

University of Dublin



TRINITY COLLEGE

***Gluten Free Food Scanner:  
An Android Application to Improve  
Access to Gluten Free Foods***

Ailbhe Redmond

B.A.(Mod.) Computer Science and Business

Final Year Project, May 2018

Supervisor: Mary Sharp

School of Computer Science and Statistics  
O'Reilly Institute, Trinity College, Dublin 2, Ireland

## 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.

---

Name

---

Date

## **Acknowledgements**

Firstly, I would like to thank my supervisor, Mary Sharp, for her support, guidance, and advice offered throughout the project development process. I would also like to thank Inmaculada Arnedillo-Sánchez, who stepped in to offer her own support and guidance when Mary was away from college.

A sincere thank you to my family and friends, whose support was invaluable both during this project and throughout my college career. Further thanks to any who testing the usability of my application and gave their honest and needed feedback.

Finally, I would like to thank David Cooney and Fergal O'Sullivan from the Coeliac Society of Ireland, who kindly provided me with the data for my database, and without whom this project would not have been possible.

## **Abstract**

This project aims to research, design, develop, and implement an Android mobile application that serves as a database of gluten free foods.

The overall goal of the project is to improve quality of life for people who need to follow a gluten free diet for any reason. Finding gluten free foods requires people to search through food ingredient labels or printed lists of foods, both of which are hugely time-consuming tasks and create frustration. A mobile phone application was chosen as determining whether a food is gluten free or not occurs most often outside the home.

Major application features include allowing users to search for items by typing in its name, viewing additional information about items, and using the phone camera to show and scan ingredient text on the phone screen. Together, these functionalities assist gluten free dieters in finding products that are safe to consume.

The gluten free food list was provided by the Coeliac Society of Ireland with their permission for use.

## Table of Contents

	<i>Page Number</i>
<b>ABBREVIATIONS</b>	<b>7</b>
<b>CHAPTER 1: INTRODUCTION</b>	<b>8</b>
1.1 Aims	9
1.2 Motivation	10
1.3 Reader's Guide	11
<b>CHAPTER 2: BACKGROUND &amp; RESEARCH</b>	<b>13</b>
2.1 Background	13
2.1.1 <i>Gluten Free Diet</i>	13
2.1.2 <i>Coeliac Disease</i>	14
2.1.3 <i>Smartphone Penetration</i>	15
2.2 Existing Applications	16
2.2.1 <i>Coeliac UK</i>	16
2.2.2 <i>Coeliac UK Gluten Free Food Checker</i>	17
2.2.3 <i>Food Maestro</i>	18
2.2.4 <i>Gluten Free Scanner UK</i>	18
2.2.5 <i>Influence of Existing Applications</i>	19
2.3 Technology	20
2.3.1 <i>Development Environment</i>	20
2.3.2 <i>Database</i>	20
2.3.4 <i>Text Recognition Libraries</i>	21
2.4 Data	22
2.5 Ethics	22
<b>CHAPTER 3: REQUIREMENTS AND DESIGN</b>	<b>25</b>
3.1 Requirements	25
3.1.1 <i>Functional Requirements</i>	25
3.1.2 <i>Non-Functional Requirements</i>	26
3.2 System Design	27
3.3 Back-End Design	28
3.4 Front-End Design	31

3.5 Prototypes	33
3.5.1 Low-Fidelity Prototypes	34
3.5.2 High-Fidelity Prototypes	35
3.6 Final Design	36
<b>CHAPTER 4: TECHNOLOGIES USED</b>	<b>38</b>
4.1 Android	38
4.2 Firebase	38
4.2.1 Firebase Realtime Database	39
4.2.2 Firebase Authentication	39
4.2.3 Firebase Admin and Cloud Functions	40
4.2.4 Firebase Notifications	40
4.3 Google OCR	40
<b>CHAPTER 5: IMPLEMENTATION</b>	<b>42</b>
5.1 Development Model	42
5.2 Database Search	44
5.3 Product Information	51
5.4 User Accounts	53
5.5 Ingredient Recognition	60
5.6 Implementation Difficulties	65
5.7 Code Repository	66
<b>CHAPTER 6: TESTING</b>	<b>68</b>
6.1 Incremental Testing	68
6.2 Usability Testing	69
6.2.1 System Usability Scale	70
6.2.2 Test Environment	70
6.2.3 Test Results	71
6.2.4 Feedback from User Testing	72
<b>CHAPTER 7: BUSINESS PLAN</b>	<b>74</b>
7.1 Value Proposition	74
7.2 Key Partners	75
7.3 Key Resources	75

7.4 Key Activities	75
7.5 Customer Segments	77
7.6 Customer Relationships	77
7.7 Channels	77
7.8 Cost Structure	78
7.9 Revenue Model	78
<b>CHAPTER 8: CONCLUSION</b>	<b>80</b>
8.1 Evaluation	80
8.2 Difficulties	81
8.3 Further Work	82
8.4 Closing Remarks	84
<b>CHAPTER 9: REFERENCES</b>	<b>86</b>
9.1 Endnotes	86
9.2 Bibliography	89
<b>APPENDICES</b>	<b>90</b>
Appendix A: Prototypes	90
Appendix B: Final Design	94
Appendix C: Ethics Application Documents	97
Appendix D: Business Model Canvas	108
Appendix E: Resources	109
Appendix F: Presentation Slides	110
<b>CODE REPOSITORY</b>	<b>CD on rear cover</b>

## Abbreviations

API	Application Programming Interface
CSOI	Coeliac Society of Ireland
HTTPS	HTTP Secure
JSON	JavaScript Object Notation
NoSQL	Non-Structured Query Language
OCR	Optical Character Recognition
SUS	System Usability Scale
UI	User Interface
UML	Universal Modelling Language



## Chapter 1: Introduction

In recent years, the term “gluten free” has become a buzzword associated with dieting and healthy eating. For many, eating gluten free is not a choice, but a necessity for everyday life. Gluten is a protein found in many grains and their derivatives, and thus is present in many food products. Determining whether or not a food is gluten free is a time-consuming and often frustrating process. In order to do so, people must read ingredients labels or refer to printed lists of gluten free products.

The Coeliac Society of Ireland, a charity dedicated to helping those with coeliac disease to follow a gluten free diet, offer one such list. Their printed book contains a list of products that have been certified gluten free for the year. Finding a product in this book takes time, is often awkward, and is no longer the most accessible method thanks to advances in technology. As searching for gluten free foods will most likely happen on the go, this project will create a mobile application that aims to offer a replacement for the existing system, improve speed of searching, and offer other functionality to help make finding gluten free foods easier for consumers.

This application will allow users who are logged in to view and search a database of gluten free foods identical to what was once contained in the printed book. Allowing users to search for a particular term will greatly increase their speed in finding products. The application will also allow users to use their phone camera to read and copy food ingredients to the screen in an easy-to-read font. The application will also search through this ingredients list and notify the user of any gluten-containing ingredients.

There are currently a number of similar applications available on the Google Play Store, but none are targeted to the Irish market, and none provide the same visualisation technology. This creates an opportunity for an app designed for Ireland to enter the marketplace and gain market share.

## **1.1 Aims**

The aim of this project is to create an application that provides a platform to help users to find gluten free foods more easily, efficiently, and effectively. This application will be developed for Android and will enable the following operations:

- Allow users to enter a search string to search for a product in the database, and view the results of that search query
- Allow users to view all of the products belonging to a particular category
- Allow users to add products to their favourites, and view a list of their favourites
- Allow users to use their phone camera to read an image of food ingredients, view those ingredients on their phone screen, and give further information about the ingredients
- Provide information about gluten free diets, food labelling laws, and other relevant issues
- Allow all of the above to be done after a login process

All of the aims are SMART – **S**pecific, **M**easurable, **A**ttainable, **R**elevant, **T**ime Bound. They have specific functionality associated with them, are measured by whether or not they are achieved, are relevant to the overall functionality of the application, and are attainable within the development time period. Separate from this list of goals, the application should also be easy for users to use, with simple flows from screen to screen.

The author also had personal aims and goals relating to the implementation of the project. These goals include:

- Learn the necessary skills to develop a smartphone application from start to finish, including:
  - Using Java and JavaScript to develop the application on the Android Studio platform
  - Building an application with an easy-to-use and aesthetically pleasing user interface using XML

- Using GitHub as a code repository
  - Learning key skills for maintaining and updating an Android application
- Apply skills learned throughout college, from both a business and computer science perspective, including:
  - Improving and expanding knowledge of Java, Javascript, and XML languages
  - Building knowledge of database structures and management beyond relational SQL databases
  - Applying development, project management, business strategy, and presentation techniques learned throughout the degree program

## **1.2 Motivation**

The primary motivations for this project are the author's own frustration with finding gluten free products in a printed book, and the desire to provide a better, technology-based solution. The advent of smartphones has created a real opportunity to create modern, mobile solutions to existing problems. In this instance, a smartphone application could greatly improve user experience by speeding up access to gluten free food offerings through the use of a searchable database. It will also allow users to take advantage of something they are already carrying (a smartphone) rather than asking them to carry a physical book with them. Technology also creates opportunities for other functionalities to be added to take advantage of other smartphone capabilities, including barcode scanning and image-to-text ingredient interpretation.

A secondary motivation was the fact that, at present, no technology-based solution for this problem exists in the Irish market. While applications exist in the UK, the products listed do not always apply to Ireland, and the applications have certain shortcomings. The Irish market has a large proportion of both gluten free dieters and smartphone users, representing a large potential market for the application.

### **1.3 Reader's Guide**

Following this introduction, this report contains the following chapters:

#### **Chapter 2 – Background & Research**

This chapter will outline the background of the gluten free diet, existing applications on the market, and technology researched for the project.

#### **Chapter 3 – Requirements and Design**

The chapter details the requirements of the application and the stages of application design. This includes sketches, mock-ups, and the finished application design.

#### **Chapter 4 – Technologies Used**

This chapter discusses the technologies used for implementing the application, detailing Android, Firebase, and Google Text Recognition API.

#### **Chapter 5 – Implementation**

The implementation chapter details the incremental software development model used in developing the application, along with in-depth explanations of the functionality added during each increment. Each function will be outlined with code snippets, and application functionality shown using screenshots.

#### **Chapter 6 – Testing**

The user testing chapter will outline techniques chosen to test the application, including user testing. It will discuss user evaluation of the application in detail.

#### **Chapter 7 – Business Plan**

This chapter discusses the business potential of the application, including potential revenue streams and partners for the future.

## **Chapter 8 – Conclusion**

This chapter will conclude the project, evaluating successes and problems, future improvements, and what was learned throughout the implementation of the project.

## Chapter 2: Background & Research

### 2.1 Background

Gluten is a protein found in wheat, barley, and rye grains. Gluten acts as a glue, helping food to maintain its shape and stick together. Gluten does not naturally occur in oats but may also be found in oat products due to cross contamination of grains in the growing, harvesting, and milling process. There are many illnesses associated with consuming gluten-containing grains, including coeliac disease, gluten intolerance, and gluten allergies.

#### *2.1.1 Gluten Free Diet*

A gluten free diet is the only treatment for many medical conditions associated with the gluten protein, including coeliac disease. In recent years, the diet has gained popularity as a method of weight loss and improving digestive health. Books such as *Wheat Belly* by Dr William Davis and testimonials from celebrities about the health benefits of the diet led to this increasing popularity <sup>(1)</sup>. Other popular diets, such as Palaeolithic and Ketogenic, also remove gluten in addition to other ingredients. The autoimmune protocol diet, used in the treatment of various autoimmune and inflammatory bowel diseases, also excludes gluten from the diet <sup>(2)</sup>.

In 2017, a Bord Bia survey found that 20 percent of an Irish nationally representative sample regularly purchased gluten free foods. Of this group, 22 percent suffered from coeliac disease, a gluten allergy, or a wheat allergy, and so were medically required to remove gluten from their diet. A further 40 percent identified as being intolerant or sensitive to gluten-containing products. The remaining 38 percent followed a gluten free diet by choice for its perceived health and weight benefits. Many of the shoppers surveyed also purchased foods for other free-from diets, such as lactose free or soy free <sup>(3)</sup>.

Food products are considered gluten free if they contain 20 parts per million or less of gluten <sup>(4)</sup>. This level is considered both low enough to be detectable in testing and to not cause

damage in coeliac disease sufferers. This labelling requirement applies to both processed and unprocessed food products. Food products cannot carry a “Gluten Free” label if they do not comply with this requirement. Because of this, gluten free foods may potentially contain gluten containing ingredients, but in low levels. This may still be a problem for those with a gluten allergy but should be suitable for coeliacs. Such ingredients include gluten free wheat starch and barley malt vinegar. If a product contains these ingredients and is labelled gluten free, it is safe for coeliacs to consume. If not, the product may contain greater than 20ppm and thus is unsafe.

### *2.1.2 Coeliac Disease*

Coeliac disease (also spelled celiac disease) is an auto-immune disease causing a reaction to the gluten protein found in wheat, barley and rye. Sufferers react abnormally to ingesting gluten, causing the small intestine to become inflamed and damaged. This causes malabsorption of nutrients. The only treatment for coeliac disease is a gluten free diet, which, for most patients, reverses any damage to the intestine. Some coeliac disease sufferers may be asymptomatic, but most display a range of varied symptoms <sup>(5)</sup>.

Coeliac disease is thought to affect approximately one in 100 people in the general population. Based on an Irish population of 4.773 million <sup>(6)</sup>, this would equate to a potential coeliac population of approximately 47,000 people. The Coeliac Society of Ireland (CSOI), a charity created to improve quality of life for coeliacs, currently has over 5,000 members <sup>(7)</sup>. They propose that for every coeliac diagnosed in Ireland, there are five to ten people undiagnosed. However, many of the symptoms are very common, or may appear unrelated, making it easy for the disease to go unnoticed. Symptoms include diarrhoea, cramping, bloating, weight loss, depression, mouth ulcers, alopecia, infertility, heartburn, fatigue, osteoporosis/osteopenia, anaemia, and general ill-health.

It is believed that several genes contribute to susceptibility to coeliac, although not all genes related to the disease have been identified yet. Because of this genetic link, direct

family members of those diagnosed with coeliac are at higher risk of also being diagnosed with the disease. First degree relatives are believed to be seven to ten times more likely to develop coeliac disease than members of the general population <sup>(8)</sup>.

### *2.1.3 Smartphone Penetration*

A Deloitte study of Irish consumers in 2017 found that, on average, 90 percent of Irish adults owned a smartphone. Ownership rates were highest among younger consumers, but rates only fell below 90 percent for the over 60s population, which still maintained an adoption rate of 83 percent <sup>(9)</sup>. A previous study by Deloitte in 2016 found that 18–24 year olds were the most frequent users of smartphones, with 87 percent of the group using their devices always or very often during daily activities such as shopping, watching television, meeting friends, or using public transport <sup>(10)</sup>.

These statistics show that smartphone applications have a large potential market in Ireland. In terms of operating systems, Android is the dominant player in the market. Android OS was first released by Google in 2008, and currently has the largest worldwide user base of mobile operating systems <sup>(11)</sup>. As of February 2018, Android smartphones made up 52.5 percent of the Irish market. Apple's iOS followed behind at 46.7 percent <sup>(12)</sup>. For this reason, the author chose to pursue creating an Android application.

In 2017, the CSOI performed a survey of members to gauge interest in a potential smartphone application. The majority of members expressed interest in a smartphone application with database searching capabilities. Members were asked about their interest in an application with barcode scanning functionality – whereby the application would determine if a product was gluten free or not by scanning its barcode – which most members also expressed interest in. However, 97.4 percent of people surveyed said they would use an application without this functionality <sup>(13)</sup>. The Society plan to create their own proprietary application but have been slow to do so due to high development costs and limited availability of funds.



## 2.2 Existing Applications

There are currently no applications comparable to the proposal available on the Irish market. However, there are several currently available in the United Kingdom. Two such applications are managed by Coeliac UK, the British equivalent to CSOI, while a further two are privately owned. The two Coeliac UK apps are targeted specifically at coeliacs and require a membership (costing £24 per year <sup>(14)</sup>) to use. The Gluten Free Scanner UK is also targeted at coeliacs and costs £3.99 to download <sup>(15)</sup>. The FoodMaestro application is free to use and offers support for a variety of allergies. All of the applications offer database search and barcode scanning functionality, and some offer additional information pages and the ability to add products to user favourites.

### 2.2.1 Coeliac UK

The first Coeliac UK application is straightforward to use, and the barcode scanner is responsive and works quickly. The use of green text to show which products are suitable for the gluten free diet should be universally understood as green is seen as a positive colour.

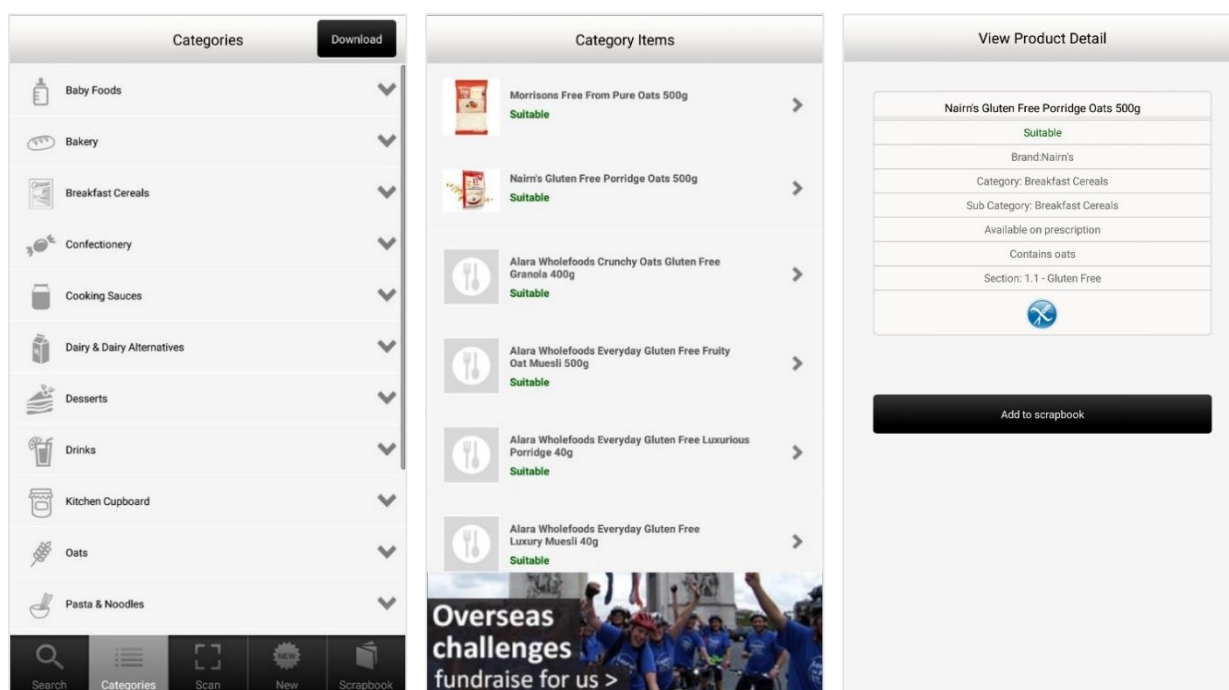
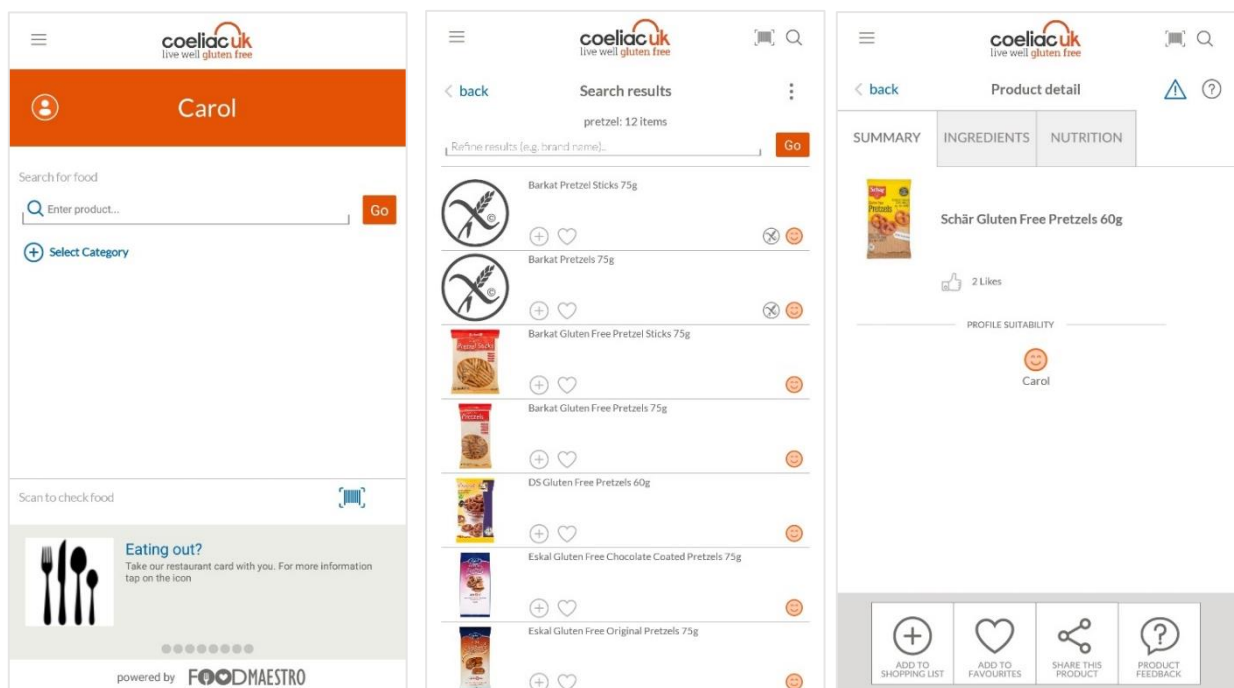


Image 2-1: Screenshots from Coeliac UK app

However, there are certain failings with the application. The user interface is generally unappealing, and font size is small and difficult to read, inhibiting usability. The bottom navigation bar is not present on all screens, making it difficult to navigate between pages. Categories are not labelled as the user passes through the application, leaving it up to the user to remember what they are doing or where they are in the application. This goes against design principles that suggest systems should make the users aware of where they are so that nothing is lost if they become distracted or are interrupted <sup>(16)</sup>. The application also loses usable space through the inclusion of large banner advertisements and empty photo boxes for products.

