KITCHENETTE RECIPES

Final Report

Ailish Kavanagh
C00206130@itcarlow.ie
Supervisor: Paul Barry
Paul.barry@itcarlow.ie

Abstract

The purpose of this project is to create an Android application where the user can mark foods that they have in the house and based on these foods the app should suggest recipes they can cook. The application should present the user the option to add their own custom ingredients and recipes. If a user selects, they have cooked a certain recipe, the app should automatically update the contents list to remove the ingredients they used. If a user wants to make a recipe, but does not have all the necessary ingredients, the app should present the user with a "shopping list" of ingredients to buy.

Table of Contents

Table of Figures

Introduction

This final report will document the process of developing the Kitchenette application. The Kitchenette Recipes application is a food management system which allows the user to enter the food they have in their home, and receive suggested recipes based on what they can currently make. There are many likeminded web applications to the Kitchenette app as discussed in the Research Manual and Functional Specification, there are none that allow user customisation or operate on an android device versus as a web application. The Kitchenette Recipes application set out with the aim of improving the upon these existing systems and creating for the user a comprehensive smart food management system for their home.

The first section of this document will outline the description of the final submitted project. It will contain a brief overview of the technologies used to develop this project and a detailed description of each application screen and its functionality. This section will also discuss the difficulties faced in development, those that were solved and those that were not completely solved.

The next section of this document will explore how the final project compares to the original specification and design for the Kitchenette application. It will define what changed over the development process, including what features or functionalities nay have been added in or excluded. This section will also discuss changes to the overall layout of the application, compared to its design in the beginning.

This document will contain the learning outcomes of this project, both technical and personal. In this section a more descriptive analysis of the technologies used to develop the Kitchenette application will be explored. It will also examine the personal learning outcomes of completing this project including time management and meeting goals.

The final section of this document will present an overall review of the completed project, both the successes and failures of developing the Kitchenette application, what would be changed if starting from scratch, and a description of what should be done next going forward.

Section 2 - Project Description

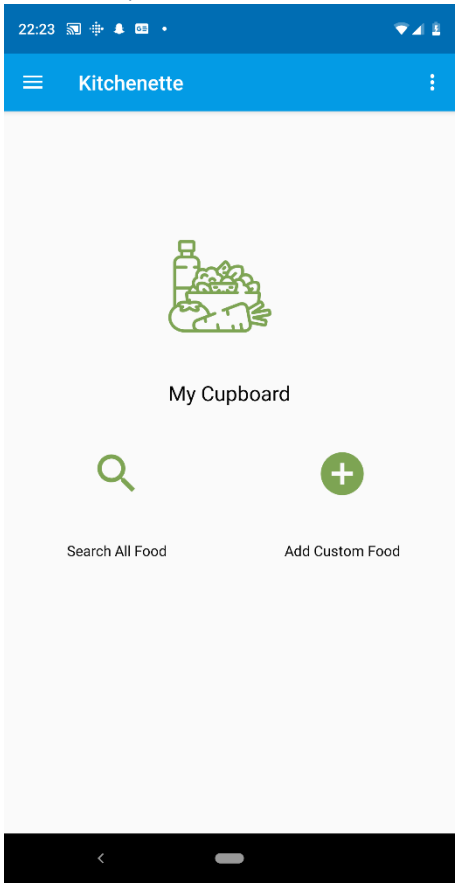
The Kitchenette application was built as an Android application using Kotlin and Android Studio as its main development environment. Kotlin is a relatively new language for Android development but is also officially supported by Google. Android Studio provides a lot of support for using Kotlin for Android application development. The graphical user interface (GUI) for this application was built using XML, a mark-up language like HTML designed to store and transport data. XML allows the user interface to be drawn with stylised layout or "blocks."

The backend of the project is an SQLite database system. SQLite is small, fast and light and ideal for app development. In most cases of SQLite, the database is created new with each install of the App. The SQLite database is preloaded into the Kotlin applications using an open source SQLite open asset helper library. The SQLite database is stored on the Google's Firebase platform. After initial install of the application, all database management is maintained within the SQLite text file itself. Before uploading to firebase, the SQLite database was managed using DB Browser for SQLite software for Windows machines. To upload to firebase a small file of JSON is installed integrated into the application. This file is pre-generated by Firebase completely and not included in this documentation as such.

Barcode scanning is done using the Barcode Detector API supported by Android development and Google. This API can recognise and read barcodes in a variety of formats including EAN, UPC and QR codes, and allows the barcode to be manipulated thereafter.

