



Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

The University of Dublin

School of Computer Science and Statistics

Design and Development of a Sharing Economy Application

Alexandra Silva



April 30, 2020

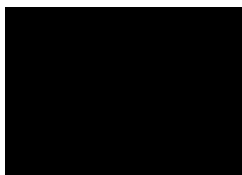
A Final Year Project submitted in partial fulfilment
of the requirements for the degree of
MAI (Computer Engineering)

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at <http://www.tcd.ie/calendar>.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at <http://tcd-ie.libguides.com/plagiarism/ready-steady-write>.



Signed:

Date: 29th April 2020

Abstract

The research conducted in this study focused on the design and development of an item and skill sharing application. This research investigates the problem of increasing consumer expenditure in Ireland despite the increase in usage of sharing economy applications. It is proposed that a novel item and skill sharing application be introduced to reduce consumer expenditure and increase sustainability in Ireland.

An extensive review of related literature and existing sharing economy applications found that the requirements of trust, privacy and pricing would have to be satisfied for the proposed application to be successful. Various methods were considered to satisfy these requirements. These methods were chosen based on a further review of related literature and existing sharing economy applications. They include; implementing the application as community-based and implementing a rating system to satisfy trust, implementing a user defined friendship model with custom granularity to satisfy privacy, and allowing users to decide upon their own fees and encouraging an atmosphere of bartering on the platform to satisfy pricing. Android Studio and Firebase were chosen to facilitate the implementation of the proposed application.

A major insight of this research is the importance of implementing a sharing economy application as community-based if it is to be successful. It is found that users are more likely to engage with a sharing economy application if they view the other users of the platform as their *neighbours*.

A major contribution of this research is the user defined friendship model. This model allows users to customize the information seen by users of the platform who are their *friends* and users who are not. A major challenge faced during the development of the proposed application was ensuring maintenance of user privacy while still showing enough user information to encourage trust. The fact that the implemented user defined friendship model solves this problem is a significant accomplishment of this research.

Acknowledgements

This MAI research project has certainly been a significant undertaking that I could not have accomplished without the support and guidance of so many different people throughout this journey.

First and foremost my utmost thanks goes to Dr. Mélanie Bouroche, who has been an incredible support to me through every moment of this journey. I am incredibly thankful for the insights and guidance provided by her throughout this project.

Another major thanks goes to Professor Stephen Barrett, your guidance is truly appreciated and had a huge impact on this dissertation.

Lastly to my loving friends and family - Mom, Ben, Aoife, and Jannet thank you for your support and patience through the stressful times and for offering me well needed advice and opinions throughout this process.

ALEXANDRA SILVA

University of Dublin, Trinity College,

April 2019

Contents

1	Introduction	1
1.1	Background	1
1.1.1	Consumerism	2
1.1.2	The Sharing Economy	2
1.1.3	Drivers of the Sharing Economy	3
1.2	Challenges	4
1.3	Motivation	6
1.4	Reader's Guide	7
2	Related Work	8
2.1	Requirements	8
2.2	State of the Art	9
2.2.1	Application Model	9
2.2.2	Methods to Ensure User Privacy	11
2.3	State of Practice	13
2.3.1	Food-Sharing Applications	13
2.3.2	Skill-Sharing Applications	14
2.3.3	Item Sharing Applications	15
2.3.4	Home-Sharing Applications	16
2.3.5	Ride Sharing Applications	17
2.4	Summary	18
3	Design	19
3.1	Application Model	19
3.2	Requirements	20
3.2.1	Trust	20
3.2.2	Privacy	21
3.2.3	Pricing	22
3.3	Application Development Environment	22
3.4	Architecture	23
3.4.1	MVC	23
3.4.2	MVP	24
3.4.3	MVVM	25

4	Implementation	27
4.1	Architecture	27
4.2	Data Processes	30
4.2.1	User Authentication	31
4.2.2	Application Database	32
4.3	Trust Establishment	33
4.3.1	Location Based Recommender System	34
4.3.2	Rating System	35
4.4	Privacy Protection Methods	36
4.4.1	Data Structure	36
4.4.2	Information Granularity	37
4.5	User Interface Design	38
4.5.1	Login and Register Pages	39
4.5.2	Home Page	40
5	Results	42
5.1	Recommender System	42
5.2	Friendship Model	44
5.3	Rating System	46
5.4	Summary	46
6	Evaluation	48
6.1	Architecture	48
6.2	Application Development Environment	49
6.3	Firebase	50
6.3.1	User Authentication	50
6.3.2	Database	50
7	Future Work	52
7.1	Database	52
7.2	Graph Database Structure	53
7.3	Messaging Service	54
8	Conclusions	55
8.1	Community Based Application	55
8.2	User Defined Friendship Model	55

List of Figures

1.1	Consumer Expenditure in Ireland (1)	6
2.1	Sharing Application Models (2)	10
2.2	Mix Zone Model (3)	12
2.3	Existing Applications Requirements	18
3.1	MVC Architecture (4)	24
3.2	MVP Architecture (4)	25
3.3	MVVM Architecture (4)	26
4.1	MVVM Architecture Dependencies	28
4.2	Items Model Class	28
4.3	Items Layout Class	29
4.4	Users ViewModel Class	29
4.5	User Profile Page	30
4.6	Firebase Authentication Register Method	32
4.7	Firebase Authentication Login Method	32
4.8	Significant Database Fields	33
4.9	Part of Location Based Recommender System	34
4.10	Weighted Average Rating Algorithm	35
4.11	Rating Calculation Function	35
4.12	JSON Representation of Friendships Data Structure	37
4.13	JSON Representation of Updated Friendships Data Structure	38
4.14	Login and Register Mock-up Pages	40
4.15	Home Mock-up Page	41
5.1	Results Ranked by Location	43
5.2	Results Ranked by Rating	44
5.3	User Information Settings Page	45
5.4	Item Page	46
7.1	Graph Database Friend of a Friend Query (5)	54

1 Introduction

In recent years, people have become more and more invested in an emerging economic model known as the sharing economy. The increase in people being involved in the sharing economy is largely due to environmental factors and the changing priorities of modern consumers. Despite the growth of the sharing economy, consumer expenditure in Ireland has continuously increased over the last decade. The purpose of this research is to introduce a new item and skill-sharing application to the sharing economy in Ireland, that will, in theory, reduce consumer expenditure and increase sustainability in Ireland. The research described in this dissertation follows the design and development of this proposed application.

Chapter 1.1 examines the background of the sharing economy by detailing both the past and current attitudes towards consumerism while analysing the differences between the economic model of the past and the emerging sharing economic model.

Chapter 1.2 details the challenges that the sharing economy faces while analysing the reasons for these challenges.

Chapter 1.3 details the motivation behind this research.

1.1 Background

This section of the paper discusses the background of the sharing economy by examining the past and current attitudes towards consumerism and the differences between the economic model of the past and the emerging sharing economic model.

Section 1.1.1 discusses the attitudes of consumers in the past. This section will also examine how and why these attitudes have changed in recent times.

Section 1.1.2 discusses the background behind the sharing economy. A new economic model that has been trending in recent years.

Section 1.1.3 discusses why the sharing economy has grown so rapidly since its emergence.

1.1.1 Consumerism

This section discusses the attitudes of consumers in the past, how these attitudes have changed in recent times, and why. In previous decades consumerism was outlined clearly between the suppliers of products (selling) and the consumers of these products (buying). This well known economic model was based on the fact that the priorities of suppliers lied in making money by selling their products, and the priorities of consumers lied in ensuring they had ownership of all the materials they needed or wanted and were able to buy these quality goods at a fair price (6). After World War II, the priorities of consumers changed. Instead of just wanting to purchase materials they needed or wanted, they began constantly purchasing the newer and better version of materials they already owned. This created an incredibly unsustainable form of consumerism in terms of the environment (6). Consumerism, defined as the idea that increasing your consumption of goods and services is desirable and that it will increase both your well being and happiness led people to believe that, “The Good Life” they saw on advertisements and television at the time, was within their reach if they just kept obtaining more material possessions. This type of consumerism was at the time, heavily encouraged by world leaders due to the undeniable fact that consumerism fuels economic growth (6). Despite the clear economic advantage of consumerism, one huge downfall of it is the negative impact it has had and is still having on our environment, by depleting natural resources and polluting the Earth. This continuing level of consumption has led us to a world where by 2050 our ocean will have more plastic in it than fish (7). One study published by the Journal of Industrial Ecology found that everything people consume is responsible for 60 percent of global greenhouse emissions (8). It is clear that if we want to change the planet and ensure that it is not destroyed we must tackle consumerism and find a new economic model to live by.

1.1.2 The Sharing Economy

This section discusses the new economic model that has been trending in recent years known as the sharing economy. The sharing economy is an emerging economic model that can be defined as a peer to peer based activity of acquiring, providing and sharing access to both goods and services and is very often facilitated by a community based online platform (9). The sharing economy has created a new form of social and economical exchange. Instead of having the classic consumers buying products from suppliers who sell the products, the sharing economy has enabled every single individual to act as both a consumer and a supplier simultaneously. The sharing economy also distorts the classical relationship among factors of production. With traditional consumerism the factors of production such as land, labor and capitol lied in the hands of the privileged few. Now with the sharing economy, these same factors of production are in the hands of the masses who now have the freedom to use their properties and resources in any way they please (9). The emergence of

the sharing economy in these recent years has been driven by many reasons such as the creation and wide adaptation of the internet, the changing priorities of modern consumers, growing environmental awareness within consumers and finally the urgent need for more sustainable practices.

1.1.3 Drivers of the Sharing Economy

This section discusses the several drivers of the sharing economy. The first driver of the sharing economy that will be discussed is the internet. Before the Internet emerged consumers were forced to purchase their products through physical retail locations. This form of purchasing allows for multiple different layers of distributors before finally reaching consumers. This meant that the consumer surpluses are removed from the consumers by the owners of the factors of production, along these many layers such as the creators of the product, the distributors of the product and the suppliers of the product etc (9). The internet differs as it allows consumers to buy their products directly from the source or through a distribution network that is much less complex. With the surpluses that consumers save they can become the owners of their own factors of production. With the internet being a convenient tool for instant communication, capital owners can communicate with each other and exchange goods and services making them both suppliers and consumers within the economy. This started with the trading of basic goods through the likes of eBay, and has since advanced to facilitate trades of labor, automobiles and temporary ownership of property through the likes of GoCar and Airbnb.

The next driver of the sharing economy that will be discussed is the changing priorities of modern consumers. The consumers of today differ majorly from the consumers of older generations, in that they care more about lower costs, instant access and flexibility, rather than simply purchasing the newest version of products they already own. This makes the sharing economy an ideal economic model for our modern consumers as sharing the use of objects or services is much more economical than purchasing them. One major aspect of consumer psychology that has changed in recent times is their desire for experiences over possessions (9). This has been caused by our changing belief in happiness. Before consumers believed that owning more material goods would make them happier while now consumers have the belief that happiness comes more from experiences rather than possessions. This is definitely proven within psychological studies and literature and it appears that consumers have begun to realize this for themselves (9). Another major change within the priorities of consumers today is their desire for community. In the past we had a culture of individual consumerism which is now being challenged by our hard wired drive for community building and a sense of belonging (9). All of these changes in consumer psychology have led to the sharing economy becoming one of the fastest growing business trends in history with more than 23 billion US dollars being invested into different sharing

economy businesses since 2010. Despite it only emerging within the past decade, the sharing model is in no way a new concept, with it simply being a new way of describing bartering which many rural communities still thrive on today (10).

The final driver of the sharing economy that will be discussed is consumers' growing sense of environmental awareness. This sense of environmental awareness has had perhaps the biggest impact on the change between modern consumers today versus consumers of the past. A growing urgency of needing to reduce our emissions and waste has led consumers to look for new ways to get the products that they need. This growth in environmental awareness has led consumers to the use of companies working within the sharing economy because of how they boost environmental protection, by encouraging people to purchase less and borrow more, therefore leading to less waste. According to the Irish Retail and Consumer Report for 2019 from PwC, 41 percent of Irish consumers are willing to pay a premium in order to buy products that are more sustainable (11). This number shows how consumers are willing to now more than ever look for ways to be increasingly environmentally sustainable when it comes to the products they consume. A 2017 study on American consumers showed that 88 percent of consumers will be more loyal to a company that supports social and environmental issues (12). This can be proven not only by the growth of the sharing economy within recent years but also by the growth of companies such as Beyond Meat who offer consumers plant based meat products that are of course more environmentally sustainable than traditional meat products. This change in the attitudes of consumers along with the undeniable environmental threat that our planet is currently under, is expected to cause an increasing growth in the sharing economy in the near future.

The drivers of the sharing economy discussed have resulted in the sharing economy's revenue of 18.6 billion US dollars in 2017. This makes it one of the fastest growing business trends in history. The sharing economy is also predicted to continue growing with a prediction that its revenue will reach over 40 billion US dollars by the year 2022 (13).

1.2 Challenges

This section will examine the challenges that the sharing economy faces along with an analysis into why the sharing economy faces these challenges despite its clear environmental advantages.

Despite the obvious advantages of the sharing economy that have been outlined, and the clear growth in the sharing economy as seen from the provided figures, the sharing economy is certainly not without its challenges. One of the biggest challenges faced by the sharing economy is trust. This challenge of trust is seen in many different ways in the sharing economy. Firstly, users must trust the application itself and the people behind the

platform. Traditionally, providers build a reputation for themselves over time that consumers can trust. They do this using multiple different techniques such as branding, advertising and customer testimonials. Another way traditional businesses can build their reputation is through receiving quality marks and standards by unbiased parties, for example the Bord Bia quality mark on Irish food. Unfortunately, because businesses in the sharing economy don't sell products or services but rather facilitate the sharing of products and services among their customers, they don't have the funds available to invest in branding and advertising until they are fully established. This means that they cannot use those traditional methods of branding and advertising to build a reputation for themselves. Sharing economy businesses also cannot build their reputation through the use of quality standards as there are no third parties that can award quality marks to sharing economy platforms (14).