### 2.2.2 Coeliac UK Gluten Free Food Checker



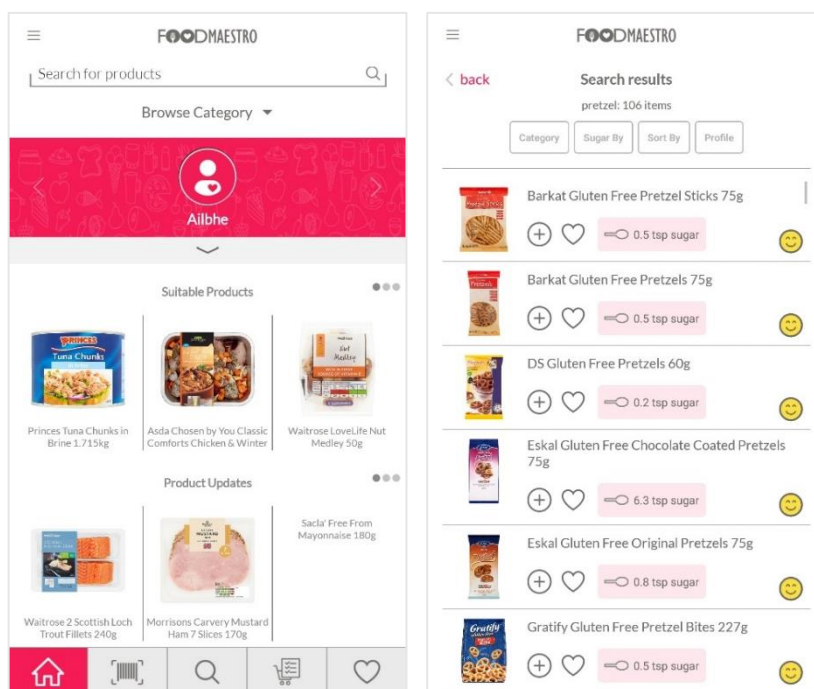
**Image 2-2:** Screenshots from Coeliac UK Gluten Free Food Checker app

The second Coeliac UK application has the same basic features with an updated user interface. It has additional features of customisable allergen profiles and the ability to refine searches by limiting them to certain categories. Navigation is handled by a dropdown menu visible from all pages instead of the moving navigation bar in the previous application, improving overall usability. The application uses the universal symbol of a smiley face to

signify product suitability. The user interface is much easier to use than the previous app version, but once again uses small text that is difficult to read.

### 2.2.3 FoodMaestro

FoodMaestro are a private company focused on developing mobile applications for allergy sufferers. The previous Coeliac UK application was developed in partnership with FoodMaestro. FoodMaestro's own app is designed for many allergies and has near identical user interface and functionality to the Coeliac UK application. Major differences are the use of a bottom navigation bar on some pages, a different homepage and different colour scheme. The navigation bar is not present on all screens, again limiting its use. Where Coeliac UK had an empty homepage, this app makes better use of this space to display suitable products based on the user's allergen profile.

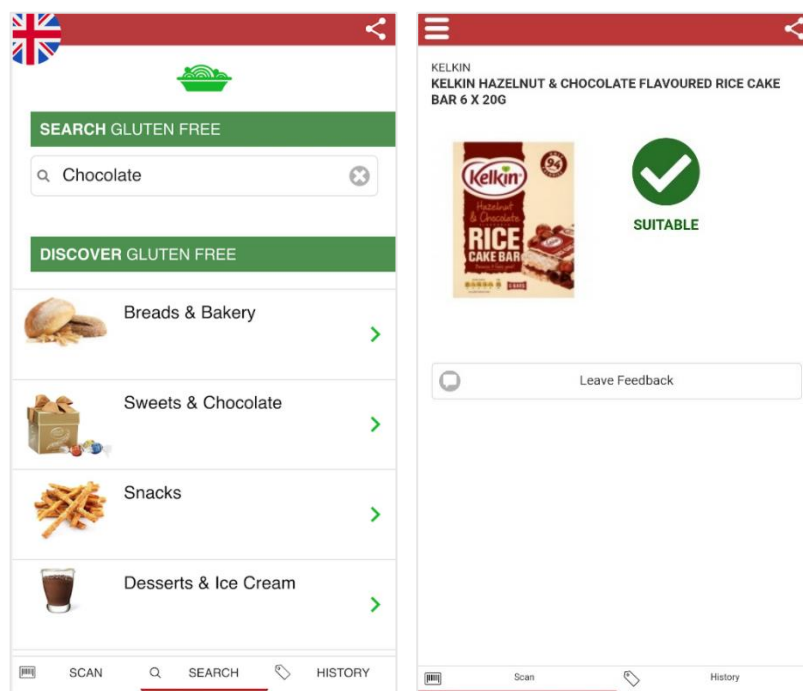


**Image 2-3:** Screenshots from FoodMaestro

### 2.2.4 Gluten Free Scanner UK

Gluten Free Scanner UK offers both paid and free versions of their application. The free version is limited to the barcode scanner, while the paid version includes a database of

gluten free products. The application has a simple user interface, with a bottom navigation bar and dropdown menu to navigate from page to page. The font size is larger than the other applications on the market, making it easier to read the information on the screen. The use of two universally recognised symbols – the colour green and a check mark – to signify suitability should be recognisable to all users. This application also has the highest user rating on the Google Play Store, with an average score of 4.6 out of 5 stars<sup>(17)</sup>. The other applications range from a rating of 2.9 to 3.4 out of 5<sup>(18–20)</sup> from a similar number of reviews.



**Image 2-4:** Screenshots from Gluten Free Scanner UK

### 2.2.5 Influence of Existing Applications

The existing applications all have features that would be useful for this project. A high-quality database search function is essential to the project's success. Without this, app functionality will be hugely limited. The existing applications also had influence over the planned user interface for the project. The use of universal symbols and colours to signify suitability for the diet is also useful and helps to give users a visual cue. All of the existing applications suffered from issues with text readability and navigation, so this project endeavours to be easier to read and navigate.

## **2.3 Technology**

In order to develop the best possible application, the author undertook research into development environments, database structures, and image recognition APIs that could be used. As previously discussed, this project will develop an application for the Android mobile operating system. In keeping with this, Android-specific development environments and complementary databases were investigated.

### *2.3.1 Development Environment*

Google's Android Studio was chosen as the development environment for the project for a variety of reasons. Previous experience using the software paired with being Google's own development environment made Android Studio an attractive option. It offers integrated Google tools, is free to use, and provides templates for user interface elements to make them easier to implement. Android Studio also offers integration with Git for version control and Google's Firebase cloud offerings, which include a database that was considered for the project.

### *2.3.2 Database*

Three major databases were considered for the project: SQLite, MongoDB, and Firebase Realtime. SQLite offers a relational database, while both MongoDB and Firebase are non-relational.

*SQLite* is a relational database system that comes as part of the Android operating system and can be embedded in applications. It works well for storing simple data objects but can be difficult and tedious to implement. The service was investigated due to the author's previous experience in SQL and relational databases. However, for this project a relational database was not necessary, so other options could be considered. The author also felt that the additional time required to connect the database to the application would take time away from developing application functionality.

*MongoDB* is a NoSQL, non-relational database. It integrates with Android through a RESTful API. The database must be hosted somewhere on the web to be accessed. Data can be uploaded in JSON format, allowing for easy data management and more flexible data structure than MySQL allows. However, the costs and complications of hosting the database online made the offering less attractive.

*Firebase Realtime* is part of Google's Firebase Cloud Services. It offers a NoSQL database that reflects data changes in real time. It is easily integrated into the Android platform and allows for data upload in JSON format. The format works well for fairly simple data structures but can become difficult to manage as complexity increases. It can be used for both web and mobile applications, allowing for potential expansion to web if needed. The database is free to use up to 1GB stored and 100 connections, which allows for freedom during development and testing without costs. Scaling for the future, Firebase offers customisable plans depending on usage, again saving on costs. Other Firebase services include authorisation and analytics, which could be used for application log in and gathering user information. The author had limited previous experience using Firebase for Android. For these reasons, the author chose Firebase Realtime Database as the database for the project.

### *2.3.3 Text Recognition Libraries*

There are three major optical character recognition APIs available for use on Android: Google, Microsoft, and AWS. A comparison of the APIs by Dataturks in 2017 found that Google Text Recognition read text more accurately than alternative libraries offered by Microsoft or AWS <sup>(21)</sup>. Google's Text Recognition API is part of their Cloud Vision library, which includes additional features such as a barcode scanner that could be used in future improvements to this application. The accuracy coupled with the use of other Google libraries helped the author to choose Google Text Recognition for the application.

## **2.4 Data**

The data used in this application is a sample of the Coeliac Society of Ireland's gluten free products database. It was given to the author with permission for use in the project. As the data belongs to the Society, it conforms to the structure used by CSOI in their printed gluten free food list. All products included in the database were guaranteed to be gluten free by the CSOI when it was shared with the author in July 2017. The sample of 470 entries represents approximately five percent of their full database.

## **2.5 Ethics**

The basis of all ethics is to "do no harm", referring to intentionally harming others through a person's actions. While there is not yet a code of ethics for developing new technologies, any technology that relates to an individual's health and wellness has associated ethical considerations. A developer must ensure that no harm is done to the user of any new technology. During planning and development of this project, certain ethical concerns arose and had to be considered in order to avoid having negative effects on users.

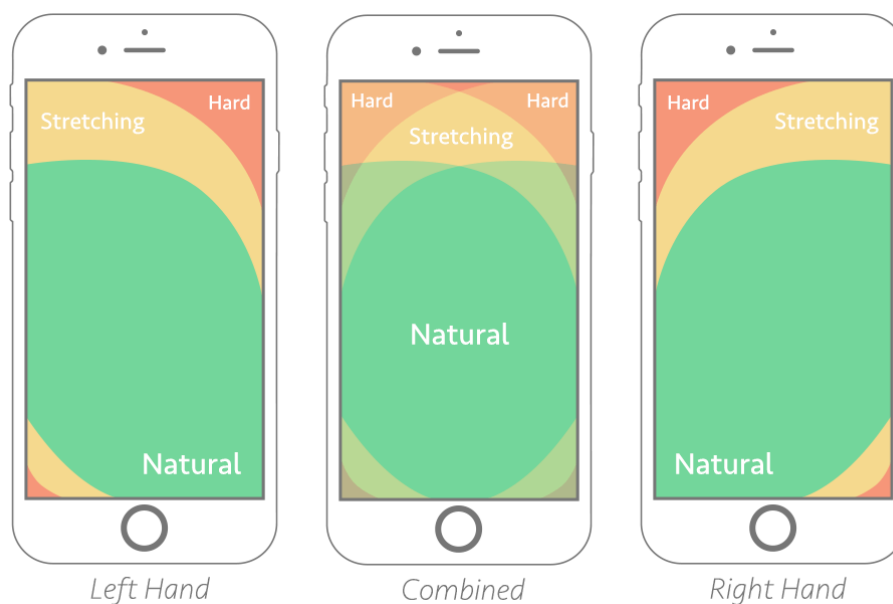
When providing a database related to health, the onus is on the developer to both guarantee the accuracy of database contents and to maintain and monitor the database to ensure it is kept up to date. In this project, where data is provided by the Coeliac Society of Ireland, continued partnership between the charity and the author will be required to maintain data accuracy. The author also has a responsibility to continually check with the Society to ensure any data changes are reflected in the application. The developer must also ensure the security of the database, such that only authorised individuals can alter information. This will be controlled by strict database access rules.

The ingredient scanner functionality requires the developer to balance between medical advice and suggestion. Limitations of text recognition APIs mean that the scan cannot be guaranteed to be correct. If the application does not completely read the ingredient string and instructs that a product is gluten free when it is not, the user could come to harm. For

this reason, the application must avoid giving clear statements such as “Contains wheat”, and instead give advice like “Ingredients mention wheat”. In this way, the user is still responsible for reading the ingredients and ensuring it is compatible with their diet.

The user account functionality must ensure the privacy and security of user information. Users must know that their data is not at risk by sharing it with the developers of an application. Having chosen Firebase Realtime Database for the application’s database functionality, the author decided to also pursue the use of Firebase Authentication to register users. As a Google service, the author felt that the security level was high. The service uses the same or similar functionality to Google Sign-in and Chrome Password Manager, two highly regarded services, increasing confidence in its security.

Application ethics can also relate to the user interface of an application. Ergonomics, the study of working efficiency, can be influenced by layout and function location.



**Image 2-5:** Thumb-mapping of mobile phone users by handedness <sup>(22)</sup>

On today’s large smart phones, certain parts of the screen require the user to stretch or strain to reach them. User interface design should take this into account and make sure major functionality is easily accessible and ergonomically friendly, increasing user efficiency



and wellbeing. This idea will be kept in mind when designing the user interface for this project, keeping major functionality in areas that are easy for users to reach.

Overall, the major ethical consideration was to balance between the responsibilities of the consumer to guard their own health and the responsibility of the developer to provide accurate information. In order to highlight this user responsibility, the author decided to include a disclaimer in the application information screen. This disclaimer instructs users that while the application is as up-to-date as it can be, errors may still occur. As such, the author advises all application users to check product packaging and ingredients themselves before consuming, even if the product appears in the database. This reminds individuals of their own role in management of their gluten free diet.

## Chapter 3: Requirements and Design

Developing clear system requirements is critical to developing a high-quality system. The requirements for this project were influenced by the research undertaken in the previous chapters which helped to develop both functional and non-functional requirements of the application.

Once clear requirements were developed, system design began. The overall application design drew on the results of research, with additional focus on Android design guidelines. Application design was also influenced by design heuristics studied during the author's Software Engineering module in Junior Sophister year and the Human Factors module in Senior Sophister year. These heuristics helped to develop system features and layout. Design steps included the creation of both low- and high-fidelity prototypes that would inspire the final application user interface.

### **3.1 Requirements**

#### *3.1.1 Functional Requirements*

Functional requirements define what system features should accomplish. Functional requirements for the project were gathered through analysing the project aims and examining existing applications on the market to see what features they had. The functional requirements for the project are as follows:

- **Ability to create user accounts**

Users should only be allowed to view database information when logged in to protect the data from misuse. A list of personal product favourites will be attached to each account. This list should be viewable from the user's profile.

- **Ability to search the database for products**

Once a user is logged in, they should be able to search for products in the product database by product name or brand. All potential results for the search should be displayed on the screen. Products should also be viewable by category.

- **Ability to view specific product information**

Products should have individual pages giving details about their name, brand, and package size. Users should be able to add products to their favourites from these product screens.

- **Ability to scan ingredients using the phone camera**

Once a user is logged in, they should also be able to use their phone camera to scan a product's ingredients. These ingredients should then be displayed on the phone screen and searched for any gluten containing ingredients.

- **Provide information about gluten free diets**

The application should also provide information about what gluten is, the gluten free diet, and other related conditions. This information should also include a disclaimer about data accuracy.

### *3.1.2 Non-Functional Requirements*

Non-functional requirements refer to additional aspects of the system that affect system quality. This includes elements such as usability, learnability, and reliability. The following non-functional requirements were identified:

- **Usability**

The application should be easy to use, with a clear layout and clear navigation between pages. It should be simple to understand and avoid any user frustration. This will be achieved through a well-structured user interface, with clear labelling and logical flow for the user to follow.

- **Learnability**

The application should be easy for users to learn and become proficient in using. This is seen as a subset of usability, and thus is achieved by ensuring high usability.

- **Testability**

In order to ensure that the application works correctly, it should be easy to test and tested throughout development. Thorough testing helps to ensure that no major problems exist within the application and improves user experience by reducing crashes and failures.

- **Maintainability**

The application code and database should be built in such a way that it can easily be maintained in the future to ensure that the application runs smoothly throughout its lifetime.

- **Extensibility**

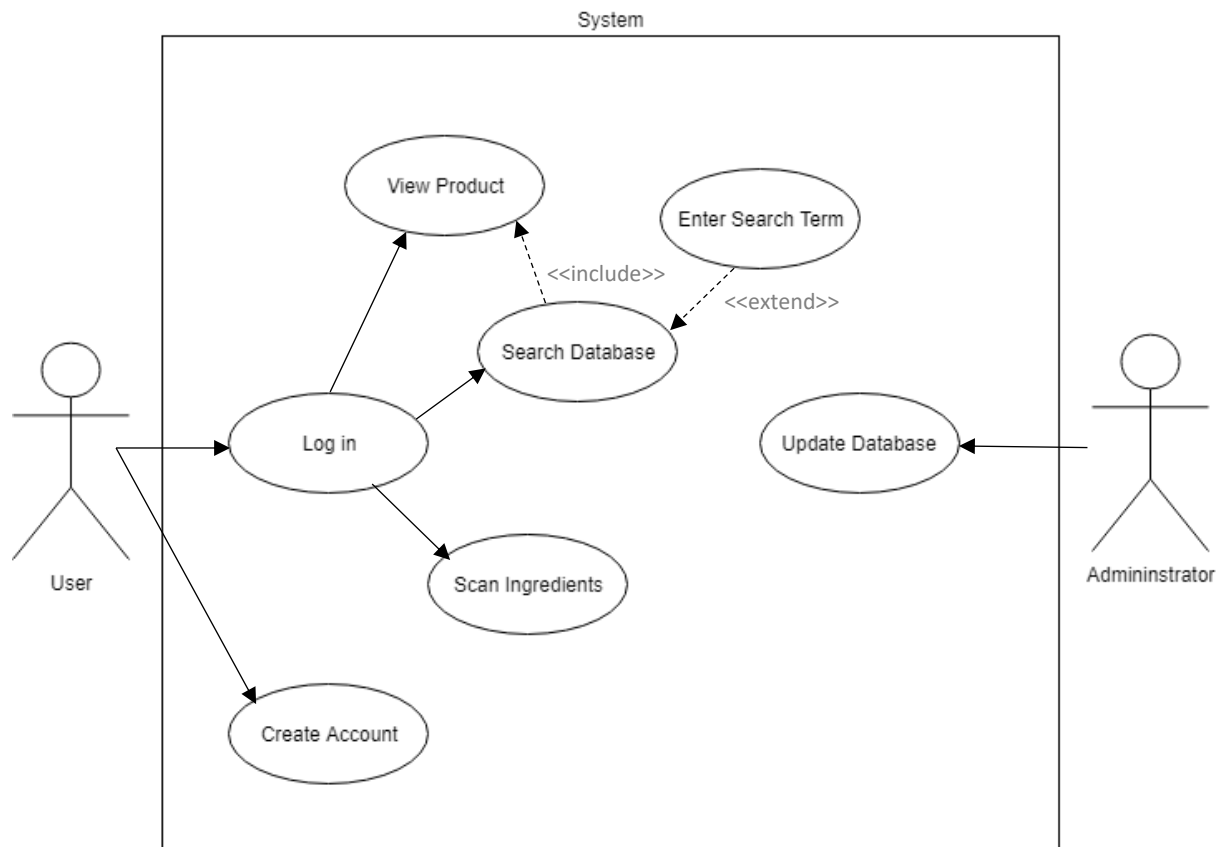
The application code and database should also be built in such a way that additional features and data can be added at any time without having to restructure the database or rebuild the system.

- **Reliability**

The application should be reliable and perform as expected when used. There should be minimal errors.

### **3.2 System Design**

Use case diagrams are a UML modelling tool used to describe system behaviour and interactions with external entities. These entities may be other systems, system administrators, or users. The system behaviour describes the goals actors wish to achieve, such as logging in or searching a database, and are usually related to the functional requirements of the system. The diagrams are helpful in tracking system requirements and translating those requirements into design. The following use case diagram details the system boundaries and major interactions between the system and its actors:

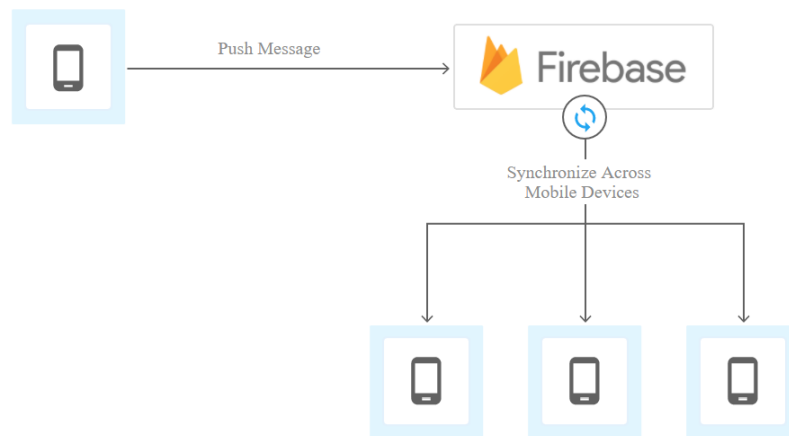


**Image 3-1:** UML use case diagram (drawn with Draw.io)

In this system, the major actors are users and an administrator. The administrator is responsible for any updates to the system database. All other major system functions relate to the users. When not logged in, users have two choices: create a new account or log in with an existing one. Once logged in, they can access major system functionality. Products can be viewed on their own, or after a database search that involves entering a specific search term. Users may also scan the ingredients of a product to view them on the screen.

### **3.3 Back-End Design**

The back-end of the application consists of a Firebase Realtime Database, which is a NoSQL cloud database. NoSQL is non-relational, which tends to allow for faster query responses than relational alternatives. On Firebase, data is stored as a JSON and parsed at runtime. Firebase structures should aim to avoid deeply nesting data as this will increase server load and query runtime.



