

USAMA MALIK  
Student Number:  
33551485

# Computing Project:

## BarCard

IS53007D: COMPUTING PROJECT  
SUPERVISOR: DAN MCQUILLAN

## Table of Contents

<b>Abstract .....</b>	<b>3</b>
<b>Acknowledgements .....</b>	<b>4</b>
<b>Introduction .....</b>	<b>5</b>
<b>Aims And motivation .....</b>	<b>6</b>
Aims .....	6
Motivations.....	6
<b>Use Cases Scenarios.....</b>	<b>7</b>
Scenario 1 .....	7
Scenario 2 .....	7
<b>Background Research .....</b>	<b>9</b>
Existing Technologies.....	9
LinkedIn .....	9
Snapchat .....	9
<b>Technology Research .....</b>	<b>10</b>
QR Codes.....	10
Business Cards.....	10
Android Development Tools.....	11
Database.....	12
Technology Research Conclusion .....	13
Back-end, Firestore – .....	14
<b>Required Tools.....</b>	<b>15</b>
<b>System Design .....</b>	<b>16</b>
System Requirements .....	16
App Flow.....	17
<b>Relevant App screenshots.....</b>	<b>19</b>
<b>Implementation.....</b>	<b>22</b>
Flutter Packages Used.....	22
Front end Implementation .....	26
<b>Testing.....</b>	<b>36</b>
Black box testing .....	36
White box Testing.....	38
<b>User Testing.....</b>	<b>40</b>
<b>Self-Evaluation .....</b>	<b>45</b>
<b>Conclusion .....</b>	<b>44</b>
Aims and Objectives achieved / failed .....	44
Project Future .....	44

## Abstract

BarCard is a multiplatform mobile application that allows for local, social or professional networking with the use of QR linked “E-Business cards”. Individuals can sign up and create a card that can be shared with other individuals using a unique QR code linked to the card. Once the card is scanned it gets saved in the history. Cards can also be added to a favourite card section. Users can also see cards near their vicinity which can help network with locals in the area.

The app is implemented using the Flutter framework, which is an open-source UI software development kit created by Google. To accompany the front-end, it seemed obvious to use Google’s Firestore back-end.

## Acknowledgements

Firstly, I want to thank my parents for all the support, and I appreciate your efforts and love in bringing me up to be a better individual.

I also want to thank Dan McQuillan, my project supervisor. I feel very grateful to have had Dan as my supervisor and am thankful for his guidance and advice which helped streamline my project. Thank you for all your support, encouragement, and motivation.

## Introduction

In a research in the United States it was found that 23% of adults don't know their neighbour<sup>1</sup>, let alone individuals in their area. Imagine the variety of professional individuals that could be in your near vicinity, teachers, doctors, accountants, lawyers, engineers, designers or maybe even a Multimillion company Director. How valuable would it be if you could connect with them and get some advice or guidance for your professional Career.

Being a student myself and talking to fellow students and new graduates, I have numerous times heard the line "Only if I knew before!". One of the biggest complaints about graduates is that they didn't or couldn't talk to anyone working in their line of study or work due to either not personally knowing anyone in that line of work or not having the means to connect with someone near them.

I plan to provide a proposed solution for this problem by providing a mobile application that allows users to contact individuals near their vicinity, and anyone in general. This will be done through the concept of E-Business cards. Each user will have a card with their details on it and a unique QR code linked to it. The purpose of the barcode will be so it can easily be shared when physically networking, the other individual has to just scan the QR code and it will automatically add the card.

---

<sup>1</sup> <https://www.pewresearch.org/fact-tank/2019/08/15/facts-about-neighbors-in-u-s/>

## Aims And motivation

### Aims

My project aims to provide an easy networking tool that stems from the traditional methods of Business cards. The concept of business cards is very well known to most individuals so the learning curve to use the BarCard will be quite easy. An individual may use this tool to broaden their networking links, or for individuals simply seeking for employability guidance or referencing to a workplace.

### Motivations

The main motivation was from my personal experiences and complaints about specifically networking for students. Knowing the necessary steps to take to be able to have a jumpstart in your early career can have a massive positive difference before graduation.

To some extent, the current most helpful guidance provided to students are their teachers, Lecturers or module leaders. This is a very reliable and good way to get guidance and professional help. In most universities in the UK they also assist up to 2 years to postgraduates, in most cases this is enough help and guidance for most students. However, for individuals who have already graduated or individuals that don't have access to tools provided by educational institutes, it is very difficult to find an individual in the same line of study or work to be able to network with.

One experience that motivated me to pursue this project was when I met a gentleman waiting for the bus stop on my commute to uni. I had seen this individual almost every day on my commute to Uni and once overheard a conversation to do with something about software development. One day I had the courage to speak to the individual, this was during a time when I was applying to Internships during the penultimate year of my studies. The gentleman was very helpful and even told me about an opportunity at his workplace.

After this interaction I was thinking of all the possible different individuals near me, working in a related workplace to my studies, that may be able provide me with guidance or opportunities that otherwise I will never know about.

One may argue that why would a stranger want to help an inexperienced graduate. However, a lot of companies provide an incentive to employees if they refer someone to a graduate scheme or employment scheme in their workplace. For example, a bonus may be given to the employee if the referred individual passes the recruitment stage and goes on to work in the company. This means that some individuals may want to help out a graduate by referring them (after getting to know them) to their workplace and also receive some kind of benefit.

## Use Cases Scenarios

Here I will provide two scenarios of two different individuals and how they may want to use my app.

### Scenario 1

A student still in the first year of their Fashion Design degree may want to use my app to search for a professional designer in their area to be able to gain mentorship from them, or maybe even ask the professional if they are willing to assist them in a project to be able to strengthen the student's portfolio.

This would be done by searching in the nearby section of my app and seeing if there is a fashion designer in their near vicinity. If there is, the student can use the email button to directly contact the professional designer.

### Scenario 2

An Entrepreneur that has just started their own business for the first time. They are still new in the market and don't have much experience. They may want to use my app to create an E-business card to be able to easily share with individuals who are interested in the product or business. They can also make their card public so other users can view their card and contact them with opportunities.

Other users can contact the new business owner to establish a contact network and later potentially enquire about large orders or potentially talk about large business deals.

## User Personas

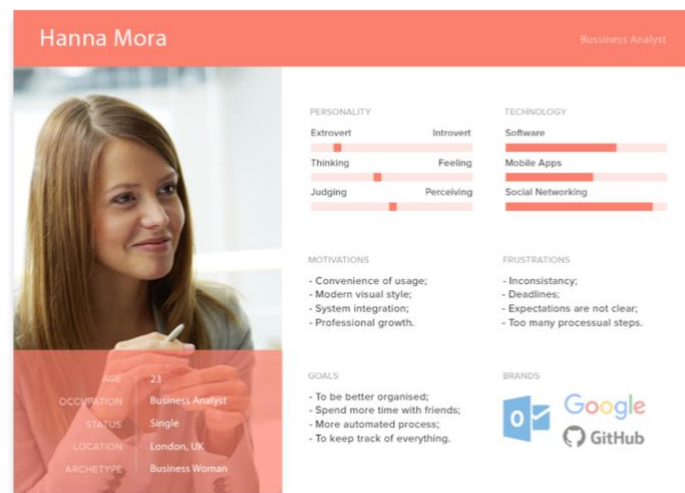


Figure 1 User Persona 1

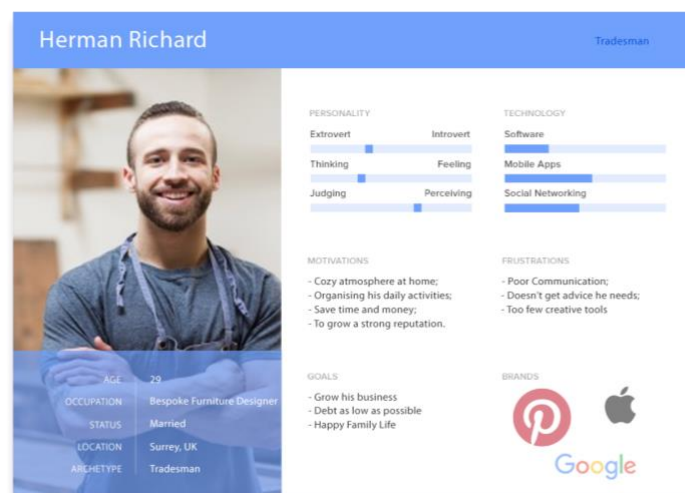


Figure 2 User Persona 2

The user personas were created by me using a photoshop template<sup>2</sup>

<sup>2</sup> <https://dribbble.com/shots/3163600-Free-Persona-Templates>



## Background Research

### Existing Technologies

#### LinkedIn

LinkedIn is a web platform that provides employment-oriented services and social networking that focuses on professional networking and career development. LinkedIn has almost 700 million users worldwide<sup>3</sup>. Its main services are job postings by employers and job seekers posting their CV's.

The reason I am including this in existing technology is because it is the biggest platform for professional networking. It is really popular for mostly employment reasons; however, it isn't much use for individuals trying to get employability advice or profession guidance. My project will be better in terms of this because individuals will be able to directly network with experienced individuals near their vicinity.

#### Snapchat

Snapchat is a very popular multimedia messaging platform; it is especially popular amongst the younger demographic (13 – 29 years old)<sup>4</sup>. Its main feature is that any message, it being a text, image or video, is only available to view for a short period, meaning once the message has been viewed the message cannot be viewed again.

Snapchat has a very unique way to add friends. One way is to simply search their username. The more interesting way is by scanning the friends "Snapcode". A Snapcode is a unique QR code that every Snapchat user has. When an individual scans another person's code using the Snapchat app, the corresponding user profile pops up which allows for easy friend request. In late 2016, a quarter of all friend add method came from Snapcodes<sup>5</sup>. Making it a popular feature.

This process is very intuitive and involving. Physically scanning the code and instantly being able to add someone in one step makes it so much easier than giving out your username and then searching the username whilst making sure you are spelling it correctly.

I used this intuitive feature as an inspiration to make the QR codes on the E-Business card for easy and intuitive sharing.

---

<sup>3</sup> <https://news.linkedin.com/about-us#1>

<sup>4</sup> <https://www.smartinsights.com/social-media-marketing/social-media-strategy/snapchat-statistics/>

<sup>5</sup> <https://marketingland.com/snapchats-snapcodes-decline-getting-brands-new-followers-eclipsed-deep-links-221355>

## Existing Technology Conclusion

The idea that I proposed for this app, is very unique and nothing similar exists in the market, thus only having two existing technologies to analyse, LinkedIn and Snapchat. These are the closest existing technologies I could find because they have certain features that are similar to what I implemented.

LinkedIn is very good for job hunting and creating an employment portfolio for employers to be able to see, however, it is mostly used for professionals that already have some experience. It cannot be used to gain experience, advice or gain valuable links for individuals that are inexperienced or want to increase their networking spectrum.

My app will allow this because it is more localised and allows for potentially physically meeting individuals after contacting them, because users will know that the person they are talking to is within the specified radius of their vicinity, for example within 1 mile or 5 miles.

## Technology Research

### QR Codes

QR Code, short for quick response code, is a type of matrix. Meaning that it is two dimensional. It consists of black squares arranged on a square label with a white background. It is easily and quickly readable with an imaging device such as a camera or smartphone and is processed using a method called Reed-Solomon error correction. This same error correction method is also widely used in technologies such as DVDs and satellites.

The data from a QR code is extracted by reading both the vertical and horizontal patterns. A QR code can store up to 3kb of data<sup>6</sup> which is enough to store 4296 Alphanumeric characters. Knowing this, I could store all of the E-business cards detail within the QR code itself, if I wanted to. However, I used the users' Unique ID to create the QR code, so once it is scanned the app gets the card information from the database using the unique id from the scanned QR code.

### Business Cards

Business cards have been used since as early as the 17<sup>th</sup> Century<sup>7</sup>, when in Europe, gentlemen would give out their card to essentially self-promote themselves.

In the modern day, the core concept hasn't changed much, they are still used as a formal introduction as a convenience and a memory aid. Worldwide over 27 million business cards are printed daily<sup>8</sup> and still hold an important part in the business world, even with the rise of smartphones and the internet.