Secondly, users have to trust the other users on the platform that they will be sharing with. For a user to share their belongings or skills with another user they must trust them, but why would anyone trust a complete stranger. To overcome this challenge sharing economy platforms tend to require that users provide certain pieces of personal data about themselves when setting up their accounts. However, this invokes another challenge faced by the sharing economy; privacy. Other methods used by sharing economy applications to encourage trust between user include, the implementation of a rating system and the process of implementing the application as a community-based application (14).

Thirdly, the items or skills that are being shared through the platform must be trustworthy. While traditional businesses can easily manage the quality of their products and services as they themselves are manufacturing or providing those products and services, businesses in the sharing economy have much more of a challenge when it comes to ensuring the quality of all products and services being shared through their platform. To overcome this challenge some businesses within the sharing economy utilise quality checks and others may implement another rating or review system for the items or services themselves (14).

Finally, another challenge faced by the sharing economy is privacy. Users on sharing economy platforms tend to want as much information as possible about the others on the platform whom they will be sharing their belongings or skills with. This is because it's easier to trust in someone who isn't a complete stranger. In contrast, when users are joining a sharing economy platform they prefer to disclose as little information about themselves as possible in order to protect their privacy (14). This obviously creates a huge challenge for sharing economy businesses as they must somehow find a balance between providing enough information about users so they can trust in one another while still protecting their privacy.

1.3 Motivation

This section discusses the motivation behind this research. As discussed the importance of the sharing economy has recently increased due to many reasons, one being, our need for more sustainable practices in terms of consumerism. There are multiple different types of sharing within the sharing economy, such as item-sharing, skill-sharing, property-sharing, ride-sharing etc. Each of these types of sharing help our society become more sustainable by reducing waste and emissions. For example, ride-sharing and parking-sharing platforms both help in reducing vehicle emissions while item-sharing platforms help in reducing manufacturing emissions. Another major benefit of the sharing economy is the huge decrease in wasted resources because people who share are allowing their products to be reused. From the outlined points it is clear that the sharing economy greatly benefits the planet. Another benefit to the sharing economy is that it helps people reconnect to their communities.

Despite the numerous sharing economy applications that exist today, consumer expenditure has still continued to grow in recent years. In Ireland for example, the consumer expenditure has done nothing but increase over the last 10 years. This is despite the continuous increase of usage in sharing economy applications such as GoCar, Airbnb and Etsy. A visual representation of the continuous increase in consumer expenditure in Ireland over the past ten years is shown in figure 1.1.



Figure 1.1: Consumer Expenditure in Ireland (1)

It is proposed that this increase is due to the current lack of a successful item-sharing or skill-sharing platform in Ireland. While some do exist, they do not compare to item-sharing and skill-sharing applications created in other countries in Europe, Asia and America. This will be the problem that is addressed in the creation of the proposed sharing economy application. The proposed application will attempt to create an application for both item-sharing and skill-sharing in Ireland, that people will genuinely use. The hope is

that the creation of this application would eventually lead to a drop in consumer expenditure and a rise in sustainability in Ireland.

1.4 Reader's Guide

The remainder of this dissertation is structured as follows.

Chapter 2 is entitled Related Work. The chapter provides an in-depth review of the requirements the proposed application must satisfy to be successful. This review involves an analysis of relevant literature and existing sharing economy applications. This analysis will look at different possible approaches to satisfying each of the relevant requirements.

Chapter 3 is entitled Design. This chapter discusses the decisions made regarding the design of the proposed application. These decisions are based on the findings of the Related Work chapter and relate to the approaches chosen to satisfy each application requirement and the tools chosen to develop the proposed application.

Chapter 4 is entitled Implementation. This chapter describes the implementation of the proposed application in great detail. It focuses on the steps involved in developing the various components of the proposed application that satisfy the relevant requirements.

Chapter 5 is entitled Results. This chapter discusses and displays the results gathered from the implementation and testing of the proposed application.

Chapter 6 is entitled Evaluation. This chapter provides an in-depth evaluation of the significant methods and tools used to design and develop the proposed application.

Chapter 7 is entitled Future Work. This chapter details some potential opportunities for the continuation and enhancement of the work completed in this research.

Chapter 8 is entitled Conclusions. This chapter details some overall conclusions regarding the research discussed in this dissertation.

2 Related Work

This chapter discusses the requirements the proposed application must satisfy to be successful. This review involves an analysis of relevant literature and existing sharing economy applications.

Section 2.1 presents the requirements that must be satisfied for the proposed application to be successful and provides an analysis on each requirement.

Section 2.2 presents a review of relevant literature. The literature presents possible approaches in designing an application model and satisfying the requirement of privacy for the proposed application.

Section 2.3 presents a review of existing sharing economy applications. This review analyses the approaches taken by existing sharing economy applications to satisfy the requirements of the proposed application.

Section 2.4 provides a brief summary of the chapter.

2.1 Requirements

In this section the requirements that the proposed application must satisfy will be discussed. The choice of these requirements is based upon the analysis of the challenges that the sharing economy currently faces. These requirements include trust, privacy and pricing.

In order for a sharing application to be successful, the users must trust each other enough to share their belongings or skills with one another. As discussed in section 1.2 platforms within the sharing economy face numerous challenges associated with trust, making this requirement especially important but also very complex. The proposed application must find a way to ensure users they can trust the platform itself, trust the other users on the platform and trust the items and skills that are being shared through the platform. It will also have to find the perfect balance of sharing enough user information so that the users can trust and contact each other, while still protecting their privacy.

This brings us to the second requirement; privacy. For users of the proposed application to willingly divulge their personal information on the platform, they will have to be ensured that their privacy is being protected. Another consideration that will have to be taken into

account is ensuring that users are still sharing enough information, to encourage trust between users on the platform.

Finally, the last requirement that the proposed application will have to satisfy is pricing. There are numerous methods used by sharing economy platforms when it comes to pricing, some allow the users to decide whether they will be sharing their items or skills for free or whether they will be charging a fee and the amount of said fee, while others keep the power in the hands of the providers of the platform. Both of these methods will have to be considered when deciding upon the best pricing system for the proposed application.

2.2 State of the Art

This section presents a literature review. The literature presented shows possible approaches in designing an application model and satisfying the requirement of privacy for the proposed application.

2.2.1 Application Model

This section outlines the research that has been done into different possible application models that could be used to build the proposed sharing application. I. Constantiou *et al* [2017] proposed four defined sharing economy models known as chaperones, gardeners, franchisers and principals. The purpose of these models is to distinguish sharing economy platforms from traditional marketplaces, supplier networks, third-party intermediaries, etc. Each model is classified along two dimensions. The first being the level of control exerted by the owner of the platform over the users of the platform and the second being the intensity level of rivalry among the users of the platform fostered by the owner. The control dimension refers to the independence of the platform users. Control is tight when the platform owner specifies, standardizes and monitors all aspects of platform participation. Control is loose when the platform owner defines only minimum standards and is more interested in supporting platform participation rather than dictating it (2).

The rivalry dimension relates to the degree to which a market mechanism is in place for the platform. Rivalry is high when the platform owner uses an algorithm to decide on how to price the service. When rivalry is high the platform is run as a market. Rivalry is low when the platform prices, if any, are based around compensating or sharing the costs of the supply side (2). Figure 2.1 shows each possible application model relating to their level of control and rivalry.

	Control	
	Loose Minimum standards or guiding principles for platform participation are set by the platform owner	Tight Platform participation is specified, standardized and monitored by the platform owner
Rivalry		
High Pricing scheme based on real-time changes in supply and demand	Chaperones Prototypical Example: Airbnb <ul style="list-style-type: none"> • Value proposition: Service differentiation • Other examples: Homeaway, Rentomo, Apprentus 	Franchisers Prototypical Example: Uber <ul style="list-style-type: none"> • Value proposition: Low costs and efficiency gains • Other examples: Lyft, Postmates, Caviar
Low Pricing scheme based on compensation of the suppliers' costs	Gardeners Prototypical Example: Couchsurfing <ul style="list-style-type: none"> • Value proposition: Self-organization and community building • Other examples: BeWelcome, BlaBlaCar, Peerby 	Principals Prototypical Example: Handy <ul style="list-style-type: none"> • Value proposition: Low costs and risk mitigation • Other examples: TaskRabbit, Zeel, Deliveroo

Figure 2.1: Sharing Application Models (2)

According to I. Constantiou *et al* [2017] there are four possible models for a sharing economy application that is based on their approach. The first is Chaperones. This model has high rivalry and loose control. This means that the platform will run as a market and participants are encouraged to act as competing micro-entrepreneurs. The loose control aspect means that while participants can be informed by the platform owner about certain details such as supply levels, the participants themselves decide on their own prices. One example of a Chaperone sharing application is Airbnb. The home owners are competing with one another by offering rentals at competitive prices, proving the platform uses a high rivalry approach. However, it is the home owners, not the platform providers, who decide on their fees, proving the platform providers exert loose control. (2).

The next sharing application model proposed by I. Constantiou *et al* [2017] is known as Franchiser. This model has high rivalry and tight control. This means that the platform owner has absolute control and authority over the platform, meaning they can dictate the prices or change the algorithms that are used to dictate the prices. A platform owner of a Franchiser application is focused on standardising the service being provided. Due to the high rivalry aspect, the platform is still run as a market but differs from the Chaperone model in that the prices are dictated by the platform owners algorithms not the participants themselves. One example of a Franchiser sharing application is Uber. The platform is run as a market with each driver competing for customers, this is due to its high rivalry. The platform exerts tight control as the platform providers control the algorithms that set the

price for each journey (2).

According to I. Constantiou *et al* [2017] the third sharing application model is known as Gardener. This model has low rivalry and loose control. This means that the main role of the platform owner is to simply cultivate communities by providing a service with minimum standardization. Low rivalry means that the platforms participants are not encouraged to compete with another but simply offer their services or items for the compensation they desire if any at all. Gardeners gain their competitive advantage from the participants willingness to be involved in the community they have cultivated. One example of a Gardener sharing application is Peerby. In Peerby the users can share whatever items they want and decide on their own fees if they are charging any at all. This is due to the platforms loose control approach. Peerby describes the sharing done on the platform as *gifting* rather than *lending*. This is done to cultivate a community rather than a competitive marketplace, proving its low level of rivalry. (2).

The final sharing application model proposed by I. Constantiou *et al* [2017] is known as Principal. This model has low rivalry and tight control. This means that the platform owner has absolute control over the platform and focuses on standardizing the service by enforcing rules and monitoring the activity of platform participants, very much like a supervisor. However, in contrast to Franchisers, Principals allow the participants to decide on their own prices. The platform owner encourages participants to act more as a community rather than a competitive marketplace. Principal sharing applications gain their competitive edge by providing incentives to participants to encourage a high level of effort on their behalf, for example, rewards, rating system etc. One example of a Principal sharing application is Handy. The platform is highly standardized by implementing rules on how participants do their jobs. This tight control allows the platform providers to standardize the services on the platform and monitor participants. The pricing scheme on the application rewards quality work with higher compensation and encourages no direct competition between participants ensuring that low rivalry is maintained. (2).

2.2.2 Methods to Ensure User Privacy

This section examines different possible methods to satisfy the requirement of privacy in the proposed application. This is done by researching the existing literature on the topic. The risks associated with the requirement of privacy will also be analysed. There are numerous possible privacy threats for the users of the proposed application which must be addressed in the design, implementation and management processes for the proposed application. One of these threats is the privacy of the users' personal location information. The reason this is a threat is because users will have to share their location information in order to find other users close to them to share their items or skills with. Another privacy threat that must be

considered is the privacy of the users' identity. This is a threat because users will eventually have to reveal their identity in partaking in item or skill sharing with other users on the platform.

One method, outlined by F. Stajano *et al* [2004] is known as the mix zone model. This model consists of a trusted middleware system between the location system and the application. The application registers two types of geographic zones, application zones and mix zones. When users are within application zones, the application can identify their location. When users are within mix zones the application cannot trace user movements as the middleware limits the location information received by the application. This is done by mixing together all users' identity inside the mix zone, so instead of the application receiving a traceable user identity associated with a location, it receives a pseudonym from the middleware. The pseudonym of any given user will change when they enter a different mix zone. Mix zones can be assigned either by users, each being asked to register at least one mix zone, or by the application itself. The aim of the mix zone model is to prevent tracking of long term user movements in order to protect user location privacy (3). Figure 2.2 shows a visual representation of this model.