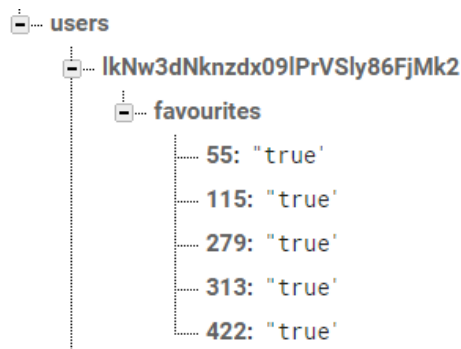
**Image 3-2:** Firebase cloud data synchronisation <sup>(24)</sup>

When new data is pushed to the Firebase database, either from a user's phone or by the application administrator, the database is immediately updated. Any connected users will see the update in real time, and disconnected users will see the update reflected when they next connect to the database. Firebase also allows for users to store an offline version of the database on their phone, so that applications can be used without Internet connection. The application will store the version of the database that exists when the request is made, which the user can then interact with. Once they connect to the Internet again, any updates will be synchronised with their local copy of the database.

This application stores and queries two data types in the database – foods and users. Foods make up the searchable product database of the app and have a unique ID at the top level, followed by brand, category, product name, size, and type attributes.



**Image 3-3:** Firebase database structure for food products



**Image 3-4:** Firebase database structure for users

User entries to the database are created automatically when a user creates an account. Each is given a key which matches its authorisation ID. Each is also given a child, favourites, which is empty on creation. When a user adds a product to their favourites, its product ID is added as a child key of favourites with true as its value. If the user removes the product from their favourites, the corresponding value is updated to be false.

Database rules define whether users can read or write to the two data types. Foods are only readable by authorised users of the application – where the authorisation token is not null. No write permissions are granted to users. If an update to the foods table is needed, it must be performed by an application administrator through the Firebase online console.

```

{
  "rules": {
    "food": {
      ".read": "auth != null",
      ".write": false
    },
    "users": {
      "$user_id": {
        ".write": "$user_id === auth.uid",
        ".read": "$user_id === auth.uid"
      }
    }
  }
}

```

**Image 3-5:** Firebase database rules

Users must be able to read and write to their own user record in order to view and add products to favourites. This is controlled by checking that the user ID in the database record – \$user\_id – matches the active authorisation token in the application.

### **3.4 Front-End Design**

In designing the front-end of the system, the author took into account Nielsen and Molich's user interface design heuristics studied during the Senior Sophister 'Human Factors' module. Nielsen and Molich proposed the idea of heuristic evaluation, whereby usability criteria – referred to as heuristics – are used to evaluate the quality of system design. Designs that follow these heuristics are considered more user friendly than those which do not. Ten major heuristics were identified to consider when designing a user interface <sup>(24)</sup>:

#### **1. Visibility of system status**

The system should always keep users informed about what is going on and their location in the system through appropriate feedback and notification within reasonable time.

#### **2. Match between system and real world**

The system should speak the user's language, using familiar words and real-world conventions to make information appear in a natural and logical order.

#### **3. User control and freedom**

The system should allow users to explore and include clearly marked exits, undo, and redo operations.

#### **4. Consistency and standards**

Words and phrases should have consistent meaning across the system and platform.

#### **5. Error prevention**

The system should be designed to prevent errors from occurring.

#### **6. Recognition rather than recall**

Users should not have to remember how to use and navigate the system, but instead be reminded by recognisable symbols and words.



## **7. Flexibility and efficiency of use**

The system should cater to both new and frequent users through both expert and novice functionality.

## **8. Aesthetic and minimalist design**

Only relevant and needed information should be displayed to avoid overloading the user with information.

## **9. Help users recognise, diagnose, and recover from errors**

Error messages should be expressed in plain language, indicate the problem, and suggest a solution.

## **10. Help and documentation**

Depending on the system, it may be necessary to provide help and documentation. Any such information should be available to users, easy to search, and help them to solve any problems that may arise.

Several of these heuristics were identified as being particularly important to the project. Firstly, showing page status was identified as an issue with existing applications on the market, and therefore should be addressed in this project. Pages should include some identification of the user's location in the app through either a bottom navigation bar or clear text on the page. Minimalist design and navigation through recognition is also important, as it will increase user satisfaction by making the application easier to use. Thorough testing and clear error messages were also identified as important to minimise errors and avoid user frustration with insufficient information about any errors they encounter.

Another principle of user interface design is ergonomics, meaning the physical characteristics of design. There are several important ergonomics to be considered when designing the user interface of the application, including:

- **Layout**

Application layout is a crucial aspect of design. Layout should be consistent across screens, with the same functionality on similar screens found in the same place.

- **Spacing**

Spacing is a subset of layout and helps to increase readability of application pages. As discussed with heuristics, it is important that users are not overloaded with information. Text spacing helps to break up the information and make it easier to read and search through. The use of whitespace in the application is also important for design. Whitespace helps to make important text stand out by adding natural order and spacing to the layout. Whitespace is also aesthetically pleasing when used correctly.

- **Colour**

The use of colour can help to emphasise or highlight certain features and draw the user's attention. Universal colour meanings can be used to help the user understand the function of something without necessarily understanding how it functions e.g. the use of green to signify suitability. When using colour, designers must also be careful to consider colour blindness, which affects approximately eight percent of men <sup>(25)</sup>. Developers should be aware of this and endeavour to include suitable colour schemes.

### **3.5 Prototypes**

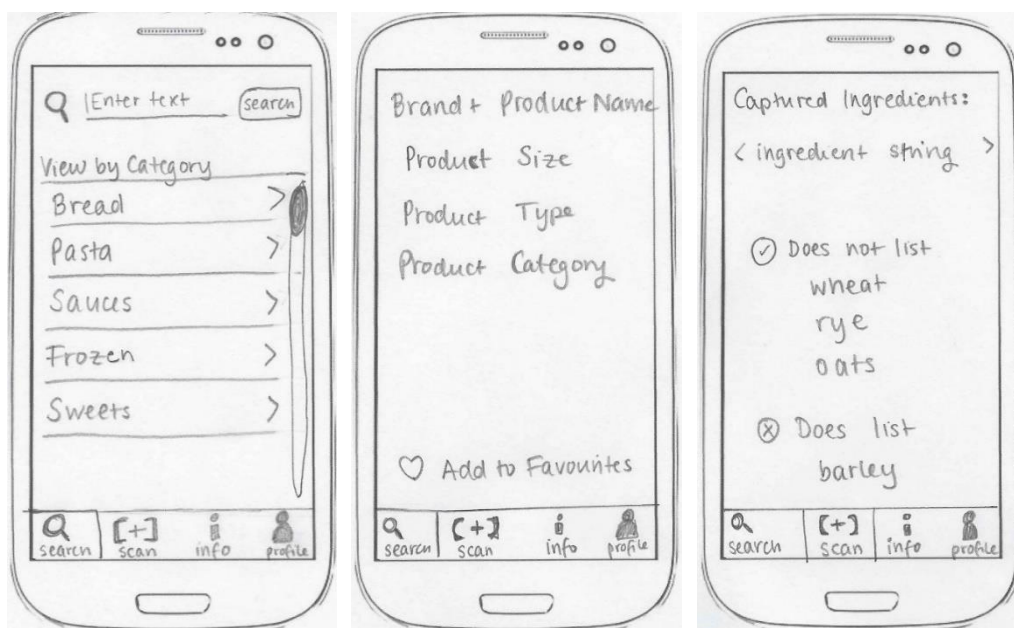
The author developed prototypes in order to refine the user interface (UI) of the application based on the heuristics and ergonomics discussed. Designing a high quality, navigable UI is critical in ensuring the success of the application. If users become frustrated due to confusing or poor navigation experiences, they will not want to use the application. Improving navigation ability and giving the user a clear indication of their location in the application would be a major improvement on current offerings on the market.

Another important consideration was the fact that this application could be used by a wide variety of age groups, as gluten free diets may be undertaken by anyone at any age. In order to improve usability for multiple age groups, the author endeavoured to use large, easy-to-read fonts. The author also chose to use text labels for any symbols and buttons in case symbols could be misinterpreted.

The major pages of the application identified through research and design were search and search results, ingredient scanning, information about diets, and user profiles.

### 3.5.1 Low-Fidelity Prototypes

Low-fidelity prototypes are used in the early stages of user interface design to quickly and cheaply make mock-ups of the system, and make any changes as need with minimal investment. Elements can easily be moved if users suggest they might be better elsewhere. Paper prototypes also allow the designer to visualise how the system will physically look and see any missing elements that might not have been included when thinking about the design. They also allow the designer to focus on the big picture of how the system might look, rather than worrying about implementation details.



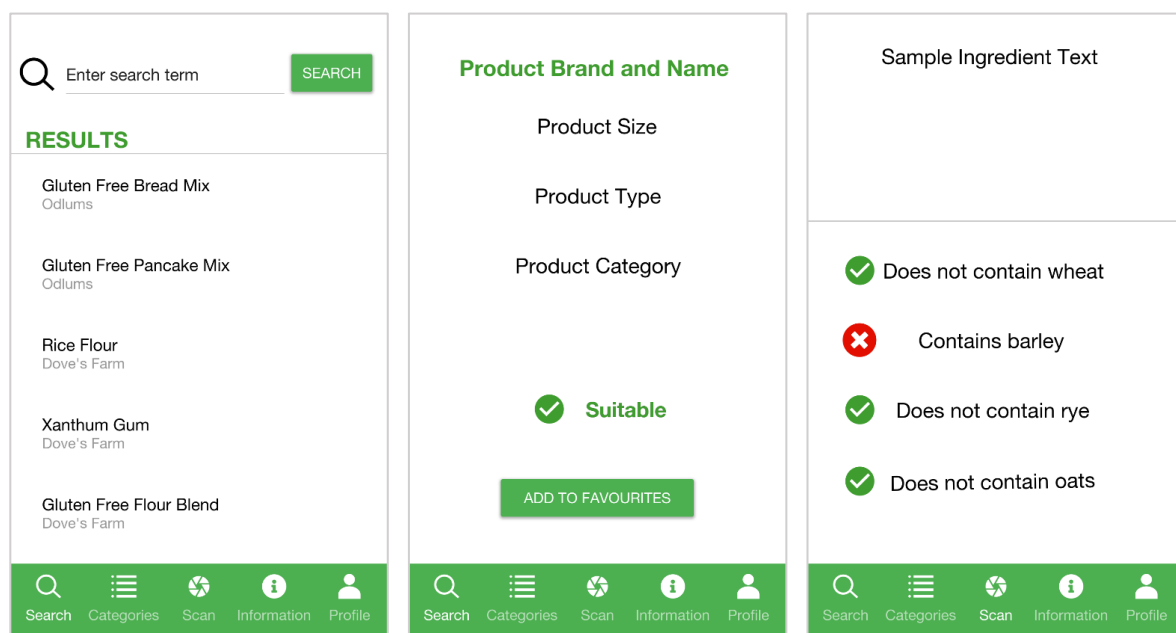
**Image 3-6:** Low-fidelity prototypes of application home page, product page, and ingredient interpretation

The above sketches are paper prototypes of the system. Navigation is handled by the bottom navigation bar and buttons on some pages. This allows for navigation to all functions from any page within the app. The selected page (or page type) is highlighted in the navigation bar when it is open. Text is kept large on all pages in order to increase readability.

Full low-fidelity prototypes can be viewed in Appendix A.

### 3.5.2 High-Fidelity Prototypes

High-fidelity prototypes are used to emulate the finished application design more closely than low-fidelity. They are often produced electronically using dummy code or online tools. They usually show some refinements based on feedback for the initial design. They allow for the use of colour, buttons, and materials that closely resemble the real design. The following high-fidelity prototypes were produced using proto.io:



**Image 3-7:** High-fidelity prototypes of application home page, ingredient interpretation, and product page

The first major change was to improve use of space. In the paper prototypes, search results were shown on a different page from entering a search term. Instead, the search page showed the various food categories with the option to click into each one. The author realised

that viewing search results on the home page would improve usability by allowing users to perform additional searches after viewing results and by eliminating unnecessary navigation. The view by category function moved to its own tab on the navigation bar to allow for easy access to the functionality.

The page showing the results of the ingredients scan was also changed. In the paper prototype, all suitable and unsuitable ingredients were listed together. In this prototype, the author decided to separate each ingredient onto its own line. Each line is also accompanied by a visual cue in the form of a green tick for suitable and a red X for unsuitable. These universal symbols for suitability should allow users to quickly interpret the results of the search.

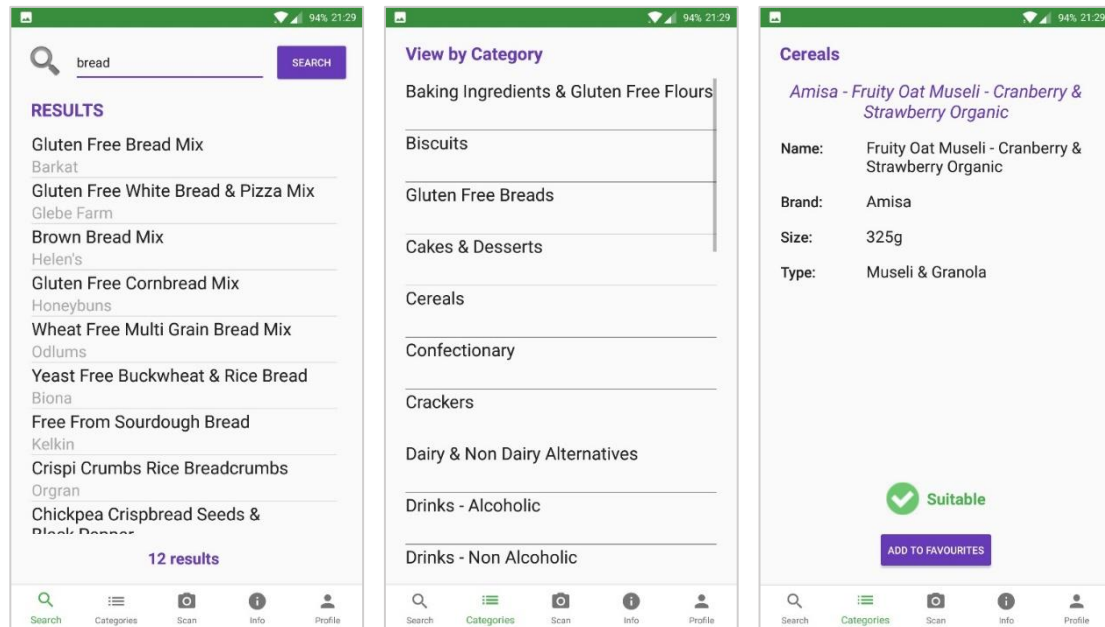
The high-fidelity prototypes also show the potential colour scheme for the application that could not be shown on paper. The application uses two major colours: white and green. Green was chosen due to its common psychological association to health and wellness. It also connects with the colour used for suitability, which minimises the number of colours used in the application. Having a limited number of colours used in the app helps the red X for unsuitable to stand out, which should improve speed of interpretation.

Full high-fidelity prototypes can be viewed in Appendix A.

### **3.6 Final Design**

The final application design made some minor changes from the high-fidelity prototypes. The author chose to introduce a contrasting colour to help elements stand out as opposed to the use of only two colours in the prototypes. The use of purple for key text and buttons provides a high contrast against the main green colour and is unlikely to cause issues with colour blindness.

The author also decided to change the background colour of the navigation bar to blend in with the background of the application. Green is then used to highlight the current page, drawing more attention than the previous coloured underline due to increased contrast.



**Image 3-8:** Final design of application home page, category view, and profile page

Other additions to the application included restructuring the product page, adding labels for product categories, and adding a count for the number of the results at the bottom of the search results list.

Screenshots of the final design of all screens can be viewed in Appendix B.

## Chapter 4: Technologies Used

After technologies were chosen for the project, further research was done to develop a deeper understanding of how they function in order to aid in their implementation.

### **4.1 Android**

Android is a Linux-based mobile software stack that allows for open-source development. Android applications compile into Android packages, which include all of the data and resource the application needs to run. Every application runs independently from any other active app. A major principle of Android applications is that of *least privilege*, where the application only has access to the components it needs to do its work, and nothing more<sup>(26)</sup>. In this project's case, the application has access to the phone camera, but cannot, for example, interact with contacts stored in phone memory.

Android applications are made up of a multitude of components that can act as entry points for the application. In this regard, they are very different from traditional computer applications that have only one entry point. Activities are a type of component and allow for interaction between the user and the application. Each activity is a single screen of the user interface such as the homepage or search results. Activities can be paused and resumed, maintaining their state when they are not active. Intents are used to move between activities and pass information, such as Strings or objects, between them.

### **4.2 Firebase**

Google Firebase offers various services for both mobile and web applications. Services fall under one of three headings: Develop, Grow, and Earn<sup>(28)</sup>. 'Develop' offers services such as the real time database, user authentication, and cloud functions that will be used for the project. The other headings offer services such as analytics, notifications, and app monetization.

#### *4.2.1 Firebase Realtime Database*

As previously discussed, Firebase Realtime Database is a cloud-based, NoSQL real-time database chosen as the database for this project. The database uses a single API for both data reading and writing. Firebase uses an asynchronous data model, which differs from many other databases. In traditional, synchronous programming, the application tries to access the database and waits until it gets a response to continue running. This can cause application lag or freezing if, for example, the database query is complex, or the Internet connection is poor. Lag creates a poor user experience and could turn users away from the application in favour of something that runs more smoothly.

Instead, Firebase uses asynchronous database calls. This means that while the application is waiting for the response from the database, any other code unrelated to the request will run. This prevents lag when waiting for a response but does create other issues. Any code that requires data from the database response must be written within the database call, as otherwise it will cause a runtime error. Thinking asynchronously can be a challenge for developers, but once the concept is understood the runtime benefits are useful.

#### *4.2.2 Firebase Authentication*

Firebase Authentication provides user interface templates and backend services for secure authentication of application users. Users can be authenticated using any of a number of services, including email, Google, Facebook, and Twitter. The sign-in experience is efficient and smooth for users, avoiding any frustration. Accounts can be managed from the Firebase console, allowing administrators to control the user base. The Authentication package includes customisable UI and log in functions that can be used in any application. Using these services allows developers to focus on developing major app functionality. This project uses email log in through Firebase Authentication to give users access to the application.



#### 4.2.3 *Firebase Cloud Functions*

Firebase Cloud Functions allow developers to trigger backend functions based on actions in other Firebase products, such as new user sign-ups using Authentication or additions to the Realtime Database. Cloud Functions are written in Javascript and deployed to Firebase via the command line. Once uploaded, functions will automatically be performed once the triggering action is called. The Firebase server will scale the amount of server space the function has access to depending on how many calls are being made at a time. Using Firebase Cloud Functions can keep key functionality on the server rather than on the client side, helping to avoid any tampering or reverse engineering of functions. For this application, Cloud Functions are used to create a new user entry in the database when an account is created.

#### 4.2.4 *Firebase Notifications*

Firebase also allows for developers to send notifications to users via Firebase Cloud Messaging. Messages can be sent to all users at once, or a targeted subset of users. The messages can also be cross-platform, notifying Android, iOS, and web users all at once. Messages can be sent immediately or at a particular time determined by the user's time zone. Firebase can also track conversion events and show if users take a particular action within the application after receiving the notification. This functionality could be used to notify users when items are added, removed, or updated in the database.

### **4.3 Google OCR**

Google's Text Recognition API is part of their Mobile Vision Library, which includes other functionality such as barcode scanning and face detection <sup>(28)</sup>. The Text Recognition API works on the basis of Optical Character Recognition (OCR). OCR is the process of converting images or printed text into machine-encoded text. The machine-encoded text can then be searched for a particular String or character. The API uses a *TextRecognizer* to read text from an image, returning a JSON of the text it extracts. Text can be detected in blocks, lines, and individual words. In this case, blocks are structures like a paragraph or column of text. The API

works for any Latin character language. While the technology is still in its infancy, the API is fairly accurate and will improve as new versions are released.

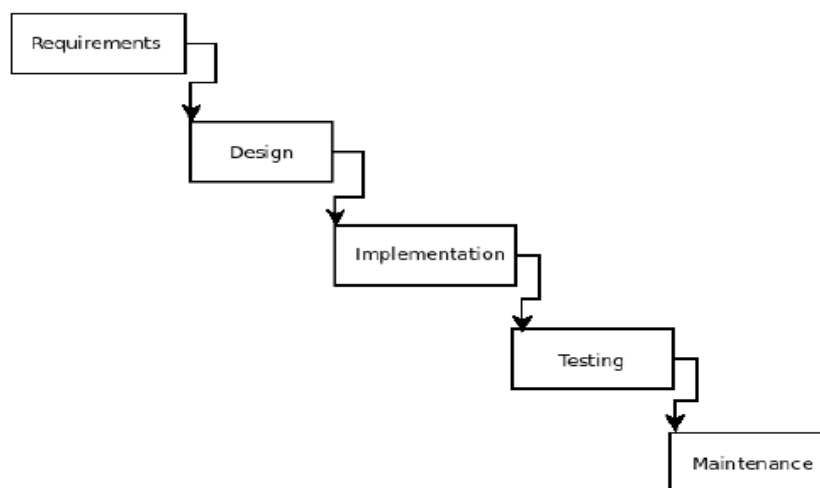
## Chapter 5: Implementation

After researching and designing the application, the author began the implementation phase. This involved choosing a development model, then implementing features based on this model. The chapter also describes the difficulties encountered during implementation. The development model was chosen based on information learned about different models in various Software Engineering modules.

Initially, the implementation steps ignored the user interface developed in Chapter 3 in favour of developing a functional application. Once the app had reached a certain level of functionality, the UI was overhauled to better reflect the prototypes.