---

<sup>6</sup>

<http://qrcode.meetheed.com/question7.php#:~:text=Standard%20QR%20Codes%20can%20hold%20up%20to%203Kb%20of%20data.>

<sup>7</sup> [https://en.wikipedia.org/wiki/Visiting\\_card](https://en.wikipedia.org/wiki/Visiting_card)

<sup>8</sup> <https://www.creditdonkey.com/business-card-statistics.html>

In Eastern-Asian countries there are even very strict business card etiquettes, for example cards should be handed with two hands as a sign of respect<sup>9</sup>, and cards are not to be shoved in the back trouser pocket, as it is a sign of disrespect.

There are different sizes of business cards in different countries, for example in the UK the size of a typical business card is 85mm by 55mm, however the size of a typical business card in Japan is 91mm by 55mm<sup>10</sup>.

## **Android Development Tools**

Traditionally Android development happened natively with the tools Google provided for their Android OS. When programming natively the code would need to be written in either Java or Kotlin. A developer had to start from scratch with only a few UI elements and other features provided by Google development kit. However, most UI would need to be created from scratch which can be quite difficult and increases the time for development considerably. One of the biggest benefits for native Android is performance, because it is closer to the OS which means multithreading can be utilised, for example running UI and animation on a different CPU thread than the thread used for data logic.

Frameworks such as React, or Flutter provide the user with basic building blocks which can be put together to create highly complex applications. These frameworks also provide animation engines to make the final product looking highly professional and smooth. These frameworks are highly dependent on plugins or packages created by their community. These can be used in a project to further reduce the development time as they provide reusability in the code. Big corporations such as Uber, Facebook and Airbnb use the React Native framework to develop their complex mobile application<sup>11</sup>.

---

<sup>9</sup> <http://www.asianbusinesscards.com/business-card-exchange-etiquette-guide-culture-tips-for-asia/#:~:text=Asian%20translated%20business%20cards%20are,in%20a%20business%20card%20case.>

<sup>10</sup> [https://en.wikipedia.org/wiki/Business\\_card](https://en.wikipedia.org/wiki/Business_card)

<sup>11</sup> <https://insights.daffodilsw.com/blog/10-amazing-apps-that-are-built-using-react-native>

## **Database**

A robust database system is extremely critical to a mobile application, for it to be as secure and reliable as possible. There are two different technologies to create a database;

**SQL** – SQL based database is relational database management system, meaning that tables and records can have a relation with other records or tables, to create a complex system of linked data. In a SQL database, many records can be queried with just a single command which makes it advantageous for large databases.

**noSQL / Document Based** – This is essentially the opposite of a SQL Database because it is not a relational database. It essentially is a semi-structured, hierarchical collection of data which can be represented with a tree structure. A very popular document database structure is the JSON format. In this format you can have an object with children for data storage, however the child can also hold more children, creating a tree with branches. This type of database is used in systems where a high number of unique documents exist.

## Technology Research Conclusion

After the research I came to a conclusion for which technologies I needed to use. The technologies I used for the implementation are;

Front-End, Flutter -

For the front-end development I used the Flutter framework. Flutter is a Google created framework for creating both Android and IOS applications with one codebase. Flutter uses the Dart programming language which is an object-orientated programming language which I am very familiar with due to having previous projects using Java, Python JavaScript. Being familiar with this type of programming language meant I already knew the basic foundation knowledge and it took me a short time to learn the slight differences such as different syntaxes.

Flutter provides a very powerful UI and Animation engine, which makes the user experience as close to native Android as possible. Flutter also provides a vast library of plugins and packages for all sorts of developmental needs, packages such as a native image picker or permission handlers, if coded from scratch would've taken up a considerable amount of time.

Flutter's development process mainly consists of "Widgets", these are essentially classes that can be reused. However, in these classes they have a build function which means when this widget is being called it is able to show UI on the screen. Widgets can also be created from scratch if you require something custom and reusable. The structure of the build function is something unique to Flutter. A widget can have a child or many children, meaning that for example an AppBar widget can have a Text widget as its child, and the AppBar can use that however it wishes, in this case it uses it to show the AppBar title. Please look at figure 1 for basic visualisation of the typical Flutter code structure.



Figure 3 - Flutter Widget Tree

## Back-end, Firestore –

For the backend I have decided to use a document-based database management system. The advantage of using a document-based database rather than a SQL database, is that it makes it easier for me to store and query data in the database because it uses the same format as the programming language code, meaning I can essentially design the database as I'm implementing the front-end. If I had used a SQL database, I would have to first design the database schema first and then integrated it in the app which would've added more time to the implementation.

I used the Firestore database, which comes with Googles development platform, Firebase. Firestore is a highly flexible and scalable database system, specifically designed for use mobile applications, although it can also be used in web applications.

Querying data with Firestore is fairly straight forward, each document has an ID, which by default is always unique, however a document can be set a specified ID manually, when storing the document in the database. This is perfect in my case as I can have a user document with a unique ID for each user. And then use that unique id for their QR code, so when the code is scanned, I can have some logic to query for that unique id and then show the users corresponding card.

In one instance I also required some documents to have a specified ID, for example when I store a card in the recent history collection when the user scans a card, in the collection the record of the recently scanned card has the same ID as the scanned cards user's unique ID. This is so I can just use that documents ID instead of first having to refer to the user's unique ID.

## Required Tools

### Hardware

- Computer / Laptop with basic programming capabilities
- Android Smartphone for development and testing (preferably 2 smartphones to test QR scanning feature)

### Software

- Android Studio for Flutter development
- Android Emulator for instant UI visualisation and quick and easy on the go testing
- Adobe Creative Suite for prototyping and some UI designing
- Dart and Flutter SDK and Android studio integration plugin

## System Design

In this section I will be talking about how I went about designing the functionality of the app.

### System Requirements

#### MVP Requirements (Most important to least)

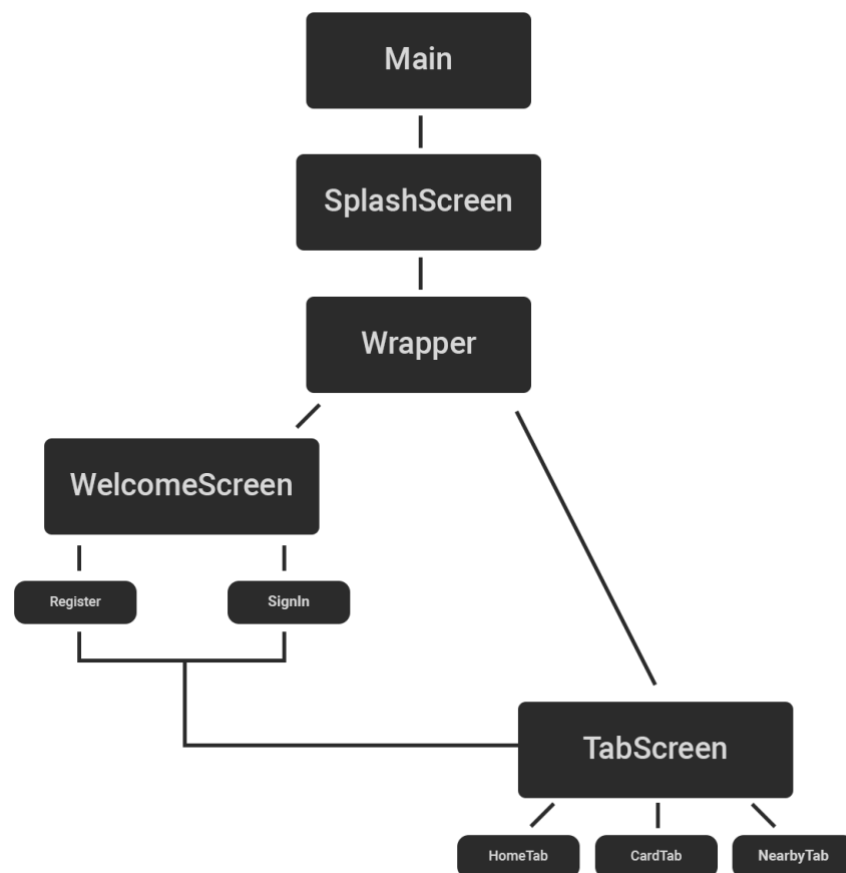
1. Be able to create an e-business with a unique QR code
2. Be able to scan a QR code
3. When a QR code is scanned it should show the corresponding card
4. Be able to see Cards near your vicinity
5. Change personal information

#### Extra Requirements (Most important to least)

1. Be able to contact a cards User
2. Be able to design the e-business card (background, text etc)
3. Be able to view scanned history
4. Add cards to a favourite section
5. Be able to have a services store for a card
6. Have a Web application so scanning QR code without the app takes you to the web app. So, users can still see the card without downloading an app.



## App Flow



*Figure 4 Diagram to show how the app is supposed to flow*

Figure 2 shows how the app is designed to flow from screen to screen. Firstly, the root of the app is the main class. This function shows you the Splash Screen which runs for a set time, then it runs the Wrapper class. The wrapper class checks if a user is logged in or not. If a user is not already logged in it takes the user to the welcome screen. The welcome screen shows the user a quick tutorial on how the app is intended to be used. After the user has seen the welcome screen it takes the user to the sign in page where they can sign in. Or if they don't have an account they can register also. The forget password section also exists here however it is not mentioned in the diagram as it is not a separate screen, rather it is a dialogue popup.

If a user is already signed in or after a user registers it takes you to the tabScreen which is the screen where a bottom navigation tab exists and the user can switch between 3 tabs, the HomeTab where the recent card history exists, the cardTab where the user's card can be customised and the nearbyTab where the card in the near vicinity of the users are shown.

## **Privacy and Transparency**

One big topic I always kept in mind when implementing and designing my app was privacy. In this age of high awareness of internet privacy, I felt it was highly necessary to try to be as transparent as possible with features in my app. My app contains 2 features which have to be highly transparent in terms of privacy. The first being the camera app, and the second the use of GPS location tracking.

Although the Android operating system does a very good job in allowing the user to choose if the app can have access to certain aspects of the phone. I still need to make sure I explain to the user why the GPS or camera is being used if they deny permission.

At first, I was planning to implement my nearby section differently. I was planning to implement it so it is a map view, and each card's exact location can be seen on the map. After discussing this plan with Dan McQuillan, my supervisor, he pointed out that this is a huge privacy problem as someone could see exactly where the card owner lives. He recommended that I try a different approach, thus being the reason I came up with my current solution. Where a card is shown within a minimum of a 1-mile radius of the current user, this way an exact location of the card is not revealed.

There is also an option in the card customisation section where the user has an option to make the card public or not. By default, every user that signs up, their card is not public. This means that if the card is not public, it will not show their card to anyone in their near vicinity, although they can still share their card if someone physically scans their QR code.

If for example a person has scanned my cards QR code in the past, and I have set my public status to false. They will not be able to see my QR code anymore. This is so they cannot share my card to anyone else. So, when the card is made not public, the only way to get their card is by asking them physically if they can scan their QR code. However, this doesn't prevent someone from taking a screenshot of the card and sharing it to someone else via a different communication method.