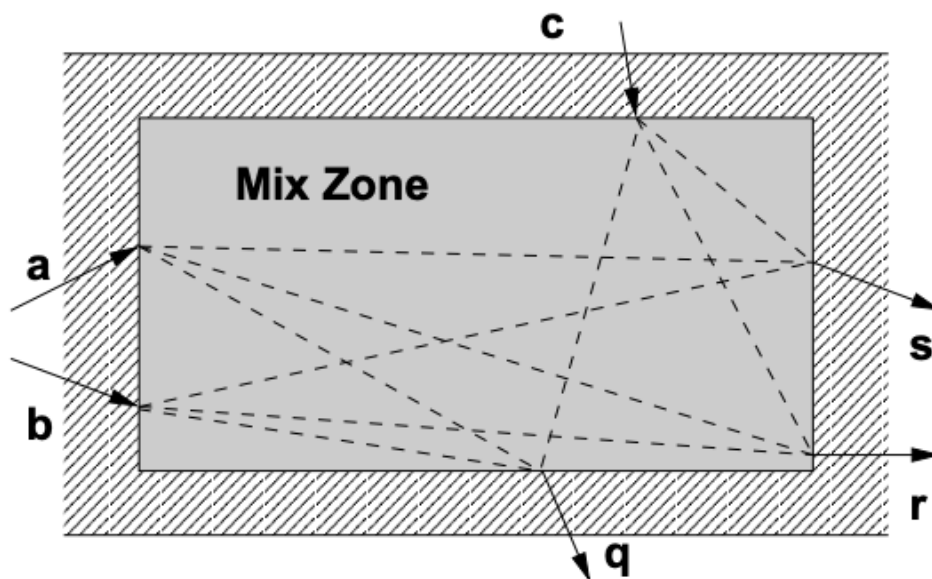


Figure 2.2: Mix Zone Model (3)

Another approach for protecting the location privacy of users on the platform involves implementing a user defined friendship model defining profile access much alike to many social media platforms such as Facebook. As outlined by M. Anwar *et al* [2009] the owner of the profile may assign an access policy to each item of information on their profile, dictating exactly who can see each item of information when they view their profile. This means that users can project different representations of themselves to different groups of

users (15). For example, this is often used to show trusted users, *friends*, on the platform more detailed information than other users. In order to implement this approach the platform must design a consent protocol, whereby a user is able to send a friendship invitation to users they wish to be *friends* with. The user they send it to must either accept or ignore this invitation. It is only when they accept the invitation that they will be able to see the more detailed information on the users profile (15). This method can also be used to protect the privacy of users' identity, as they can opt to only reveal their identity to users they are *friends* with

Due to the constant exploitation of personal information online nowadays, the assurance of information privacy has been compromised. There is major risk regarding the privacy of a users location information. This is because this disclosed information includes not only their current location but also past visited places and daily routes which could all be used to generate a location based profile on a specific user. Other information disclosed by the user could also be used to understand their patterns of use and their behaviours. These are all major risks due to not only hackers who could access the information illegally but also third parties who legally have access to this information and could potentially exploit it. The responsibility of keeping data safe and secure and ensuring privacy for users, lies with the owners of the application that are collecting and storing this user data.

2.3 State of Practice

This section presents a review of existing sharing economy applications. This review focuses on the approaches each application has taken to satisfy the requirements of trust, privacy and pricing. The existing applications reviewed are presented in five different categories, food-sharing, skill-sharing, item-sharing, home-sharing, and ride-sharing applications.

2.3.1 Food-Sharing Applications

This section will provide an analysis into the different approaches taken to satisfy the relevant requirements by existing food sharing applications. Olio is an application that was launched in the United Kingdom in 2015. Olio was co-founded by Tessa Clarke and Saasha Celestial-One. The inspiration behind the creation of Olio was the founders' desire to have a simple way to share food with their neighbors and their belief in the idea that small actions can lead to big changes. The founders both believed that Olio could build a more sustainable future by encouraging people to share food instead of throwing it away. People using Olio are helping in reducing food waste, this in turn helps reduce emissions as less food products need to be bought if they are being shared. The Olio application has over 1.4 million registered users in 49 countries. Last year at the Sustainable City Awards in the United Kingdom, the Olio application won within the Managing Resources Smart Technology

category. Since launching, Olio has had a very positive impact with over 2.4 million portions of food being shared already (16). In short, Olio is a food-sharing platform. Olio allows neighbors to connect with each other along with local businesses to share food that would otherwise be thrown away such as food nearing its sell-by date in stores, spare home-grown produce or the groceries in your fridge that you know are going to waste. Olio establishes trust between its users by creating a sense of community within their platform. This community-based aspect of the application encourages users to trust one another as they are sharing with their own neighbours. One way Olio implements the community-based aspect of their platform is through their location based recommender system. When users search for food on the platform the results are all contained nearby to the users location.

Olio protects the privacy of their users in many ways such as providing their users with a privacy policy that ensures each user that the platform will not share any of their information with external parties unless legally necessary. The platform also ensures their users location privacy be protected by ensuring that other users can only see how far away another user is from them, they cannot view their actual location or address (16).

In terms of pricing Olio encourages an atmosphere of bartering or swapping rather than payments. To share food with their neighbors, users simply have to add a photo and description of the available food that they wish to share along with where and when the food is available for collection. For users that are looking for food, they simply have to browse the listings of available food near them, when they find some food that they are interested in they just have to send a request to the user to arrange the pick up time (16).

2.3.2 Skill-Sharing Applications

This section will provide an analysis into the different approaches taken to satisfy the relevant requirements by the existing skill-sharing application, Anytimes. Anytimes is an application and website that was launched in Japan in 2013. Anytimes was founded by Chika Tsunoda. Chika's main inspiration behind the creation of Anytimes was her belief that the Japanese work environment needed both diversification and flexibility. Chika wanted to provide an alternative to the traditional Japanese-style work environment, especially for women since Japan's gender wage gap is a whopping 25.5 percent. The company gained 2.1 million US dollars of funding in 2015 and has since that point partnered with numerous companies and local governments across Japan. Its plan now is to expand into more countries in Asia over the next few years (17). Anytimes is very similar to the item-sharing application Peerby in that it encourages users to share with their neighbors, the difference between Anytimes and Peerby is that it allows users to share their skills with one another instead of their items. Anytimes is an application that connects people through skill-sharing. For users to share their skills with others they simply have to post an ad of the services that

they can offer to people on the platform. For users to borrow skills from others they can simply submit a request for the service that they need and wait for a user with the necessary skills to respond. Similarly to Peerby and Olio, Anytimes uses the idea that they are a community based platform to encourage trust between their users.

Anytimes protects their users identity by only showing usernames on the platform, unless the user decides they want to show their full name. When registering for the application users do not have to provide their location information. They only share their general location when requesting a service by another user. They then only share their exact address with the user when they feel comfortable to do so, protecting their location privacy on the platform (18).

The platform like the majority of other sharing economy platforms allows their users to decide upon their fees. However the platform does encourage a system of bartering much like Olio. Payments can be offered by the user needing the service or requested by the user offering the service. Payments to the provider of the service are held in escrow by the company until after the service has been provided (18). Anytimes allows people to outsource services for much cheaper. It also provides people with a simple platform to earn some extra money using their skills. Similarly to Peerby it is the users who are offering the services that decide on their price.

2.3.3 Item Sharing Applications

This section will provide an analysis into the different approaches taken to satisfy the relevant requirements by the existing item-sharing application Peerby. Peerby is a start-up application and website that was launched in Amsterdam in 2012. Peerby was founded by Daan Weddepohl. Daan had the vision of creating a platform that solved the common problem of living in a city and not knowing who your neighbors are. He created Peerby to connect neighborhoods and encourage people to share their belongings with one another, not only for the obvious sustainability reasons but also to cultivate communal relationships for people living within cities, to connect big city dwellers the way small town inhabitants are usually connected. Peerby has over 250,000 users worldwide, making it the largest sharing economy platform in Europe. In 2016 Peerby received 2.2 million dollars, all raised from a crowdfunding campaign by its users, this money has been put towards the worldwide expansion of the company which has proved successful so far (19). In short Peerby is an item-sharing platform. Peerby allows people to borrow the items that they need from others within their neighborhood. It also allows them to lend the items that they have to their neighbors. According to Peerby 80 percent of items we own are used no more than once a month, this process of sharing allows us to put our items to better use when we don't need them. Peerby establishes trust between users through this sense of community, by

advertising their platform as a neighbour to neighbour item-sharing service.

In terms of privacy, Peerby does not implement any privacy protecting methods. Users have to share their full name, their contact details and their address. The platform does not implement any complex methods to protect the privacy of their users' location, it simply adds a margin of error of two hundred metres so that users can only see each others approximate address (20). It is unclear if the algorithm for adding this margin of error is randomized, or the same for every user. Randomized would make it more difficult to hack.

Similarly to Olio, Peerby allows users to decide on their fees while still encouraging users to swap items rather than charging a fee. On Peerby, to lend items out users simply have to post an ad of their item on the platform and to borrow items the users can search their area to find what is available near them or explicitly search for what they need. When found the user simply sends a sharing request to borrow the item. The user lending the item out decides on the price if there is one. This saves people from spending a lot of money on things they will only be using for a limited amount of time. The other benefits of using Peerby includes allowing people to connect within their community and perhaps most importantly, it allows people to live in a way that is far more environmentally sustainable.

2.3.4 Home-Sharing Applications

This section will provide an analysis into the different approaches taken to satisfy the relevant requirements by the existing home-sharing application Airbnb. Airbnb is an online marketplace that was founded in America in 2008 by Brian Chesky. It is essential for the safety of the platforms users, that Airbnb ensure that each user can trust one another. One of the processes that Airbnb implements in order to ensure the safety of its users is background checks. The platform runs users names through different public databases such as sex offender registries and criminal records to ensure that individuals are safe to use their site. Though background checks such as these can't always identify a persons past crimes they will help in keeping the majority of dangerous individuals away from the platform if their records are available. Despite these checks the platform still encourages users to practice caution and follow provided safety tips. The safety tips provided by the platform include, getting to know the guest before allowing them to use your home, protecting your finances by making and receiving payments through the Airbnb's secure payment platform etc (21).

Unlike Olio, Airbnb does share user information with outside sources but only when the user has consented for them to do so, as discussed in the platforms privacy policy. Airbnb does take measures to not only protect the location privacy of their users but also the identity privacy of their users by only showing users' first name. To protect their users'

location privacy the platform shows an approximate location, only revealing the exact address when a booking has been confirmed (22).

Airbnb allows its users to decide upon their fees. However, the platform does provide advice on how to come up with an appropriate fee for your property. The users must also pay a host fee to the platform. This fee changes depending on the relevant country but is usually around three percent (23).

2.3.5 Ride Sharing Applications

This section will provide an analysis into the different approaches taken to satisfy the relevant requirements by existing ride-sharing applications. Uber is an American multinational company offering many services such as ride-sharing. The company was founded in 2009 by Travis Kalanick and Garrett Camp. Uber uses transparency in order to establish trust between its users. It uses GPS technology to show you exactly where your car is and both how long and what route it will be taking before it arrives to you. Uber, like many other applications today also uses a rating system to establish trust. Riders rate each other based on their ride-sharing experiences and their accumulated ratings show up alongside their information. For, example if you are the driver and a certain rider is always late when you are collecting them or if they misbehave in your vehicle you can give them a low rating. The combination of this GPS technology, electronic payment, rating system etc. makes people willing to get into the backseat of strangers' cars as if they were not strangers at all (24).

Uber requires users to provide their full name and contact information when signing up to the site. As stated in their privacy policy, this user information along with information collected as users engage with the platform such as location and transaction data, will be shared with Uber subsidiaries, affiliates, service providers and business partners. One measure implemented by Uber to protect the location privacy of their users, is that they only collect location data when the user is running the application (25).

Unlike every other application discussed Uber decides upon their fees themselves. To do this they use a dynamic pricing algorithm. The algorithm decides on rates based on numerous variables, the most important being the time and distance of the route. Other variables considered include traffic, the current demand for a driver in the area and peak hours (26).

Figure 2.3 shows a comparison of each of the analysed existing sharing economy applications, in terms of the approaches they have taken to satisfy the requirements of trust, privacy, and pricing.

		Requirements			
		Type	Trust	Privacy	Pricing
Applications	Peerby	Item Sharing	Community based platform.	Adds margin of error to user location.	Free to use. Users decide on fees.
	Olio	Food Sharing	Community based platform. Offers safety guidelines to users.	Shows users distance from one another instead of address.	Free to use. Users decide on fees.
	Uber	Ride Sharing	Rating system on drivers. Background checks on drivers. GPS tracking system on drivers.	Only collects user's location data when they are running the app.	Free to use. Platform algorithm decides on fees.
	Airbnb	Home Sharing	Background checks on homeowners. Offers safety guidelines to users.	Address only provided to users when booking confirmed. Doesn't share users last names.	Free to use. Users decide on fees. Users pay 3% of the fee to the platform.
	Anytimes	Skill Sharing	Community based platform.	Only provide address to other users when comfortable. Only shares usernames.	Free to use. Users decide on fees.

Figure 2.3: Existing Applications Requirements

2.4 Summary

As evidenced by the state-of-the-art and the state-of-practice, there are numerous possible approaches to satisfy the requirements of trust, privacy, and pricing that could be implemented in the proposed application. The next chapter of this paper will discuss the design decisions made for the proposed application. These decisions are based on the findings presented in this chapter.

3 Design

This chapter discusses the decisions that have been made for the proposed application regarding the application model and the methods used to satisfy each requirement. These decisions are all based around the analysis presented in chapter 2 of different possible application models and different possible methods to satisfy each requirement.

Section 3.1 discusses the decisions made regarding the application model of the proposed application.

Section 3.2 discusses the approaches that will be implemented in the proposed application to satisfy the requirements of trust, privacy, and pricing.

Section 3.3 presents the decisions made regarding the application development environment that will be used to develop the proposed application.

Section 3.4 presents the possible architectures that could be implemented with the chosen application development environment and details which of these architectures will be implemented and why.

3.1 Application Model

This section discusses the application model used to design the proposed application based upon the research that has been presented. There were numerous challenges in choosing the application model for the proposed application. Firstly, ensuring the users of the platform would have the freedom to make their own choices on how to sell or swap their items and services, while still encouraging a system of bartering on the platform. The challenge here was understanding how to get the users to swap their items and skills free of charge without taking away their right to charge a fee. Secondly, ensuring that the users of the platform can trust each other. Lastly, discouraging negative and opportunistic behavior on the platform. The application model that will be implemented balances these challenges in the best way possible. As stated in section 2.2.1 there are 4 possible application models, Chaperones, Franchisers, Gardeners, and Principals. There are two parameters involved in the classification of these models, rivalry and control. The proposed application will implement the tight control approach. This means the platform providers will have the ability to monitor the platform and control who uses the platform. This approach was chosen because it will encourage users to put their trust in the platform as they know it is being monitored.

Having the users trust the platform is essential if we are to encourage more people to join the application. The application will implement the low rivalry approach. This means that the platform will not run as a marketplace. Users will have the right to choose the prices of their items and skills. We will use this approach because it will allow users to make their own choices on how to sell or swap their items and skills. Hopefully, this approach will also encourage a sense of bartering on the platform. The application model will, therefore, fall under Principals. According to I. Constantiou *et al* [2017], the Principal model exerts tight control over the application to standardize it. They do this by enforcing rules and monitoring the platform. Principals gain a competitive edge by giving their users incentives. Platforms that fall under the Principals model share the values of competing at low or no cost through tight control and mitigating the risk of opportunistic behavior by the users (2). Users caught partaking in opportunistic or dangerous behavior on the platform will be banned giving users further assurance that they can trust in the platform and its users.