### 5.1 Development Model

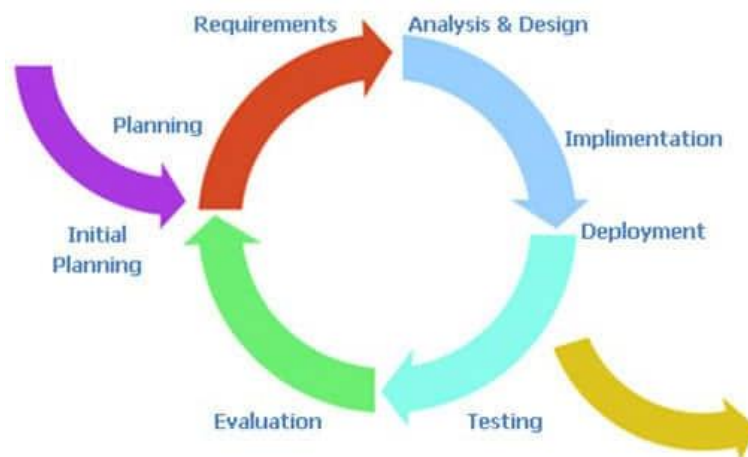
There are many well-known software development models, each with their own advantages and disadvantages. Models help to define the structure a project will take, as well as lay out clear expectations for deliverables. Using a clearly defined structure also helps to keep the project more focused, with a clear roadmap for how it should proceed. Two major models were considered when developing this project: waterfall and iterative.



**Image 5-1:** Waterfall model of software development <sup>(29)</sup>

The waterfall model is perhaps the most well-known development model. It was made popular due to its sequential nature and ease of understanding for both technical and business users. Completing each stage before proceeding ensures that it is completed to the highest quality possible. The simplicity of the model makes it easy to explain to non-technical stakeholders and helps to set expectations for delivery. However, each stage occurring in order prevents returning to previous stages should requirements change during design, or major implementation issues arise during testing. In order to deal with these issues, the cycle has to begin again from the beginning, which has high associated time and monetary costs.

This disadvantage led to the development of other, more flexible software development models. One such example is the iterative model of software design, which creates multiple waterfall cycles. Each iteration has its own planning, design, coding, testing, and deployment stage. At the end of each iteration, the project should be in a releasable state. Each phase should build upon the last, adding new functionality without removing anything.



**Image 5-2:** Iterative model of software development <sup>(30)</sup>

During each cycle, the project is tested and evaluated, helping to ensure the best quality finished product. If major problems arise, the next increment can address these problems where possible. This process helps to ensure that problems are discovered before all coding stages are complete and allows them to be fixed during later stages of development. Incremental design is also useful where there is a possibility of project

requirements changing during the development processes, as these changes can potentially be integrated during the next iteration.

Requiring incremental releases of software is another major advantage of the model. This allows developers to add features and functionality at each step, splitting functionality goals into smaller chunks as opposed to one major development cycle. Developing in this way allowed the author to create a clear development plan for each iteration, address any problems as they arose on a smaller, iteration-level scale, and to achieve a releasable product at the end of iteration.

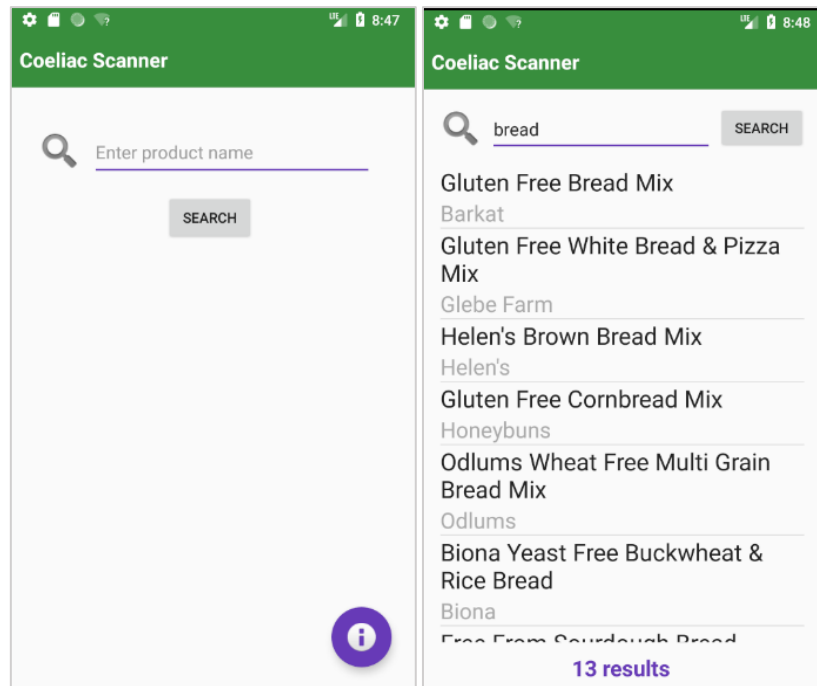
The author identified four iterations of the project: database search; product information and searching by categories; user log in and adding products to favourites; and ingredient text scanning. For the early iterations, the author chose to make a functional user interface over one that matched the prototypes exactly. This choice was due to the limited number of functions introduced in the early stages of development. The prototyped bottom navigation bar is advised for use with three to five top level destinations but is not useful for a limited number of destinations. In the early stages, navigation was instead handled using buttons.

## **5.2 Database Search**

Database search represented the minimum viable product for the application, and so was chosen for the first iteration. Developing the database search required integrating Firebase Realtime Database with the Android application, querying the database, and displaying query results on the application screen.

In order to integrate Firebase functions with the application, Firebase dependencies had to be added to the application's *build.gradle* file. These dependencies allow for the use of objects such as *FirebaseDatabase* and *DatabaseReference*. To fully synchronise the project

with Firebase, the application also had to be added to the author's Firebase account through the online console or through integration in Android Studio. The author chose to use the Firebase console. Connecting the app required the creation of a Firebase project on the console, which then links to the Android application by including the application package name on the console, and adding a custom generated *google-services.json* file to the application folder.



**Image 5-3:** First iteration app home page and search results

The home screen of the application includes the main search functionality. Users can type in a search term, then press the search button and be brought to the search results. Tapping “Search” stores the term entered into the search bar and passes it to the results screen. Once the screen is created and the string is retrieved, the query to retrieve foods from the database begins, using the following format:

```
mFirebaseDatabase = FirebaseDatabase.getInstance();
mDatabaseRef = mFirebaseDatabase.getReference();

mDatabaseRef.child("food").addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot)
    {
        for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
            Product newProd = snapshot.getValue(Product.class);
        }
    }
});
```

**Image 5-4:** Firebase query to access products

The *mFirebaseDatabase* holds an instance of the Firebase database. *mFirebaseDatabase.getReference()* stores a reference to the root of the database in *mDatabaseRef*. This reference is usually a URL that can be used by authorised users to access the database online. The reference is used in all Firebase queries before viewing deeper levels of data. The database structure was discussed previously and can be viewed in Chapter 3, Section 4.

*mDatabaseRef.child("food")* points to the top level of the foods stored in the database, before any product IDs. The *ValueEventListener* on the food child listens for changes to data in the location children. This could be triggered by a food record being updated in or added to the database, or by looping through the data as done with the *onDataChange(DataSnapshot)*. This function will loop through all of the children of food (i.e. all of the products in the database), with the snapshot taking on the attributes of each product. In this way, information from the database can be stored in instances of a local *Product* class.

```
public class Product {
    public String name;
    public String brand;
    public String category;
    public String type;
    public String size;
    public String key;

    public Product ()
    {
    }

    public Product (String name, String brand, String category, String type, String size)
    {
        this.name = name;
        this.brand = brand;
        this.category = category;
        this.type = type;
        this.size = size;
    }

    public void addKey(String key)
    {
        this.key = key;
    }
}
```

**Image 5-5:** Product class

The *Product* class has the same format as products in the database, including using the same labels for product attributes. If the case or text is different, an instance of the class cannot be created directly from the *dataSnapshot*. Creating a new product using the *snapshot.getValue(Product.class)* calls the second *Product* constructor and assign the database values to the attributes. The product key must be manually assigned because it is not a feature of the product structure in the database.

```
for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
    Product newProd = snapshot.getValue(Product.class);
    newProd.key = snapshot.getKey().toString();
    //check if the product pulled from firebase matches the search string and,
    // if not present in list, add it
    if (((newProd.name.trim().toLowerCase()).contains(searchString))
        || ((newProd.brand.trim().toLowerCase()).contains(searchString)))
        && (!listContains(resultsList, newProd)))
    {
        resultsAdapter.add(newProd);
    }
}

resultsCount = resultsList.size();
if (resultsCount == 0) {
    noResultsText.setVisibility(View.VISIBLE);
}
else {
    resultsCountText.setText(resultsCount + " results");
    resultsCountText.setVisibility(View.VISIBLE);
    resultsLabel.setVisibility(View.VISIBLE);
}
```

**Image 5-6:** Searching results of Firebase query and outputting data

Once accessed, the product information from the database is checked against the search term, labelled as *searchString*. Any strings handled by the system have the *String.trim()* method called to remove any trailing whitespace characters that would impede matching. The search string is trimmed before being passed to the method, while the database strings are trimmed at runtime. While good database management can ensure that strings are well structured and do not need to be trimmed, including the method mitigates the risk of a mistake being made when constructing the database.

If either the product name or product brand contain the search term, the product is added to a custom list adapter for search results. This list adapter is responsible for formatting the results to display the product name and brand in the results *ListView*. After all matching



products have been added to the list, the results are counted. If this count is zero, no results are displayed, and the screen shows a hidden “No Results Found” view. If results are found, the results count is made visible beneath the results list.

All of this functionality must be handled within the Firebase query due to its asynchronicity. If the application attempts to display the information outside of the query, the application will either crash or display nothing. The *ListView* itself is created and drawn outside of the query but is empty until the query is completed. The *ListView* must be drawn first as otherwise a new one will be drawn every time a product is added to the list.

To facilitate this database query, users need to be able to read from, but not write to, the database. The following database rules allow anyone to read from the database without requiring log in. No one is permitted to write to the database. These rules were temporary until user accounts were established.

```
{
  "rules": {
    ".read": true,
    ".write": false
  }
}
```

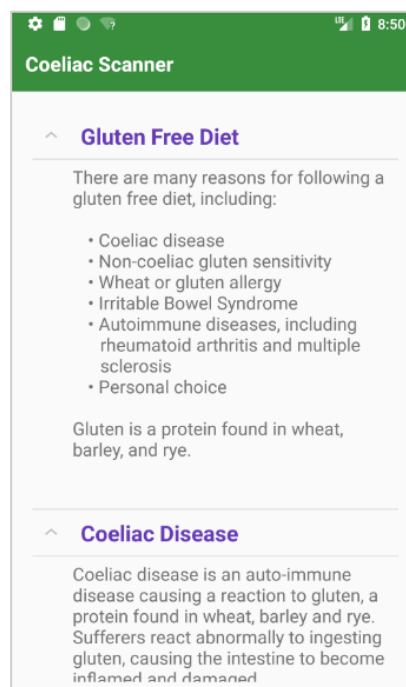
**Image 5-7:** Database rules for iteration one

When the search button is pressed, any previous results list is cleared, and any results text removed from the screen. This prepares the view to display the next set of results. In order to update the list adapter, it must be told that the data has changed after the list is cleared. This notifies the adapter to change the presentation of the list on the screen.

```
//searchButton onClickListener to search the database
searchButton.setOnClickListener(
    (v) → {
        resultsList.clear();
        resultsAdapter.notifyDataSetChanged();
        noResultsText.setVisibility(View.INVISIBLE);
        resultsCountText.setVisibility(View.INVISIBLE);
        resultsLabel.setVisibility(View.INVISIBLE);
        searchString = searchBar.getText().toString().trim().toLowerCase();
        getData();
    }
);
```

**Image 5-8:** Code snippet managing clicks on the search button

The author also chose to include basic information pages about gluten, gluten free diets, and coeliac disease during this iteration. This information page also includes the disclaimer about using the application. The information screen is accessed via the floating action button on the home screen.



**Image 5-9:** First iteration app information screen

The *FloatingActionButton* can be seen in Image 5-3. This style of button was chosen to highlight the information page on the home screen. Using a contrasting colour and symbol helps the button to stand out on the screen. The information screen uses an expanding list to show the information text. This minimises the amount of information initially displayed, only

showing the section headings. Once the user has selected a section to view, it expands to fill available screen space, and can be minimised once read.

Implementing the expandable *ListView* requires a custom list adapter. This adapter is responsible for passing and formatting data correctly for the list headers and children. Data adding is handled in the Information activity, while the formatting is handled in the list adapter class.

In order to display the list in the activity, the application must first find the *ListView* in the layout. Then, the relevant data is prepared in a separate function and added to the list adapter on its creation. Creation of the adapter also formats the list to use the correct layouts. The adapter is then attached to the *ListView*. In the list data function, the data is stored as a *HashMap* of pairs of list headers and children. The headers are initially assigned to an *ArrayList* before being put into the *HashMap* as a pair. All of the strings are stored within the project's *string.xml* file. In this way, strings can be reused across pages if needed. This also ensures that if strings need to be modified, it only needs to be done in one location.

```
@Override
public View getChildView(int groupPosition, final int childPosition,
                        boolean isLastChild, View convertView, ViewGroup parent) {

    final String childText = (String) getChild(groupPosition, childPosition);

    if (convertView == null) {
        LayoutInflater infalInflater = (LayoutInflater) this._context
            .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = infalInflater.inflate(R.layout.information_item, root: null);
    }

    TextView txtListChild = (TextView) convertView
        .findViewById(R.id.lblListItem);

    txtListChild.setText(childText);
    return convertView;
}
```

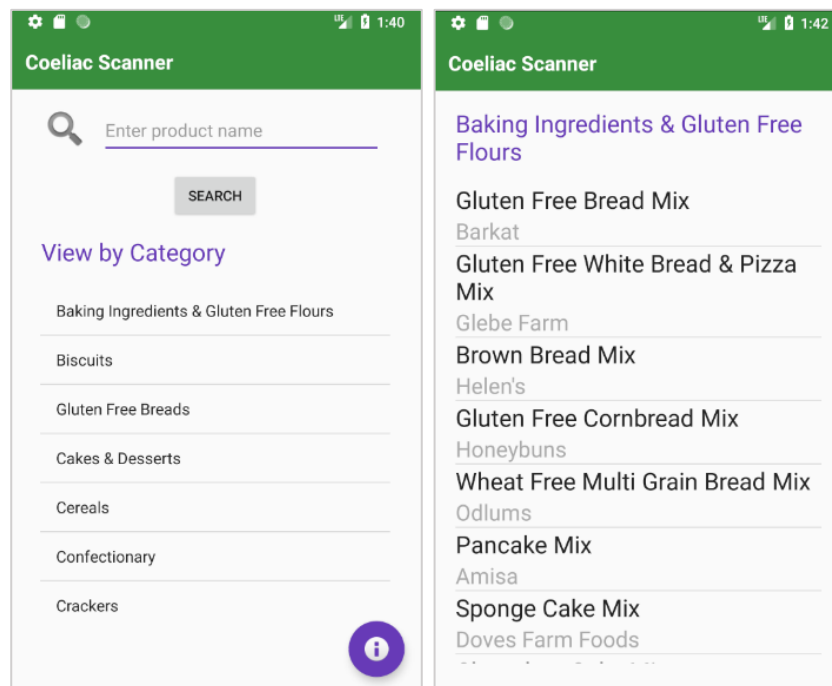
**Image 5-10:** Method for formatting text in the *ExpandableListAdapter*

The *ExpandableListAdapter* class attaches the correct layouts to the *ListView* headers and children using two functions: *getChildView* and *getGroupView*. Group refers to the

information headers, while child refers to the children. Two methods are used to allow for the use of two custom layouts. These layouts are used allow the headers to stand out against the children when the list is expanded, as shown in Image 5-9. Both methods are called within the *ExpandableListView* constructor and so run when the adapter is created in the activity.

### 5.3 Product Information

The second iteration added the ability to view products by categories as well as by searching, and to view product information screens. These features were chosen for the second iteration as the author felt they complimented the existing functionality well. The ability to view product information is a particularly important and valuable addition to searching the database. The user interface remained focused on being functional rather than directly emulating the prototypes.



**Image 5-11:** Second iteration app home page and category view

Viewing products by categories requires the application to show the list of available categories, then find the products in the database that match that category and display them. Initially, the list of categories was added to the main search page to minimise the amount of navigation required.

The list of possible categories for products in the database is stored in the application's `string.xml` file. In order to display it on the screen, the list must be stored as a string array in the activity, then added to the *ListView* through an *ArrayAdapter*, which formats the array for the view.

Once the list is drawn to the screen, an *OnItemClickListener* is declared. This allows the application to react when one of the items on the list is tapped. When a category is tapped, its name is added to a new page intent for a new activity to display the corresponding products. The new activity is then started.

The passed category name is retrieved and stored in the new activity. The application then connects to and queries the Firebase database to find all products belonging to the category. The list of products is displayed in the same way as search results – each matching product is added to an adapter which displays the values in a *ListView*.

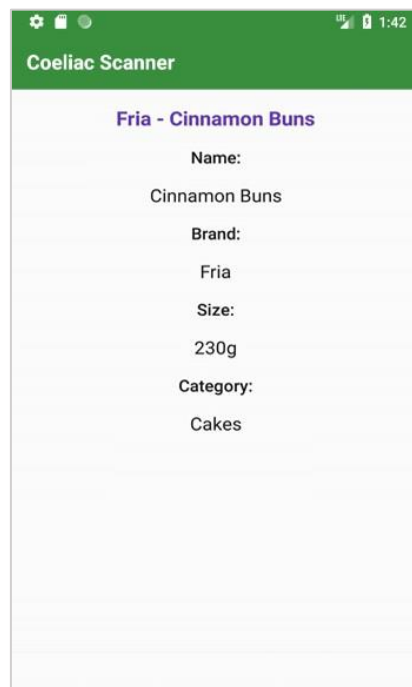
```
public void getData()
{
    mFirebaseDatabase = FirebaseDatabase.getInstance();
    mDatabaseRef = mFirebaseDatabase.getReference();

    mDatabaseRef.child("food").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot)
        {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                Product newProd = snapshot.getValue(Product.class);
                newProd.key = snapshot.getKey().toString();
                if (category.equals(newProd.type.toString()))
                {
                    resultsAdapter.add(newProd);
                }
            }
        }
    });
}
```

**Image 5-12:** Firebase query to retrieve products from a category

Product pages were added to the application to provide users with additional information about the products in the database. The product name and brand displayed in the *ListView*s are perhaps the most useful pieces of information for users searching for

products, but additional information such as product size and subcategory are also useful for shoppers.



**Image 5-13:** Second iteration product screen

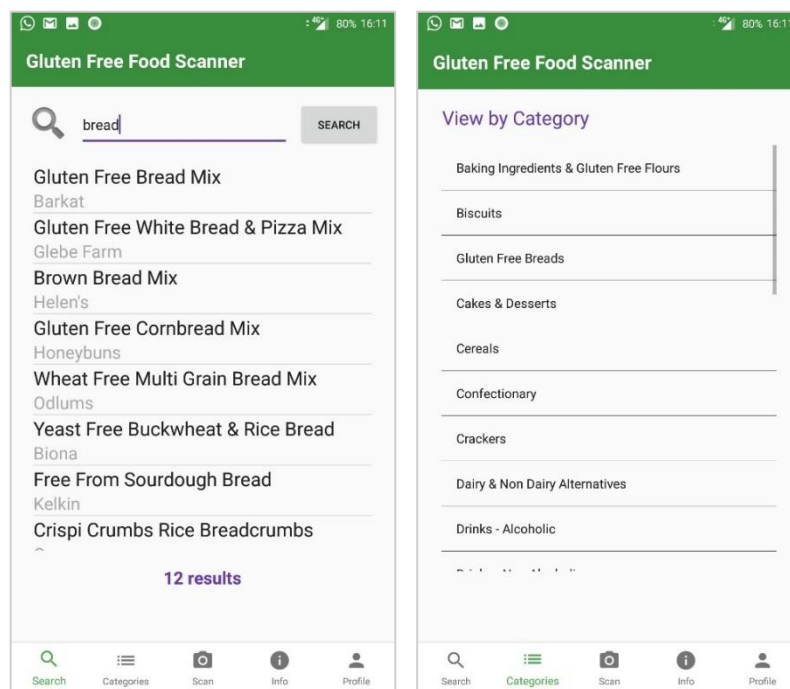
The product page displays information in a simple list format made up of individual *TextViews*. In order to pass the product information to the product screen, the lists on the search results and categories page must have an *ItemOnClickListener*. Once a product is clicked, its information is passed to a new page intent. This information can then be accessed from the product screen by accessing the intent.

## **5.4 User Accounts**

The third iteration added user accounts to the application by integrating Firebase Authentication. The iteration also added the ability to add products to a user's favourites.

At this time, the author undertook a user interface overhaul to align the UI more closely with the application prototypes. The addition of user profiles added a fourth top level destination, making the use of a bottom navigation bar suitable. The author chose to include

the “Scan” menu option at this point despite the function not being included in this iteration as it would be easy to remove if the next iteration did not proceed.



**Image 5-14:** Third iteration UI overhaul of home page and category view

The search functionality and view by category were split into two separate pages. This allows the search results to be displayed on the same screen as the functionality, streamlining the user experience. New searches can also be performed from the same screen. Moving the category view to a separate page also allows for more space to be given to the list, allowing the user to see more categories at a time.

The search functionality was chosen as the default application location. The author felt that this would be the most used functionality in the app and so should be the default. As shown in the ergonomics diagram in Chapter 2, Section 5, the bottom right-hand corner of the screen is the most easily accessed for the majority of users, so the most used functionality is placed there. The profile functionality, which will most likely be used least, is placed in the least accessible section of the navigation bar.

The use of further Firebase libraries required the addition of new dependencies to the application's *build.gradle* file. These dependencies allow for the use of Firebase authentication tokens and built-in authentication user interfaces.

Firebase Authentication allows users to sign in to an app using a variety of methods, including email address and social media. The author chose to use email log in only, as some potential users may not have social media accounts. The authentication code can be added to any launcher activity – for this application, it was added to the main search page. The *FirebaseAuth* token and *AuthStateListener* act as gatekeepers for the app, checking whether or not users are logged in. If they are already, the application starts as normal on the main activity. If they are not, the application begins the 'Sign In' Activity.