## Relevant App screenshots

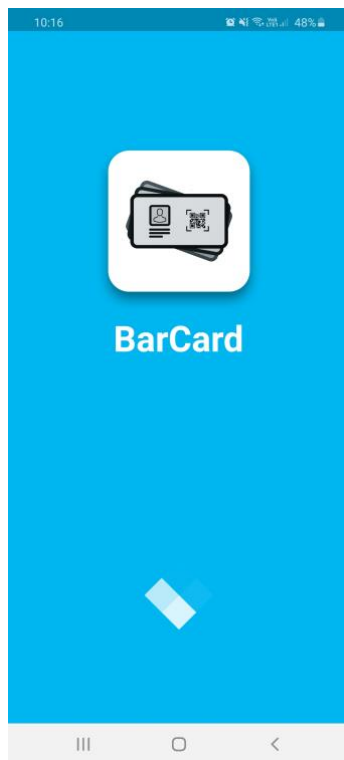


Figure 6 SplashScreen

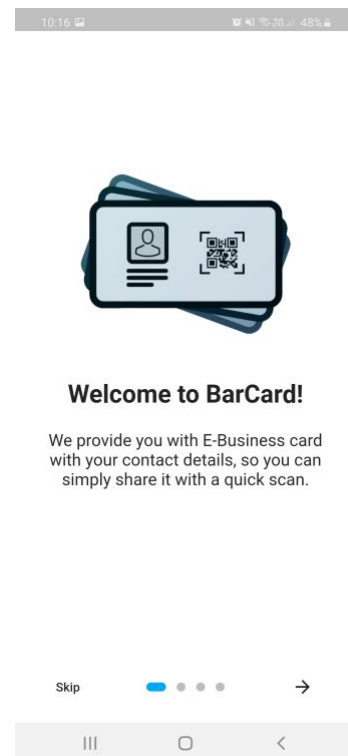


Figure 5 Welcome Screen before login

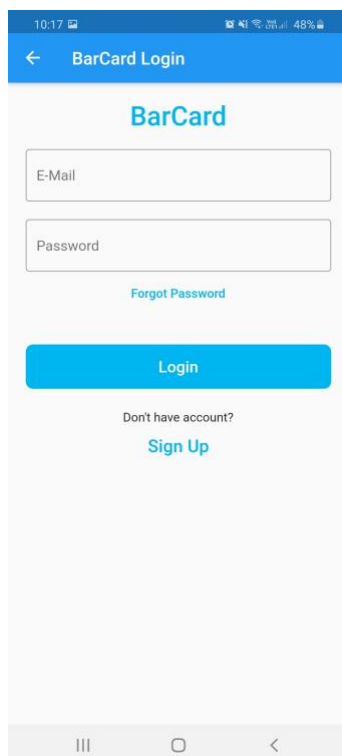


Figure 7 Login Screen

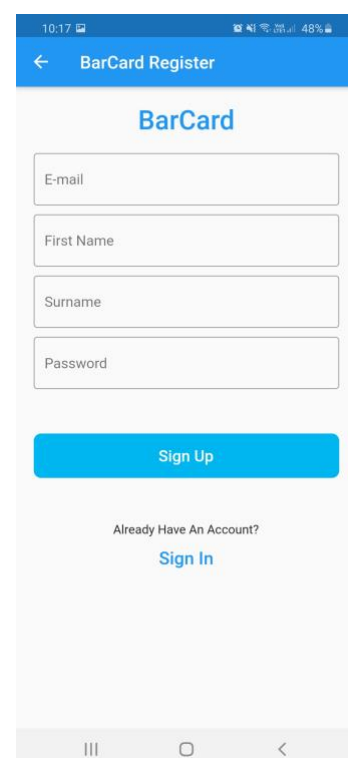


Figure 8 Sign up Screen

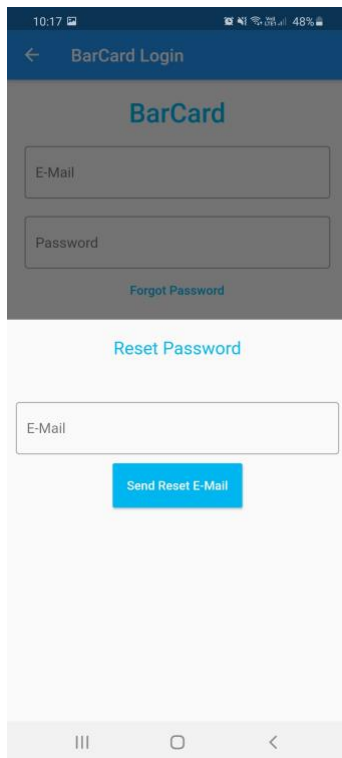


Figure 10 Forget Password field

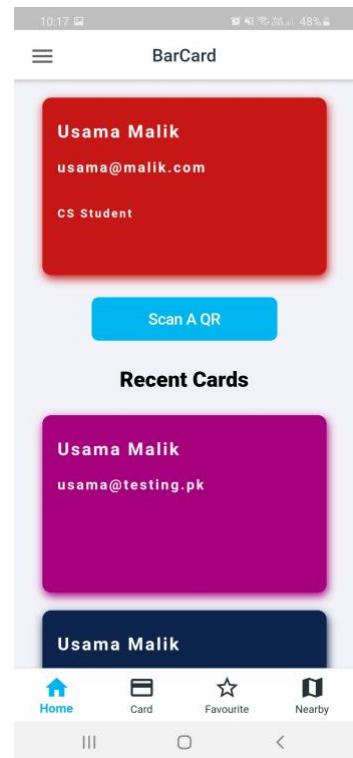


Figure 9 Home screen

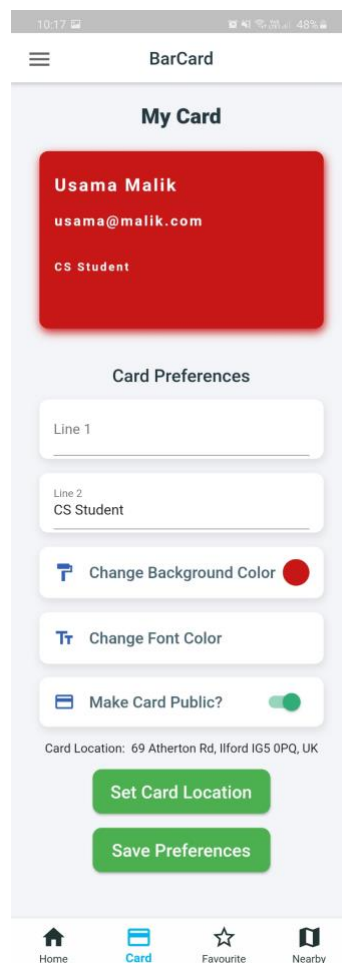


Figure 11 Card Customisation screen (scrolled screenshot)



Figure 13 Favorite Cards section

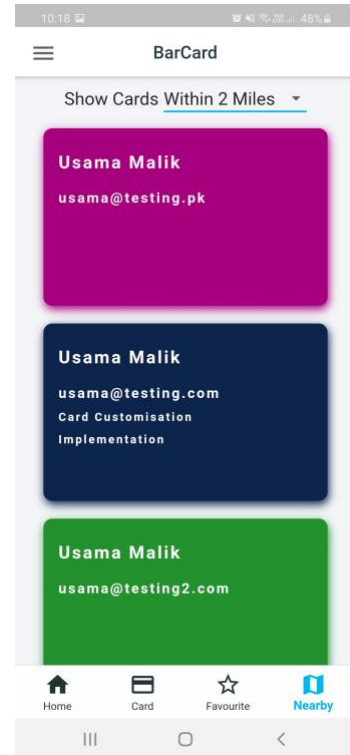


Figure 12 Nearby Cards Section

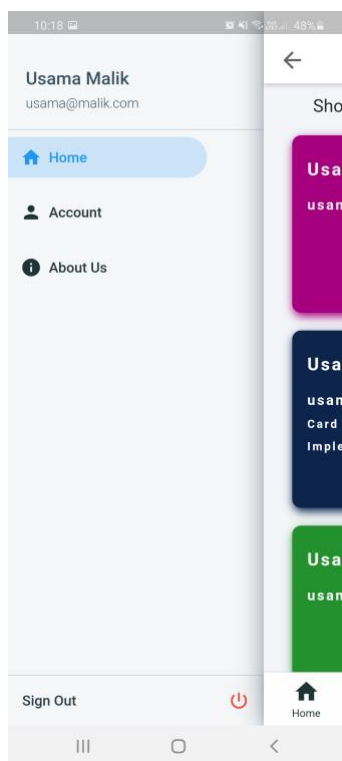


Figure 15 Navigation Drawer

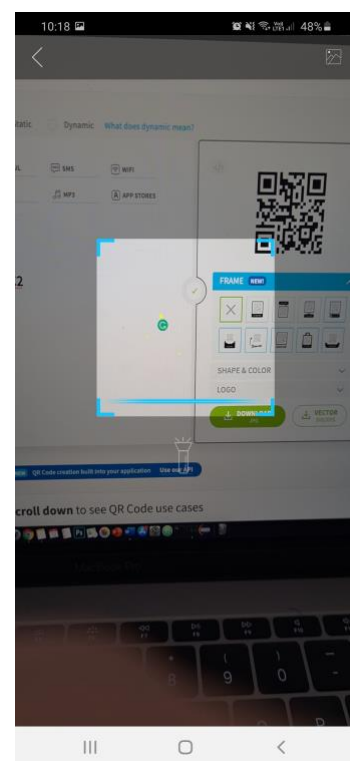


Figure 14 Barcode Scanner screen

## Implementation

The implementation of the application was personally the most enjoyable for me. It took around 3.5 months to implement the application, over this period I focused on the MVP requirements first and tried my best to include as many extra system requirements as possible.

When I was implementing, User interface (UI) and User Experience were always kept in mind, so the app always feels and looks professional, and is as close to existing design practices as possible. So, users feel confident in using my Application and find it intuitive to use.

All parts of the System are functioning and have ample error handling to make the user experience as smooth as possible.

This section will describe in detail, how I implemented and designed the main features of the application. I break it down into 3 different sections, frontend, backend and the mid layer (also known as the business layer). The frontend will focus on mostly Flutter and UI creation, the mid layer will briefly describe the database queries and the CRUD operations, and the backend will describe the database design and structure in Firestore.

***Please note that all the code I have written myself can be found in the “lib” folder, and I have mentioned in the code or in this report whenever I am using a package or third-party code.***

***The full implementation can be found:***

***<https://github.com/usama-malik89/Computing-Project/tree/master/Implementation>***

### Flutter Packages Used

A brief explanation on each package I used, some are provided by the Flutter dev Team and some are by Third parties.

#### **firebase\_auth<sup>12</sup>**

Used for signing up and authenticating users when logging in. Returns a user object globally whenever a user is logged. This can be used to determine if a user is logged in, or access user details globally (when I say globally, I mean anywhere in the code or app).

---

<sup>12</sup> [https://pub.dev/packages/firebase\\_auth](https://pub.dev/packages/firebase_auth)

### **cloud\_firestore<sup>13</sup>**

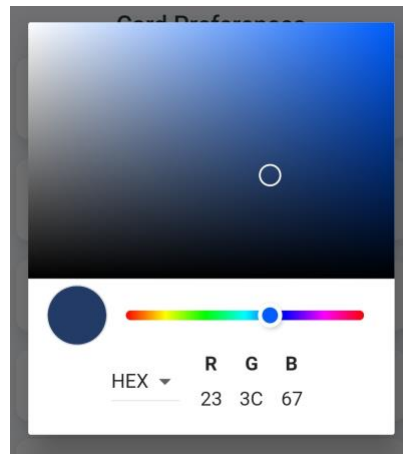
Used for mainly location-based functions. Provides with classes such as GeoPoint which is the variable type the location of the card is stored in the database using longitude and latitude.

### **provider<sup>14</sup>**

This is a package that allows for global access to a variable. I use it to have the logged in user accessible anywhere in the code, so things such as logged in users username and unique id can be easily accessible.

### **flutter\_colorpicker<sup>15</sup>**

This is used to show a colour picker for a user to be able to customize their card background and font colours.



*Figure 16 Shows colour picker popup for a user to select any colour they wish*

### **flip\_card<sup>16</sup>**

This is used to be able to flip any widget so it has two sides. This provides the animation to be able to do so. This was needed so maximise screen space so the E-Business card could have a backside for added functionality.



*Figure 17 Flipping a widget using flip\_card*

<sup>13</sup> [https://pub.dev/packages/cloud\\_firestore](https://pub.dev/packages/cloud_firestore)

<sup>14</sup> <https://pub.dev/packages/provider>

<sup>15</sup> [https://pub.dev/packages/flutter\\_colorpicker](https://pub.dev/packages/flutter_colorpicker)

<sup>16</sup> [https://pub.dev/packages/flip\\_card](https://pub.dev/packages/flip_card)

**qr\_flutter<sup>17</sup>**

This package allows for easy QR code generation. You can input any string and it generates a QR code image.

**qrscan<sup>18</sup>**

Allows for the camera to be used as a QR code scanner, automatically opens the scanner when pressing a button.

**introduction\_screen<sup>19</sup>**

Allows for easy slider introduction screen, so when users first open the app they can get an idea of what the app is. Sort of a basic tutorial on how to use the app.

**location<sup>20</sup>**

This package allows me to get the location of the device using GPS. This will prompt the user for permission to use the device's location, the user then has an option to grant or deny permission.

**url\_launcher<sup>21</sup>**

This is used to be able to open the users' preferred email application to be able to send an email to a user.

**geo\_firestore<sup>22</sup>**

This package is used mainly for its ability to query the database to only return users within a certain radius in the vicinity.

