                    address is invalid')
526             return false;
527         };
528
529     console.log('Adding deal (' + _dealName + ') - Please check
            Metamask');
530     console.log('Adding deal with (' + App.account + ') account');
531     console.log("the expiration date is : " + _expirationDate2);
532
533     App.contracts.LoveEconomy.deployed().then(function(instance) {
534         return instance.getBusinessDetails(App.account, {from:
                App.account});
535     }).then(function(businessDetails) {
536
537         if (businessDetails[2] != true) {
538             console.log('This is not an active business.
                    Customer cannot be loaded')
539         } else
540             _businessContractAddress = businessDetails[1];
541         console.log("the business address is :" +
                _businessContractAddress);
542         // now get instance of the business contract
543         return App.contracts.LocalBusiness.at(
                _businessContractAddress);
544     }).then(function(businessInstance){
545         console.log('Adding deal ' + _dealName );
546          businessInstance.addDeal(_dealName, _dealDetails,
                _expirationDate, _price, {
547                 from: App.account,
548                 gas: 5000000});
549      })
550     // .then(function(result){
551     //      console.log(result.logs[0].args.businessName + ' added
            ');
552     //      console.log(result.logs[0].args.businessAddress + '
            added');
553     //      console.log(result.logs[0].args.tokenContractAddress +
             ' added');
554     //      console.log(result.logs[0].args.dealName + ' added');
555     //      console.log(result.logs[0].args.dealDetails + ' added
            ');
556     //      console.log(result.logs[0].args.price + ' added');
557     //      console.log(result.logs[0].args.expiryDate + ' added')
            ;
558     //      console.log(result.logs[0].args.active + ' added');
```

```
559       //      dealContractAddress = result.logs[0].args.
              tokenContractAddress;
560       //      console.log('Deal contract address: ' +
              dealContractAddress);
561
562       //      return dealContractAddress;
563       //      // log the error if there is one
564
565       // })
566       .then(function(){
567            alert("A new deal has been added");
568       }).catch(function (error) {
569            console.log(error);
570       });
571
572    },
573
574
575 // displaying product details when page loads
576
577 // displaying the number of tokens already sold for a deal
578 TokenDisplay: function() {
579
580 // $("#loadingGIF").show();
581 $('#deals_loadingGIF').modal({ show: true });
582
583 var LoveEconomyInstance;
584 var _businessContractAddress;
585
586 var _businessName;
587
588       // get current metamask logged in
589       // refresh account info
590 App.displayAccountInfo();
591
592 App.contracts.LoveEconomy.deployed().then(function(instance) {
593       LoveEconomyInstance = instance;
594
595       LoveEconomyInstance.getAllBusinesses({from: App.account}).then
              (function(businessAddresses) {
596            console.log("there are " + businessAddresses.length + "
                  businesses");
597            businessAddresses.forEach(businessWalletAddress => {
598                 return LoveEconomyInstance.getBusinessDetails(
                        businessWalletAddress, {from: App.account}).then(
                        function(businessDetails){
599
```

```
600                     if (businessDetails[2] != true) {
601                     console.log('This is not an active business.
                            Customer cannot be loaded')
602                     } else {
603
604                     _businessName = businessDetails[0];
605                     _businessContractAddress = businessDetails[1];
606                     console.log("the business address is :" +
                            _businessContractAddress);
607                     // now get instance of the business contract
608                     App.contracts.LocalBusiness.at(
                            _businessContractAddress).then(function(
                            businessContractInstance){
609                       businessContractInstance.getAllDeals({from
                              : App.account}).then(function(
                              dealAddresses) {
610                         console.log("there are " +
                                dealAddresses.length + " deals");
611                         dealAddresses.forEach(
                                dealContractAddress => {
612                           return businessContractInstance.
                                  getDealDetails(
                                  dealContractAddress, {from:
                                  App.account}).then(function(
                                  dealDetails){
613
614                             App.showTokenDetails(
615                                   dealDetails[0],
616                                   dealDetails[1],
617
618                                   );
619                             })
620
621                         });
622                     })
623
624                     })
625                     }
626                 });
627
628
629         });
630
631         });
632
633     })
634
```

```
635 },
636
637
638 showTokenDetails: async function (_businessName, _tokenAddress) {
639       var businessName = _businessName;
640       var dealContractAddress = _tokenAddress
641
642       console.log("the token address is " + dealContractAddress);
643
644       let dealsSold = await App.contracts.DealsToken.at(
              dealContractAddress).dealsBought({from: App.account});
645
646       let dealName = await App.contracts.DealsToken.at(
              dealContractAddress).dealName.call();
647
648       var tokenDetails = $("#tokenDetails");
649
650
651       // add date combination here
652
653       console.log("the business name is " + businessName + "the deal
              name is " + dealName +
654                           "deals sold " + dealsSold)
655       // Render candidate Result
656       var tokenTemplate = "<tr><th>" + businessName + "</th><td>" +
              dealName + "</td><td>" + dealsSold + "</td></tr>"
657                                 // the above line sets the id of the
                                       button as the function input when the
                                       button is pressed.
658       tokenDetails.append(tokenTemplate);
659
660       $('#deals_loadingGIF').modal("hide");
661 },
662
663     //////////////////////
664    // User site code //
665   //////////////////////
666
667    // displaying all the deals and their details
668
669 displayActiveDeals: async function() {
670
671 if (App.loading) {
672       return;
673 };
674 App.loading = true;
675
```