3.2 Requirements

This section of the chapter discusses the methods implemented to satisfy the platform requirements of trust, privacy, and pricing. These methods were chosen based on the research presented in chapter 2.

3.2.1 Trust

This section discusses the methods chosen to satisfy the requirement of trust in the proposed application. To research methods of satisfying this requirement of trust, an analysis of the methods used by existing applications in the sharing economy was performed. This analysis can be found in section 2.3. As stated in section 2.1 there are many challenges associated with trust regarding sharing economy applications. To satisfy this requirement, the users of the application must be able to trust the platform itself, trust the other users on the platform, and trust the items and skills that will be shared through the platform. As discussed in section 3.1 the tight control approach implemented in the application model will encourage users to trust the platform because the application will be monitored. Chapter 2, presented several different methods for garnering trust between users. It is difficult to say which of these methods is the most effective as each user could be affected differently by each method. For this reason, a hybrid of the methods presented to encourage trust between users will be implemented on the platform. Firstly, the proposed application will be implemented as a community-based application. This will be done through the branding and advertising of the proposed application, as well as through the implementation of a location-based recommender system that ensures users are sharing with other users within their community. The platform will also implement as much transparency as possible

without violating user privacy, in the hope that being able to see relevant information about each other will initiate trust between users. A rating system will also be implemented in the proposed application so users can rate each other positively or negatively based on their experiences. Lastly, users will be presented with a list of safety tips for them to use. The hope is that by implementing numerous different methods to establish trust between the users, every user will respond to at least one of these methods. Finally, the platform must implement methods to ensure that users can trust the items and skills shared through the platform. Unfortunately, there is no way to standardize the quality of goods and services shared through the proposed application. However, a rating system will be implemented, and users will be assured that any user caught sharing faulty goods or services will be banned from the platform.

3.2.2 Privacy

This section discusses the methods chosen to meet the requirement of privacy in the proposed application. To research methods of satisfying this requirement of privacy, an analysis of relevant literature and existing sharing economy applications was performed. This literature analysis can be found in section 2.2 while the analysis of existing applications can be found in section 2.3. The challenge in satisfying this requirement is that the platform must still show enough user information so that users can trust one another, while still maintaining their user privacy. There are two types of privacy that must be considered, location privacy and identity privacy. A user-defined friendship model will be implemented in the proposed application to tackle the threat of location privacy. Users will be able to define their level of location granularity for their *friends* and for the other users on the application. For example, user A can decide that only their *friends* can view their exact address and that every other users can only see their general location. User A can decide that user B is a *friend* after multiple positive sharing experiences with user B. This type of model has been used since the conception of social networking sites such as Facebook, allowing users to share more with their *friends* and less with everyone else. As outlined by M. Anwar *et al* [2009], the same friending policy as Facebook will be used in the proposed application. Users will be allowed to send their friendship invitation to other users. When it is accepted, the two users will become *friends* and will be allowed to have more access to each others location information (15). This same user defined friendship model will also be used to protect the identity of users. Users will be allowed to share less of their personal information with unknown users and more personal information with their *friends*. For example, users can decide to share their first and last name with their *friends* and share only their first name with everyone else. This model ensures that users are still revealing enough information so other users can trust them, while still allowing them to maintain their privacy. This model also protects the freedom of the user because they can decide what personal

information they wish to share and with whom.

3.2.3 Pricing

This section discusses the methods chosen to satisfy the requirement of pricing in the proposed application. To research methods of satisfying this requirement of pricing, an analysis of the methods used by existing applications in the sharing economy was performed. This analysis can be found in section 2.3. Two approaches to meet this requirement emerged from this analysis. The first approach allows users to decide on their own fees, while the second approach allows the platform to choose the fees. The second approach is well suited for applications that are trying to establish a marketplace. Setting the fees for your platform allows the provider more control over the platform, but also gives the provider the power to say what items and skills are the most valuable instead of the user. This approach is also well suited for platforms that want to generate a profit. The purpose of the proposed application is to encourage people to participate more in the sharing economy. Therefore, the first approach is more appropriate for the proposed application. Users will decide on their fees. However, the proposed application will also follow in the steps of Peerby and Olio in that it will encourage users to swap goods and services with one another rather than renting them out and charging fees.

3.3 Application Development Environment

This section presents a comparison of two possible application development environments that could be used to develop the proposed application. These environments include IntelliJ IDEA and Android Studio. From the results of this comparison, the decision of which environment will be used to develop the proposed application will then be presented.

IntelliJ IDEA is an integrated development environment that is written in java and is used to assist the development of computer software and applications. This environment prides itself on maximizing developer productivity with the implementation of numerous helpful features. The first of these features to be discussed is its ability to detect duplicates. The environment compares code fragments and notifies you if there is a similar code fragment that can replace it, improving the efficiency and redundancy of the developers code. Additionally, the development environment provides inspections that find mistakes in the code. The feature then offers the user solutions to these mistakes. This feature ensures the correctness of the developers code. The environment provides features that improve the ergonomics of the development environment, one of which is helpful shortcuts. IntelliJ IDEA offers keyboard shortcuts for rapid selection, switching between tool windows, and the editor and many more. This results in the process of developing to be more efficient for the developer. Another feature dedicated to the ergonomics of the development environment is

the user interface. All the lists, trees, and popups in the environment provide the developer with a quick search that instantly takes you to a set of relevant items (27).

Android Studio is the official integrated development environment for Google's Android operating system. Android Studio and IntelliJ IDEA share similarities as Android Studio is built upon IntelliJ IDEA but is designed specifically for Android application development. Android Studio offers even more features than IntelliJ IDEA that are specific to android application development, such as its emulator. The emulator enables the simulation of android devices on a machine, allowing developers to test their application on multiple devices and API levels. Testing an application on various devices and API levels is essential to ensure consistency. These emulated devices provide almost all of the same capabilities as the physical device would, such as location services, access to applications such as google maps and google play store, and even the ability to simulate incoming phone calls and text messages. Another unique Android Studio feature is its layout editor. The layout editor allows you to build layouts visually by dragging user interface elements into the visual design editor instead of writing the XML by hand. This feature can save the developer a lot of time and also gives a much clearer understanding of what the user interface is going to look like before even viewing it on the emulator (28).

From the research done into IntelliJ IDEA and Android Studio and the comparison of both of these options, it is clear that Android Studio offers more useful features and capabilities for the specific developments of an Android application. Additionally, the developer of the proposed application has more experience with Android Studio, making it the most appropriate choice.

3.4 Architecture

This section presents a comparison of the possible architectures that could be applied to the proposed application. These architectures include Model View Controller (MVC), Model View Presenter (MVP), and Model View ViewModel (MVVM). These are the most typical architectures used in Android Studio applications. From the results of this comparison, the decision of which architecture to implement in the proposed application will then be presented.

3.4.1 MVC

The MVC Android Studio architecture splits the code into three components, model, view, and controller. The model represents a set of classes that describe the business logic and data operations for the application. The view represents the user-interface components of the application. The view is responsible for displaying the data received from the controller

to the user. The controller is responsible for processing incoming requests from the view. It reacts to users' input and processes the incoming data with the help of the model. It then passes the results back to the view to present to the user. MVC is the most widely used architecture for Android Studio applications for numerous reasons, some of which include; The architecture offers support for fast and parallel development due to the separation of concerns. This separation also increases the testability of the code. Developers can unit test the model with ease due to its lack of referencing to Android classes. Modifications to the user-interface do not affect the entire model, meaning updating the appearance of your application in the future can be done more efficiently. The disadvantages of the MVC architecture include; The dependence of the view on the model becomes a disadvantage when the view grows more complex due to the increasing number of classes required. Considering the view depends on the controller and the model, any alterations to view logic require updates to several classes (4). Figure 3.1 shows a visual representation of the MVC architecture.

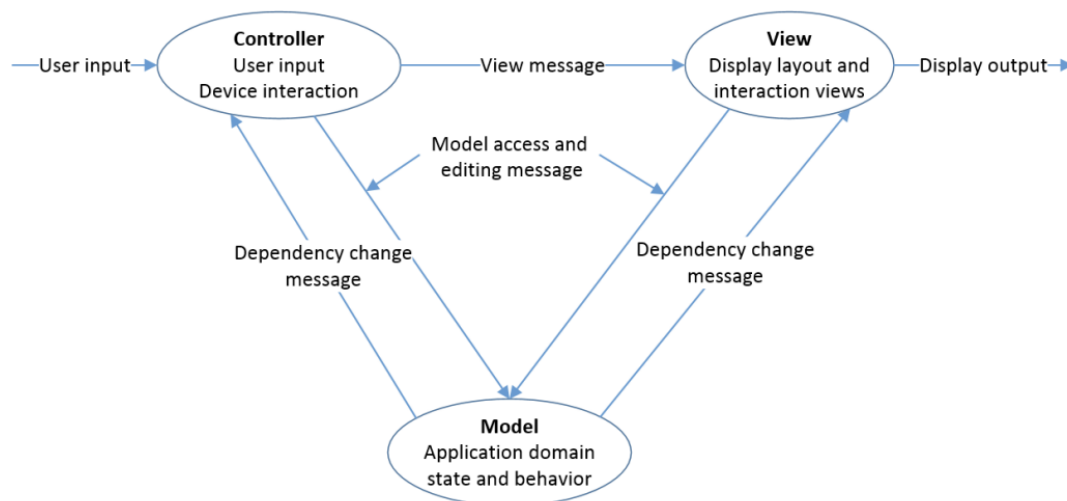


Figure 3.1: MVC Architecture (4)

3.4.2 MVP

The MVP Android Studio architecture splits the code into three components, model, view, and presenter. Similarly to MVC architecture, the model represents a set of classes that describe the business logic and data operations for the application, and the view represents the user-interface components of the application. The view is responsible for displaying the data received from the presenter to the user. The presenter responds to user-interface events. It uses use cases to perform tasks on model objects and passes the results to the view. Unlike the controller, the presenter is decoupled from the view and only communicates to it through the user-interface. Android Studio developers prefer MVP architecture over MVC architecture because the view is dependent on the presenter, not the model and

controller. This lessens the complexity of the user-interface logic. Another advantage of the MVP architecture is that the application is more reusable because the components, model, view, and presenter are independent and can, therefore, be replaced. The disadvantages of the MVP architecture include; The size of the code when using this architecture can become excessive and therefore increases the complexity. This architecture also involves an extra learning curve. Due to the higher complexity, size, and learning curve, the architecture isn't suitable for small applications (4). Figure 3.2 shows a visual representation of the MVP architecture.

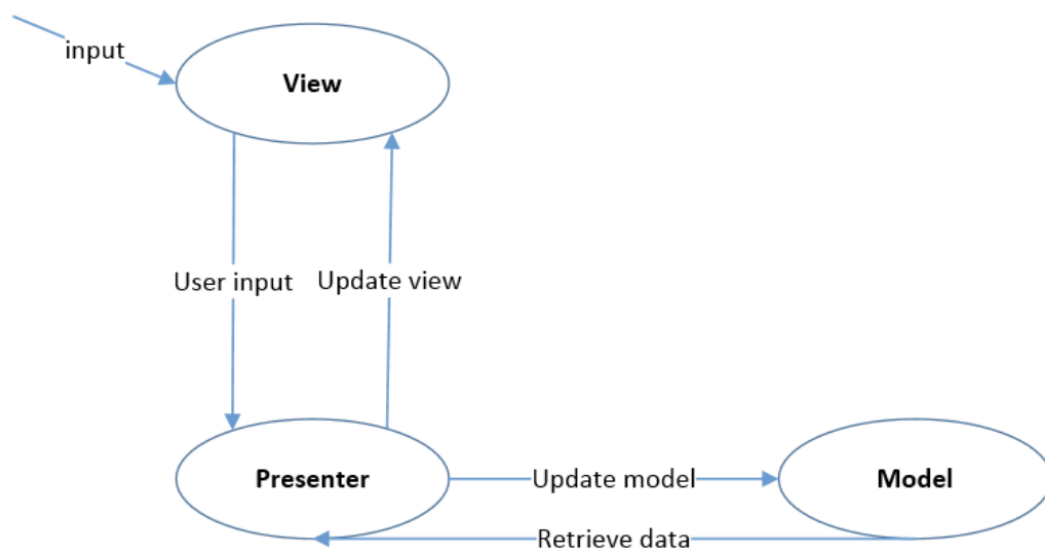


Figure 3.2: MVP Architecture (4)

3.4.3 MVVM

The MVVM Android Studio architecture splits the code into three components, model, view, and viewModel. Similarly to MVC and MVP architecture, the model represents a set of classes that describe the business logic and data operations for the application, and the view represents the user-interface components of the application. The view is responsible for displaying the data received from the viewModel to the user and transforming the models into user-interfaces. The viewModel exposes methods, commands, and other properties that are needed to maintain the state of the view, manipulate the model based on view events and trigger events in the view. The view has a many to one relationship with the viewModel. This means that it can map various views to one viewModel. MVVM architecture has become the most popular since its conception. One of the main reasons for this is its testability. MVVM breaks the dependence between application logic and the user-interface even more so than MVP, making testing more accessible. Another benefit of MVVM is how streamlined the data process is. The application will only submit the necessary data, thereby

boosting its overall performance. This boost means faster loading times for your users. Despite being the most recommended architecture, MVVM is not without its drawbacks. For simple user-interface operations, the architecture can add unnecessary complexity. Therefore, as the application grows, generalizing your view model becomes more difficult. There is also an increase in memory consumption, particularly with larger applications (4). Figure 3.3 shows a visual representation of the MVVM architecture.