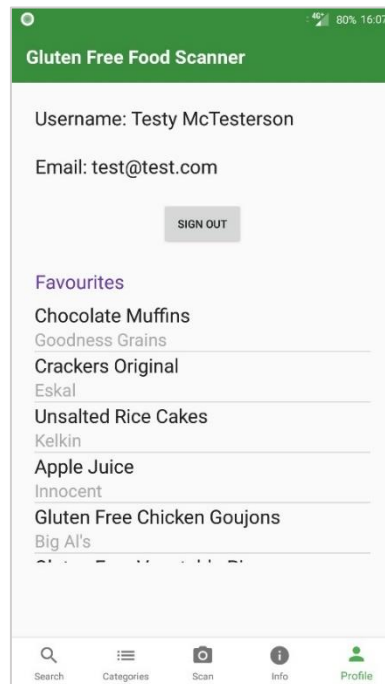
```
//login
mUsername = "Anonymous";
mFirebaseAuth = FirebaseAuth.getInstance();
mAuthStateListener = (AuthStateListener) (firebaseAuth) → {
    FirebaseUser user = firebaseAuth.getCurrentUser();
    if (user != null)
    {
        //user is signed in
        onSignedInInitialise(user.getDisplayName());
    }
    else
    {
        onSignedOutCleanup();
        //user is signed out
        startActivityForResult(
            // Get an instance of AuthUI based on the default app
            AuthUI.getInstance().createSignInIntentBuilder()
                .setIsSmartLockEnabled(false)
                .build(),
            RC_SIGN_IN);
    }
};
```

**Image 5-15:** Firebase Authentication log in

The Sign In activity has two possible routes – where an account exists, and where one does not. Once a user's email is entered, it is checked against the authentication record to find any existing accounts. If an account exists, the user is prompted to enter their password. If it does not, the user is prompted to create an account. The main activity then begins.



The user profile page displays a user's profile information and a list of their favourite products. When a user is logged in via Firebase Auth, their authentication information is available from any activity in the application. This allows account information and log out functionality to be present on a separate screen from the log in functionality. As log out should not be used very often, it is included in the profile page.



**Image 5-16:** Profile page

In order to access account information, a *FirebaseAuth* instance must access the current user's information. The user information can then be displayed on the screen.

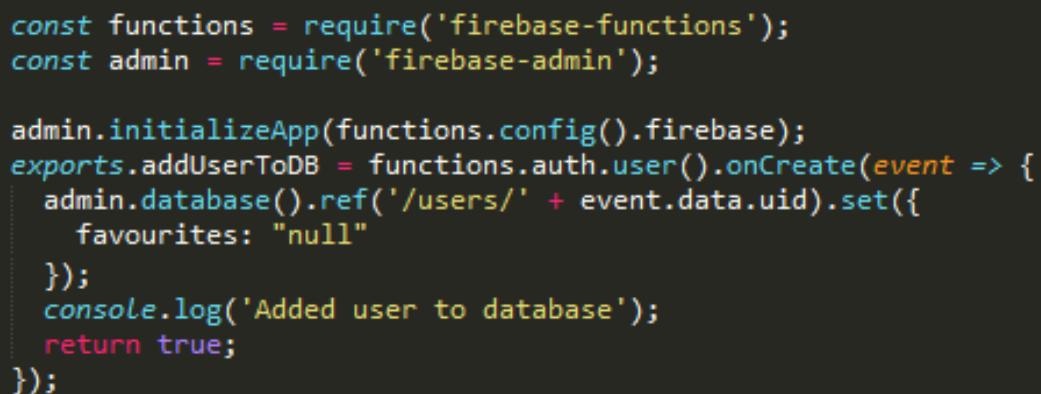
```
//FirebaseAuth
user = FirebaseAuth.getInstance().getCurrentUser();
usernameText = (TextView) findViewById(R.id.usernameText);
useremailText = (TextView) findViewById(R.id.useremailText);
usernameText.setText("Username: " + user.getDisplayName());
useremailText.setText("Email: " + user.getEmail());

signOutButton = (Button) findViewById(R.id.signOutButton);
signOutButton.setOnClickListener((view) -> {
    AuthUI.getInstance().signOut(context: UserProfile.this);
    Intent intentMain = new Intent(packageContext: UserProfile.this, MainActivity.class);
    startActivity(intentMain);
});
```

**Image 5-17:** Firebase Authentication sign out

Sign out functionality is also handled by Firebase Auth. When the 'Sign Out' button is tapped, the application calls the Authentication sign out method, which signs out the current user and clears its record from the disk cache. The application then returns the user to the main search activity and shows the sign in screen.

In order to allow users to add products to their favourites, the author needed to create editable records for each user in the database. Firebase offers Cloud Functions that can be triggered on events such as users creating an account or changes to the database. These functions are written in Javascript and deployed to the Firebase console. The following function is triggered when a new user account is created through Firebase Authentication. When a new user is created, it is added to the database using the user's authentication ID – *event.data.uid* – as its key. The user is given an empty favourites attribute that can be added to later.

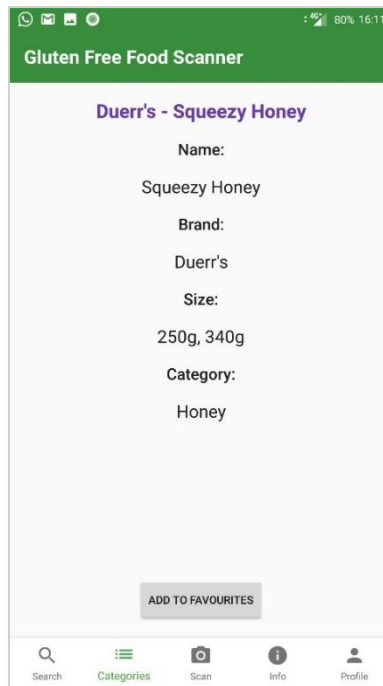


```
const functions = require('firebase-functions');
const admin = require('firebase-admin');

admin.initializeApp(functions.config().firebase);
exports.addUserToDB = functions.auth.user().onCreate(event => {
  admin.database().ref('/users/' + event.data.uid).set({
    favourites: "null"
  });
  console.log('Added user to database');
  return true;
});
```

**Image 5-18:** Firebase Cloud Function to create a user record in the database on account creation

In order to add products to a user's favourites, a button was added to product pages. The button must be able to tell whether or not a product is already in the user's favourites in order to reflect the correct state.



**Image 5-19:** Updated product page including favourites button

```
//determine if the product has been favourited or not and draw relevant button
userID = FirebaseAuth.getInstance().getCurrentUser().getUid();
mFirebaseDatabase = FirebaseDatabase.getInstance();
mDatabaseRef = mFirebaseDatabase.getReference().child("users").child(userID).child("favourites");
mDatabaseRef.child(key).addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        if (dataSnapshot.getValue() != null || !dataSnapshot.getValue().toString().equals("false")) {
            favourite = true;
        }
        updateFaveButton();
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.d( tag: "READ_FAILED", msg: "Read failed");
    }
});
```

**Image 5-20:** Firebase query to determine if a product is the user's favourites in the Product activity

To do this, the application iterates through the user's favourites as stored in the database. If the value of the key in the database is non-null and 'true', the product is included in the user's favourites and the button is updated to read "Remove from Favourites". If the product is not in the user's favourites, the button reads "Add to Favourites".

The button uses an *onClickListener* to add or remove products from the user's favourites. If the product has not been favourited, its key is added to the favourites list with a value of true, and the button is updated. If the user removes the product from their favourites, the key's value in the database is updated to be false.

```
mFirebaseDatabase = FirebaseDatabase.getInstance();
String userID = FirebaseAuth.getInstance().getCurrentUser().getUid();
mDatabaseRef = mFirebaseDatabase.getReference();

//find the keys of the user's favourites
mDatabaseRef.child("users").child(userID).child("favourites")
    .addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
                if (snapshot.getValue().toString().equals("true")) {
                    favouritesIndex.add(snapshot.getKey().toString());
                }
            }
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
            Log.d( tag: "READ_FAILED", msg: "Read failed");
        }
    });
```

**Image 5-21:** Firebase query to create a list of user favourites in the Profile activity

In order to display the user's favourites on their profile screen, the application must create a list of favourited product keys, then query the products that match those keys from the database and display them on the screen. Displaying the query to the screen uses the same method and adapter as used for search results and category view.

The addition of Firebase Authentication allowed for database rules to be updated to only allow authorised users to interact with the database. As discussed in Chapter 3, Section 3, and shown in Image 3-5, database rules were established to allow logged in users to interact with the database. Authenticated users are allowed to read data from, but not write to, the food table and are allowed to read and write only to their own user record. This will allow users to read and write to their own favourites list but will not allow them to view any other user's information.

### **5.5 Ingredient Recognition**

The final iteration was the implementation of the ingredient text scanner. In order to implement the feature, the Google Text Recognition API had to be integrated with the project, with new activities to use its functionality. The Text Recognition API is a part of Google's major Vision library. The entire Vision library must be included in the application *build.gradle* file to access any of the API's functionality. Compiling the vision library also requires the inclusion of additional Android support and design libraries.

The TextRecognition API requires access to the phone's camera to function. In order to do this, the whole application must request permission for the camera. To do this, a requirement is added to the *AndroidManifest* file for the application. As the application only needs to access the camera, but not to take or save pictures, only camera access is required.

The application uses a *SurfaceView* object to access the phone's camera feed. Once drawn on the screen, the *SurfaceView* shows the user a direct feed from their camera. Functions to read the text from what the camera sees are handled in the *startCameraSource()* method. Once this method has been called, a *StringBuilder* object for the activity will build a value based on what the camera is reading, and its value can be interpreted.

In the *startCameraSource()* method, a *CameraSource* object is initiated. This manages the camera activity and adjusts what is seen in the *SurfaceView*. On creation, several attributes of the *CameraSource* are set. The most important attribute is the phone's autofocus. Requiring the use of autofocus ensures that the app will be able to focus on text without input from the user.

```

startCameraSource();

scanButton.setOnClickListener(
    (v) -> {
        if (stringBuilder != null) {
            String ingredients = stringBuilder.toString();
            if (ingredients.toLowerCase().contains("ingredient")) {
                Intent intent = new Intent(getApplicationContext(),
                    IngredientInterpreter.class);
                intent.putExtra( name: "INGREDIENT_STRING", ingredients);
                startActivity(intent);
            } else {
                Toast.makeText( context: scanIngredient.this,
                    text: "Couldn't find ingredients, please try again",
                    Toast.LENGTH_SHORT).show();
            }
        }
    }
);

```

**Image 5-22:** Starting the camera and pausing the StringBuilder to interpret the String

In order to control the string being read by the camera feed, a button is used to pause the StringBuilder. Once the button is clicked, the StringBuilder's value at that time is converted into a String and interpreted. If the string contains the word "ingredient", the ingredient interpretation activity can begin using the string. Otherwise, the application alerts the user that no ingredients were found in the string through the use of a pop-up Toast message.

```

@Override
public void receiveDetections(Detector.Detections<TextBlock> detections)
{
    final SparseArray<TextBlock> items = detections.getDetectedItems();
    if (items.size() != 0 ){
        mTextView.post(() -> {
            stringBuilder = new StringBuilder();
            for(int i=0; i<items.size(); i++)
            {
                TextBlock item = items.valueAt(i);
                stringBuilder.append(item.getValue());
            }
            mTextView.setText(stringBuilder.toString());
        });
    }
}

```

**Image 5-23:** Text detection method

Once the *CameraSource* has been initialised and permission to use the phone camera granted, the application can begin reading text from the image. Any block of text that is detected is passed into a *SparseArray*, which acts as a *HashMap* of integers to Objects. If this *SparseArray* has elements, the text can be displayed on the screen. In order to do this, a *StringBuilder* combines detected text blocks from the *SparseArray* into a single block. This can be accessed from outside the *startCameraSource()* method and is used to pass the ingredients to the interpretation screen.

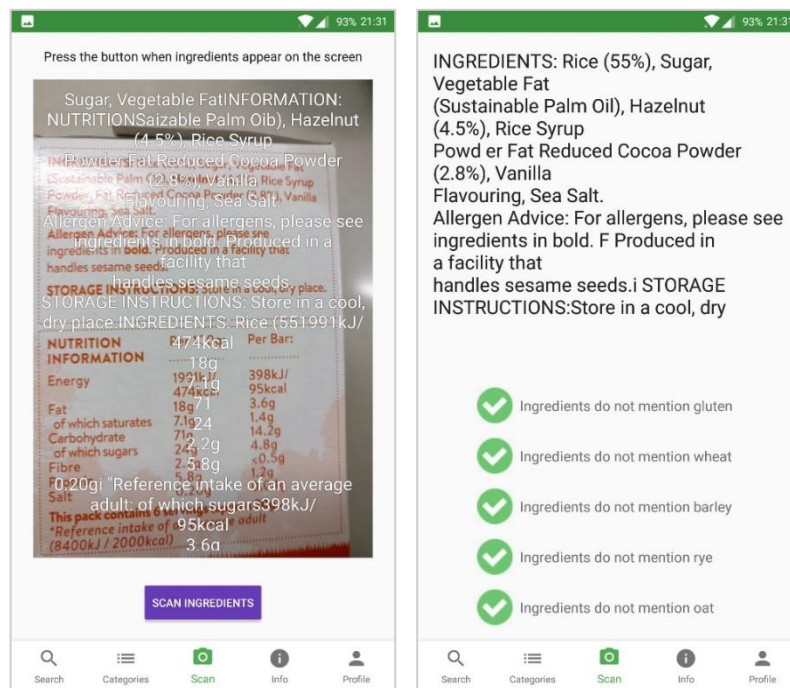


Image 5-24: Screenshots of text detection screens

The *StringBuilder* is displayed on the scanning screen to allow users to see what is being read, making it easier for them to ensure that ingredients are included in the string being scanned.

The ingredients interpretation screen searches through the passed ingredient String for mentions of gluten, wheat, barley, rye, or oats. The screen displays the scanned ingredients trimmed to begin at the first instance of "ingredient" in the string, which is assumed to be the ingredient list. This string is displayed to allow users to check that it has scanned correctly. Limitations of the API will be discussed in detail in Section 6 of this chapter.

Each gluten-containing ingredient is given its own line in the interpretation, with the same order maintained every time. If the application finds a mention of the ingredient in the string, a red X will be displayed next to its interpretation. If it is not present, a green tick is displayed. The use of this universal symbol should help users to quickly understand the interpretation.

The string is scanned for each ingredient in separate methods. Rye, barley, and oats are all handled in one method, while wheat and gluten each have their own methods as they are more complicated to interpret.

```
private void ingredientsCheck(TextView textView, String ingredient)
{
    if (lowerIngredients.contains(ingredient))
    {
        textView.setText(" Ingredients mention " + ingredient);
        textView.setCompoundDrawablesWithIntrinsicBounds(R.drawable.x38, top: 0, right: 0, bottom: 0);
    }
    else
    {
        textView.setText(" Ingredients do not mention " + ingredient);
        textView.setCompoundDrawablesWithIntrinsicBounds(R.drawable.check38, top: 0, right: 0, bottom: 0);
    }
}
```

**Image 5-25:** Sample text interpretation method

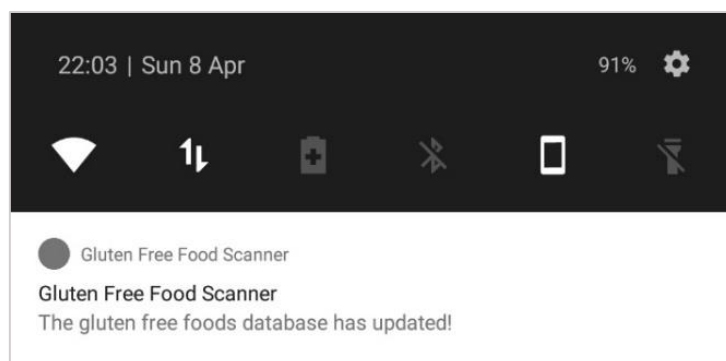
For barley, rye, and oats, the application simply scans through the ingredient string to find if it is present. The result of this check will determine what results will be displayed in the *TextView* for that ingredient.

Checking for wheat is marginally more complicated as grains such as “buckwheat” are gluten free but contain the String “wheat”. In order to avoid this issue, the application searches for “wheat” in the ingredient string. While this could cause issues if the API misreads the string and misses a space in front of the ingredient, the author felt this was the best compromise for finding wheat in the string. The alternative method would be to find each occurrence of wheat in the string and check whether “buck” precedes it. While this would potentially be more accurate, it has a great risk of becoming excessively complex. The presence or absence of “wheat” will determine what String is displayed in the *TextView*.



Ingredient Strings have the potential to include both “gluten free” and “gluten”, so both of these cases, plus an additional check for no instances, needed to be handled. If the string contains the term “gluten free”, the *TextView* reflects this by saying the ingredients include a gluten free label. In the vast majority of cases, this will mean that the product is gluten free, but the author still chose to avoid giving direct medical advice with this statement. After checking for “gluten free”, the method checks if the ingredients list does or does not contain “gluten” and updates the *TextView* to reflect the results of this check.

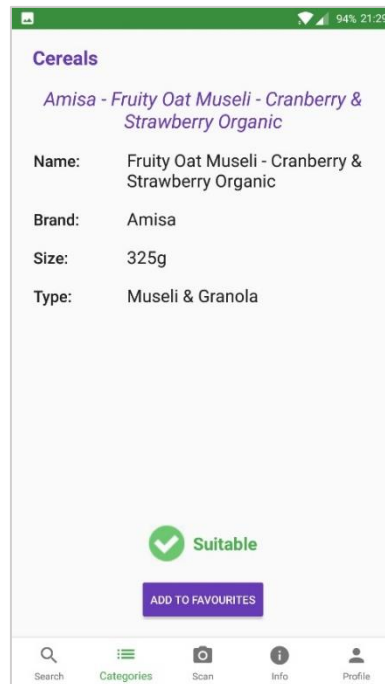
In addition to ingredient scanning, this iteration also included the addition of notifications through Firebase Cloud Messaging. Notifications can be sent to all users of the application via the Firebase console. Implementing this functionality required the addition of a Firebase messaging dependency to the app’s *build.gradle* file. Once included, application users can receive notifications in their notification drawer. The application will use the default application sound and tapping on it will bring the user to the application homepage. Notifications are created in the online Firebase console, and do not have an automatic function at this time.



**Image 5-26:** Firebase Cloud Messaging notification in the notification drawer

During this iteration, further tweaks were made to the user interface to improve usability and the overall look of the application. A minor change to all pages was removing the top navigation bar. The bar offered no functionality and took up valuable screen space. Removing the bar streamlined the overall look of the application and avoided confusing users who may have looked to it for some functionality.

Another minor application change was to alter the look of all buttons. In order to help the buttons to stand out against the rest of the application, the buttons were changed to use the contrast colour of the app. This change should help users to find buttons more easily against the rest of the screen.



**Image 5-27:** Updated profile page

The most major change was to the product page. The initial layout of the product page was functional, but not hugely attractive. In order to improve readability, the product page changed to use a *TableLayout*, adding natural order to the text. This makes it easier for users to interpret and lets them access information more quickly. The author also chose to use the same green tick notation from the ingredient interpretation screen on the product page to help users feel more confident about their choices. While all products in the database are suitable for a gluten free diet, the addition of the label will provide visual reinforcement of this fact.

## **5.6 Implementation difficulties**

There were two major causes of implementation issues: understanding asynchronous database calls and limitations of the Google Text Recognition API.

As discussed, in order to correctly interact with results from querying the asynchronous Firebase database, any processing or display functions must occur within the call to the database itself. Due to previous experience with synchronous databases, understanding this concept presented a new challenge for the author. Initial misunderstandings of asynchronous data calls led the author to become frustrated, as results of the query were visible in the development console but were not displayed on the screen. Once the author researched the database structure further and developed a deeper understanding of this query structure, this difficulty was overcome.

Limitations of the Text Recognition API also created issues during implementation. Major issues related to being unable to limit what text was being read at any given time. The library simply reads any text it can see without any way to limit this. This can lead to nonsensical text as blocks are read out of order and combined in new ways. To avoid this issue, the author hoped to allow users to limit what text is read by capturing a single image and passing that to the interpreter. However, this created additional complications by requiring read and write permissions for the application to the phone's memory. Without this, the picture could not be captured. As an alternative, the author chose to add a button to the scanning screen that allows the user to stop the scan when they are happy with the string being read. This was the best compromise to deal with the issues raised and provides a functional solution to the problem but cannot ensure that the text being read is accurate.

## **5.7 Code Repository**

Throughout development and implementation of the application, the author used GitHub as a code repository. The author used two branches – master and dev – to track progress made during implementation. The master branch was kept clean for commits at the end of each iteration, marking the release version for that cycle. The dev branch allowed commits at any time and recorded progress at every stage. This allowed the author to keep backups of every version of the project and would allow the author to revert to a previous version if necessary.

The author found the use of GitHub very important to the implementation of the application. It allowed the author to track progress and to experiment with adding new features, knowing that a working version was always available to be downloaded.

While the author had used GitHub in previous projects, knowledge of the use of branches was limited. Throughout the course of the project, the author gained valuable experience in branching and overall use of GitHub, overcoming some initial difficulties with using the system.

## **Chapter 6: Testing**

Throughout the development process, the author undertook various forms of testing the application. This included incremental testing of each feature as it was added, and usability testing once the application reached its final stages.

Testing of functionality as part of the incremental development model allowed the author to ensure that new features were working well before the cycle was completed. This helped to guarantee the overall quality of the application at each stage of development.

The author also sought ethical approval from the college to allow user testing of the application. Once this approval was granted, the author invited volunteers to participate in a usability study. Feedback from user testing was used to inform final changes to the application user interface and helped to highlight potential areas for future work, discussed in Chapter 8 of the report.

### **6.1 Incremental Testing**

As discussed in Chapter 5, development of this application followed an iterative model. This required the author to test the application throughout the development process, evaluating each feature as it was added. This implementation model helped to ensure that no new features could be added to the application before previous features were working as required.