```
676 $('#token_loadingGIF').modal({ show: true });
677
678 var LoveEconomyInstance;
679 var _businessContractAddress;
680 var exchangeRate = await App.ETHtoZAR();
681 console.log("the exchange rate is currently: " + exchangeRate);
682
683     // get current metamask logged in
684     // refresh account info
685     App.displayAccountInfo();
686
687     App.contracts.LoveEconomy.deployed().then(function(instance) {
688         LoveEconomyInstance = instance;
689
690         LoveEconomyInstance.getAllBusinesses().then(function(
              businessAddresses) {
691             console.log("there are " + businessAddresses.length
                  + " businesses");
692             businessAddresses.forEach(businessWalletAddress => {
693                 return LoveEconomyInstance.getBusinessDetails(
                      businessWalletAddress, {from: App.account}).
                      then(function(businessDetails){
694
695                     if (businessDetails[2] != true) {
696                     console.log('This is not an active
                          business. Customer cannot be loaded')
697                     } else {
698
699                     _businessContractAddress = businessDetails
                          [1];
700                     console.log("the business address is :" +
                          _businessContractAddress);
701                     // now get instance of the business
                          contract
702                     App.contracts.LocalBusiness.at(
                          _businessContractAddress).then(function
                          (businessContractInstance){
703                       businessContractInstance.getAllDeals
                            ({from: App.account}).then(
                            function(dealAddresses) {
704                           console.log("there are " +
                                dealAddresses.length + "
                                deals");
705                           dealAddresses.forEach(
                                dealContractAddress => {
706
```

```
                                    return businessContractInstance.
                                        getDealDetails(
                                        dealContractAddress, {from:
                                        App.account}).then(function(
                                        dealDetails){
                                          App.showDeal(
                                               dealDetails[0],
                                               dealDetails[2],
                                               dealDetails[3],
                                               dealDetails[4],
                                               dealDetails[5],
                                               dealDetails[1],
                                               exchangeRate
                                          );
                                    })

                                });
                    })

                    })
                    }

                });
          });

          });
     })
},


showDeal: function (ownerName, dealName, dealDetails, price,
    expiryDate, tokenAddress, exchangeRate) {
      var businessName = ownerName;
      var name = dealName;
      var details = dealDetails;
      var dealPrice = price;
      var date = expiryDate;
      var dealDetailsTable = $("#dealDetailsTable");
      var ZARprice = price * exchangeRate;
      var tokenContractAddress = tokenAddress;

      // add date combination here

      console.log("the business name is " + businessName + "the deal
          name is " + name +
                    "deal details are " + details + "the deal price
                        is " + dealPrice + "the expiry date is " +
```

```
                        date +
747                   "the token contract adddress is: " +
                        tokenContractAddress)
748     // Render candidate Result
749     var dealTemplate = "<tr><th>" + ownerName + "</th><td>" +
            dealName + "</td><td>" + dealDetails +
750                         "</td><td>" + price + "</td><td>" +
                            ZARprice +
751                         "</td><td>" +  expiryDate +
752                         "</td><td>" +
753                         "<button type='button' id = " +
                            tokenContractAddress + " onclick='App.
                            BuyThisItem(this.id)' >Buy</button>" +
                            "</td></tr>"
754                         // the above line sets the id of the
                            button as the function input when the
                            button is pressed.
755     dealDetailsTable.append(dealTemplate);
756
757     $('#token_loadingGIF').modal("hide");
758 },
759
760 // return ETH to rand exchange rate function
761 // get currency exchange
762
763 currencyConverter: async function () {
764
765     const ETHexchange = await App.ETHtoZAR();
766     console.log(ETHexchange);
767     document.getElementById("ZARrate").innerHTML = "1 ETH = " +
            ETHexchange + " Rand";
768
769 },
770
771 ETHtoZAR: async function() {
772
773     var getFullURL = function(url, options){
774          const params = [];
775          for (let key in options) {
776              params.push(`${key}=${options[key]}`);
777          }
778          return url + '?' + params.join("&");
779      }
780
781      var apiKey = "048784013
            e3c60bcdeadec8128a13f600a84fc9f9d0840007a49edcb4af6020d";
782      var baseUrl = "https://min-api.cryptocompare.com/data/price"
```