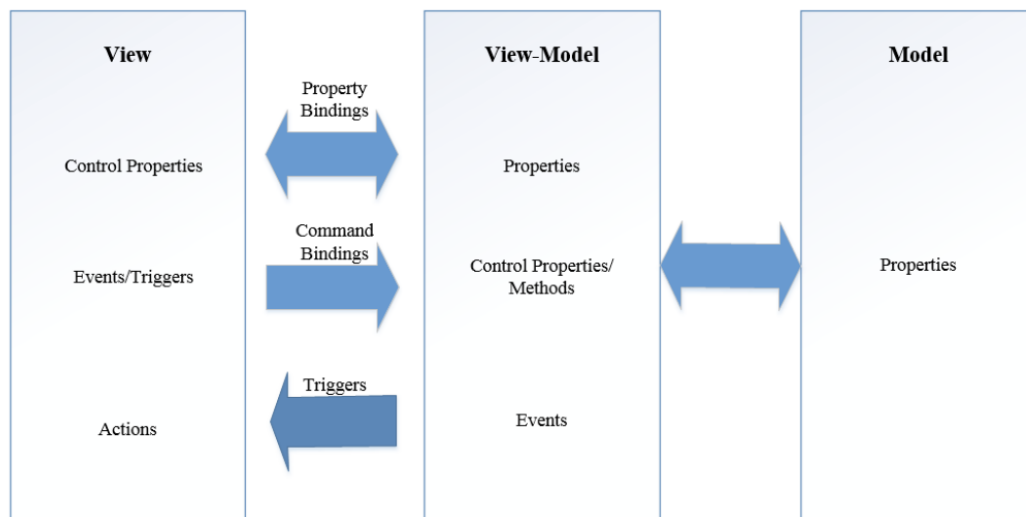


Figure 3.3: MVVM Architecture (4)

In this case the MVVM architecture will be implemented. Though it has its disadvantages, most apply to large applications and, therefore, won't apply to the proposed application.

4 Implementation

This chapter of the paper discusses the implementation of the various components of the proposed application. This includes the challenges that were encountered during the implementation of each component and how these challenges were handled.

Section 4.1 discusses the implementation of the Model View ViewModel (MVVM) architecture in the proposed application.

Section 4.2 discusses the implementation of the tools necessary to process, prepare, and store data that the proposed application needs.

Section 4.3 discusses the implementation of methods needed to satisfy the requirements of trust in the proposed application.

Section 4.4 discusses the implementation of methods needed to satisfy the requirement of location and identity privacy in the proposed application.

Section 4.5 discusses the design of the proposed applications user-interface.

4.1 Architecture

This section details the implementation of the Model View ViewModel (MVVM) architecture in the proposed application. As discussed in section 3.4.3 the MVVM architecture splits the code into three components, model, view, and view model. The first step in implementing this architecture involves separating the class structure into the three components of Model, View, and ViewModel. In the default architecture of Android Studio applications the data, logic, and user-interface are all kept within fragments and activities. This architectural structure presents challenges as the logic of the application gets confused with the presentation code, introducing bugs that are both difficult to find and difficult to solve. It is for this reason that the MVVM architecture is implemented. The first class of this architecture is known as the model, this class contains the data used in the application. The second class of this architecture is known as the view, the view classes are written in the layout files. The final class of this architecture is known as the ViewModel, the ViewModel classes handle and store user-interface related data. After correctly structuring the files the dependencies must be added to the Gradle along with the data binding library. These dependencies can be seen in figure 4.1.

```

}android {
    dataBinding {
        enabled true
    }
}

dependencies {
    implementation 'android.arch.lifecycle:viewmodel:2.3.0'
    implementation 'android.arch.lifecycle:compiler:2.3.0'
}

```

Figure 4.1: MVVM Architecture Dependencies

The data binding library is needed to create the link between the model and the user-interface, where the model holds all the information needed to be displayed. Without the data binding library, one would need to update all user-interface widgets after the data relevant to these widgets changes. This uses up a lot of time and space. Therefore, the implementation of data binding significantly reduces the application logic relevant to the user-interface and eliminates the need for methods such as *findViewById()* and *setText()*.

The model classes are implemented to hold all the data retrieved from the external database. These classes should contain no logic, no events, no complex calculations or rules. They should only contain properties and property validation. An example of one of the implemented model classes is seen in figure 4.2.

```

public class Items {

    public String name;
    public String description;
    public String price;
    public String image;

    public Items() {}

    public Items(String name, String description, String price, String image) {
        this.name = name;
        this.description = description;
        this.price = price;
        this.image = image;
    }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getDescription() { return description; }

    public void setDescription(String description) { this.description = description; }

    public String getPrice() { return price; }

    public void setPrice(String price) { this.price = price; }

    public String getImage() { return image; }

    public void setImage(String image) { this.image = image; }
}

```

Figure 4.2: Items Model Class

The layout files represent the Views of the application. Due to data binding they must be updated to contain a layout root tag with an internal data tag, where the ViewModel binds the data to the View. The data tag can contain multiple variable tags, these tags refer to properties that are used by the user-interface component. The variable tag must contain the name and the type of the property. Buttons in the layout files use *onClick()* functions to invoke listener methods inside the relevant ViewModel. Image views and text views are updated by extracting the new text or image from their ViewModel. Edit text components update the data values in the Model via the relevant ViewModel. An example of a *TextView* can be seen in figure 4.3.

```
<TextView
    android:id="@+id/DisplayName"
    android:textColor="@color/whiteTextColor"
    android:layout_width="150dp"
    android:layout_height="40dp"
    android:layout_centerHorizontal="true"
    android:layout_gravity="center"
    android:gravity="center"
    android:textSize="20dp"
    android:layout_marginTop="230dp"
    android:text="@{userViewModel.displayName}" />
```

Figure 4.3: Items Layout Class

The ViewModel classes handle and store user-interface related data. They hold all the necessary business logic and data operations of the application. The methods called on by the Views are all implemented here. It also updates the Model classes on any relevant data changes. The ViewModel allows data to survive configuration changes such as screen rotations. Figure 4.4 shows a snippet of the implemented users ViewModel.

```
public class userViewModel extends ViewModel {
    String displayName;
    private List<User> users;

    private List<User> getCurrentUser() {...}

    displayName = currentUser.getName();
}
```

Figure 4.4: Users ViewModel Class

Figure 4.5 displays the implemented user page showing the user defined display name and default profile image. The default profile picture as shown in figure 4.5 will be displayed until the user uploads their own profile image.

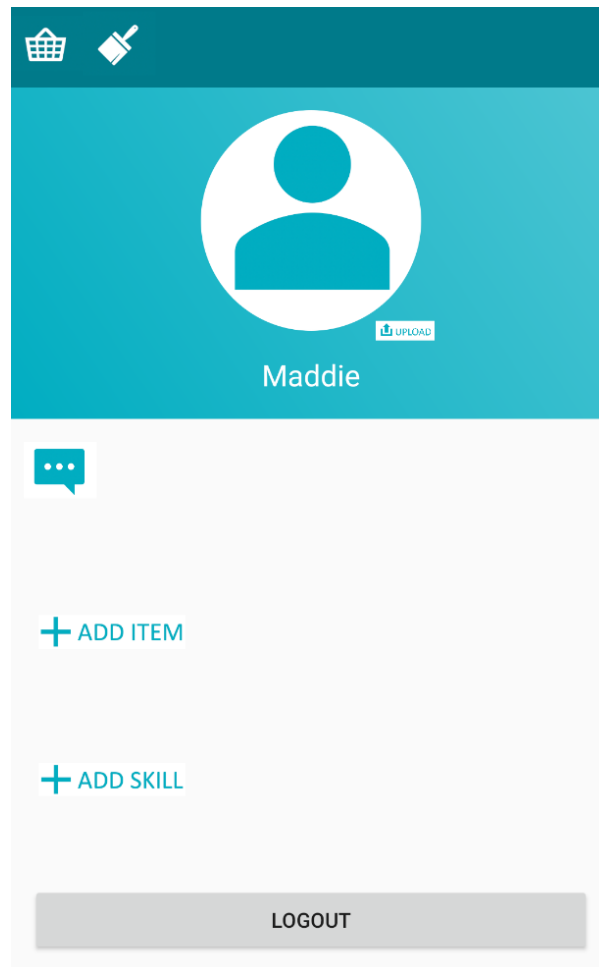


Figure 4.5: User Profile Page

4.2 Data Processes

This section discusses the processes and tools used to both prepare and store the data needed for the proposed application.

Section 4.2.1 presents the implementation of the tool used to facilitate user authentication in the proposed application.

Section 4.2.2 presents the implementation of the database used to store the data related to the proposed application.

4.2.1 User Authentication

This section discusses the implementation of the user authentication, along with the challenges faced during implementation. A major challenge associated with user authentication is ensuring that user data is safe yet accessible when needed. Another challenge can also be the implementation of user authentication itself. This is due to the involvement of processes such as encryption or cryptography. There are various user authentication services currently available. One of these authentication services is the OAuth2 protocol. This protocol provides a value called an auth token that represents the users' identity. OAuth2 is a very flexible protocol that relies on SSL (Secure Sockets Layer), which is used to ensure that cryptography protocols are used as they keep data safe. Another widely used authentication service is Firebase Authentication. Firebase Authentication provides backend services to authenticate users to an application. The special aspect about the Firebase User Authentication service is that it allows developers to implement secure logins simply and quickly. It supports authentication using passwords, phone numbers, and popular identity providers such as Google and Facebook.

While both of these services would be successful in providing user authentication for the proposed application and would both successfully solve the challenges relating to user authentication, Firebase User Authentication will be implemented in the proposed application. There are multiple reasons for this decision. Firstly, Firebase User Authentication works seamlessly with other Firebase services that will also be used in the development of the proposed application. Another reason for this decision is the simplicity of implementation. With the implementation of Firebase User Authentication, the proposed application supports authentication via email and password. For each user, it stores their email, password, and user id. The unique user id is generated automatically by Firebase when a new user registers to the proposed application.

The first step in the implementation process of Firebase User Authentication is the addition of the Firebase dependency to the module Gradle file. To register new users, the application uses the *createUserWithEmailAndPasssword()* method shown in figure 4.6. To sign in existing users, the application uses the *signInWithEmailAndPassword()* method shown in figure 4.7.

```

firebaseAuth.createUserWithEmailAndPassword(userEmail, userPassword).addOnCompleteListener(RegisterActivity.this, new
    OnCompleteListener<AuthResult>() {
        //Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (!task.isSuccessful()) {
                Toast.makeText(RegisterActivity.this, getApplicationContext(),
                    "Register was not successful: " + task.getException().getMessage(),
                    Toast.LENGTH_SHORT).show();
            }
            else {
                firestore = FirebaseFirestore.getInstance();

                FirebaseUser user = firebaseAuth.getCurrentUser();
                String userID = user.getId();
            }
        }
    });

```

Figure 4.6: Firebase Authentication Register Method

```

firebaseAuth.signInWithEmailAndPassword(userEmail, userPassword).addOnCompleteListener(LoginActivity.this, new
    OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (!task.isSuccessful()) {
                Toast.makeText(LoginActivity.this, "Login not successful!", Toast.LENGTH_SHORT).show();
            }
            else {
                startActivity(new Intent(LoginActivity.this, UserActivity.class));
            }
        }
    });

```

Figure 4.7: Firebase Authentication Login Method

4.2.2 Application Database

This section discusses the implementation of the application database and the challenges faced during the implementation process. The proposed application must store more data on each user than Firebase Authentication supports. The implementation of the Firebase Database solves this problem as it enables the storage of further user information. Database implementation poses some of the same challenges as user authentication, such as data safety and the challenge of implementing encryption or cryptography. Availability is another challenge regarding the implementation of a database. If the database goes down, the organization using the database goes down, making availability essential. Ensuring availability involves working with several servers to enable multiple levels of backup. However, this distributed approach can cause your application to become slow. A good database must maintain performance while still ensuring availability.

Firebase offers two database options, Firebase Realtime Database and Firebase Cloud Firestore. Firebase Realtime Database is the original database offered by Firebase. The database is efficient and low-latency. Firebase Cloud Firestore is the newest database offered by Firebase. This database builds upon the success of the original with the addition of a more intuitive data model, richer and faster queries, and the ability to scale further than its predecessor. While both options would work for the proposed application, the Cloud Firestore was implemented due to its additional features. Cloud Firestore solves the

challenge of availability as it houses its data across multiple data centers in different regions. This ensures global scalability and strong availability. Scaling is also automatic in Cloud Firestore. Cloud Firestore stores data as collections of documents with a similar structure to JSON.

Initially, the proposed applications database consisted of one collection named users. Every piece of data related to each user such as their friendships, their added items, and their added skills was then nested under each user. The nested nature of the initial database structure, created unnecessary complexity when implementing queries. To fix this issue the database was restructured. The improved database consists of four collections named users, items, skills, and friendships. The information in items, skills, and friendships is related to the relevant user through their unique user id. The subset of the most significant fields in these collections is shown in figure 4.8.

Field Name	Description	Relevance
<i>userID</i>	Unique identifier for users	Identifies user details, items and skills posted by user and friends of user
<i>profileImage</i>	Unique identifier for image in storage	Needed to display image from storage
<i>itemID</i>	Unique identifier for items	Identifies items, facilitating management and displaying
<i>skillID</i>	Unique identifier for skills	Identifies skills, facilitating management and displaying
<i>rent</i>	Boolean value identifying if item/skill is available for renting	Allows for filtering of items/skills based on value
<i>swap</i>	Boolean value identifying if item/skill is available for swapping	Allows for filtering of items/skills based on value

Figure 4.8: Significant Database Fields

4.3 Trust Establishment

This section discusses the implementation of methods used to establish trust between users of the proposed application. As discussed in section 3.2.1, implementing the proposed application as a community-based platform is one of the ways trust will be established

between the users. This is discussed in section 4.3.1. Another way trust will be established on the platform is through the implementation of a rating system. This is discussed in section 4.3.2.