While functions were mainly tested on completion, some testing also occurred as the feature was being developed. This testing included checking that database queries were functioning correctly, or that the ingredient interpreter was reading the correct text from the interpreted string. This type of testing often required the use of dummy code or input to replace functionality that was not yet complete. While this added some extra time to the

development process, it helped to ensure that any faults or problems with individual functions were discovered early in their development.

When functions were completed, each one was tested both as a complete unit on its own and as an integrated part of the whole application. Complete testing of the feature in isolation allowed the author to ensure it was functioning as required which helped to be sure that, when the function was integrated with the rest of the project, any issues were arising from the integration of parts rather than the functionality itself. Testing the whole application after this integration allowed the author to ensure that any integration issues were solved as they arose, and that no functions were in conflict with each other.

At each step of development and testing, the author also revisited the design heuristics and ergonomic ideas discussed in Chapter 3. This helped to ensure that the application conformed to the ideals that contributed to its initial design. While the early stages of implementation focused on functional design, the design in later iterations evolved to reflect these concepts. The overall look of the application was also considered during user testing, as users were asked to rate the application style and colour scheme.

## **6.2 Usability Testing**

Usability testing aims to evaluate how users interact with and experience an application. While there is no precise definition of what usability means, user surveys can help developers to deepen their understanding of how users feel about using their application. Usability testing can also help to detect issues that may have been missed by developers who are intimately familiar with an application. These issues may include bugs and crashes, or a misunderstanding of the meaning intended by icons.

In order to perform tests on users, the author sought ethical approval from the college. The application included a participant information sheet, an informed consent form, and a usability questionnaire. The full set of documents can be viewed in Appendix C.

#### *6.2.1 System Usability Scale*

Many scales exist for testing usability of applications. The System Usability Scale (SUS) was invented by John Brooke in 1986 as a fast and inexpensive method to evaluate system usability <sup>(31)</sup>. Since its inception over 30 years ago, the SUS has been used consistently and is well regarded as a tool for measuring system usability against industry standards.

The SUS is made up of ten statements. Participants rank each statement on a Likert scale ranging from *1 – Strongly Disagree* to *5 – Strongly Agree*. Statements alternate from being positive about the application to being negative. Users should be asked to complete the survey before any debriefing takes place to ensure they give the most honest answers about their opinions.

Each user's answers are processed by the surveyor to fall in a range from zero to four. These values are then added together and multiplied by 2.5 to give an overall usability score out of 100. The average SUS score is 68. Scores under 68 indicate that some problems exist with the system that are impeding usability – distance from 68 indicates how severe these problems are. Scores above 68 indicate that users found the system easy and pleasant to use.

#### *6.2.2 Test Environment*

In order to minimise the effects of variety on the results of the usability study, the author developed a standard procedure for each tester. Participants were invited into a quiet room with the tester. The participant information sheet was read aloud. Participants were informed about the purpose of the study and invited to ask any questions that might arise during the testing process. Participants were required to read and sign the consent form

before participating in the study. Those who chose to participate were free to leave the study at any time.

To test the application, participants were invited to create an account, then explore the application and test all the features. A variety of food products both present and absent from the app database, both gluten containing and gluten free, were provided to participants to test the application with. Participants were invited to take as long as they felt necessary to experience the features of the application.

Once participants had completed the testing phase, they were invited to complete the usability questionnaire. Questions included demographic factors, technical questions, and the System Usability Scale. Any questions that the participants had were answered, then participants were debriefed and thanked for their participation.

### *6.2.3 Test Results*

Results from the usability survey were anonymised and processed in an Excel spreadsheet.

In total, ten users took part in the survey. Of these participants, seven belonged to the 18-24 age category. Three suffered from some food intolerance, though none followed a gluten free diet. All participants agreed that they would recommend the application to friends who follow a gluten free diet. Half of users stated they would pay for such an app if following a gluten free diet, proposing prices between €2 and €10. No users experienced crashing during the testing process, and only one suffered from lag.

As part of the survey, users were asked to rate the application on a scale from one to ten. From this, the application received an average rating of 8.4. Usability scores from the SUS ranged from 77.5 to 95, with an average usability score of 87. This is well above the average



score and signified a highly usable system. The standard deviation of this usability score was 6.5, signifying minimal variation from the mean. This value indicates that 68.2 percent of answers fell between 81.5 and 93.5, or plus or minus one standard deviation from the mean.

#### *6.2.4 Feedback from User Testing*

Overall, feedback from users during the testing process was highly positive. No users were confused by the icons used for navigation, and all found navigating in the app fairly intuitive. The only issues discussed by users related to the ingredient text scanner:

*“Seems difficult to use on non-high-contrast text; cannot manually refocus camera”*

– Anonymous User #1

Both of these issues relate to limitations of the Text Recognition API. In order to function well, the library needs to use the autofocus ability of the phone camera. Manual focus is not compatible with the structure used, so removing autofocus would result in the app only being able to focus on ingredient lists at a single distance from the phone. The API also struggles to read certain fonts or text colours on particular backgrounds. This is a limitation of the API that the author could not improve on, but which may improve over time as the API is developed further.

*“I was slightly confused by the scanning function when words kept popping up”*

– Anonymous User #2

In order to provide users with feedback for what text is being scanned, the author chose to display the text being read on the screen. Providing this text ensures that users do not become frustrated when trying to scan ingredients. Without it, a user could attempt to scan ingredients but not be told whether or not the camera is currently detecting the text.

The user would press the scan button and be told that ingredients were not detected but would not be able to tell when the reader is detecting ingredients. This would result in user frustration at being unable to find the right time to press the button and may result in users abandoning the application.

While this tester reported finding the text confusing, the author felt that the perceived benefits and potential to annoy users without the functionality meant that it should stay in the application. No other users reported issues with the text which led the author to believe it would not be a major issue moving forward.

*“Scan instructions need to be more precise”*

– Anonymous User #7

Initially, the scan instructions read: *Use your camera to read ingredients*. After one user reported finding these instructions confusing, the author decided to rephrase them to make it clearer that users were required to press the button to scan. The updated instructions now read: *Press the button when ingredients appear on the screen*. This instruction addresses the issue with not knowing to press the button, and also lets the user know that they should wait for ingredient text to be visible before trying to scan.

Otherwise, users were generally very positive about the ingredient scanner functionality, saying “it works really well”, is “really cool”, and is a “fantastic ability”. Other feedback for the app as whole was very positive and reflected the high rating given to the app.

## Chapter 7: Business Plan

There are various models available to help entrepreneurs to develop business plans. One of the most popular is the Business Model Canvas, which was proposed by Alex Osterwalder as a method of quickly developing the key sections of a business plan. This is particularly useful for start-ups on the verge of entering the market. This model focuses on factors such as key resources, activities, partners, and relationships with customers. Its goal is to help businesses to identify what makes their product special in the market and asks the business to identify costs and potential revenue streams. A full-page, more detailed version of the Business Model Canvas can be found in Appendix D.

Key Partners	Key Activities	Value Proposition	Customer Relationships	Customer Segments
	Key Resources		Channels	
Cost Structure			Revenue Streams	

**Image 7-1:** Business Model Canvas

This chapter will address each element of the canvas and identify their role in developing the project further.

### 7.1 Value Proposition

The value proposition refers to the product or services the application offers to create value for its users. The major benefit of this application is to improve quality of life for followers of gluten free diets by making it easier for them to find gluten free foods. The application provides users with an easy-to-use database search, the ability to view all foods belonging to a particular category at once, a quick ingredient interpreter, and information about gluten free diets. This will help consumers to take control of their diet and develop a deeper understanding of the foods they can safely eat.

## **7.2 Key Partners**

Key partners refer to any external partners a business needs to develop or deliver their product or service. Partnership may be temporary or may last for the entire lifecycle of the application.

In order to develop this project, the author sought the support of the Coeliac Society of Ireland in providing a section of their gluten free foods database. To continue producing the application and to expand the database to full production size, the author would seek continued partnership with the CSOI. This partnership would allow the author to take advantage of their existing subscriber base to target new customers, discussed in Section 7, and their existing relationships with food producers to develop new revenue streams, discussed in Section 9.

The author would also consider partnering with well-known gluten free food bloggers to promote the application once complete. While this is potentially expensive, it could help the application to reach a large number of consumers through the blogger's followers

## **7.3 Key Resources**

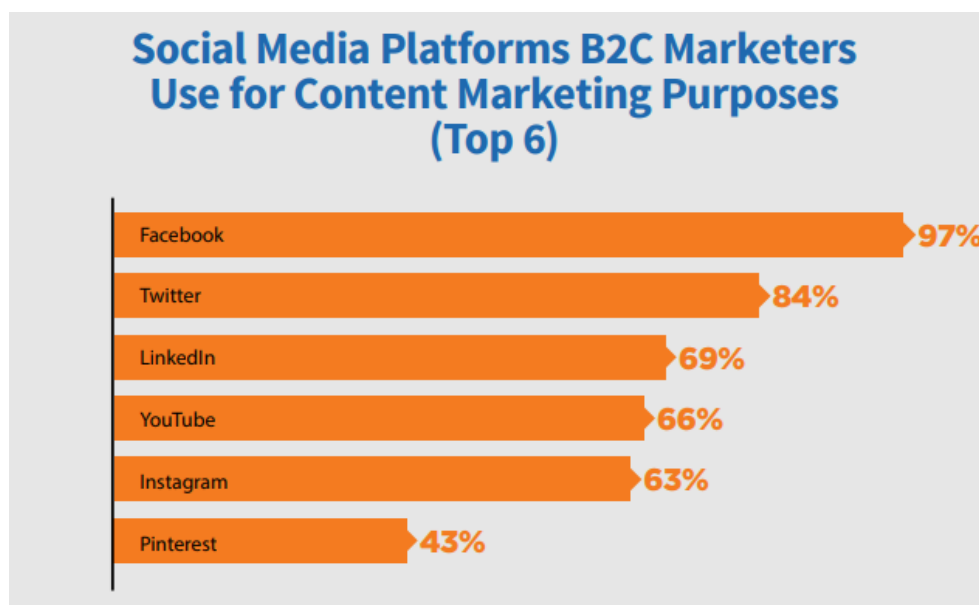
Key resources are those that the business must leverage to create value for the customer and strategic advantage for the company. This application's unique resource is its data. Through partnership with the CSOI, this application has access to a well-maintained and accurate database of gluten free foods. The CSOI is not currently offering a mobile application using their database, instead offering a printed book updated yearly. This project is therefore a unique technology source of the data for consumers.

## **7.4 Key Activities**

Key activities are those activities the business must carry out to create value. Successful implementation of the application is the first major value creating activity.

Following this, the major activities are app maintenance and updates and maintaining a strong social media presence.

Social media marketing has been recognised as an important part of modern business. 86 percent of American business-to-consumer businesses surveyed in 2018 use content marketing – the practice of creating and distribution relevant and consistent content to attract and retain a customer base – to great effect on social media <sup>(32)</sup>. A well-structured social media plan can help businesses to quickly grow their consumer bases. The same survey found Facebook and Twitter to be the most popular social networks for this type of marketing, as shown in Image 7-2.



**Image 7-2:** Social media platforms for B2C content marketing <sup>(33)</sup>

Instagram would also be a useful social network to use for a food-based app, as many food bloggers use Instagram to communicate with their followers. Curating a consistent and brand-relevant feed would allow the brand to directly interact with application users by sharing their posts and allow those users to share the app's own posts. Posts could include gluten free recipes, promoted products, or information about new application features.

### **7.5 Customer Segments**

Customer segments refer to the specific groups of customers the application is aimed at. This project is targeted at gluten free dieters of all forms, but particularly those who strictly follow such a diet. Those who are strictly gluten free will be more likely to use and pay for such an application. The author also recognises that while the application aims to target dieters of all ages, older populations may be less likely to use a smartphone application.

### **7.6 Customer Relationships**

An important part of developing a business plan is understanding how the business will build and maintain relationships with its customers. This project aims to build the relationship by providing support for customers undertaking a gluten free diet and becoming a trusted source of information for those customers. The relationship will be maintained by keeping the application up-to-date and as accurate as possible, adding new functionality, and interacting with customers through social networks, as discussed in Section 7.

### **7.7 Channels**

Channels address how the business will reach its customers. The application will be distributed through the Google Play Store, and so will be available to all Android users who wish to try the application.

Advertising channels would take advantage of partnerships with the Coeliac Society of Ireland and gluten free food bloggers as discussed in Section 2. Advertising in their publications would help the application to reach new audiences. The application would also try to take advantage of word-of-mouth advertising on social media through shared blog posts, sharing of the application's own posts, and posts to social groups on Facebook and other networks.

## **7.8 Cost Structure**

Costs facing a business may be both fixed and variable. The major costs facing the application are those relating to hosting and maintaining the application database, which will vary over the application lifecycle. For its current size and user base, the Firebase database is free to use. As the application expands, it will be forced to move to paid plans, whose cost can be customised depending on the size of the database and number of users accessing the data.

Other costs include listing the application on the Google Play Store, which requires a fixed \$25 fee to register for a publisher account <sup>(34)</sup>. If the app charges for use through the Play Store, Google will also charge a 30 percent fee per transaction <sup>(35)</sup>.

Another potential cost the application may encounter is if a direct partnership with the Coeliac Society cannot be established. If not, the application will face costs of licensing the data from the Society.

## **7.9 Revenue Model**

In order to offset the business costs and generate a profit, a business must have a revenue model. For this application, it could include a charge to use the application and allowing food producers to promote products in the app.

Applications can charge a single fee for use or a yearly subscription fee. As discussed in Chapter 2, Coeliac UK uses a subscription model for their applications. Membership of the charity includes a fee for using the app. Food Maestro is free to use, while Gluten Free Food Scanner UK charges a one-time fee for use.

Half of users who participated in usability testing for the application were willing to pay a one-time fee ranging from €2 to €10, signifying that users would potentially be willing to pay to use the application. In order to offset the yearly database hosting fees, it would be prudent to pursue a subscription model that charges users each year as opposed to a single fee.

Depending on the partnership deal negotiated with the Coeliac Society of Ireland, charity members could be offered an account for the application as part of their yearly membership fee. Part of this fee would then be given to the developers to offset costs.

Partnering with food producers could also generate revenue for the application. Food producers could be invited to pay a fee to advertise their product on the application or to promote it in search results, similar to promoted results in Google searches.



## **Chapter 8: Conclusion**

This report has outlined the steps taken to achieve the goal of creating a functional Android application that improved and updated the current method of finding gluten free foods in a printed book. Steps to achieve this included research, design, implementation, and user testing. This chapter will evaluate the development process, discuss the problems encountered, and detail potential future work on the application.

### **8.1 Evaluation**

The final version of the application successfully achieved all the functional requirements stated when development began. It is capable of viewing and searching a food database, viewing detailed product information, reading ingredients label via the phone camera and scanning for the presence of gluten-containing ingredients, provides information about gluten free diets to users, and requires the user to create an account to use any of this functionality.

The application also satisfies the non-functional requirements established. It is highly usable, as shown by user testing and a System Usability Scale score of 87. Users in testing felt comfortable using the application, and all were able to use all of the application functionality. The application is easy to maintain, and the code and database is structured in such a way that it can easily be extended to accommodate additional functionality or a larger database.

The application design conforms to the design heuristics and ergonomics outlined during the design phase. Those heuristics identified as most important – visible system status, use of recognition, minimalist design, and error recovery – were all achieved. The results of the user testing again highlighted this achievement, as users were able to navigate the application easily and did not encounter errors.

Overall, the application succeeded in providing a new, easy-to-use, solution to a long running problem faced by many followers of gluten free diets. The application improves upon existing applications in the marketplace, provides unique functionality in the form of the ingredient scanner, and is a new offering on the Irish market.

## **8.2 Difficulties**

There were two major difficulties encountered when developing this application. The first related to understanding asynchronous database calls, which was a new concept for the author. The second related to limitations of the Google Text Recognition library.

While the author had previous experience in using databases, this experience was solely in synchronous access. In this structure, database queries are processed one at a time, and code after the query does not run until the query has completed and the response has been returned. On the other hand, asynchronous access processes database queries in tandem with any following code and returns the response once it is ready. This avoids application lag when waiting for a response.

While the author was aware of this quality, she expected that results of asynchronous queries could be used elsewhere in code in the same way as synchronous queries. However, any functions that use the data must be called from within the query. This includes displaying the results on the screen. This misunderstanding led to confusion when data was being returned correctly, but not being displayed on the screen. Once the author developed a deeper understanding of this technology, the issue was resolved, and data displayed correctly.

The limitations of the Google Text Recognition library impact on the functionality of the application. While the text recognition functions well, certain factors must still be considered. The library does not currently recognise certain fonts, very small font sizes, and

can struggle when text is not high-contrast or packaging is glossy. These limitations mean that the application cannot read text from every type of product. Another major limitation is that it is not possible to select a small section of text viewed by the camera to read – instead, the API reads all text in its view. The author attempted to overcome this issue by trimming the read string around “Ingredient” and some terminator but found there is no consistent terminator on product packaging. Some products follow their ingredients with “Allergy Advice/Information”, some with “Suitable for Vegetarians”, and others follow the products with no specific text. The author chose to trim the string at “Ingredient” but allow it to trail past whatever else it reads. While this is an imperfect solution, it is functional and allows the ingredient recogniser to work well.

### **8.3 Further Work**

The author identified several areas where the application could be extended in the future, including improvements to the current application and new developments.

- **Expanded database**

The current application runs on a small fraction of the actual Coeliac Society of Ireland database. In the future, to provide the best service to users, the application database would need to expand to include all of the products certified as gluten free by the Society or an alternative list of gluten free foods.

- **Barcode scanning**

Other applications on the market offer barcode scanning functionality, allowing the user to scan an item barcode to search for the item in the database. The author investigated implementing this functionality but discovered that while many libraries exist for scanning barcodes, there is no universal database of barcodes available. As such, there would be no way to ensure that the information the barcode scanner extracts directs the user to the correct product. This would be useful functionality, and the author would like to pursue implementing it in the future.

- **Offline capabilities**

The most common use location for this application would likely be in the supermarket while shopping for foods. Often, supermarkets have poor phone and WiFi signals, making it difficult to use an online database. Adding the ability to download the database and store it locally on the user's phone would make the application more usable in this context.

- **Highlighting allergens in scanned text**

As of 2014, EU law requires that the top 14 food allergens be highlighted in some way on packaged food ingredient labelling <sup>(36)</sup>. Most often, allergens are written in bold or in capitals. While the ingredients recreated on the application screen are not on food packaging and do not have to comply with this legislation, it may be beneficial for consumers for it to do so. The author would consider formatting the read ingredient string to highlight these allergens in the future.

- **Automatic notifications on database update**

If a food record is altered or removed from the database, it is important to inform consumers as quickly as possible so as not to put their health at risk. As users provide their emails on account creation, it would be possible to email all users to notify them of updates – the system the Coeliac Society of Ireland currently uses. However, it would be more efficient and beneficial to users to send an automatic push notification to their phone when a record is updated. Currently, the app implements manual push notifications.

- **iOS implementation**

While Android phones represent the majority of the Irish market, Apple iOS is also a popular operating system. The author would like to extend this implementation to iOS in order to serve the greatest possible number of consumers.

- **Other allergens**

In the future, it may be possible to extend this implementation to other allergens in partnership with other charities. Finding allergen free foods can be challenging for consumers, so expanding the implementation could be hugely beneficial. Updating the application for different allergens would require a separate database and different ingredient interpreter and information Strings.

#### **8.4 Closing Remarks**

This report has outlined the successful development of an Android application for managing a gluten free diet. The author is hugely satisfied with the work done on this project. The end result is a functional application that could be majorly beneficial to the 20 percent of Irish consumers who follow this diet.

The author learned valuable skills throughout this project. Most importantly, the skills required to develop all parts of a project as an independent developer, including project management, database administration, and development. Each iteration helped the author to gain new insight into the development of mobile applications and improve overall coding ability. The process also allowed the author to administer user testing, previously only studied in theory. This experience gave the author a deeper understanding of the testing process and gave valuable feedback about app design and functionality that can be taken forward to future projects.

The author also developed a business plan for the project, requiring deep analysis of the potential of such an application in the market. This allowed the author to draw on lessons learned in the business modules studied for her degree, including Entrepreneurship and Marketing and expand on the lessons of those modules.

Overall, the author would rate the project as a success. The final product is releasable and conforms to a high standard of usability and design. The completion of the project led to the gain of a well-rounded knowledge of Android applications, project development and management, and business strategy in practice.

## Chapter 9: References

### 9.1 Endnotes

- (1) Telis, G., 2013. *The truth about gluten-free, paleo and other diet books*. [Online] Available at: [https://www.washingtonpost.com/national/health-science/the-truth-about-gluten-free-paleo-and-other-diet-books/2013/08/05/2aeb5874-eef9-11e2-bed3-b9b6fe264871\\_story.html?utm\\_term=.d31f8d74f156](https://www.washingtonpost.com/national/health-science/the-truth-about-gluten-free-paleo-and-other-diet-books/2013/08/05/2aeb5874-eef9-11e2-bed3-b9b6fe264871_story.html?utm_term=.d31f8d74f156) [Accessed 22 March 2018].
- (2) Konijeti, G. G. et al., 2017. Efficacy of the Autoimmune Protocol Diet for Inflammatory Bowel Disease. *Inflammatory Bowel Diseases*, 23(11), pp. 2054-2060.
- (3) Bord Bia Insight Centre, 2017. *Consumer Insight Into Gluten Free in Ireland*, Dublin: The Thinking House.
- (4) Coeliac UK, 2018. *Law on gluten free*. [Online] Available at: <https://www.coeliac.org.uk/gluten-free-diet-and-lifestyle/food-shopping/law-on-gluten-free/> [Accessed 22 March 2018].
- (5) Ludvigsson, J. F. et al., 2013. The Oslo definitions for coeliac disease and related terms. *Gut*, 62(1), pp. 43-52.
- (6) World Bank, 2018. *Population, total*. [Online] Available at: <https://data.worldbank.org/indicator/SP.POP.TOTL?locations=IE> [Accessed 21 March 2018].
- (7) Coeliac Society of Ireland, 2018. *Prevalence and Diagnosis*. [Online] Available at: <https://www.coeliac.ie/join-now/professional/prevalence-and-diagnosis/> [Accessed 22 March 2018].
- (8) Bai, J. C. et al., 2013. World Gastroenterology Organisation Global Guidelines on Celiac Disease. *Journal of Clinical Gastroenterology*, 47(2), pp. 121-126.
- (9) Deloitte, 2017. *Global Mobile Consumer Survey – The Irish Cut*, Dublin: Deloitte.
- (10) Deloitte, 2016. *There's no place like phone: half of Irish consumers check smartphones in the middle of the night*. [Online] Available at: <https://www2.deloitte.com/ie/en/pages/about-deloitte/articles/Theres-no-place-like-phone.html> [Accessed 22 March 2018].