**geocoder<sup>23</sup>**

Used to be able to turn map coordinates into an address String

**flutter\_spinkit<sup>24</sup>**

This provides different loading animations

---

<sup>17</sup> [https://pub.dev/packages/qr\\_flutter](https://pub.dev/packages/qr_flutter)

<sup>18</sup> <https://pub.dev/packages/qrscan>

<sup>19</sup> [https://pub.dev/packages/introduction\\_screen](https://pub.dev/packages/introduction_screen)

<sup>20</sup> <https://pub.dev/packages/location>

<sup>21</sup> [https://pub.dev/packages/url\\_launcher](https://pub.dev/packages/url_launcher)

<sup>22</sup> [https://pub.dev/packages/geo\\_firestore](https://pub.dev/packages/geo_firestore)

<sup>23</sup> <https://pub.dev/packages/geocoder>

<sup>24</sup> [https://pub.dev/packages/flutter\\_spinkit](https://pub.dev/packages/flutter_spinkit)



### **permission\_handler<sup>25</sup>**

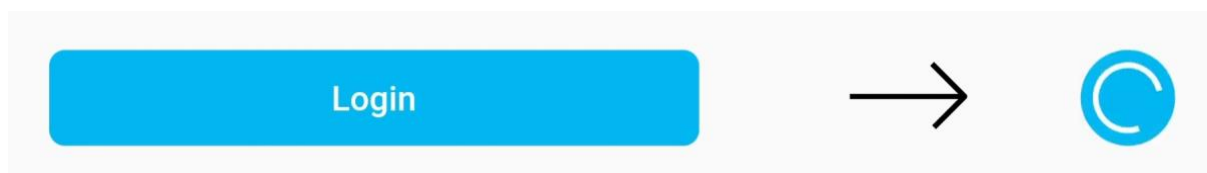
This package allows for easy permission handling. Permission handling is needed to make sure the devices GPS and camera can be used by the app, with the consent of the user. If permission is denied and the user tries to use the scanner for example, I can use this package to check if the permission has been granted, if it hasn't, I then show a dialogue popup to explain why the permission is needed

### **flutter\_sticky\_header<sup>26</sup>**

Used to make headers stick to the top of the screen when scrolling, so users know which section they are on, even when they have scrolled all the way to the bottom.

### **progress\_button<sup>27</sup>**

Used for the signup and login buttons. It turns the button into a loading indicator animation to show the user that something is being processed. This package also has an error state, meaning if for example the sign in fails for some reason I can change the state of the button to "error" which creates a shaking animation to the button, similar to IOS native buttons.



*Figure 18 Button changes to loading animation when pressed*

---

<sup>25</sup> [https://pub.dev/packages/permission\\_handler](https://pub.dev/packages/permission_handler)

<sup>26</sup> [https://pub.dev/packages/flutter\\_sticky\\_header](https://pub.dev/packages/flutter_sticky_header)

<sup>27</sup> [https://pub.dev/packages/progress\\_button](https://pub.dev/packages/progress_button)

## Front end Implementation

As mentioned before Flutter makes it incredibly easy to create professional looking UI elements with just a few lines of code. The main sections I will talk about for the front end will be the Splash screen, login/signup screen (including validation), Card UI element, card customization and nearby cards section.

### Splash Screen

A splash screen is essentially a loading screen that gets shown whenever you first open the app. Initially I did not plan to include this, however when getting feedback, someone pointed out that whenever the app first opened the login screen was being shown for a few seconds before showing the home screen. This was happening even though the user was logged in. This made the app feel slow and made it look like the app was stuttering.

The root of the problem was that when the app opens, the Wrapper class checks if the user is logged in by checking if the Firebase Auth returns a user or null, and then shows the corresponding screen, the login page or home screen.

```
class Wrapper extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    final user = Provider.of<User>(context);  
  
    //return home or authenticate  
    if (user == null) {  
      return WelcomeScreen();  
    } else {  
      return NavigationHomeScreen();  
    }  
  }  
}
```

Can be found: Wrapper.dart lines 7-19

by Usama Malik

Figure 19 Whole code can be found in Wrapper.dart file

The problem is that it can take a few seconds for FirebaseAuth to return a user for the Wrapper class to determine which screen to show. So, to solve this issue I first display a simple Splash screen at the root of the app, which then navigates to the wrapper class after a set time of 4 seconds, as seen in the short code below (the whole implementation of Splash Screen can be found in SplashScreen.dart file)

```
home: SplashScreen(navigateTo: Wrapper(),),
```

by Usama Malik

Figure 20 Can be found main.dart file, line 22

Whilst the splash screen is being shown, FirebaseAuth returns a user if one is logged in, so the corresponding screen can be shown, without a stutter.

I do realise showing the splash screen for a set number of seconds is bad practice as FirebaseAuth may return data faster than that. Unfortunately, I couldn't figure out a way to only show the splash screen when the data is being loaded and navigate away when the data is loaded, so I stuck with this solution.

### Login & Signup Screen

The Login and Sign up screen are very similarly implemented, because they were implemented using the help of a tutorial on firebaseAuth<sup>28</sup>. Both have a form widget with text fields;

```
126 Form(
127   key: _formKey, //key to be able to refer to this form
128   child: Column(
129     children: <Widget>[
130       Container(
131         padding: EdgeInsets.all(10),
132         child:
133           //Form Field for email
134           TextFormField(
135             validator: (val) => val.isEmpty //input validator to make sure something has been input
136               ? 'Please enter a valid email!' //error message
137               : null,
138             controller: userInput, //controller to be able to refer to the input data
139             //UI styling
140             decoration: InputDecoration(
141               border: OutlineInputBorder(),
142               labelText: 'E-Mail',
143             ), // InputDecoration
144           ), // TextFormField
145       ), // Container
146       Container(
147         padding: EdgeInsets.fromLTRB(10, 10, 10, 0),
148         //Form Field for password
149         child: TextFormField(
150           validator: (val) => val.length < 7 //input validator to make sure password is longer than 6 chars
151             ? 'Password must be longer than 6 characters!' //error message
152             : null,
153           obscureText: true, //to hide the input as this is a password
154           controller: passInput, //controller to be able to refer to the input data
155           //UI styling
156           decoration: InputDecoration(
157             border: OutlineInputBorder(),
158             labelText: 'Password',
159           ),
160         ),
161       ),
162     ],
163   ),
164 ),
```

by Usama Malik

Figure 21 Can be found in file SignIn.dart, Lines 126 - 169

<sup>28</sup> <https://petercoding.com/firebase/2020/02/25/using-firebase-auth-in-flutter/>

A Form widget takes in two properties, a key and child. The key is used to be able to refer to it later on in the code if you need to. In this case the form has a Column as its child at line 130, and the Column has two TextFormField widgets (line 134) which takes in 3 properties, validator, controller and decoration.

The validator property is as the name suggests, for form validation. At line 135 you can see that the FormField for email is checking if it is empty, if it is, it returns an error message at line 136, otherwise it returns null at line 137, meaning no error has occurred. The controller property is used to be able to refer to the data in the field later.

When the user then presses the login button the logic below gets run;

```
191 //tries signing in user with texfield data
192 dynamic result = await _auth.signInWithEmail(userInput.text, passInput.text);
193
194 //if the sign in fails error message is shown
195 if (result == null) {
196   setState(() {
197     err = 'Invalid Credentials!';
198     bState = ButtonState.normal; //sets the button's state to normal
199   });
200 } else { //otherwise the current screen is closed and home screen is shown
201
202   setState(() {
203     bState = ButtonState.inProgress; //sets the button's state to loading animation
204   });
205   Navigator.pop(context);
206 }
```

by Usama Malik

Firstly at line 192 I use the FirebaseAuth method signInWithEmail() and pass in 2 parameters, the email input and password input. If the login fails, FirebaseAuth returns the result as null, this allows me to do a check on line 195, which shows an error message if login fails and if it is successful it takes the user to the home screen.

### Card UI element

The Card UI is the most important part of the app as it is the main aspect of the project. At the beginning I was struggling to create the UI that resembled a business card, so I watched a tutorial “Credit Card - Speed Code + Tutorial”<sup>29</sup>, a small part of this tutorial shows how to make a credit card UI in Flutter. I used this to get the basic shape of the credit card and use it to make my E-Business card.

To benefit from code reusability, I have made the card element in its own class, that takes in 2 parameters, the UID and margin. There are a few different variations of the card back, this needed to be done because different cards needed different actions depending on which section of the app they were on.

---

<sup>29</sup> <https://www.youtube.com/watch?v=WTHgpyXMC0>

For example, the user's card only shows the QR code on the backside, whereas cards in the nearby section show the QR code, favourite button and contact button on the back.

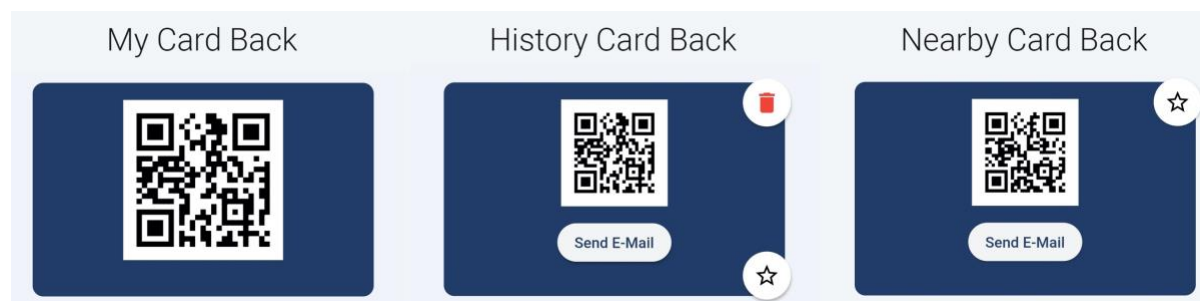


Figure 22 Different Card Variations depending on which section of the app they are on

Initially I had just one class for the card and a parameter was being passed in to differentiate the cards, this way a lot of logic was being done within the class, and 800+ lines of code were present in one class. There was a huge performance issue due to this, the card would flicker, and the animation was really slow. After some research on Flutter I learned that separating the code in different classes drastically improves performance. On Flutter's dev website it states *"Avoid overly large single Widgets with a large build() function. Split them into different Widgets based on encapsulation"*<sup>30</sup>.

To fix this issue I simply did what Flutter Dev's recommended, I separated every variation of the card into its own widget. This solved the problem and made the app run much smoother.

Each card widget does the data handling within itself, the card widget only takes in a uid and pulls the necessary data such as card text and styling from the database using the methods from DatabaseService class in database.dart file.

The implementation of the card can be seen below, (please note that the code below is not the actual code but the shorter code just for visualisation, the whole code for the different variations can be found in the bCards folder).

---

<sup>30</sup> <https://flutter.dev/docs/perf/rendering/best-practices>

```

1  @override
2  Widget build(BuildContext context) {
3      return FutureBuilder<List<String>>(
4          future: _userData,
5          builder: (BuildContext context, AsyncSnapshot<List<String>> snapshot) {
6              dynamic children;
7
8              if (snapshot.hasData) {
9                  children = FlipCard(
10                     direction: FlipDirection.HORIZONTAL,
11                     front: frontBCard(),
12                     back: backCard(),
13                 );
14             } else if (snapshot.hasError) {
15                 children = FlipCard(
16                     direction: FlipDirection.HORIZONTAL,
17                     front: errorCard(),
18                     back: errorCard()
19                 );
20             } else {
21                 children = FlipCard(
22                     direction: FlipDirection.HORIZONTAL,
23                     front: loadingCard(),
24                     back: null,
25                 );
26             }
27
28             return children;
29         },
30     );
31 }
32 }

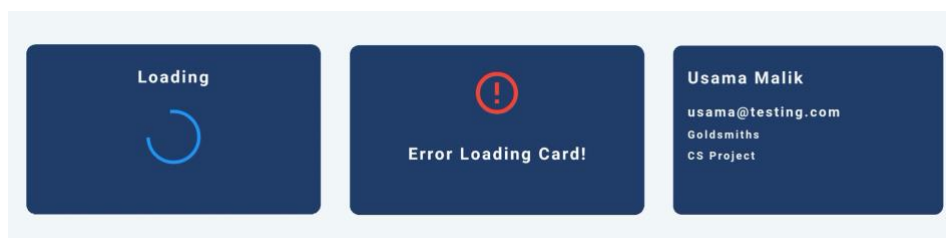
```

by Usama Malik

Figure 23 Shortened code for Card widget, full implementation can be seen in any card variations in bCards folder

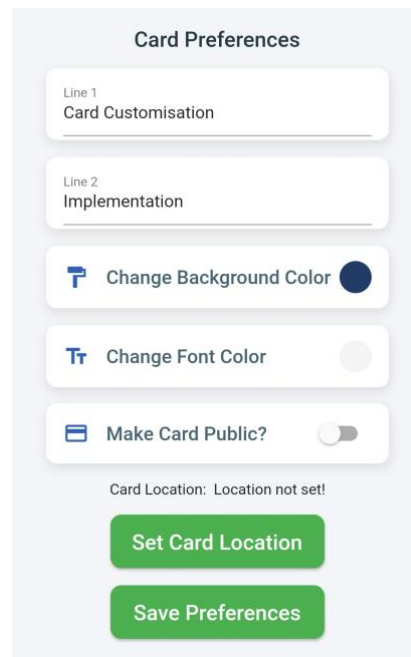
Every time you want to draw something to the screen in Flutter it needs to be done in the build function which needs to be overridden, this can be seen in the code above. Then it uses the FutureBuilder widget which takes in a future function, in this case the `_userData` function which gets initialised with the user data from the database using another function. A future function in Flutter allows you to run work asynchronously to free up any other threads that should not be blocked. Like the UI thread. This way when a large task is being done the performance of the UI doesn't get affected.