4.3.1 Location Based Recommender System

This section discusses the implementation of the location based recommender system. Community-based applications typically bring users together by giving them a common goal. In this case, the goal is to reduce consumer expenditure and increase sustainability in Ireland through the activity of sharing or swapping items and skills through the proposed application. This goal must be used in the branding and advertising of the application in order to bring together people with this common interest. Maybe the most important aspect of a community-based application is the implementation of a location based recommender system. This means that when a user is looking for an item or skill, the search results should all be relatively close to the user who is searching. To implement this system, the users address is retrieved when they are searching for an item or skill. The address is then compared to the addresses of the users within the search results. Only results within a certain distance are displayed to the user. As of now that distance is two kilometres. Of course this can be easily increased or decreased depending on the results. As shown in figure 4.9, when searched it is ensured that the items or skills displayed are within the threshold distance of two kilometres.

```
firebaseFirestore.collection("Items").get().addOnCompleteListener(new
    OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot doc : task.getResult()) {
                    user = doc.getUser();
                    userAddress = user.getAddress();
                    distance = getDistance(currentUserAddress, userAddress);
                    if (distance < 2) {
                        image = doc.getId();
                        String name = doc.getString("itemName");
                        String description = doc.getString("itemDescription");
                        String price = doc.getString("price");
                        itemsList.add(new Items(name, description, price, image));
                    }
                }
            }
        }
    });
```

Figure 4.9: Part of Location Based Recommender System

By default these results will be ranked based on the distance to the relevant user. The closest search result will be ranked first, while the furthest will be ranked last.

4.3.2 Rating System

This section discusses the implementation of the rating system. The rating system assists in establishing trust between users of the platform and also establishes trust between the users of the platform and the items and skills being shared on the platform. The algorithm used for the rating system is shown in figure 4.10. As discussed by Fahmi, A *et al* [2018] this algorithm is known as a weighted average algorithm. Each rating (1 to 5) is weighed with the number of votes it has received. The new rating equals the sum of these weights divided by the total number of ratings [28].

$$\frac{(5 \times \text{Five Star Ratings}) + (4 \times \text{Four Star Ratings}) + (3 \times \text{Three Star Ratings}) + (2 \times \text{Two Star Ratings}) + (1 \times \text{One Star Ratings})}{\text{Number of Ratings}}$$

Figure 4.10: Weighted Average Rating Algorithm

The rating is displayed to users of the platform to allow them to make an informed decision about which items and skills to use and which items and skills to not use. The rating can also play a part in the recommender system. By default, the order of the items or skills displayed is based on the distance, closest first, furthest last. However, the user can manually choose to sort the results by rating instead. In this case items and skills are ranked by rating, highest rating first, lowest rating last. To rate an item or skill the user must visit the individual item or skill page. A rating bar is implemented at the bottom of these pages. The user must award their rating and then press the rate button below it. An example of this page is shown in the results chapter of this paper at 5. When an item or skill is rated the application calls a function that recalculates the overall rating and returns this new rating to the view. This function is shown in figure 4.11.

```
protected float getRating(int rating) {
    totalRatings += 1;
    if(rating == 5) {
        fiveStarRatings += 1;
    }
    else if(rating == 4) {
        fourStarRatings += 1;
    }
    else if(rating == 3) {
        threeStarRatings += 1;
    }
    else if(rating == 2) {
        twoStarRatings += 1;
    }
    else if(rating == 1) {
        oneStarRatings += 1;
    }
    float newRating = ((5*fiveStarRatings) + (4*fourStarRatings) + (3*threeStarRatings) +
        (2*twoStarRatings) + (1*oneStarRatings)) / totalRatings;
    return newRating;
}
```

Figure 4.11: Rating Calculation Function

4.4 Privacy Protection Methods

This section discusses the implementation of the user defined friendship model in the proposed application. As stated in section 3.2.2, the challenge of meeting the user privacy requirement is ensuring that while user privacy is maintained, the platform still shares enough user information so users can trust one another. The user defined friendship model was chosen as the method to ensure both location privacy and identity privacy for the users of the application. Section 4.4.1 discusses the structuring of the database in terms of the friendship model. Another aspect of the friendship model involves allowing users to customize what information their friends on the platform see and what information everyone else sees. The implementation of this is discussed in section 4.4.2.

4.4.1 Data Structure

This section discusses the structuring of the user friendship data in the application database. The Firebase Cloud Firestore database offers three options in terms of data structuring, adding Documents, having multiple collections, and having sub-collections within documents. Nesting your data in documents is easily implemented and streamlines the data structure. However, this limits the scalability of the data structure, especially if the data is going to expand over time. Since user friendship data will certainly expand over time, this option is unsuitable for the application. Creating sub-collections within documents gives you full query capabilities and ensures the size of the parent document remains the same. However, implementing any deletions to sub-collections is difficult. Since deletions will most likely be made by users deleting *friends*, this option is also unsuitable. The final option is creating multiple collections at the root level. This structure will be used as it offers the most flexibility and scalability. This structure also enables powerful querying within each collection. A JSON representation of the implemented data structure for user friendships can be seen in figure 4.12.

```

{
  "friendships": {
    "userA_id": {
      "userB_id": true,
      "userC_id": false
    },
    "userB_id": {
      "userA_id": true,
      "userC_id": true,
      "userD_id": false
    }
  }
}

```

Figure 4.12: JSON Representation of Friendships Data Structure

As seen from figure 4.12 the root collection is named *friendships*. Each document in the root collection represents each user of the platform and is named using the unique user id of each user. Under each document the unique user id's of other users on platform are used as keys. If the value of the key is true, this means the two users are friends. If the value of the key is false, this means that the users are not *friends* and that a friendship invitation was sent by one of these users and was rejected by the other. If a user id does not exist in the document it means the users are not *friends* and neither of them have ever sent a friendship invitation to the other. When browsing another users profile the code checks if the other user is a *friend* or not. If not, the add friend button will appear on the users profile. When this button is clicked a friend request is sent to the user, the user will appear as friends only when the request is accepted. In the case of a false value between the two users, no add friend button will appear as the friendship invitation has already been sent and rejected.

4.4.2 Information Granularity

This section discusses the implementation of the information granularity options. This model allows users to to customize what information their *friends* on the platform see and what information everyone else sees. The information granularity options allow users to increase granularity for users they are not *friends* with, and decrease granularity for users they are *friends* with. For example, a user can choose to show their full name and exact address to *friends* while showing only their first name and general location to users who are not yet their *friends*. The implementation of this involved adding a button on the user

profile page that brings the user to an information settings page. It is here that the user can edit what information is seen by other users on the platform depending on their friendship status. An example of this implemented edit page is shown in figure 5.3 in chapter 5. To implement the custom information granularity, extra fields of data are needed in the friendship database structure. These fields include `firstName`, `lastName`, `addressFriends`, `addressOthers`. The values of these fields are strings. The value of `firstName` and `lastName` will either be *friends* or *everyone*. This value indicates whether the users first name or last name can be shown to all users of the platforms or only their *friends*. The value of `addressFriends` and `addressOthers` will either be *distance*, *location*, or *address*. This value indicates whether the users distance, general location, or exact address will be displayed to all users of the platform or their *friends*. Figure 4.13 shows the edited friendship data structure that allows for custom information granularity.

```
{
  "friendships": {
    "userA_id": {
      "firstName": "everyone",
      "lastName": "everyone",
      "addressFriends": "location",
      "addressOthers": "distance",
      "userB_id": true,
      "userC_id": false
    },
    "userB_id": {
      "firstName": "everyone",
      "lastName": "friends",
      "addressFriends": "address",
      "addressOthers": "distance",
      "userA_id": true,
      "userC_id": true,
      "userD_id": false
    }
  }
}
```

Figure 4.13: JSON Representation of Updated Friendships Data Structure

4.5 User Interface Design

This section discusses the process of designing some of the user-interface pages of the proposed application. In order to do this a mock up of the application had to be created. 4.5.1 discusses the design of the login and register pages of the proposed application. 4.5.2 discusses the design of the home page of the proposed application. The mock ups of every page of the proposed application were created using MockFlow. MockFlow allows users to brainstorm ideas for the design of their applications user interface without having to implement them on their application. This allows users to plan their user interface more

efficiently. MockFlow consists of a full featured and intuitive editor with thousands of prebuilt components and layouts (29).

4.5.1 Login and Register Pages

This section discusses the design of the login page and the register page. The first step taken to create the mock-up was deciding on the color scheme, to make this decision, complementary and monochromatic color combinations were researched. Due to the eco-conscious nature of the application, the colours blue and green were chosen first. Multiple combinations of these colours were tested before deciding on the core colour for the application. It was decided that white would be used against the core colour and that monochromatic colours would be used throughout the application. The next step taken to create the mock-up was deciding on the name of the proposed application. It was important to ensure that the name of the application was both memorable and reflective of the core values of the application. After lots of brainstorming, SharingsCaring was chosen as the name of the application. This name was chosen because it reflects the purposed of the proposed application while relating to a well known colloquial expression, *Sharing is Caring*. The next step in designing the proposed application was designing the logo of the application. The logo was created using PowerPoint design, by testing different design ideas using the name and the chosen color scheme. With the color scheme, name and logo of the application chosen, the next step was to start designing the mock-up application login and register pages. The login and register pages were designed using a classic layout. The login page utilises the designed logo. The register page utilises the name of the application and the colour scheme. The mock ups of both the login page and the register page can be seen in figure 4.14.

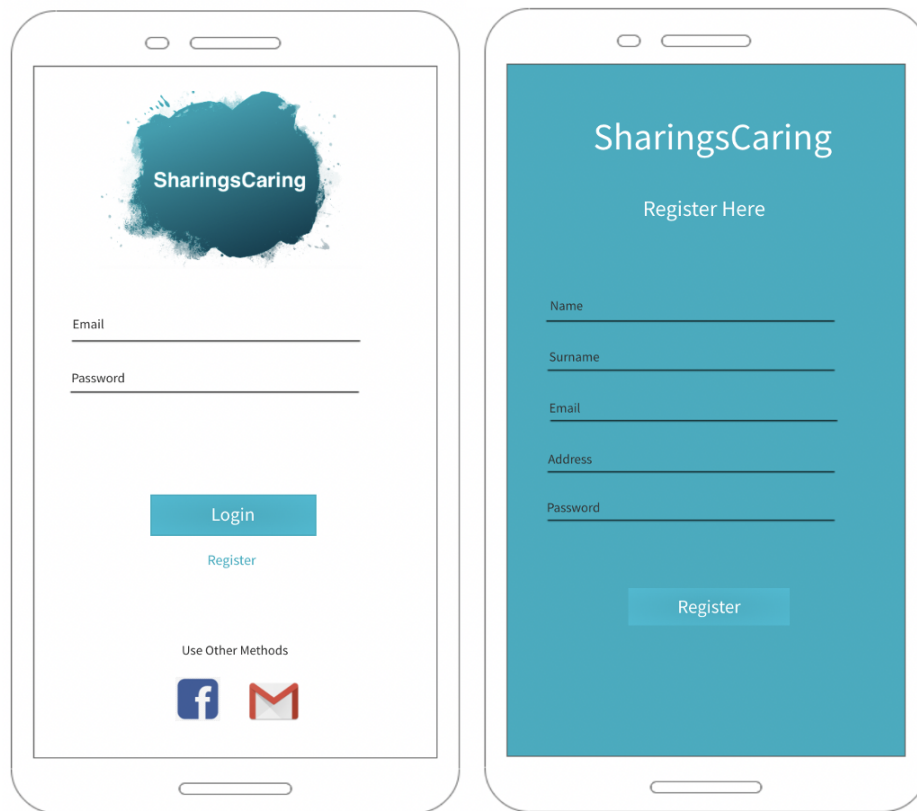


Figure 4.14: Login and Register Mock-up Pages

4.5.2 Home Page

This section discusses the design of the home page of the proposed application. The home page of the application consists of a list of either items or skills that a user can search through to find what they're looking for. The core colour is used for the search bar view at the top of the page, and a darker monochromatic version of the core colour is used for the toolbar above it. The search bar view consists of a search bar and two radio buttons that allow the user to decide whether to rank results by distance or rating. The toolbar consists of two buttons. The first will bring the user to their user profile page. The second will bring the user to the items home page or the skills home page depending on the home page they are currently on. The design of each item or skill in the list includes an image. For items the image will be of the item itself. For skills the image will be the user profile image of the user providing the skill. The item or skill name and description is also included in the design of each item or skill in the list. These are displayed in the applications core colour. Finally, depending on the chosen ranking system, the distance from the user searching or the rating of the item or skill is shown in light grey at the bottom of each result in the list. The mock up design of the home page is displayed in figure 4.15.

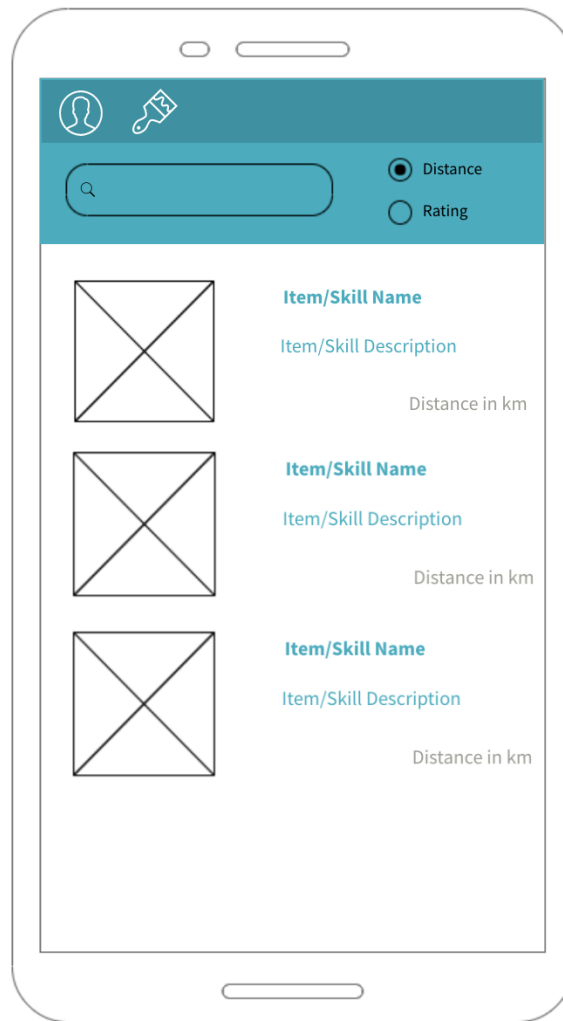


Figure 4.15: Home Mock-up Page

5 Results

This chapter discusses the results obtained from the implementation and testing of the proposed application.