- (11) Statista, 2018. *Global market share held by smartphone operating systems from 2009 to 2017*. [Online]  
Available at: <https://www.statista.com/statistics/263453/global-market-share-held-by-smartphone-operating-systems/>  
[Accessed 22 March 2018].
- (12) StatCounter, 2018. *Mobile Operating System Market Share in Ireland - February 2018*. [Online]  
Available at: <http://gs.statcounter.com/os-market-share/mobile/ireland>  
[Accessed 22 March 2018].
- (13) Coeliac Society of Ireland, 2017. *Food List App*. [Online]  
Available at: <https://www.coeliac.ie/app/>  
[Accessed 22 March 2018].
- (14) Coeliac UK, 2018. *Join Us*. [Online]  
Available at: <https://www.coeliac.org.uk/join-us/>  
[Accessed 20 March 2018].
- (15) Scan Gluten Free, 2018. *Gluten Free Scanner UK - Coeliac healthy diet*. [Online]  
Available at:  
<https://play.google.com/store/apps/details?id=com.scanglutenfreeuk&hl=en>  
[Accessed 21 March 2018].
- (16) Doherty, G., 2017. *Users*, lecture notes distributed in CS4051 Human Factors at Trinity College Dublin, Ireland in October 2017.
- (17) Scan Gluten Free, 2018. *Gluten Free Scanner UK - Coeliac healthy diet*.
- (18) Coeliac UK, 2018. *Gluten free food checker*. [Online]  
Available at:  
<https://play.google.com/store/apps/details?id=com.foodmaestro.coeliacuk>  
[Accessed 20 March 2018].
- (19) Coeliac UK, 2018. *Coeliac UK - Gluten free app*. [Online]  
Available at: <https://play.google.com/store/apps/details?id=com.locassa.coeliac>  
[Accessed 20 March 2018].
- (20) FoodMaestro Ltd., 2018. *FoodMaestro*. [Online]  
Available at:  
<https://play.google.com/store/apps/details?id=com.foodmaestro.foodmaestro>  
[Accessed 20 March 2018].
- (21) Datururks, 2017. *Image text recognition APIs showdown*. [Online]  
Available at: <https://dataturks.com/blog/compare-image-text-recognition-apis.php>  
[Accessed 16 March 2018].



- (22) Ingram, S., 2016. *The Thumb Zone: Designing For Mobile Users*. [Online]  
Available at: <https://www.smashingmagazine.com/2016/09/the-thumb-zone-designing-for-mobile-users/>  
[Accessed 30 March 2018].
- (23) Google Cloud, 2018. *Build an Android App Using Firebase and the App Engine Flexible Environment*. [Online]  
Available at: <https://cloud.google.com/solutions/mobile/mobile-firebase-app-engine-flexible>  
[Accessed 30 March 2018].
- (24) Doherty, G., 2017. *Users*.
- (25) Colour Blind Awareness, 2018. *Colour Blindness*. [Online]  
Available at: <http://www.colourblindawareness.org/colour-blindness/>  
[Accessed 28 March 2018].
- (26) Android Developers, 2018. *Application Fundamentals*. [Online]  
Available at: <https://developer.android.com/guide/components/fundamentals.html>  
[Accessed 30 March 2018].
- (27) Firebase, 2018. *Firebase Products*. [Online]  
Available at: <https://firebase.google.com/>  
[Accessed 30 March 2018].
- (28) Google Developers, 2018. *Text Recognition API Overview*. [Online]  
Available at: <https://developers.google.com/vision/text-overview>  
[Accessed 30 March 2018].
- (29) Nallur, V., 2016. *SW Development Processes*, lecture notes distributed in CS3012 Software Engineering at Trinity College Dublin, Ireland in November 2016.
- (30) Ghahrai, A., 2017. *Incremental Model*. [Online]  
Available at: <https://www.testingexcellence.com/incremental-model/>  
[Accessed 30 March 2018].
- (31) Brooke, J., 1996. SUS - A Quick and Dirty Usability Scale. In: P. W. Jordan, B. Thomas, B. A. Weerdmeester & I. L. McClelland, eds. *Usability Evaluation in Industry*. London: Taylor & Francis, pp. 189-194.
- (32) Content Marketing Institute, 2018. *B2C Content Marketing - 2018 Benchmarks, Budgets, and Trends - North America*, New York: Content Marketing Institute.
- (33) Ibid.

- (34) Google Play, 2018. *Google Play Console*. [Online]  
Available at: <https://play.google.com/apps/publish/signup/>  
[Accessed 9 April 2018].
- (35) Google Play, 2018. *Transaction Fees*. [Online]  
Available at: <https://support.google.com/googleplay/android-developer/answer/112622?hl=en>  
[Accessed 9 April 2018].
- (36) Food Standards Agency, 2013. *Advice on Food Allergen Labelling*, London: Food Standards Agency.

## **9.2 Bibliography**

Bord Bia Insight Centre, 2017. *Gluten Free Stories from Ireland*, Dublin: The Thinking House.

Doherty, G., 2017. *Evaluation*, lecture notes distributed in CS4051 Human Factors at Trinity College Dublin, Ireland in November 2017.

Doherty, G., 2017. *Prototyping*, lecture notes distributed in CS4051 Human Factors at Trinity College Dublin, Ireland in November 2017.

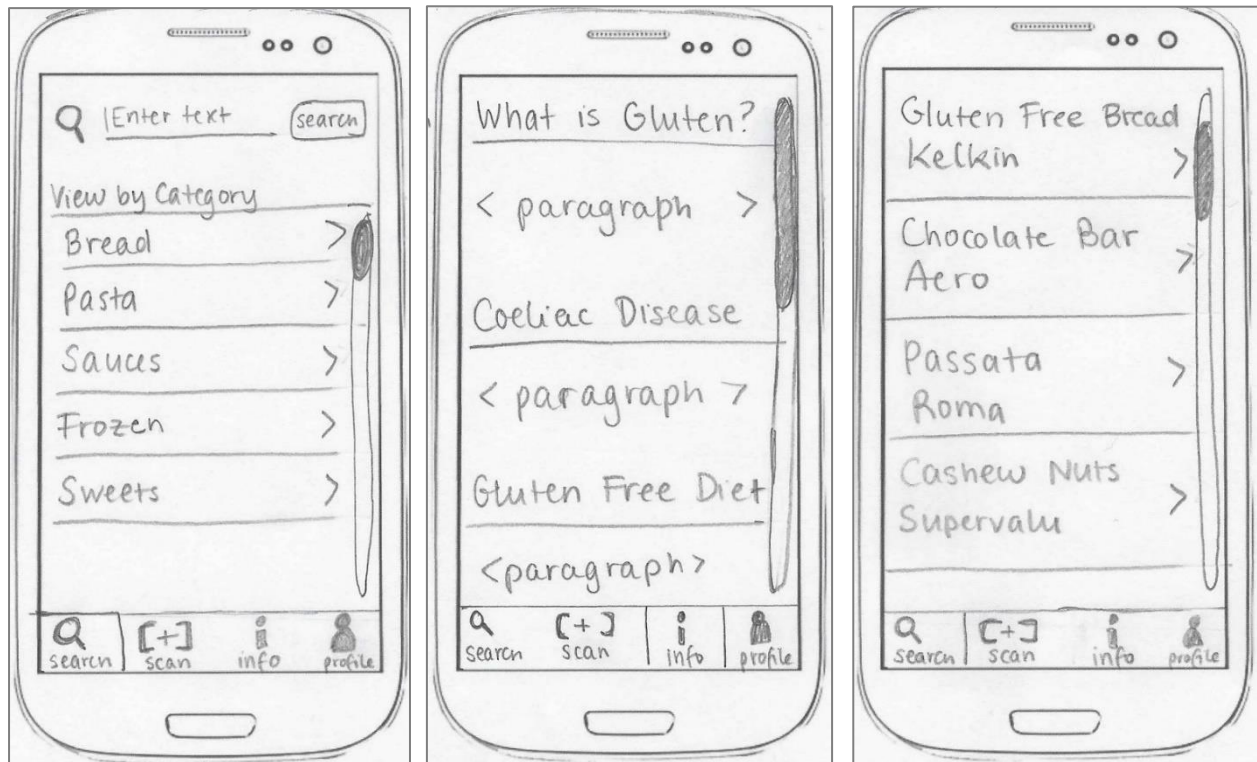
Google Developers, 2018. *Optical character recognition (OCR)*. [Online]  
Available at: <https://cloud.google.com/vision/docs/ocr>  
[Accessed 30 March 2018].

Strategyzer, 2018. *Business Model Canvas*. [Online]  
Available at: <https://strategyzer.com/>  
[Accessed 8 April 2018].

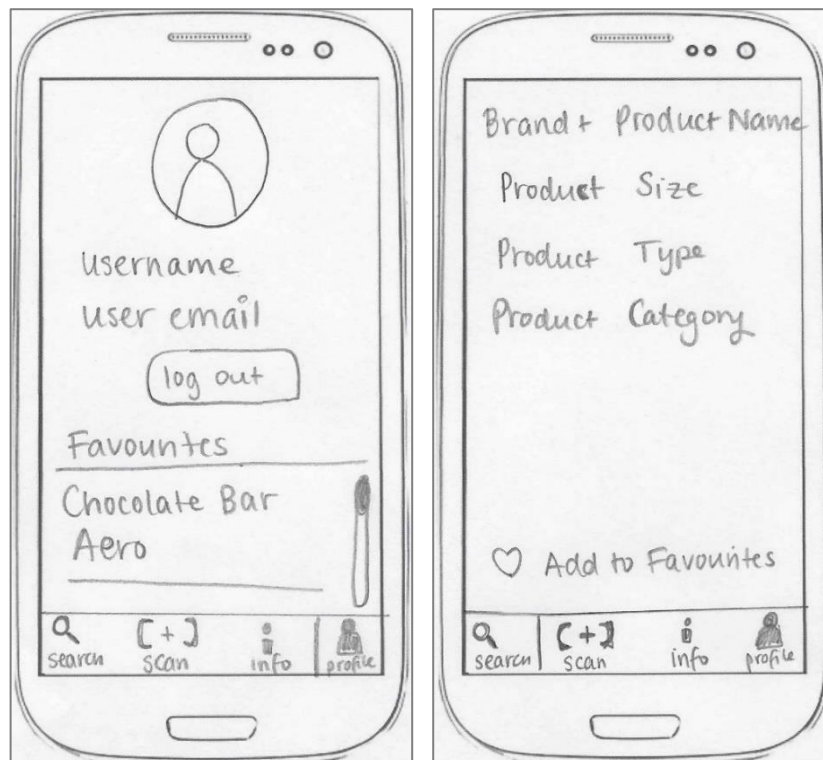
## Appendices

### Appendix A: Prototypes

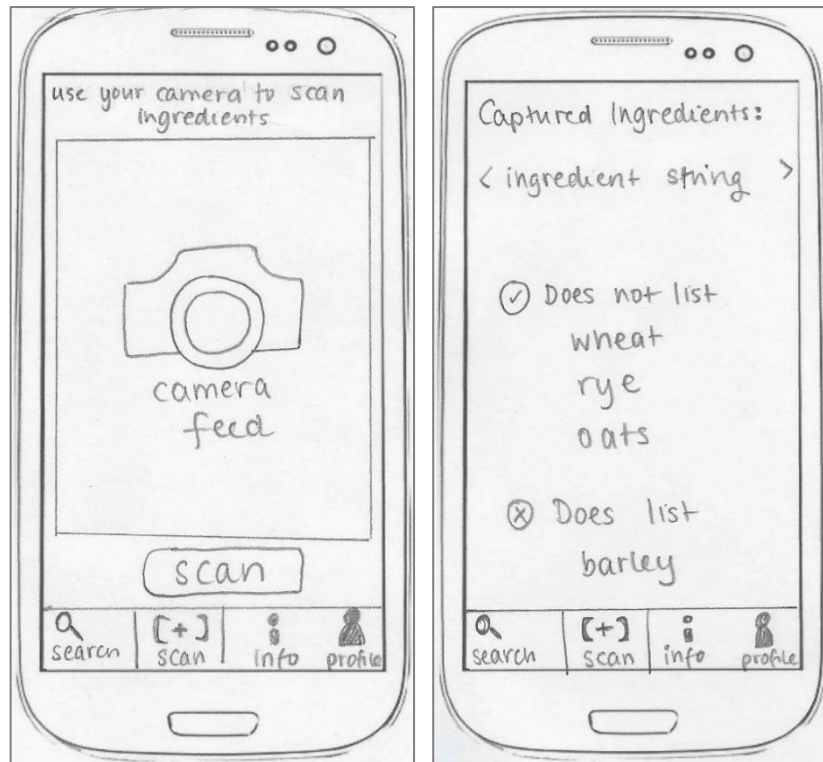
#### Low-Fidelity



**Image A-1:** Low-fidelity home page, information screen, and product list



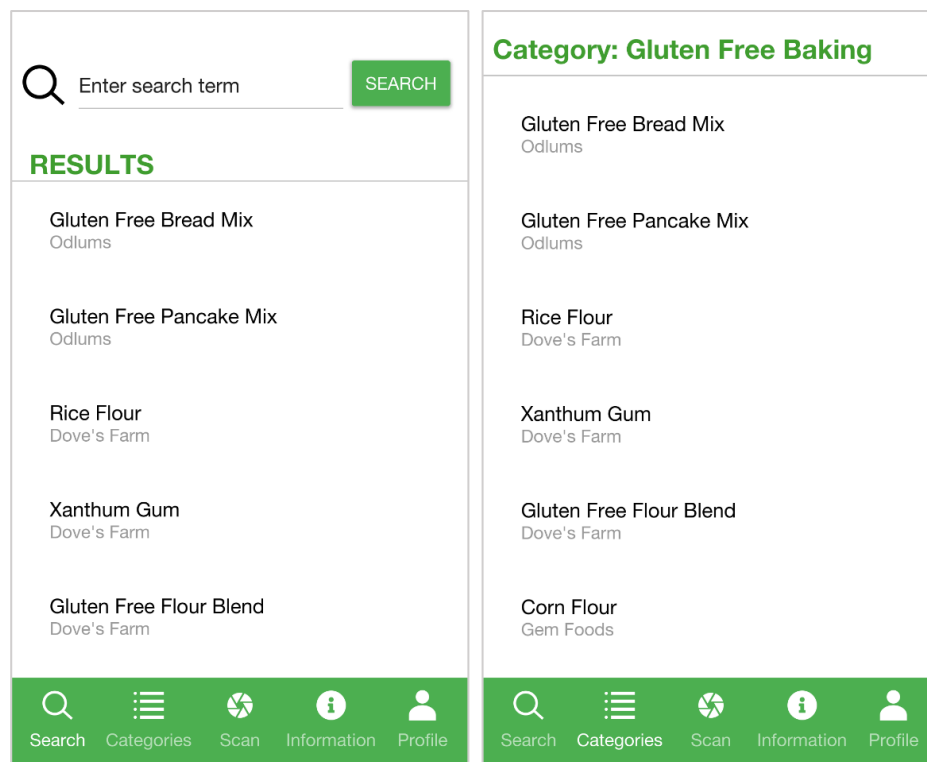
**Image A-2:** Low-fidelity product page and profile



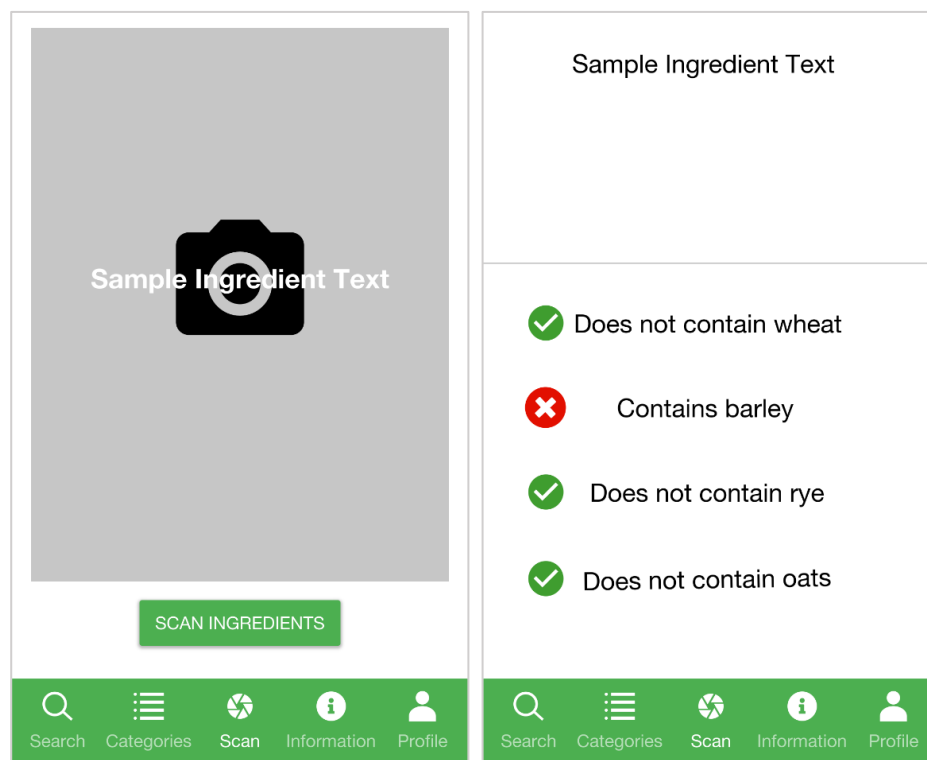
**Image A-3:** Low-fidelity camera screen and ingredient interpreter

Low-fidelity prototypes were drawn by hand.

## High-Fidelity



**Image A-4:** High-fidelity home page and category view



**Image A-5:** High-fidelity camera screen and ingredient interpreter

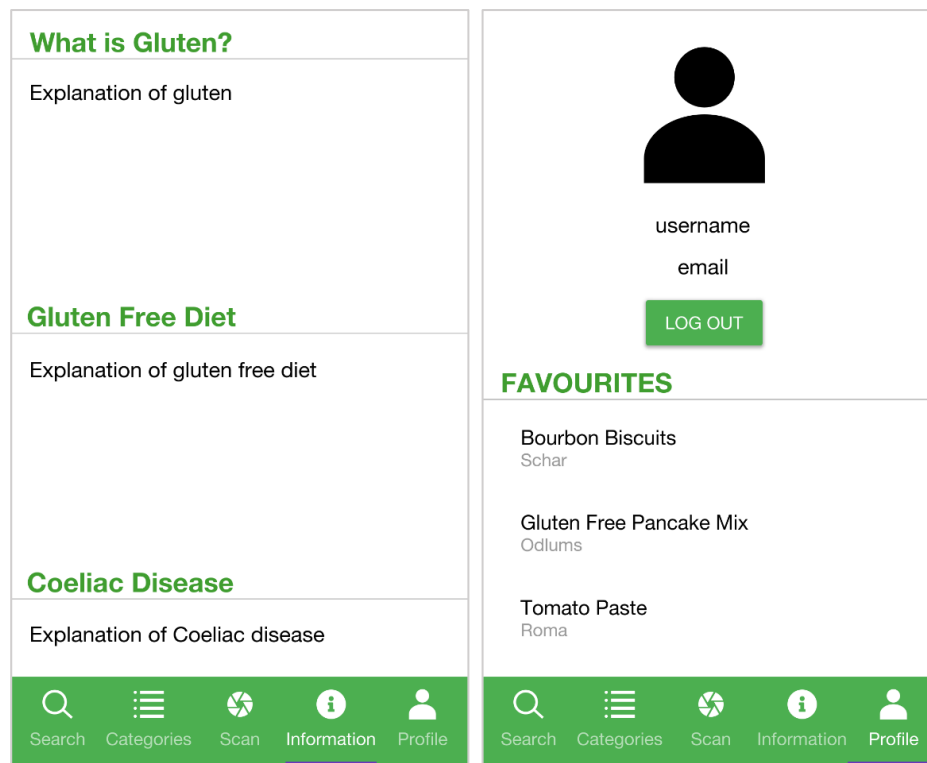


Image A-6: High-fidelity information screen and profile page

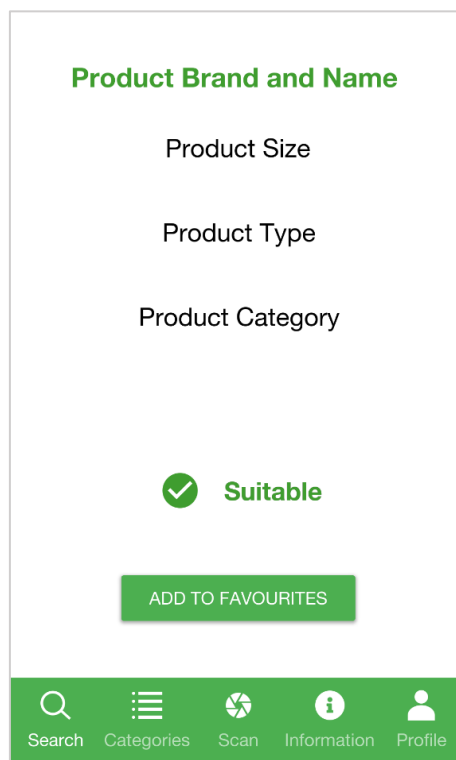


Image A-7: High-fidelity product page

High-fidelity prototypes were produced using proto.io.