Then there is some logic for error handling, so if an error occurs when retrieving data from the database it can show an error message. If the database snapshot has data, the code block from line 8 – 13 gets run which creates a flipCard with the correct frontCard and backCard. If an error has occurred or no data is present in the database snapshot, then the code block from line 14-19 get run which simply shows an error card. If the data is still being loaded from the database, then the code block from line 20 – 26 which shows a Loading card.



## Card Customisation

Customising the card consists of changing the information text on the card, changing the font colour, changing the background colour, making the card public or private and changing the card location.



The screenshot displays the 'Card Preferences' interface. It features two text input fields: 'Line 1 Card Customisation' and 'Line 2 Implementation'. Below these are two color selection controls: 'Change Background Color' with a blue circle and 'Change Font Color' with a light grey circle. A toggle switch for 'Make Card Public?' is currently turned off. At the bottom, there is a status message 'Card Location: Location not set!' followed by two green buttons: 'Set Card Location' and 'Save Preferences'.

*Figure 24 Card Customisation Screen*

The code for the customisation can be found in the CardTab.dart file. All the customisation elements such as Text, background colour, font colour and public status are all class variables which then get stored in the database once the “Save Preferences” button is pressed. This is done by using the setUserPref() method in the DatabaseServices class, this method takes in all the customisation variables and stores it in the database to the corresponding users UID.

When the change Background or Font colour buttons are pressed, I use the colorpicker package to show a colour picker to the user, and the live colour change is shown so the user can see how the card looks like with the new colour scheme.

To set the card’s location the user just has to press the “Set Card Location”. This uses a the saveLocation function in the CardTab.dart file on line 149. This gets the devices current location and saves the map coordinates in the database.

### Nearby Section

The nearby section all the public cards within a specified mile radius is shown to the user. This was achieved by the help of the location package and the GeoFirestore package. The location package helps get the device's location and also get the coordinates of the location. Once the location coordinates have been received, geoFirestore stores it in my database, I have to specify which collection to store it in and which record with the specified UID. The geoFirestore package also creates a geohash value from the coordinates and saves it in the database.

Geohashing is a method to encode/decode map coordinates (Long / Lat) into a single string. It divides the world map into grids, each grid can either be a letter or number. Then each grid has smaller grids with each grid having their first index of the String being its parent's letter and the second character in the string being wherever on the grid the location is. Then these smaller grids have even smaller grids within them, this can carry on depending on the accuracy you require. A geohash string with length 10 is accurate to an area of  $4\text{m}^2$ , which is more than accurate enough for my use case.





Then I created a function, MapQuery() on line 24 of the NearbyTab.dart file. This function uses GeoFirestore to make a query to the database.

```
24 Stream mapQuery(radius, uid) async* {
25   //Initialises GeoFirestore package with the userCollection being taken in as a parameter
26   GeoFirestore geoFirestore = GeoFirestore(data.userCollection);
27
28   //Initialises location package
29   Location location = new Location();
30
31   //getting devices location
32   var pos = await location.getLocation();
33
34   //Using the devices coordinates to set the center of the query location
35   final queryLocation = GeoPoint(pos.latitude, pos.longitude);
36
37   //gets all the cards within the radius and adds to the list
38   final List<DocumentSnapshot> documents =
39     //uses getAtLocation method with the devices location being the center,
40     //and input radius to query all the cards in the database within this radius
41     await geoFirestore.getAtLocation(queryLocation, radius * 0.6214);
42   //filters the list to make sure the users own card doesn't get shown
43   List<DocumentSnapshot> filtered = documents.where((doc) {
44     return doc.documentID != uid;
45   }).toList();
46   //filters the list again to make sure cards that aren't public don't get shown
47   List<DocumentSnapshot> secondFilter = filtered.where((doc) {
48     return doc.data['isPublic'] == true;
49   }).toList();
50
51   //returns list with the cards
52   yield secondFilter;
53 }
```

by Usama Malik

On line 38, in the above function, you can see that the geoPackage is used to query from the database. The query method takes in two parameters and, the query location and the radius. The query location is essentially the centre point of where the query will search for in the database, and the radius is used to specify in which radius to search the cards for. As you may notice the radius is multiplied with 0.6214, this is because the method assumes the radius is in km. However, I want to specify it in miles, thus being the reason I had to multiply the radius so it can be turned into miles first.

After a list has returned, I have to apply 2 filters, one filter is to make sure the users own card doesn't get included in the list, and the second filter to make sure non-public cards don't get shown to the user. These filters can be seen on line 43 and 47 respectively.

A simple drop-down menu is also shown on the top of the screen, to be able to specify the radius in which the cards should be shown.

## Back end – Firebase

The implementation for the structure of the database is very similar to how an object tree would look like in a JSON formatted document.

The Structure of the database is as followed;



The root collection in the database is the UserCollection, which contains every record of a user, and each user record has a unique ID. The user record contains 10 properties and 2 collections;

1. bColor for storing the background colour for the card
2. email for the user's email address
3. fColor for storing the font colour for the card
4. firstName for the first name of the user
5. geohash for storing the geohash string of the location
6. isPublic for storing the Boolean to determine if the card is public or not
7. line1 for storing the custom line1 of the card
8. line2 for storing the custom line2 of the card
9. location for storing the GeoPoint object with map coordinates
10. surname for the user's surname
11. cardFav is a collection for storing the favourite card records
12. cardhistory is a collection for storing the recent scanned card history

The cardFav collection has all the records of the cards the user has marked as favourite, and each record has a property for timestamp, to store when the action took place so it can be sorted in ascending order.

The cardHistory collection is exactly the same as cardFav collection, however, it contains the records of the recently scanned card history, which also has the timestamp property.

## **Mid Layer**

The mid layer, also known as the business layer, is the layer that joins the front end and back end layers together so they can communicate with one another. In a traditional web application this is usually a layer consisting of a PHP layer which handles the data and query requests with a SQL database. However, it is different with the Flutter and Firebase combination development. Because both developed by Google, they also provide a Flutter package that provides the basic CRUD methods, to be able to communicate with the Firebase database.

I would consider the DatabaseService class in the Database.dart file to be my mid layer because I have written all the functions that need to communicate with the database in this class. The majority of the functions are to get data from the database, however some functions store data in the database. All the functions are "Future" functions, which is syntax for a dart specific function. This type of function allows the function to only return data when there is data present. For example, if I am wanting to retrieve the background colour of the card from the database, because of the speed limitations of the server and the internet, it can take time for it to retrieve the data from the database, with the future function if the data is still in "transit" the function will not return anything, however once the data has been retrieved the function will return it.

All this happens on a secondary thread meaning the main thread where the animation and UI engine may exist, will not be affected due to different processes taking place when using the app, allowing a smooth User experience throughout the app.

## **Bugs**

There were several bugs during the development stage. The main way I fixed bugs was through searching online and look at answers in the Stack Overflow community. Unfortunately, the bugs bellow were the only bugs I had recorded, I did encounter many bugs, however I was able to fix them straight away due to the hot reload function allowing me to test newly added code straight away.

### **Text Disappearing when scrolling**

Whenever the screen was scrolled, the data such as name and email on the card would disappear whilst scrolling but would appear as soon as it stopped scrolling. After some looking online, I figured out what the problem was. Essentially when the card class was initialised all the data fields are initialised as empty strings and then when it receives the data from the database it changes it to relevant information such as name and email. But I guess the way flutter handles animation, it uses the initialised UI as the placeholder whenever a user is scrolling.

There was an easy fix, essentially, I used a method called "FutureBuilder", this provides different states such as if database object hasData, hasError or loading. This way it only draws the card when the data has loaded, otherwise it shows a loading circle, or an error card if nothing gets returned from the database.

## QR scanner error card

In the first iteration of the QR code scanner there was a small bug, whenever an invalid QR code was scanned it would still add it to the recent section, but the card would be shown as "Error Loading card". I fixed this so whenever a scan has happened, a function first checks if a card exists in the database. If it does only then it gets added in the recent section.

## Alpha Testing

Testing is a very important part of any software development. It ensures that the system and the containing functions work exactly how they are expected to work.

The majority of the testing I have done during the development stage, as I was implementing. This was because Flutter provides easy testing by using an Android simulator. This is essentially a smartphone emulator on the PC that can be used exactly how an Android device would be used. On top of the emulator, Flutter also provides hot-reloads which made testing the app so much easier. Hot-reload is what allows the app to be updated instantly on the emulator as soon as I press the save button in Android studio so essentially showing me a live version of the app as I am coding. For example, if I add a text widget in the code and hit the save button, the app changes instantly on the emulator allowing me to see the changes instantly.

This saves a lot of development time, because if I had to rebuild the app every time I wanted to see a change it would take so much time that the development time would likely double or triple because to build the app it takes around one minute on my laptop, compared to a 1 second reload with Hot reload.

After the implementation was complete, I carried out further tests, black box and white box testing. Also known as Behavioural and Code Based Testing.

## Black box testing

Blackbox testing exists to test functionality of the app without looking at the implementation, so without looking at the internal code structure<sup>31</sup>. This testing is essential because it replicates how a user would use the app because a user would not know the structure of the code.

Test Number	Action Description	Expected Result	Actual Result
1	Sign In	When signing in with correct credentials it should show loading animation and then go to the home page	The loading animation was shown for half a second and then taken to the home page PASS

---

<sup>31</sup> <https://www.guru99.com/black-box-testing.html>

2	<b>Register</b>	When registering with a valid email and password it should create the account and login straight away, taking the user to the home page	When registering, the loading animation was shown and was taken to the home page PASS
3	<b>Opening QR code Scanner</b>	When the user presses the "Scan a QR code" button, the camera should open	When the button was pressed the camera opened PASS
4	<b>Scanning a valid QR code</b>	When a user scans a valid QR code the corresponding card should get shown	When the QR code was scanned the camera app closed and the corresponding card got shown. PASS
5	<b>Scanning a random QR code</b>	When scanning a random QR code, an error card should be shown on the screen	When scanning a random QR code the camera closed and a error card was shown PASS
6	<b>Saving Card preferences</b>	When the "Save Preferences" button is pressed the values should get updated and the confirmation popup should be shown	When the button was pressed the confirmation was shown, and the data got updated in the database PASS
7	<b>Saving card location</b>	When the "Save card location" is pressed the cards location should get set to the current address, and the address should get shown on the screen	When the button was pressed a confirmation was shown, and the corresponding address was updated after a few milliseconds. PASS
8	<b>Viewing nearby cards</b>	When viewing nearby cards, only the cards that are set as public by their user and are within the specified radius should be shown	When in the nearby cards section, the cards in the 5 mile radius got shown. PASS
9	<b>Saving card to favourites</b>	When pressing the star button on a card, it should get saved in the favourite section	After pressing the star button on a card, the card was present in the favourite cards section PASS
10	<b>Removing card from favourites</b>	When pressing the delete button on a card, it should get removed from the favourite section	After pressing the delete button on a card, the card was removed from the favourite cards section PASS
11	<b>Signing out</b>	When the sign out button is pressed the	The welcome screen is shown when the sign out button was pressed

		welcome screen should be shown	
--	--	--------------------------------	--

## Testing Methods

All the tests were carried out by me on my personal smartphone or the emulator. Tests 1 and 2 were straight forward, for both I first input nothing in the fields and pressed the login button. Then I input the incorrect email address (e.g. usama@) and a short password length. Then finally I input the correct credentials. All of these cases were successful.