Section 5.1 discusses the results of the implemented recommender system.

Section 5.2 discusses the results of the implemented user defined friendship model.

Section 5.3 discusses the results of the implemented rating system.

Section 5.4 provides a brief summary of the chapter.

5.1 Recommender System

This section discusses and displays the results of the implemented recommender system. The recommender system is a means of establishing the community-based aspect of the proposed application and satisfying the requirement of trust. The recommender system establishes the community-based aspect of the proposed application by ensuring all results shown to the users are within a two kilometre radius of their address. This encourages users to share with people within their community. The recommender system aids in satisfying the requirement of trust as users are more likely to trust a platform that is providing them with helpful and accurate results.

The testing of the implemented recommender system was done by verifying the accuracy of the results provided on a user search. To be accurate, all results should be within a two kilometre radius of the user who is searching. Figure 5.1 shows the results of a sample user search. As seen from the figure the results are accurate as all are within the two kilometre radius.

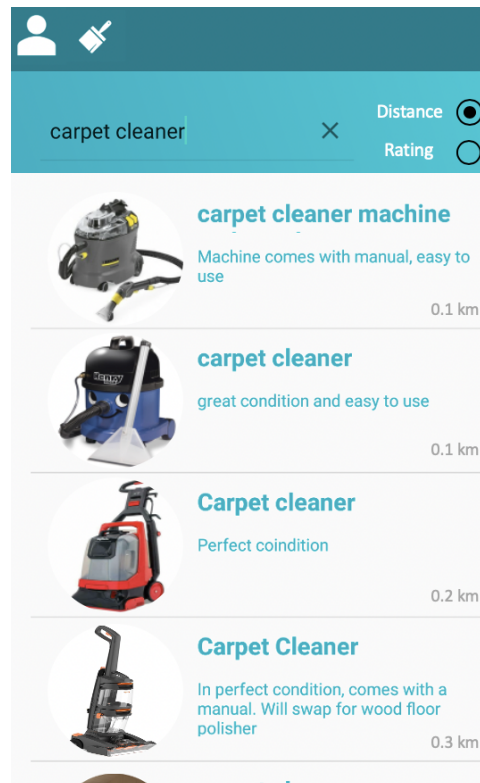


Figure 5.1: Results Ranked by Location

The default ranking of the search results should be based on the distance to the user, with the closest result ranking first and the furthest result ranking last. The accuracy of the results is important in developing trust between the users of the proposed application and the application itself. As seen from figure 5.1 the results shown are ranked correctly as the closest result is ranked first and the following results are ordered correctly based on the distance.

While the results of a search will always remain within the two kilometre radius of the relevant users location, the ranking of the results does not have to be based on the distance. The user can decide to manually change the ranking system to be based on ratings instead of distance. In this case the result with the highest rating should be ranked first and the result with the lowest rating should be ranked last. By default, any result without a rating will rank below all the other results. As seen in figure 5.2 the results shown are ranked correctly as the result with the highest rating is ranked first and the following results are ordered correctly based on their ratings.

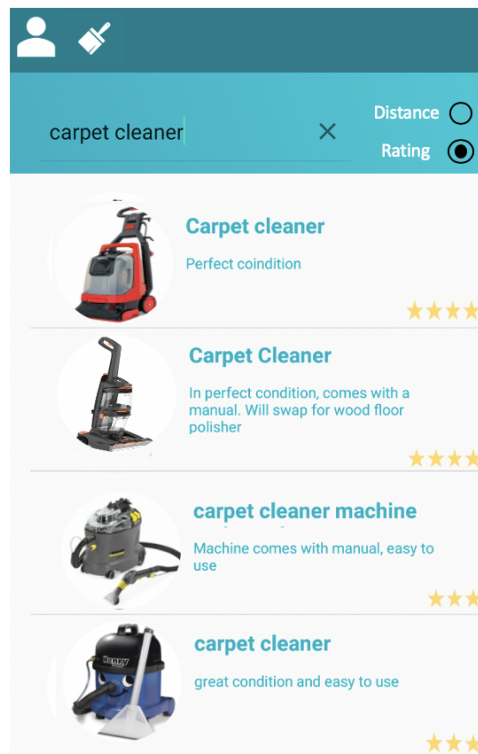
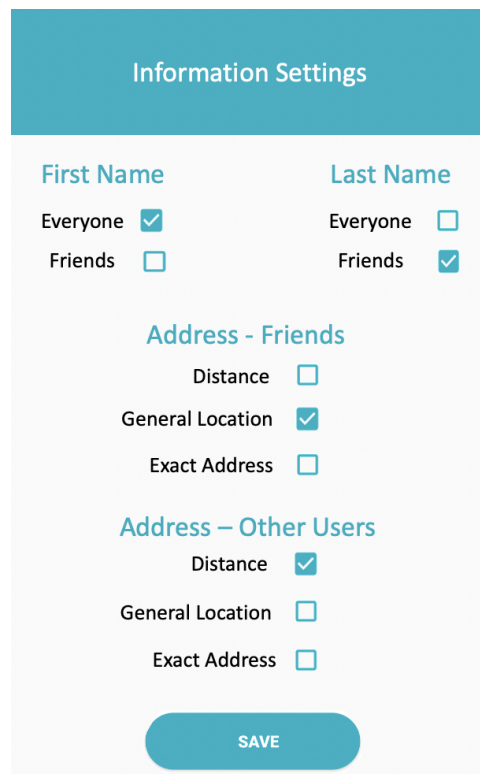


Figure 5.2: Results Ranked by Rating

5.2 Friendship Model

This section discusses and displays the results of the implemented user defined friendship model. The implemented friendship model is a means of satisfying the requirement of privacy. Users can hide certain pieces of information from other users of their choice, protecting their location and identity privacy. The user should be able to customize the information seen by users on the platform who are their *friends* and users who are not. This is done via an edit button on the users profile page. On click of this button the user will be brought to an information settings page. Here, the user should be able to customize who can see their first name and who can see their last name. The options should allow for *friends* or *everyone*. The user should also be able to customize the granularity of the location information seen by their *friends* on the platform versus the granularity of their location information seen by other users of the platform. The options are to show their exact address, their generalized location or the distance from their address to the relevant users address. Figure 5.3 shows an example of this edit page filled out. As seen from this figure, the user is able to accurately customize the information seen by their *friends* on the platform versus everyone else.



The image shows a mobile application settings screen titled "Information Settings". It is divided into two columns for "First Name" and "Last Name". Under "First Name", "Everyone" is selected with a checked checkbox and "Friends" is unselected. Under "Last Name", "Everyone" is unselected and "Friends" is selected. Below these are two sections for address sharing: "Address - Friends" and "Address - Other Users". Each section has three options: "Distance", "General Location", and "Exact Address". In the "Address - Friends" section, "Distance" is unselected, "General Location" is selected, and "Exact Address" is unselected. In the "Address - Other Users" section, "Distance" is selected, "General Location" is unselected, and "Exact Address" is unselected. At the bottom of the screen is a blue rounded button labeled "SAVE".

Information Settings	
First Name	Last Name
Everyone <input checked="" type="checkbox"/>	Everyone <input type="checkbox"/>
Friends <input type="checkbox"/>	Friends <input checked="" type="checkbox"/>
Address - Friends	
Distance <input type="checkbox"/>	
General Location <input checked="" type="checkbox"/>	
Exact Address <input type="checkbox"/>	
Address - Other Users	
Distance <input checked="" type="checkbox"/>	
General Location <input type="checkbox"/>	
Exact Address <input type="checkbox"/>	
SAVE	

Figure 5.3: User Information Settings Page

As the recommender system is location based, by default all items and skills will show the distance from the relevant user. If the user has decided to share their address with finer granularity, such as their generalized location or their exact address, this will only be shown after clicking into the individual item or skill. Figure 5.4 shows an example of an item page. In this case the user has decided to share their full name and the distance between their address and the viewers address.

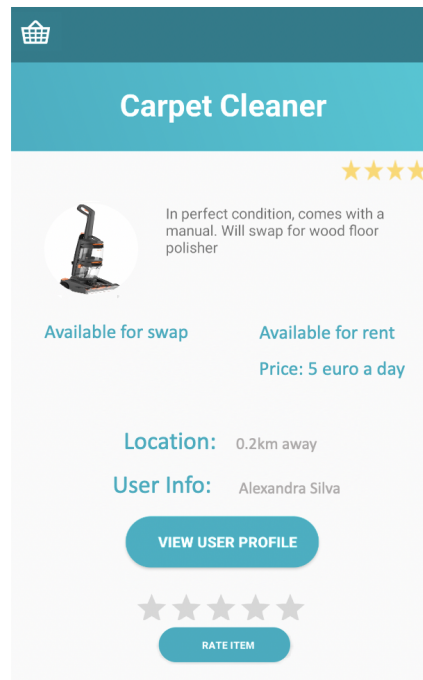


Figure 5.4: Item Page

5.3 Rating System

This section discusses and displays the results of the implemented rating system. The implemented rating system is another means of satisfying the requirement of trust in the proposed application. Users can decide what items or skills to use from the platform based on the positive or negative experiences of other users. This encourages users to trust the items and skills being shared through the platform. The user should be able to view the rating of each item and skill from the home page when ranking by rating. The user should also be able to see this rating when viewing an individual item or skill. Finally, the user should be able to give an individual item or skill a rating out of five stars. As shown in figure 5.2 the user can see the ratings of results when ranking items or skills by rating. Figure 5.4 displays an individual item page and shows that the user can both view the current rating of the item and contribute their own rating using the rating bar at the bottom of the page.

5.4 Summary

Due to extenuating circumstances involving the outbreak of the novel coronavirus, results of users experience with the proposed application could not be obtained. The results discussed in this chapter, instead relate to the degree at which the implementation of the proposed

application has reached its goals and how accurately the implemented application performs in terms of the requirements discussed in this paper.

From the results presented in this chapter it can be said that the requirement of trust has been satisfied. This is due to the accuracy of the results obtained from the implemented recommender system and rating system. It can also be said that the requirement of privacy has been satisfied. This is due to the accuracy of the results obtained from the implemented user defined friendship model.

Overall, the results presented show significant insights and achievements from the implementation of the proposed application.

6 Evaluation

This chapter presents the evaluation of the methods used to design and implement the proposed application. The architecture, application development environment, user authentication, and database will be evaluated. The evaluation considers both the positives and negatives of each method.

Section 6.1 presents the evaluation of the architecture used to build the proposed application.

Section 6.2 presents the evaluation of the application development environment chosen to develop the proposed application.

Section 6.3 presents the evaluation of the user authentication and database services chosen for the proposed application.

6.1 Architecture

This section presents the evaluation of the architecture used to build the proposed application. The chosen architecture is known as MVVM (Model View ViewModel). As discussed in section 3.4.3 the MVVM Android Studio architecture splits the code into three components, model, view, and viewModel. The model represents a set of classes that describe the business logic and data operations for the application, the view represents the user-interface components of the application, and the viewModel exposes methods, commands, and other properties that are needed to maintain the state of the view, manipulate the model based on view events, and trigger events in the view. This architecture separates business logic and data operations from the user-interface components, causing less bugs. It is for this reason that the MVVM architecture was chosen above the other architecture options.

Unfortunately, when implemented this architecture ended up being overkill for the proposed application. The user-interface of the proposed application is not exceedingly complicated. For this reason, having to implement numerous classes to do simple user-interface operations felt counter intuitive. This process added a lot of unnecessary development time, affecting the efficiency of the implementation phase. Another issue experienced regarding the MVVM architecture relates to debugging when using data binding. Data binding is declarative which made it harder to debug than the imperative style

of user-interface programming traditionally used in Android Studio applications. With the imperative style, the developer can easily add breakpoints to their code and use single stepping to understand the nature of the bug and how to fix it. With the declarative style, the developer cannot set breakpoints or perform single stepping. The debugging has to take on a trial and error nature which is inefficient and does not ensure a solution. With this architecture implemented, low complexity bugs took an unnecessarily long amount of time to solve. For these reasons, the MVVM architecture was abandoned soon after its implementation.

After this, the MVP architecture was used. As discussed in section 3.4.2 this architecture splits the code into three components, model, view, and presenter. The model represents a set of classes that describe the business logic and data operations for the application, the view represents the user-interface components, and the presenter responds to user-interface events. It uses use cases to perform tasks on model objects and passes the results to the view. Unlike the controller, the presenter is decoupled from the view and only communicates to it through the user-interface. This architecture was far more well suited to the proposed application and was very easy to debug in comparison to the MVVM architecture.

6.2 Application Development Environment

This section presents the evaluation of the application development environment chosen to develop the proposed application. The chosen application development environment is Android Studio. The use of Android Studio as the application development environment for the proposed application had both its advantages and disadvantages.

One major advantage to the use of Android Studio in developing the proposed application were the available learning resources. These resources include the Android developer guides, Codelabs, and guided courses. Out of these the Android developer guides were used during the development of the proposed application and proved to be very helpful. The guides cover every topic referring to the development of an Android Studio application and provide solutions to common problems faced by Android developers. Another advantage to using Android Studio was the intelligent code editor. This editor offers advanced code completion, refactoring, and code analysis. When typing Android Studio displays numerous intelligent suggestions in a dropdown list. The developer can then simply click or press tab to insert the code. This allowed for better code to be written in the proposed application. It also facilitated faster and more efficient application development.