## Appendix B: Final Design

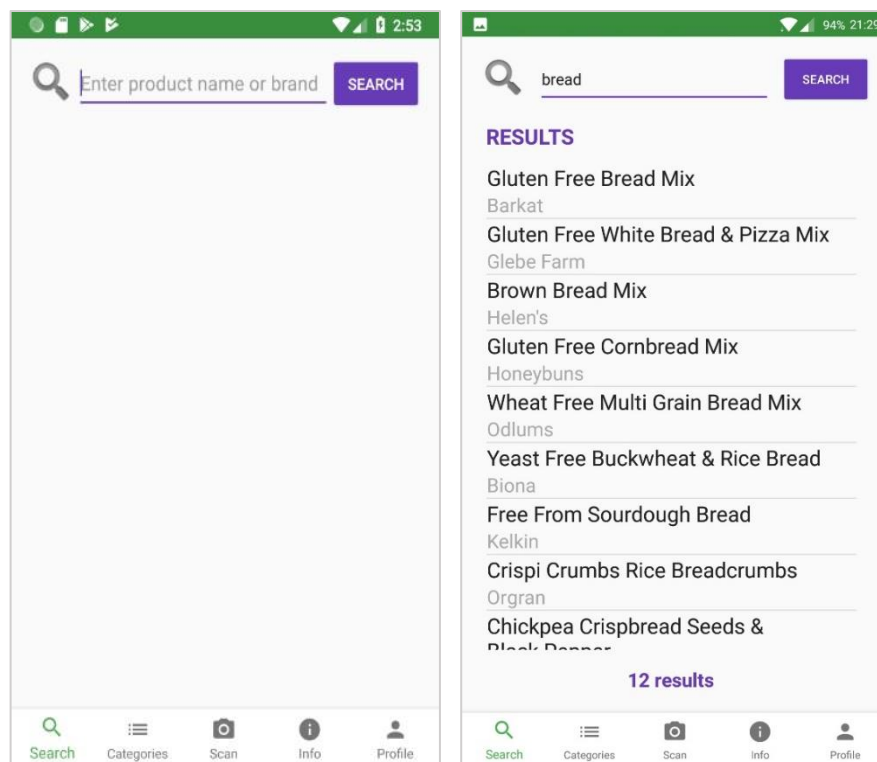


Image B-1: Final design of home page and search results

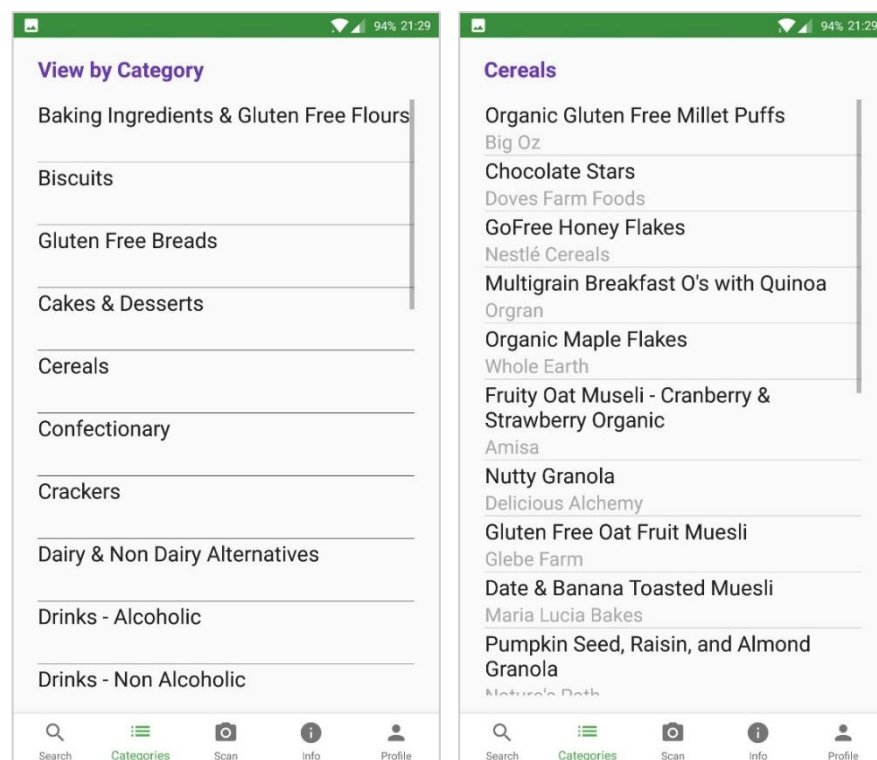


Image B-2: Final design of category list and view

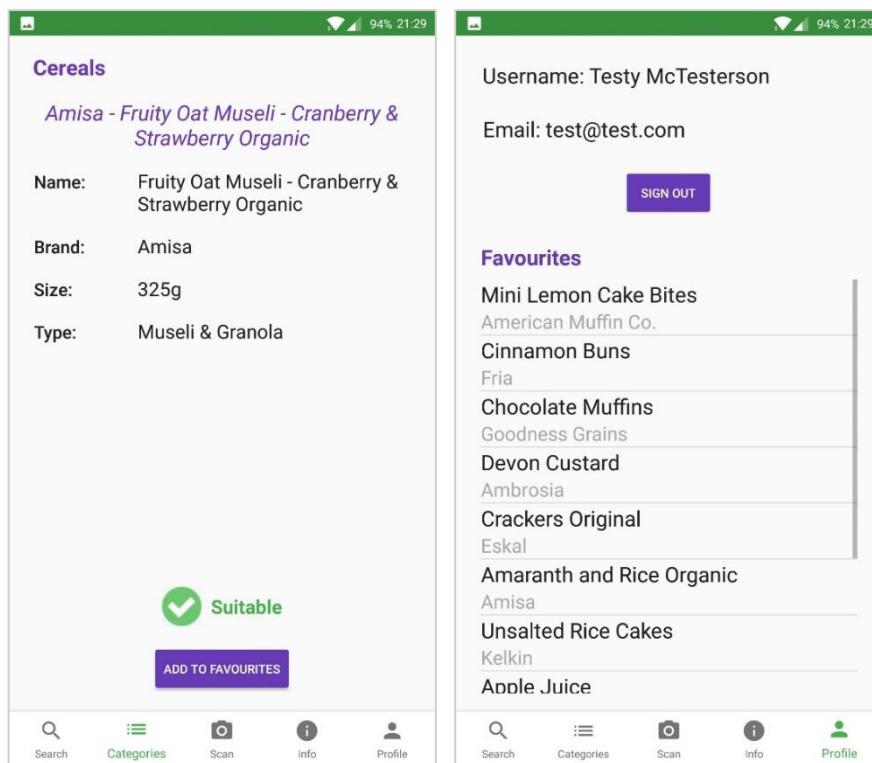


Image B-3: Final design of product screen and profile

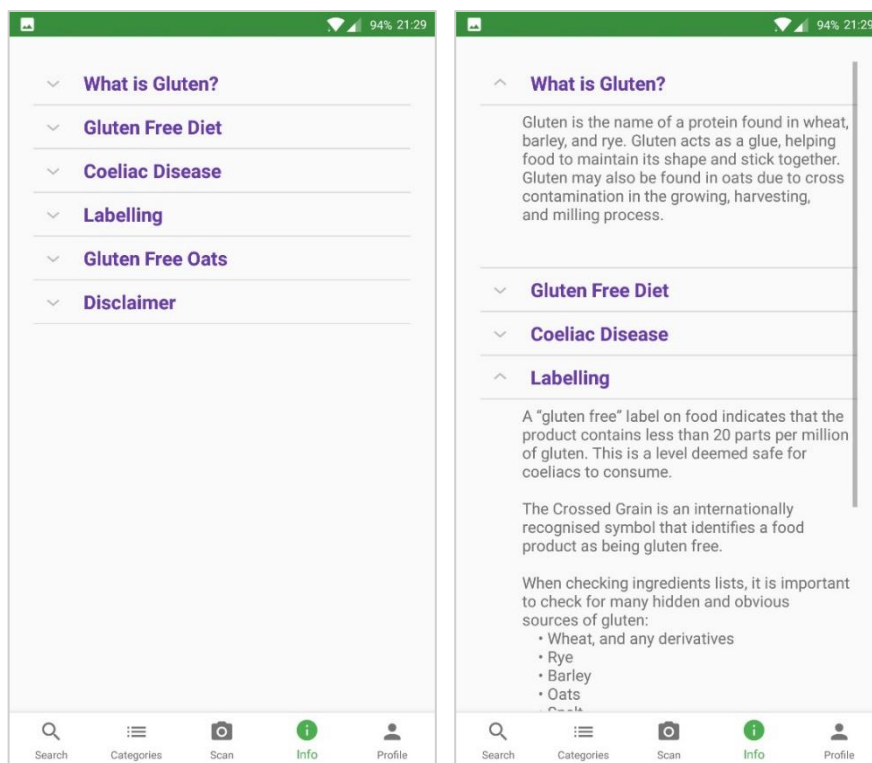


Image B-4: Final design of information screen – collapsed and expanded





## **Appendix C: Ethics Application Documents**

### **TRINITY COLLEGE DUBLIN OUTLINE OF RESEARCH PROPOSAL**

---

**Project Title:** Mobile Application for Managing a Gluten Free Diet

#### **Purpose of Project**

This study is being carried out in order to test the usability of an Android application developed as part of my final year project. This application aims to assist people who must follow a gluten free diet to find foods suitable for them to consume.

#### **Description of Methods and Measurements**

One questionnaire will be given to users after they have tested the application. The results of the questionnaire will be entered into a spreadsheet in order to be evaluated. Audio recordings will be taken during testing, as users will be encouraged to speak aloud while they interact with the application. Users will not be videoed while interacting with the application. The audio recordings will be reviewed, and any conclusions drawn from them will be presented in the project documentation. No audio recordings will be made available to anyone other than the researcher, nor will they be replayed in any public forum or during presentation of the research. Audio recordings will be used to gain a deeper understanding of the user's thought processes and experiences when using the application.

#### **Participants**

Participants will include:

- Classmates and students at Trinity College
- Friends and close relatives from home
- Family friends

Participants will be over 18 years of age. No person under the age of 18 will be allowed to participate. For application testing and the following questionnaire, I will be taking approximately 20 participants. This number was chosen to help broaden the potential demographics of respondents. Participants will be recruited in person or by telephone/text message. Participants from the above group will be invited to participate, with the expectation that at least 50 percent will decline. As such, I will ask between 40 and 45 people to test the application.

### **Debriefing Arrangements**

Each participant will be given an information sheet and a consent form regarding participation in this research. They will be invited to ask any questions that arise during their participation, and any questions will be answered to the best of my ability.

### **Ethical Considerations**

A potential conflict of interest could arise due to participants being known to me. This will be mitigated by encouraging the participants to answer as honestly as possible, and by anonymising the data after collection. By taking these steps, risks should be minimised.

In order to mitigate the risks of users becoming sick due to information included in the application, it will include a disclaimer that the most accurate information about a food can be found on its label, and that application information is only as up-to-date as it can be.

### **Relevant Legislation**

I have familiarised myself with the Data Protection Act and the College Good Research Practice guidelines.

## TRINITY COLLEGE DUBLIN

### INFORMATION SHEET FOR PROSPECTIVE PARTICIPANTS

---

This study is being carried out in order to test the usability of my final year project, an Android application that allows users to find out if a food is gluten free.

**I confirm that (where relevant):**

- I have familiarised myself with the Data Protection Act and the College Good Research Practice guidelines available at: [http://www.tcd.ie/info\\_compliance/data-protection/legislation/](http://www.tcd.ie/info_compliance/data-protection/legislation/)
- I will provide you with an information sheet that provides the main procedures.
- I will obtain informed consent for participation.
- I will ask you for your consent to be observed.
- Your participation is voluntary, and you may withdraw at any time without reason and without penalty.
- Audio recordings of your interaction with the application will be taken with your permission.
- No audio/videos/photographs will be taken without your permission.
- All data will be anonymised before use and will never be identified as belonging to you.
- You have the option of omitting questions you do not want to answer.
- You are asked to answer questions as honestly as possible.
- Your data will be treated with full confidentiality and that, if published, it will not be identified as yours.
- You will be debriefed at the end of your participation if requested.
- I have verified you are 18 years of age or older and competent to supply consent.
- If you, or anyone in your family has a history of epilepsy then you are proceeding at your own risk.
- In the unlikely event that illicit activity is reported to me during the study I will be obliged to report it to appropriate authorities.

- I will act in accordance with the information provided.

**Relevant procedures and expected duration of tasks:**

- 1. Testing the application:** You will be asked to test the application. This will take approximately 20 minutes.
- 2. Testing/Usability Questionnaire:** Following this, you will be asked to complete a questionnaire on the testing, comprising of 21 questions. This questionnaire will take approximately 20 minutes.

If you complete the testing of the application, it is expected that you will complete the testing questionnaire. You are free to not do so, but it will aid the researcher if you do.

**Risks/Benefits:** There are no risks in participating in this study for this project. Benefits include first user experience with the application and design. At the end of the project all participants will have the opportunity to view the final product.

**Provisions for debriefing before and after participation:** For debriefing I will use the SHARP methodology – an acronym for addressing each stage of participation.

**Before**

**S**et learning objectives.

**After**

**H**ow did it go?

**A**ddress concerns.

**R**evue learning points.

**P**lan ahead.

**Anonymity and preservation of data:** Anonymity of the participant and any third-parties will be upheld. This is of the utmost importance to the researcher. Anonymity will be preserved in analysis, publication, and presentation of results and findings. All data on the participant will be held on an encrypted drive, protected by a random password and recovery key that will be in my possession. All relevant data will be discarded of in full at the end of the academic year.

**Audio recordings:** No audio recordings will be made available to anyone other than the researcher, nor will any such recordings be replayed in public or in any public forum or presentation of the research. The reason that audio may be used is to better gain an understanding of how the application will be used by the user. You will be encouraged to talk aloud to yourself about what you are trying to do and how you are going about it.

**Participant selection:** You were selected through family, friends, and extended family and friends. You were eligible for selection due to your relationship with the researcher. You are asked to answer questions honestly, without consideration for any relationship between participant and researcher.

**Publication:** Individuals results may be aggregated anonymously, and research reported on aggregate results.

**TRINITY COLLEGE DUBLIN**  
**PARTICIPANT CONSENT FORM**

---

This study is being carried out in order to test the usability of my final year project, an Android application that allows users to find out if a food is gluten free. I am a final year student studying Computer Science and Business at Trinity College Dublin.

Relevant procedures and expected duration of tasks:

Testing the application: 20 minutes

Usability questionnaire: 20 minutes

**DECLARATION:**

- I am 18 years or older and I am competent to provide consent.
- I have read or had read to me a document providing information about this research and this consent form.
- I have had the opportunity to ask questions and all my questions have been answered to my satisfaction and I understand the description of the research that is being provided to me.
- I agree that my data is used for scientific purposes and I have no objection that my data is published in scientific publications in a way that does not reveal my identity.
- I understand that if I make illicit activities known, these will be reported to appropriate authorities.
- I understand that I may stop electronic recordings at any time, and that I may at any time, even subsequent to my participation have such recordings destroyed (except in situations such as above).
- I understand that, subject to constraints above, no recordings will be replayed in any public forums, or made available to any audience other than the current researchers/research team.

- I freely and voluntarily agree to be part of this research study, though without prejudice to my legal and ethical rights.
- I understand that I may refuse to answer any question and that I may withdraw at any time without penalty.
- I understand that my participation is fully anonymous and that no personal details about me will be recorded.
- I understand that if I or anyone in my family has a history of epilepsy then I am proceeding at my own risk.
- I have received a copy of this agreement.

**Participant Name:** \_\_\_\_\_

**Participant Signature:** \_\_\_\_\_

**Date:** \_\_\_\_\_

#### **STATEMENT OF INVESTIGATOR RESPONSIBILITY**

I have explained the nature and purpose of this research study, the procedures to be undertaken, and any risks that may be involved. I have offered to answer any questions and fully answered such questions. I believe that the participants understand my explanation and have freely given informed content.

Researcher's contact details:

**Name:** Ailbhe Redmond

**Phone:** 086 8135099

**Email:** [redmonai@tcd.ie](mailto:redmonai@tcd.ie)

**Investigator signature:** \_\_\_\_\_

**Date:** \_\_\_\_\_



**TRINITY COLLEGE DUBLIN**  
**USABILITY QUESTIONNAIRE**

---

Please answer the following questions honestly. Each question is optional. Feel free to omit a response to any question; however the researcher would be grateful if all questions were responded to.

Please do not name any third parties in any open text field of the questionnaire. Any such replies will be anonymised.

The researcher would like to thank you for your participation.

*Each question is optional. Feel free to omit a response to any question; however the researcher would be grateful if all questions were responded to.*

**Demographic Information**

---

Please tick the box that corresponds to your age

18-24	25-29	30-49	50-64	65+	Rather not say

Do you have any food allergies or intolerances?

Yes

No

If you answered yes above, please list them below:

---

---

---

### Technical Questions

---

I experienced crashing while using this application

Yes

No

I experienced lag while using this application

Yes

No

On a scale of 1 – 10 with 10 being the most positive, please rate the application.

---

Please give a reason for this ranking.

---

---

---

Would you recommend this application to friends or family following a gluten free diet?

Yes

No

When following a gluten free diet, would you be willing to pay for this app?

Yes

No

If you answered yes above, how much would you be willing to pay?

---

### Usability

---

**Questions based on the System Usability Scale** © Digital Equipment Corporation, 1986.

The numbering scale used is as follows:

- 1 – Strongly Disagree
- 2 – Disagree
- 3 – Neither Agree nor Disagree
- 4 – Agree
- 5 – Strongly Agree

I think that I would like to use this app frequently.

1      2      3      4      5

I found the app unnecessarily complex.

1      2      3      4      5

I thought the app was easy to use.

1      2      3      4      5

I think that I would need the support of a technical person to be able to use this app.

1      2      3      4      5

I found the various functions in this app were well integrated.

1      2      3      4      5

I thought there was too much inconsistency in this app.

1      2      3      4      5

I would imagine that most people would learn to use this app very quickly.

1      2      3      4      5

I found the app very cumbersome to use.

1      2      3      4      5

I felt very confident using the app.

1      2      3      4      5

I needed to learn a lot of things before I could use this app.

1      2      3      4      5

I liked the colour scheme.

1      2      3      4      5

108 | Page

## **Appendix E: Resources**

Android's Material Design guidelines

<https://developer.android.com/design/index.html>

Web tool for designing Android icons – used to generate the app's launcher icon

<https://jgilfelt.github.io/AndroidAssetStudio/>

Database of open source material design icons – icons used for the app's navigation bar

<https://material.io/icons/>

Homepage for all Firebase services

<https://firebase.google.com/>

Documentation for use and implementation of Firebase Database

[https://firebase.google.com/docs/reference/android/com/google/firebase/database  
/package-summary](https://firebase.google.com/docs/reference/android/com/google/firebase/database/package-summary)

Udacity course in implementing Firebase Realtime Database and Authentication

<https://classroom.udacity.com/courses/ud0352>

Firebase tutorial on implementing Cloud Functions

<https://www.youtube.com/watch?v=EvV9Vk9iOCQ>

Overview of TextRecognition API

<https://developers.google.com/vision/text-overview>

Sample code library for Google Vision Library

<https://github.com/googlesamples/android-vision/tree/master/visionSamples>

Youtube tutorial on implementing a BottomNavigationBar for Android

<https://www.youtube.com/watch?v=EbcdMxAlr54>

Forum for advice and discussion of coding problems

<https://stackoverflow.com/>

## Appendix F: Presentation Slides



Trinity College Dublin  
Coláiste na Tríonóide, Baile Átha Cliath  
The University of Dublin



### Gluten Free Food Scanner

An Android Application to Improve Access to Gluten Free Foods

Ailbhe Redmond  
BA (Mod.) Computer Science and Business

16<sup>th</sup> April 2018

### Overview

- Background & Research
- Requirements and Design
- Development
- Implementation Issues
- Evaluation
- Business Plan
- Further Work
- Conclusions
- Demonstration

Trinity College Dublin, The University of Dublin

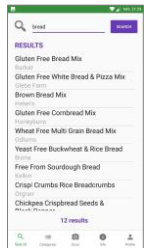
### Background & Research

- Motivation
- Existing Applications
- Research
- Technologies
  - Android
  - Firebase
  - Google Text Recognition API
- Data

Trinity College Dublin, The University of Dublin

### Requirements & Design

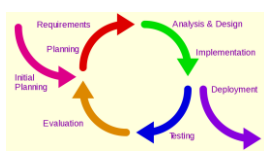
- Functional and Non-Functional Requirements
- Front-End Design
  - Prototypes
- Back-End Design



Trinity College Dublin, The University of Dublin

### Development

- Development Model
- Database Search
- Product Information
- User Accounts
- Ingredient Scanner



Trinity College Dublin, The University of Dublin

### Implementation Issues

- Asynchronous database
- Limitations of Text Recognition API

Trinity College Dublin, The University of Dublin

## Business Plan

Key Partners	Key Activities	Value Proposition	Customer Relationships	Customer Segments
	Key Resources		Channels	
Cost Structure		Revenue Streams		

Trinity College Dublin, The University of Dublin

7

## Evaluation

- Achieved functional and non-functional requirements
- Usability testing
  - Average 8.4/10 rating
  - SUS usability score of 87

Trinity College Dublin, The University of Dublin

8

## Further Work

- Expanded database
- Barcode scanning
- Offline capabilities
- Highlighting allergens in scanned text
- Automatic notifications on database update
- iOS implementation
- Other allergens

Trinity College Dublin, The University of Dublin

9

## Conclusions

- Lessons learned
  - New technologies learned
  - Real world development skills
- Reflection
  - Applied skills learned throughout college
  - Successful implementation of the application

Trinity College Dublin, The University of Dublin

10

## Demonstration

Trinity College Dublin  
Coláiste na Tríonóide, Baile Átha Cliath  
The University of Dublin

## Questions

Thank you for your attention

Trinity College Dublin  
Coláiste na Tríonóide, Baile Átha Cliath  
The University of Dublin