For tests 3 and 4 I simply followed the steps to complete the test actions and checked if it was working as expected. For test 5 I generated random QR codes on my laptop<sup>32</sup> and scanned them to see what the outcome was.

For test 8, viewing nearby card, I used a GPS spoofer<sup>33</sup> on my smartphone to fake the location of my device. This way I could create different cards at specific locations. So, for example I created a card just outside the 5 mile radius of my actual location. Then I turned off the GPS spoofer and checked if the card shows up if I select the 5 mile radius option.

## White box Testing

White box testing exists to test the functionality of the important code, and check if the important functions behave like they are expected to.

Test Number	Function & Inputs	Expected Output	Actual Output
1	regEmail( String email, String password, String fName, String sName )	Success: Returns the logged in users, user object  Fail: Returns Null	Works as expected, when I input correct parameters, It successfully works. And an occurs when incorrect parameters have been input. The error message is shown by the parent function if this function returns null PASS
2	signInEmail( String email, String password )	Success: Returns the logged in users, user object  Fail: Returns Null	Works as expected, When correct credentials have been passed as parameters, it successfully logs in. Otherwise an error message is shown by the parent function, because this function returns null

<sup>32</sup> <https://www.qr-code-generator.com/>

<sup>33</sup> [https://play.google.com/store/apps/details?id=com.incorporateapps.fakegps.fre&hl=en\\_GB](https://play.google.com/store/apps/details?id=com.incorporateapps.fakegps.fre&hl=en_GB)

3	resetPassword( String email )	<p>Success: Sends a reset password email to the specified email, using FirebaseAuth package</p> <p>Fail: No Fail case, because if wrong email is input the email will not be sent anyways</p>	Works as expected, I tested it on 4 different, correctly formatted email addresses. When an email that doesn't exist in the database is inputted, the system still shows success, however no email is sent PASS
4	OpenQRScanner()	<p>Success: Returns a String of whatever QR code was scanned</p> <p>Fail: Returns Null</p>	Works as expected, When I scanned a QR code, the correct string is returned in the Console. And if I close the camera without scanning anything then nothing is returned and no error occurs
5	coordToAddress( coord )	<p>Success: Returns a String of the address the card is set to</p> <p>Fail: Returns the either the town, city or country of the card location</p>	Works as expected, when a correct input is passed as the parameter, the function is able to get an address and returns the address line, if it can't then it returns just the town name, city name or the country name. In no case does it return nothing
6	_saveLocation( uid )	<p>Success: Uses the current location of the device and stores the map coordinates in the database</p> <p>Fail: Doesn't store anything in the database</p>	Works but slight problem. When the device doesn't have an internet connection and saves the location, a success dialogue is still shown. This is misleading because without a connection the database cannot be updated. Other than that, it works correctly
7	updateUserData( String fName, String sName, String email, bool isPublic, String bColor, String fColor	<p>Success: Uses the parameters and stores the values to the corresponding keys in database</p> <p>Fail:</p>	Has the same issue like test 6, it still shows a success dialogue even if no internet connection exists. Apart from this it works fine

	)	Doesn't store anything in the database	
8	mapQuery( radius, uid )	Success: Should return all cards within the specified radius, the list should not contain the current users own card  Fail: Shows error message	Works as expected, It returns all the nearby cards, excluding the current user's card. If no internet connection exists, then an error message is shown to the user.
9	recentQuery( uid )	Success: Should return all cards in the recent collection in the users record in the database  Fail: Shows error message if no card is returned	Works as expected, returns the list of all the cards in the recentCards collection.  If nothing is returned an error message is shown

## Beta / User Testing

Beta testing, otherwise known as user testing is essential to be able to determine the ease of use and the user acceptance level of a system.

Due to the unforeseen Covid-19 outbreak, I was not able to get a sufficient number of users for accurate user testing. I had to conduct testing on a small sample of users that consisted of my family and close friends. With my friends, the tests had to be done in a less traditional way. Whenever I was at a stage where I needed to conduct user testing, I compiled the app and sent them the .APK file, which allows my friends to install the app on their devices. This way they could test what I asked them to test, and they could provide me with feedback.

However, with my family (which I live with in the same home as), I could do more detailed tests as I could get the necessary screen ready for the tester and ask them to perform a certain task, and they could give me instant feedback.

After the implementation was finished, I requested my friends and family to fill out a short survey. I gained 9 responses, 4 of which were from my family and 5 friends.

I will briefly describe the responses; the Full questionnaire and response charts can be found in the appendix.

When asked about how their experience was when registering on the app, 88% of users said that they found it relatively easy or very easy, only 1 person didn't find it easy or hard, and no one found it hard registering on my app.



When asked about any performance slowdowns, 2 people said that they did experience some sort of performance slowdown, one of these individuals went further and said it was in the nearby section they experienced the slowdown.

When asked about if at any point whilst using they were confused, every participant answered no.

When asked about the ease of designing the card, I had mixed responses. Most users said it was relatively easy, however a majority of the users also found it relatively hard, meaning there is some issue, and the card customisation screen isn't clear.

When asked about what further customisation options they would like to see, I had 4 responses. 2 comments were related to the customisation of the text layout and font. One was about being able to change the background image, and one comment mentioned about social media links.

When asked about the QR scanner, the majority of the users found it relatively easy to use.

5 of the participants answered Yes when asked if they would use the app in the future, however 2 answered no and 2 were indecisive.

50% of the users would use the app because of its business card feature, 25% of the participants said they would use it for meeting new people in their area and 25% of the users said they would use it for educational or professional guidance.

These survey results are very valuable to understand the usability of the app. After looking at the responses I can see that there are a couple of issues that need fixing, for example the customisation screen had mixed responses. I can use these responses to set myself a goal for future development. However, the user sample size was only 9 participants, meaning this research can only be taken with a grain of salt. A better representation would be if I had many participants with different skill levels and backgrounds.

## System Failures

I would say the app 90% works as intended, there was never a moment during testing where the app crashed or done something unexpected. There are a few minor issues, however these issues don't affect the functionality of the app.

### Tab Reloading

One of the issues in the application is that whenever the user changes tabs, for example from the home tab to the nearby tab, the screen has to reload again. This slightly slows down the user's experience, especially if the network connection is slow to load the data for the cards it can take a second or two. I tried fixing this issue by a fix I found on Stack Overflow<sup>34</sup>, however this did not seem to work so this issue went unfixed.

### Misleading Success Dialogues

Another issue in the application is that when there is no internet connection and the "Save Location" or "Save Card Preferences" button is pressed, a success dialogue is shown, although nothing actually changes in the database due to no connection.

### Nearby Section Card Loading

When changing the tabs to the nearby tab, the query to get the cards in the near vicinity of the devices location can take a few seconds to load, whilst this is being loading, a loading animation is being shown. The loading times are very inconsistent for some reason. I tried looking for a solution online. However, I didn't find anything relevant. I suspect it may be due to the query being done by a package.

### Data Encryption

After the development concluded and testing started, I realised that my app uses sensitive data, which is the card's location. I had ensured that the exact location of the card does not get revealed to other users, however what I forgot was the data being stored could be vulnerable to a malicious attack and this sensitive data could be stolen. I could easily prevent this by encrypting the data with 128-bit AES encryption before storing and then decrypting it when retrieving the data. This would ensure that even if there was an attack on the database, the sensitive data will not be readable.

---

<sup>34</sup> <https://stackoverflow.com/questions/49087703/preserving-state-between-tab-view-pages>

## System Requirements Evaluation

In this section I will briefly talk about all the system requirements I was able to implement and which ones I wasn't.

Below is the same list as the list in the system requirements section, but met requirements are highlighted green and marked as MET, and requirements that aren't met are highlighted red and marked as NOT MET;

### MVP Requirements (Most important to least)

1. Be able to create an e-business with a unique QR code (MET)
2. Be able to scan a QR code (MET)
3. When a QR code is scanned it should show the corresponding card (MET)
4. Be able to see Cards near your vicinity (MET)
5. Change personal information (NOT MET)

### Extra Requirements (Most important to least)

6. Be able to contact a cards User (MET)
7. Be able to design the e-business card (background, text etc) (MET)
8. Be able to view scanned history (MET)
9. Add cards to a favourite section (MET)
10. Be able to have a services store for a card (NOT MET)
11. Have a Web application so scanning QR code without the app takes you to the web app. So, users can still see the card without downloading an app (NOT MET)

Almost every requirement was met, however some of the least important requirements were not met. Requirement 5 was not met due to my mistake of redirecting my focus on the extra requirements. I should've focused on the MVP requirements first before going onto the extra requirements.

Requirement 10 was also not met due to time constraints. I was planning to include this so users can have a sort of marketplace for services they provide, however because it being a low priority requirement I did not have enough time to be able to implement this. The last requirement that wasn't met was requirement 11 due to, again, time constraints. Initially I was planning to implement the mobile app and web app simultaneously, side by side. However, I quickly realised that I will have to focus on one first and once that is finished, start implementing the other. So, I decided to focus on the mobile app as it was more important. If I had more time, or a group of developers this could've easily been achieved, especially because after I finished the implementation, Flutter released a new version which provided Web support for Flutter<sup>35</sup>, meaning that I could've reused the exact code from the mobile app, and used it to create a Web app using flutter

---

<sup>35</sup> <https://flutter.dev/web>

## Conclusion

### Aims and Objectives achieved / failed

This project aimed to provide an easy networking tool so that an individual may use it to broaden their networking links or seek employability guidance or referring to a workplace.

After concluding my testing, in my opinion, I believe I have created the very tool I was aiming to create. The main concept of the app was to provide a local networking tool for individuals seeking to broaden their networking links. An individual can use my app and go to the nearby section to view, all the business cards that exist, in a 1-mile radius or bigger radiuses, and they can contact them via email. Although this would only be plausible if one assumes that my app has tens of thousands or more users.

### Project Future Development

Although I have finished developing the app, it was only due to the project deadline and time constraints. However, there is still huge potential in further developing my Application and potentially releasing the app on the App Store. The multiple areas that still need further development, that I am aware of currently are;

#### Card Customisation

After user testing it is apparent that the card customisation is slightly confusing and lacks some functionality. In the future I could further work on this and provide a clear method for a user to customise the card and layout. Features such as background image / art and custom text placement would be really useful for individuals looking to just use the app for its business card functionality. I could implement a feature where the user can drag and drop the text wherever they wish on the card.

I also realised that one of the reasons it may be slightly confusing to use is due to the "Save Preferences" button. The only way I could get the customisation saved was by having the user press a button, which saved the data in the database. In the future I could do it so whenever any value is changed it saves it in the database without having to press any additional buttons.

#### Card Marketplace

One of the features I intended to include when planning, was a section for a user to advertise any services they provide. This would allow other users to buy services from individuals such as tradesmen by contacting them directly. In the future I could add this feature to make my app more attractive for users to use.

#### Web App

In the planning stages, I was planning to also develop a web app. This was so when a QR code is scanned by an Individual with my app, it will redirect them to the web app and show the Business card on there, with a prompt saying something along the lines of "Download the BarCard app to save this card and many more features!". This would make my QR codes universal, unlike now, where it provides useful information only if it is scanned through my app.

### **Built in Chat**

One of the features I could develop in the future could be a chat section within the app, instead of communicating through emails. This could be useful because everything would happen in the app and no other app would have to be used. This also has the added benefit of hiding the users email address if the user wishes to.

## **Project Evaluation**

In this project, I was able to create a fully working, relatively challenging application that met most of my functional requirements and is a tool to solve the problem I was aiming to solve in the first place.

Initially, I was confident about mobile development because of my experience from my penultimate year group project, which was also a mobile application, although with a different development technology. However, when doing further research I realised the technology I had learned prior, was inefficient, inferior to other technologies and not capable of complex system designs. So, I had to learn something different, something that provided a more native OS experience for the user, thus being the reason for using the Flutter framework. This allowed me to greatly improve my knowledge and skills related to mobile development, by learning from my mistakes and learning from bug fixing.

Initially I had created a Gantt chart, for me to follow, so I could stay on track to finishing the implementation. I have to be honest and say although initially I was strictly trying to follow it, I quickly realised the time scales for every feature was off by a huge margin. This thought me to be more realistic in the future when planning, before development.

Overall the project went well, and I gained a lot of valuable lessons concerning Software Development. I learned how to manage my time to maximise my development efficiency. There were a few mistakes I had made along the process, for example, I was spending a lot of time on additional functional requirements when the minimum viable requirements were still left unfinished. I should've finished the MVP requirements first and then gone on to implementing the extra requirements.