The disadvantages of Android Studio that became clear during the development of the proposed application all relate to the performance of the emulator. Firstly, the performance

of the Android Studio emulator was incredibly slow. Having to wait for the emulator to run the application added a lot of unnecessary time in the development and testing of the proposed application. Another issue was that the emulator showed numerous bugs that did not exist when running the application on a physical device. This was a problem as a lot of development time was spent trying to fix a bug in the application code that was actually being caused by the emulator itself. Finally, the emulator crashed numerous times and would only work again if a full deletion and reboot of the emulator was done. Again, this wasted a lot of development time and caused undue stress.

Overall, Android Studio itself performed well during the development of the proposed application. The issues faced all related to the use of the emulator. For this reason, if the proposed application were to be developed again, Android Studio would still be used but the emulator would not.

6.3 Firebase

This section presents the evaluation of the user authentication and database services chosen for the proposed application. Firebase was chosen as a third party service provider for both the user authentication and the database.

Section 6.3.1 presents the evaluation of the Firebase user authentication service.

Section 6.3.2 presents the evaluation of the Firebase database service.

6.3.1 User Authentication

This section presents the evaluation of the Firebase user authentication service. As discussed in section 4.2.1 Firebase User Authentication was chosen for the proposed application due to its ability to work seamlessly with other Firebase services and its clean implementation process. The experience of using Firebase User Authentication was positive overall. The service was simple enough to implement and did not cause any bugs. The service was also easily customized and allowed for the use of Facebook and Google to authenticate users. The integration of the user authentication service with the Firebase database was also seamless and did not cause any issues.

6.3.2 Database

This section presents the evaluation of the Firebase database service. As discussed in 4.2.2, Firebase offers two separate database services, Realtime Database and Cloud Firestore. Cloud Firestore was chosen for the proposed application due to its additional features such as its ability to scale further than Firebase Realtime Database. The experience of using

Cloud Firestore as the application database was positive overall. The service itself did not cause any issues during development. The service was accessible and easily customized for the needs of the proposed application. Cloud Firestore is serverless, meaning there is no need to set up a server to manage access to your data. This allowed for a seamless experience during implementation. This serverless nature also meant that Cloud Firestore was perfect for developing the proposed application quickly.

In conclusion, Firebase was an ideal third party service to use during the development of the proposed application.

7 Future Work

A sharing economy application that facilitates both item and skill sharing was successfully designed and developed during this study. However, despite the promising results obtained from implementation there is still more work to be done to both improve upon the application and to implement additional services. This chapter discusses a number of identified opportunities for future work.

Section 7.1 discusses a possible alternative database that could be implemented in the future to improve the proposed application.

Section 7.2 discusses the possibility of implementing a graph database to store the friendships of application users in the future.

Section 7.3 discusses the necessary implementation of a messaging service for the proposed application in the future.

7.1 Database

This section of the paper discusses an alternative option for the application database that could be implemented in the future. In this study the Firebase Cloud Firestore database was used. This database was perfect for the prototype of the proposed application as it is free to use and can be implemented quickly and with ease. However, a different database would have to be used if the application were to be launched in the future. There are many reasons for this. Firstly, Firestore places a limit of 1MB on the amount of data that can be stored in a single document. This may become an issue for the storage of user friendships in the future. Another possible issue with Firestore in the future involves its document write frequency limit. Firestore limits document writes to 1 per second. This limit restricts the databases ability to scale under load and could consequently lead to failed database transactions. To solve these problems a database such as the Amazon Relational Database Service (RDS) could be used in the future.

Amazon RDS allows for both horizontal and vertical scaling. Vertical scaling enables handling of a higher load in your database. It is implemented by increasing your instance type. Horizontal scaling improves the performance of a read-heavy database. It is done by using read replicas and adding a load balancer between the application and the servers (30). RDS also provides Amazon CloudWatch metrics for database instances at no additional

charge. The RDS Management Console can be used to view key operational metrics, including compute, memory, and storage capacity utilization, input and output activity, and instance connections. The availability of these metrics would be necessary if the proposed application were to be launched in the future. Finally, Amazon RDS provides a cost efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups (31). To conclude, Amazon RDS would be an appropriate replacement for the database of the proposed application in the future. There are several reasons for this including, its enhanced scalability, capacity, security, and additional metric features.

7.2 Graph Database Structure

This section discusses the possible future implementation of a graph database to store the friendships of the application users. Graph databases are commonly used in social networks applications. This is because social networks themselves are graphs, making a graph database a perfect fit. A graph database differs to a relational database as it stores not only the data (nodes) but also the relationships between the data (edges). The data, in this case the users, would be described as the nodes and the relationships, in this case the friendships between users, would be described as the edges. Any related nodes, such as users who are *friends* are physically linked in the database, making the relationships between nodes just as accessible as the data itself. A major benefit of implementing a graph database to store the friendships of users, is the friend of a friend query scenario. In a relational database it is quite challenging to deal with this type of query where each traversal along an edge in a graph database would need a high cost join in a relational database. Performing this query in a graph database is much more efficient as traversals across edges are low cost (5). Figure 7.1 shows an example of an inexpensive solution to a friend of a friend query in a graph database.

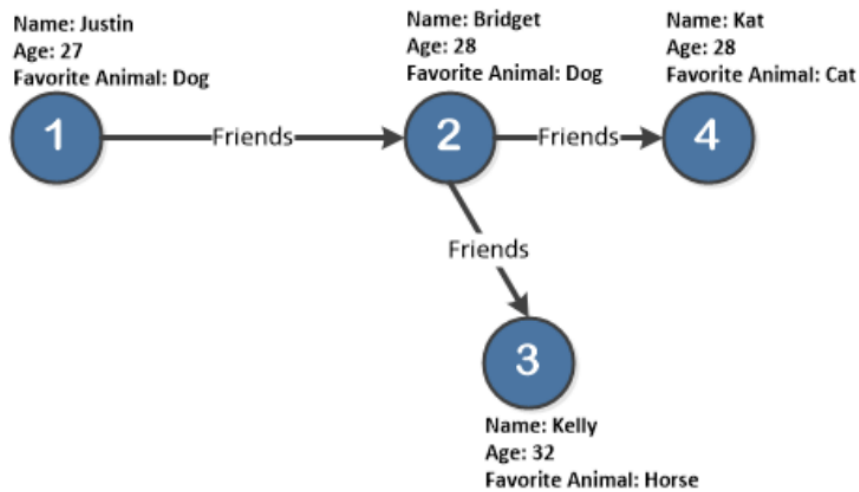


Figure 7.1: Graph Database Friend of a Friend Query (5)

Being able to easily perform a friend of a friend query is important as it allows for the implementation of a valuable feature that is common in social network applications. This feature involves recommending *friends* to users. This recommendation is based upon the *friends* of the *friends* they have on the platform.

7.3 Messaging Service

This section discusses the future implementation of a messaging service in the proposed application. A messaging service is essential for the proposed application due to its community based nature. Community based applications aim to bring people together and this is often done by allowing them to collaborate and interact through a messaging service (32). For this reason, a messaging service must be included in the future work of the proposed application. There are many ways to implement an instant messaging service. One of which involves the use of Ejabberd and Erlang.

An instant messaging service requires a server such as Ejabberd and a programming language such as Erlang. Ejabberd is a free and open source instant messaging server. The server is a robust, scalable and extensible XMPP server. Erlang is a programming language used to build scalable real-time systems with requirements on high availability. One of its many uses include instant messaging. Erlang uses concurrent processes to structure its programs. These processes have no shared memory and communicate through asynchronous message passing. Each process has a unique identifier to address the process and a message queue to store the incoming messages (33).

8 Conclusions

Although it is clear that this study would have benefited from additional time where, additional features such as the messaging service could have been implemented, the application could have been tested more thoroughly with the use of actual users and more results could have been obtained, there was still a significant amount of knowledge gained from the implementation, testing and evaluation of the proposed application thus far. This chapter provides the significant achievements and findings drawn from this study.

8.1 Community Based Application

One of the most significant findings from this study is the importance of the community-based aspect in a sharing economy application for it to be successful. Many of the existing sharing economy applications researched in this study were community based. These applications all seemed to sing high praise to the community-based aspect of their applications, in establishing trust between their users. It was found that users were more likely to share with people they thought of as *neighbours*. It was also found that the number of users on the platform grew more through word of both when using a community-based structure. It is the opinion of the author that ensuring the community-based nature of the proposed application is essential for it to be a success. In the case of the proposed application, this community-based aspect was successfully implemented through the use of a location based recommender system.

8.2 User Defined Friendship Model

The implemented user-defined friendship model is a significant achievement of this study. The friendship model allows users to customize the information seen by users on the platform who are their *friends* and users on the platform who are not. One major challenge faced when developing the proposed application was ensuring that user privacy be maintained while still showing enough user information to encourage trust between users. This model solves that challenge. By default it shows enough information to encourage trust while still maintaining a high level of privacy. The user can then decide themselves if they wish to share further information with users they deem trustworthy.

In conclusion, the proposed application has been successfully designed, developed and

tested. While there is still work to be done in the future in order to facilitate the launch of the proposed application, the work done thus far has demonstrated numerous insights and achievements.

Bibliography

- [1] Statista. Ireland consumer spending, . URL <https://tradingeconomics.com/ireland/consumer-spending>.
- [2] Marton A. Constantiou, I. and V.K Tuunainen. Four models of sharing economy platforms. *MIS Quarterly Executive*, 16(4):231–248, 2017.
- [3] A.R. Beresford and F Stajano. Mix zones: User privacy in location-aware services. *In IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second*, 2004.
- [4] T Lou. A comparison of android native app architecture–mvc, mvp and mvvm. *Eindhoven University of Technology*, 2016.
- [5] J.J Miller. Graph database applications and concepts with neo4j. *In Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA*, 2324 (36), 2013.
- [6] G.A Frantz. Consumerism, conformity, and uncritical thinking in america. 2000.
- [7] Lotze H.K. Jubinville I. Wilcox C. Worm, B. and J Jambeck. Plastic as a persistent marine pollutant. *Annual Review of Environment and Resources*, 42:6, 2017.
- [8] Stadler K. Steen-Olsen K. Wood R. Vita G. Tukker A. Ivanova, D. and E.G Hertwich. Environmental impact assessment of household consumption. *Journal of Industrial Ecology*, 20(3):530–532, 2016.
- [9] J Schor. Debating the sharing economy. *Journal of Self-Governance and Management Economics*, 4(3):2–12, 2016.
- [10] J. Wallenstein and U Shelat. Hopping aboard the sharing economy. *Boston Consulting Group*, 22:2–5, 2017.
- [11] PwC. Pwc irish retail consumer report: Investing in experience. page 3, 2019.
- [12] Cone. Cone communications csr study. pages 7–10, 2017.
- [13] Statista. Revenue of platform providers in the sharing economy worldwide in 2017 and 2022, . URL <https://www.statista.com/statistics/878844/>

global-sharing-economy-revenue-platform-providers/
#statisticContainer.

- [14] C. Codagnone and B Martens. Scoping the sharing economy: Origins, definitions, impact and regulatory issues. *Institute for Prospective Technological Studies Digital Economy Working Paper*, 1:3–24, 2016.
- [15] Anwar M. Fong, P.W. and Z Zhao. A privacy preservation model for facebook-style social network systems. *In European Symposium on Research in Computer Security*, 2009.
- [16] Olio. About olio. URL <https://olioex.com/about>.
- [17] V Jenora. Introducing anytimes, japan’s hottest skill-sharing app. URL <https://hivelife.com/anytimes-skill-sharing-app>.
- [18] ANYTIMES. Anytimes: We believe in neighbours helping neighbours. URL <https://www.any-times.com/en>.
- [19] Agriculture Consumers, Health, Food Executive Agency (Chafea) on behalf of Directorate-General for Justice, and Consumers. Exploratory study of consumer issues in online peer-to-peer platform markets. task 4 – case study: Peerby. 2017.
- [20] Peerby. Peerby. URL <https://www.peerby.com>.
- [21] Airbnb. Id and verification, . URL <https://www.airbnb.ie/help/article/1308/does-airbnb-perform-background-checks-on-members>.
- [22] Airbnb. Privacy policy, . URL https://www.airbnb.ie/terms/privacy_policy.
- [23] Airbnb. Terms and policies: Fees, . URL <https://www.airbnb.ie/help/article/1857/what-is-the-airbnb-service-fee>.
- [24] Uber. Safety, . URL <https://www.uber.com/ie/en/ride/safety/>.
- [25] Uber. Uber privacy notice, . URL <https://www.uber.com/legal/en/document/?country=united-states&lang=en&name=privacy-notice>.
- [26] Uber. How uber’s dynamic pricing model works, . URL <https://www.uber.com/en-GB/blog/uber-dynamic-pricing/>.
- [27] JetBrains. IntelliJ idea. URL <https://www.jetbrains.com/idea>.
- [28] Android Developers. Android studio. URL <https://developer.android.com/studio>.

- [29] MockFlow. Mockflow. URL <https://mockflow.com/>.
- [30] Herbst N. Ivansek S. Brataas, G. and J Polutnik. Scalability analysis of cloud software services. *In 2017 IEEE International Conference on Autonomic Computing (ICAC)*.
- [31] S Mukherjee. Benefits of aws in modern cloud. *Available at SSRN 3415956*, 2019.
- [32] Salgado L.C.D.C. Afonso, O.P. and J Viterbo. User's understanding of reputation issues in a community based mobile app. *In International Conference on Social Computing and Social Media*, 2016.
- [33] J Armstrong. Communications of the acm. *ACM*, 53(9), 2010.
- [34] Abdullah S. Amin F. Fahmi, A. and A.J.P.U.J.M Ali. Weighted average rating (war) method for solving group decision making problem using triangular cubic fuzzy hybrid aggregation. *Punjab Univ J Math*, 50(1), 2018.