```
783
784         var options = {
785             api_key: apiKey,
786             fsym: "ETH",
787             tsyms: "ZAR"
788         };
789
790         var fullURL = getFullURL(baseUrl, options);
791
792
793         const response = await axios.get(fullURL);
794         const object = await response.data;
795
796         const ETH = await object.ZAR;
797
798         console.log(ETH);
799        return(ETH);
800
801 },
802
803
804 // error as there is a require function for the buyer to be a
       active user : check that this works
805 BuyThisItem: function (tokenContractAddress) {
806        var _tokenAddress = tokenContractAddress;
807        var buyDealInstance;
808
809        console.log("the token address is: " + tokenContractAddress)
810             // get current metamask logged in
811          // refresh account info
812          App.displayAccountInfo();
813
814        //    // if the name is not provided or invalid address was
           provided
815        //    if ((web3.utils.isAddress(App.account) != true)) {
816        //         // we cannot add a business
817        //         console.log('Cannot load user because name or
           address is invalid')
818        //         return false;
819        //    };
820
821        console.log('Purchasing deal (' + _tokenAddress + ') - Please
           check Metamask');
822
823        App.contracts.DealsToken.at(_tokenAddress).then(function(
           instance) {
824             buyDealInstance = instance;
```

```
825
826           return buyDealInstance._dealPrice({from: App.account}).
                 then(function(_dealprice) {
827               console.log("the deal price is " + _dealprice);
828               buyDealInstance.purchaseDeal({
829                    from: App.account, // this needs to be the
                        users account, as only active users can
                        pruchase tokens
830                    gas: 5000000,
831                    value: _dealprice * 10**18,
832               });
833           })
834
835      }).then(function() {
836           console.log('new deal purchased');
837           // alert("You have just purchased a new deal!");
838      }).catch(function(error) {
839           console.log(error);
840      });
841
842 },
843
844
845
846 };
847
848 // initialize the app everytime the window loads
849 $(function() {
850      $(window).load(function() {
851        App.init();
852      });
853    });
854
855 // show and hide checkbox in User sign-up
856    $(function() {
857
858      // Get the form fields and hidden div
859      var checkbox = $("#chkDiscountCode");
860      var hidden = $("#dvDiscountCode");
861
862      hidden.hide();
863
864      checkbox.change(function() {
865
866        if (checkbox.is(':checked')) {
867          // Show the hidden fields.
868          hidden.show();
```

```
869           console.log("check box was clicked")
870
871         } else {
872           // Make sure that the hidden fields are indeed
873           // hidden.
874           hidden.hide();
875
876         }
877       });
878     });
```

Appendix C

Using the LoveEconomy Prototype

Installation (Development Environment)

In order to run the LoveEconomy prototype, a development environment of the following is required (note that these instructions are given for the terminal environment on Mac OS):

1. **node.js:** Install node dependencies by running the following line in your terminal:

```
$npm install
```

2. **Truffle framework:** Truffle is one of the most popular development, testing and deployment framework for the Ethereum blockchain. To install Truffle globally, run the following line in your terminal:

```
$npm install -g truffle
```

3. **Ganache:** Ganache is a personal blockchain. Download Ganache from `http://truffleframework.com/ganache/` and install it.

4. **Web3.js:** web3.js allows you to interact with local or remote Ethereum nodes using https. It can be installed by running:

```
$npm install web3
```

5. **MetaMask:** The MetaMask extension needs to be installed in your browser in order to use the LoveEconomy prototype. One can install MetaMask by following the instructions given here: `https://metamask.io/`

Once all dependencies have been installed, the GitHub repo of the LoveEconomy needs to be cloned from: `https://github.com/julipahl/LoveEconomy_2`. Once the repo has been cloned, access its folder using your terminal and complete the

following steps (at this point Ganache needs to be running successfully on your desktop):

1. Confirm that the smart contract compile successfully

   ```
   $truffle compile
   ```

2. Run tests for the smart contracts to ensure that they are working as expected. There are 22 tests that need to pass successfully.

   ```
   $truffle test
   ```

3. Deploy smart contracts to Ganache.

   ```
   $truffle migrate --reset
   ```

   The reset variable ensures all scripts in the migrations folder will be run. If, for some reason, there are problems with deploying the contract, ensure that the truffle-config.js file is updated with the correct NetworkID, Host and Port values from your local Blockchain in Ganache.

Once the above steps are successful, run:

```
$npm run dev
```

and open the http://localhost:3000/ page in your browser.

Once the localhost browser is open, ensure that MetaMask is correctly set up for the private blockchain. This is done by going into Settings > Networks > Add Network. Then, add a private network name and set the "New RPC URL" to: http://127.0.0.1:7545 . One can add new accounts to this network if need be, to interact with the LoveEconomy. These can be imported from Ganache by clicking on "Import account" and entering the private key given in Ganache.