The fact that I was able to create this application makes me very proud and shows that I can bring together the knowledge I have gained from my BSc Computer Science course or before that, and create something that can potentially be brought into the App Store for the public to be able to download.

## Bibliography

[1] Davis, L. and Parker, K., 2020. *A Half-Century After 'Mister Rogers' Debut, 5 Facts About Neighbors In The U.S.*. [online] Pew Research Center. Available at: <<https://www.pewresearch.org/fact-tank/2019/08/15/facts-about-neighbors-in-u-s/>> [Accessed 11 June 2020].

[4] Smart Insights. 2020. *Snapchat Statistics 2020 | Smart Insights*. [online] Available at: <<https://www.smartinsights.com/social-media-marketing/social-media-strategy/snapchat-statistics/>> [Accessed 11 June 2020].

[5] Peterson, T., 2020. *Snapchat'S Snapcodes Decline In Getting Brands New Followers, Eclipsed By Deep Links*. [online] Marketing Land. Available at: <<https://marketingland.com/snapchats-snapcodes-decline-getting-brands-new-followers-eclipsed-deep-links-221355>> [Accessed 11 June 2020].

[6] Taylor, L., 2020. *QR Codes - How Much Data Can A QR Code Store?*. [online] Qrcode.meetheed.com. Available at: <<http://qrcode.meetheed.com/question7.php#:~:text=Standard%20QR%20Codes%20can%20hold%20up%20to%203Kb%20of%20data.>> [Accessed 11 June 2020].

[8] P, K., 2020. *Business Card Statistics*. [online] CreditDonkey. Available at: <<https://www.creditdonkey.com/business-card-statistics.html>> [Accessed 11 June 2020].

[28] Haddad, P., 2020. *Using Firebase Authentication In Flutter*. [online] Peter Coding. Available at: <<https://petercoding.com/firebase/2020/02/25/using-firebase-auth-in-flutter/>> [Accessed 11 June 2020].

[31] Guru99.com. 2020. *What Is BLACK Box Testing? Techniques, Example & Types*. [online] Available at: <<https://www.guru99.com/black-box-testing.html>> [Accessed 11 June 2020].

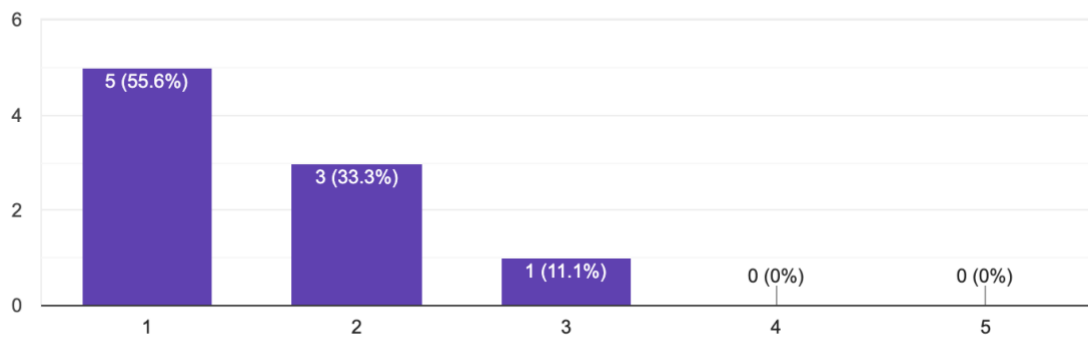
## Appendix

Github Link:

<https://github.com/usama-malik89/Computing-Project/tree/master/Implementation>

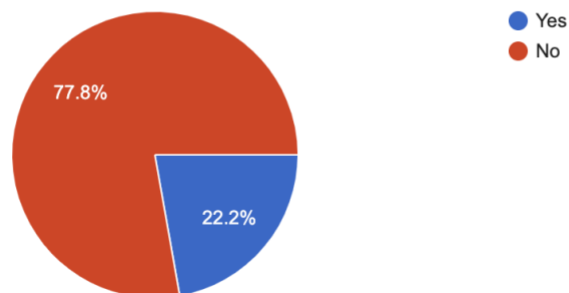
How was your experience when registering in the app?

9 responses



When using the app, did you ever feel any slowdowns?

9 responses



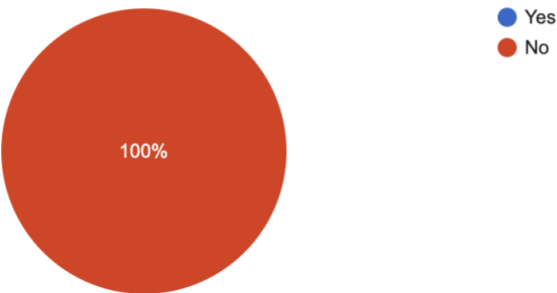
If you felt the app was slow, where was this?

1 response

when loading the cards in nearby

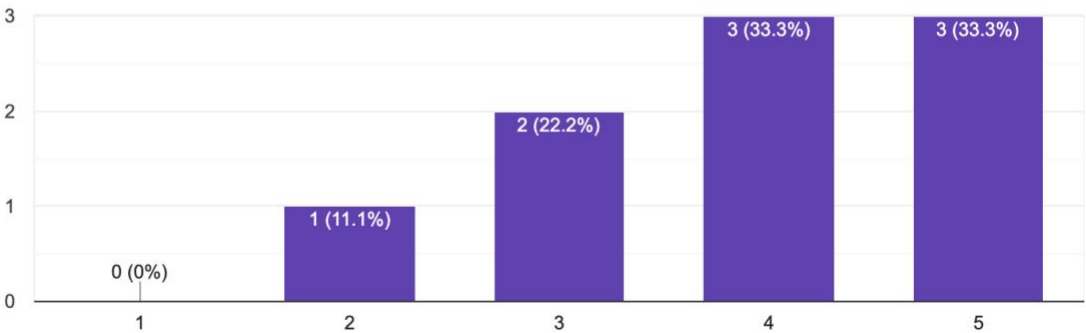
At any point of using the app did you feel confused?

9 responses



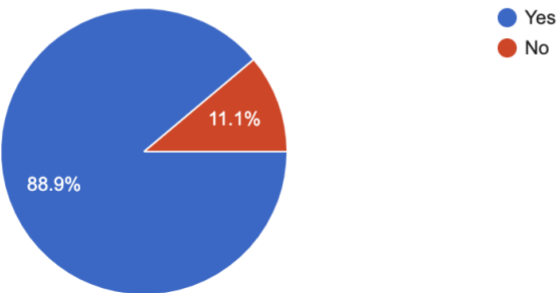
How easy was it to design your card?

9 responses



Do you think the card has enough customisation options?

9 responses



What customisation options would you want?

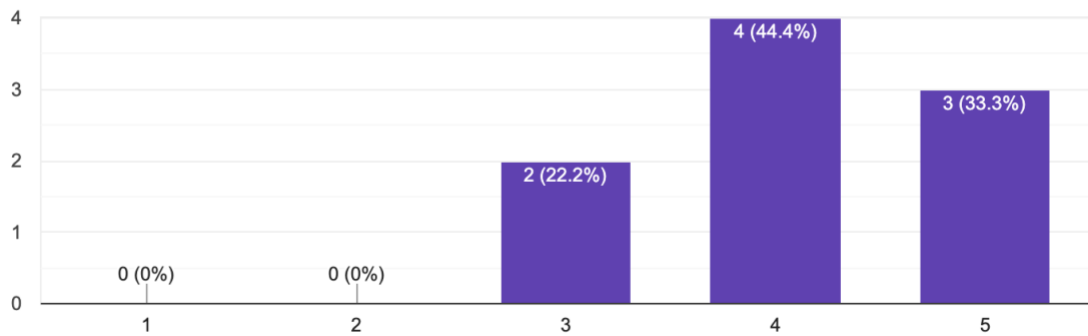
4 responses

- Change the text layout
- Customise my own card with a background image
- Font of text
- Maybe add social media information on the card



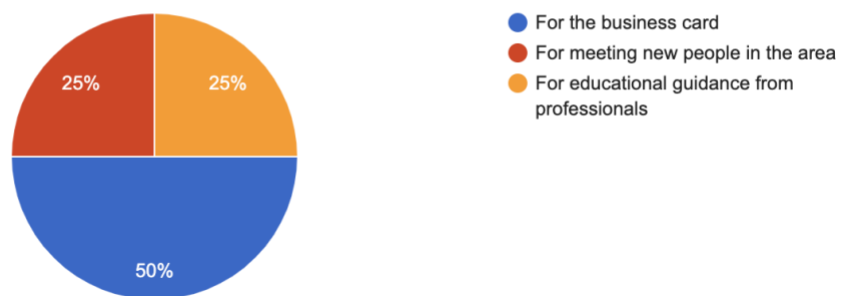
How difficult was it to scan a QR code?

9 responses



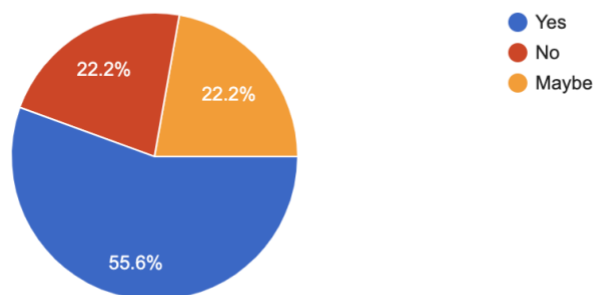
If you see yourself using the app, For what reason would you use it?

8 responses



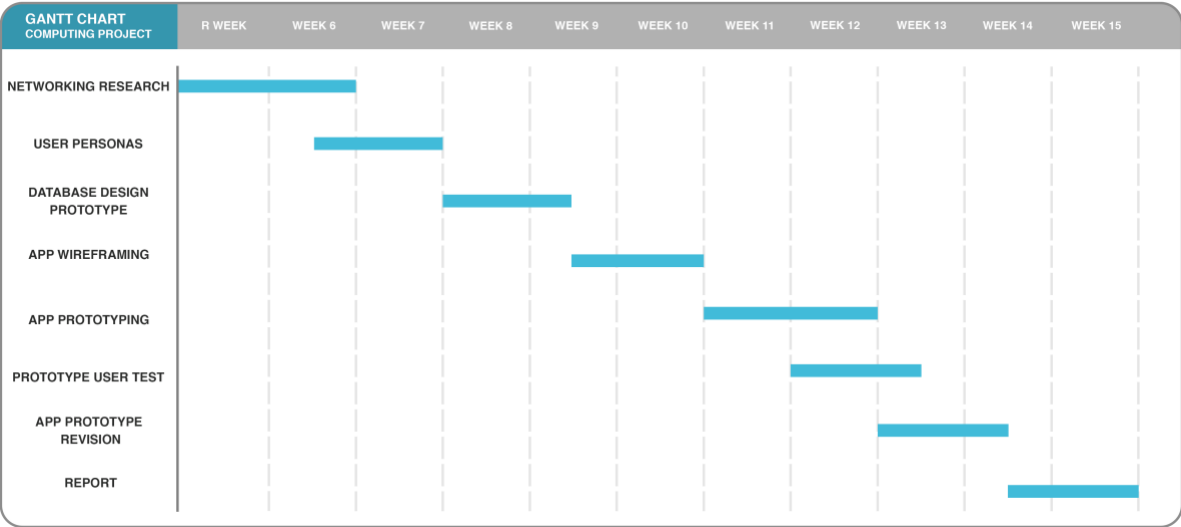
Do you see yourself using the app in the near future?

9 responses

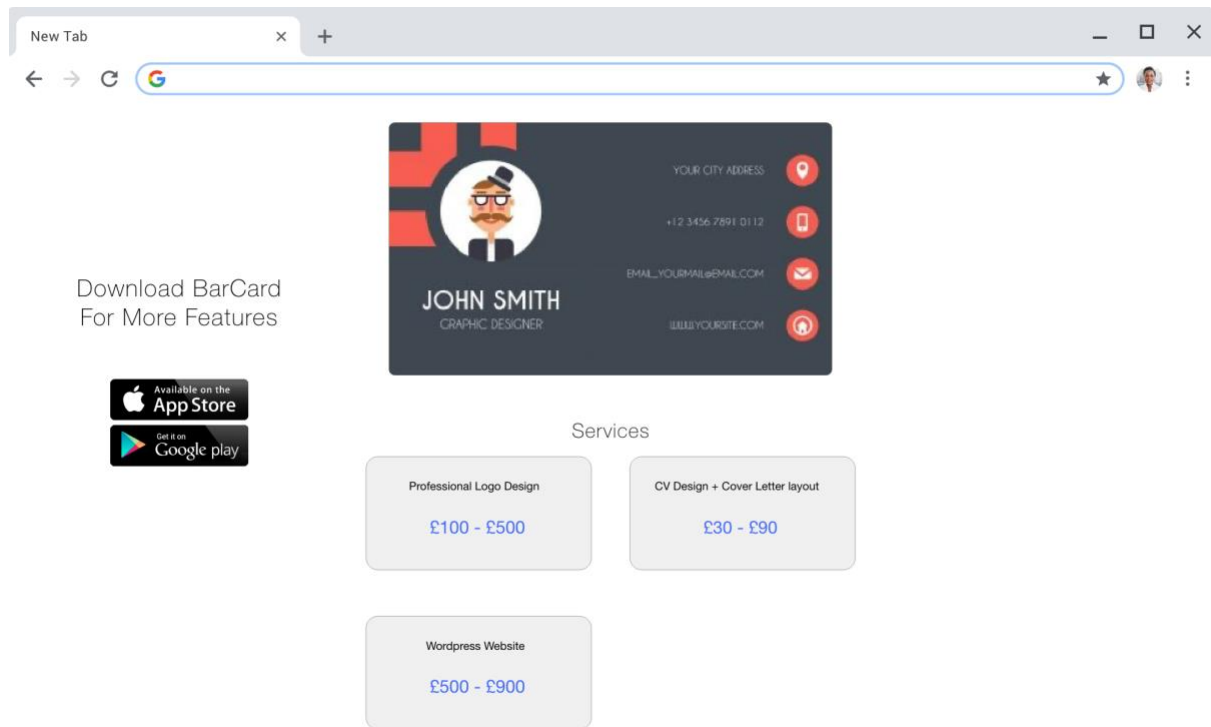




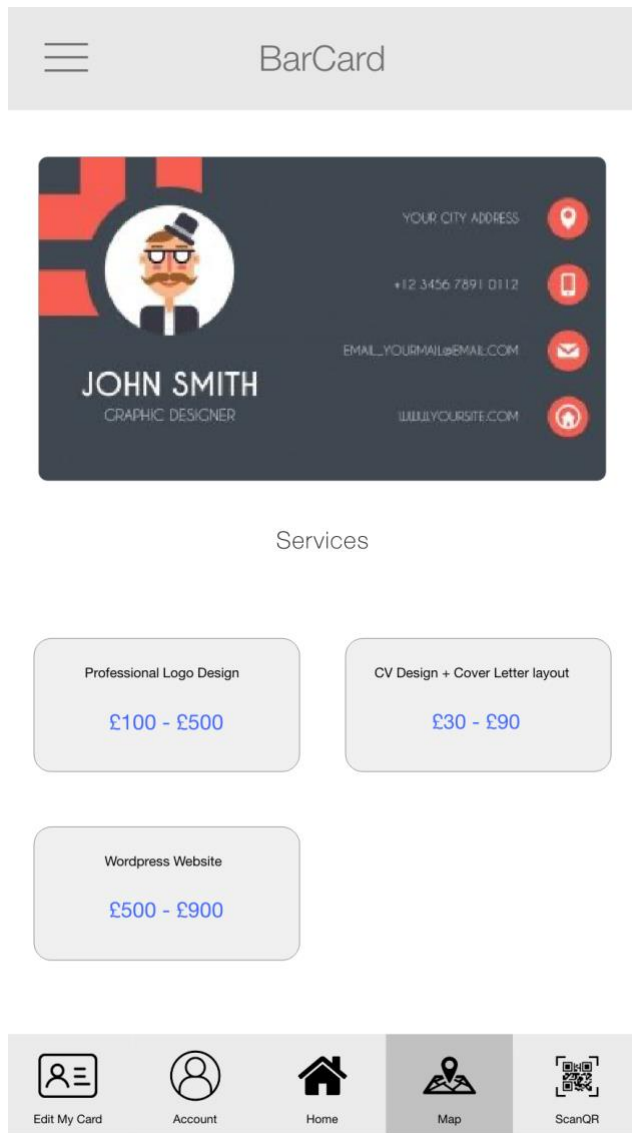
Appendix Figure 1 Logo Iterations



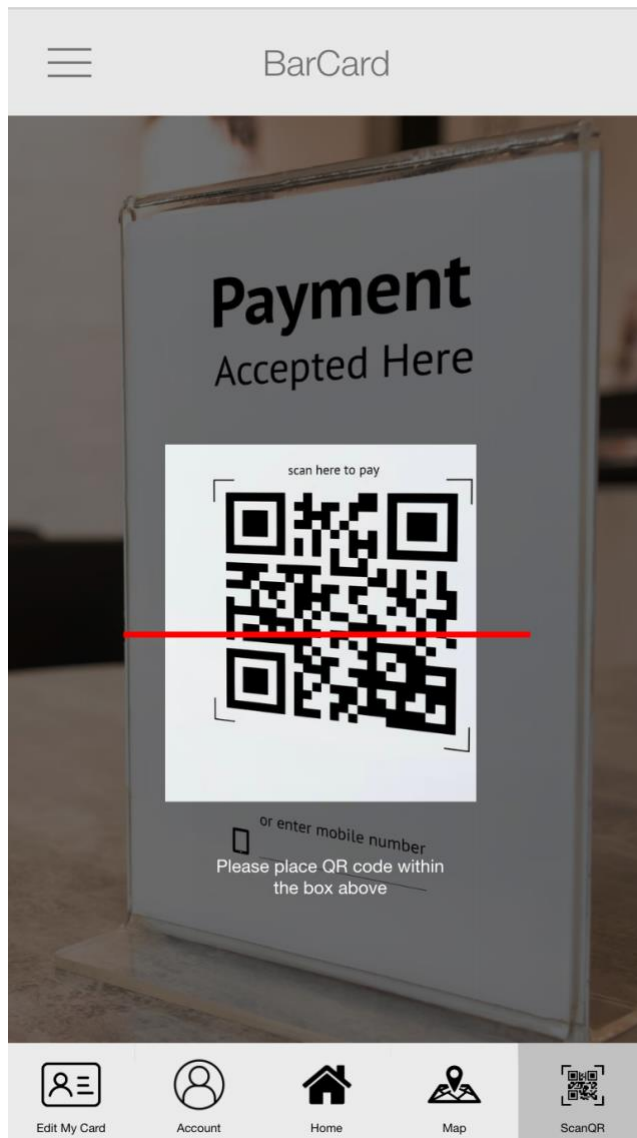
Appendix Figure 2 Initial Gantt chart



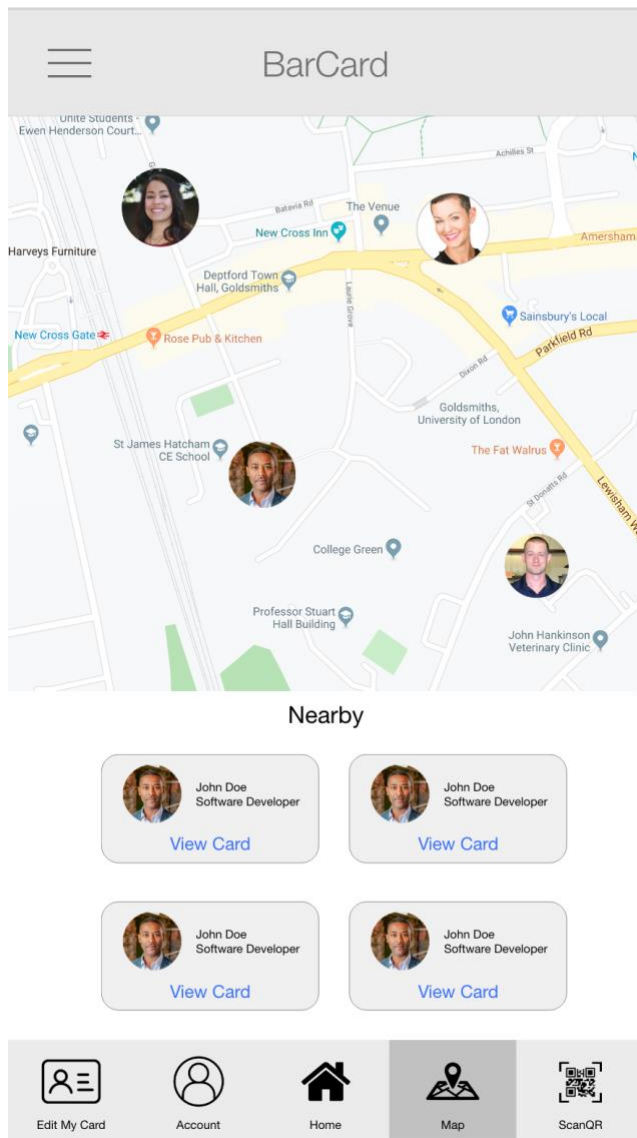
Appendix Figure 3 Initial Prototype Design 1



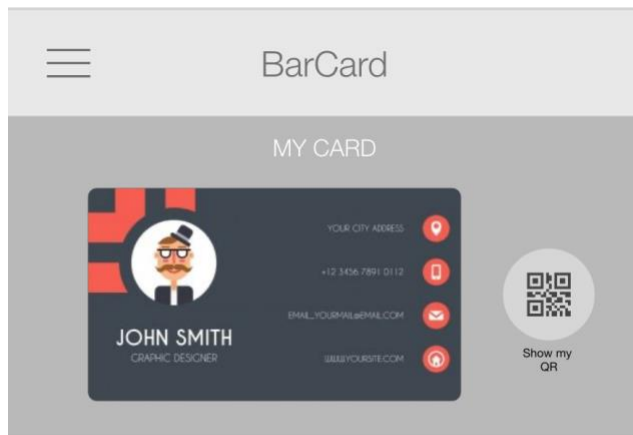
Appendix Figure 4 Initial Prototype Design 2



Appendix Figure 5 Initial Prototype Design 3



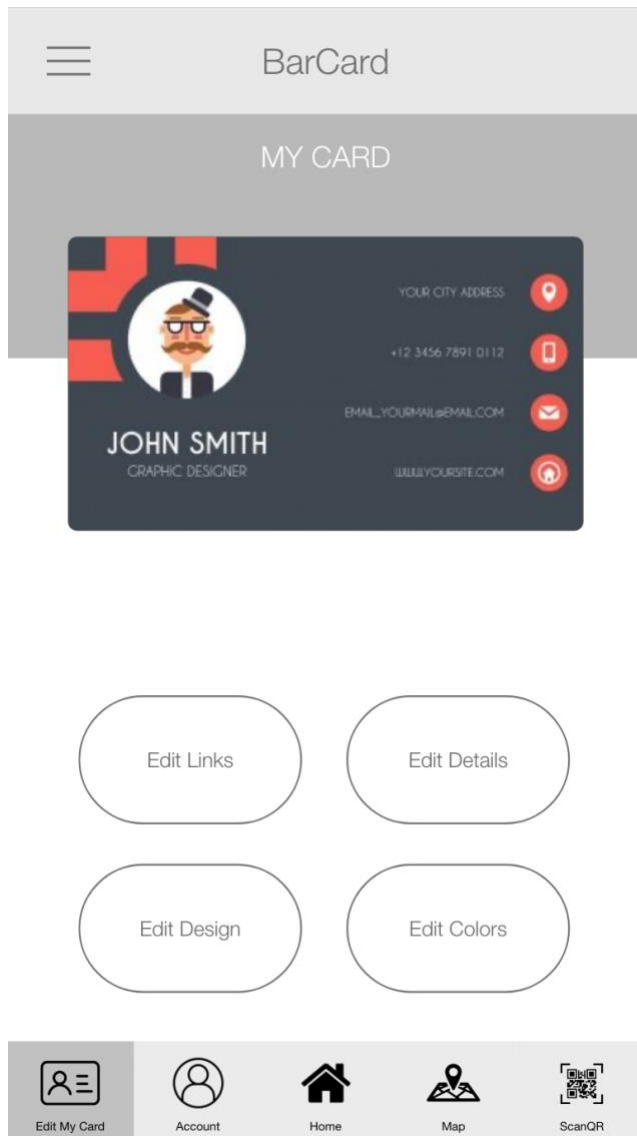
Appendix Figure 6 Initial Prototype Design 4



Saved Cards



Appendix Figure 7 Initial Prototype Design 5



Appendix Figure 8 Initial Prototype Design 6