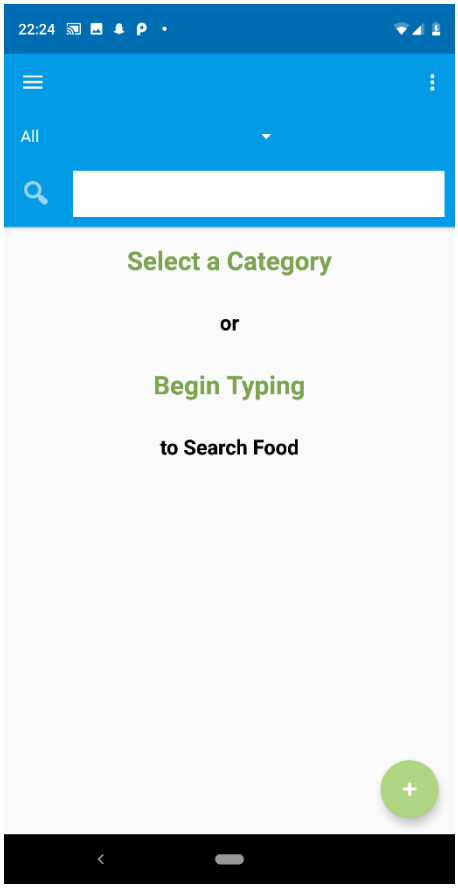
2.1 - Application Screenshots

2.1.1 - Cupboard and Food Functionality



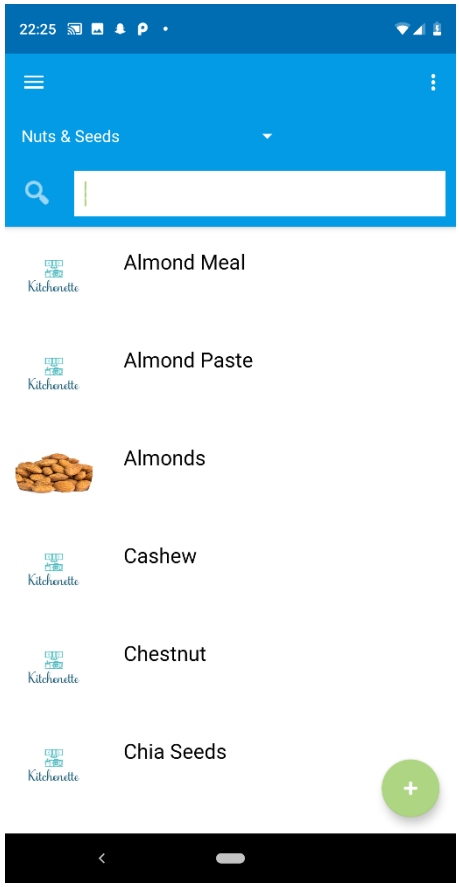
This the main landing page of the application and allows the user to navigate through the cupboard functionality. From here, the user can search through all food in the application's database, add a new custom food, or view the food in their own cupboard.

Figure 1 Main Cupboard Landing Page



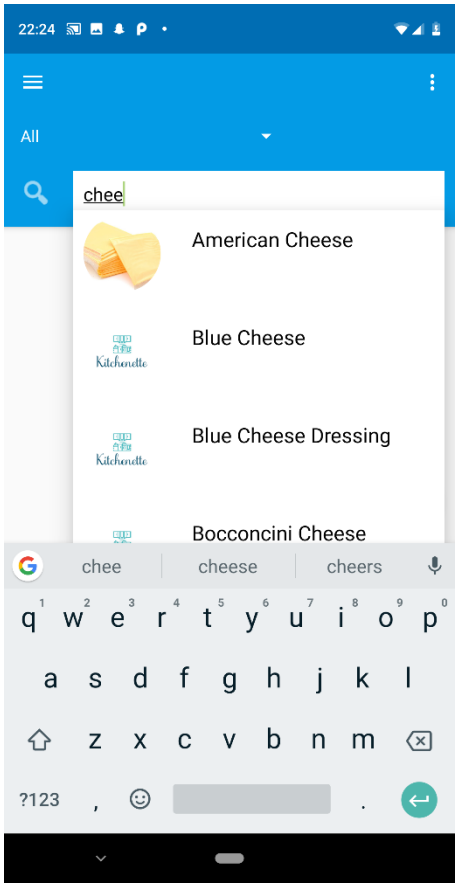
The Search All Food Landing page just displays to the user a simple message on how to navigate through food items in the database. From here the user can search all food in a specific category or begin typing in the search bar to find the food they are looking for. If the user cannot find the food they are looking for, they can add a new custom food by clicking the plus button in the bottom right corner.

Figure 2 Search all food landing page



This screen shows the user searching through a specific category, in this case "Nuts & Seeds". All food in this category will be displayed in alphabetical order. The user can still narrow their search at any stage using the search bar.

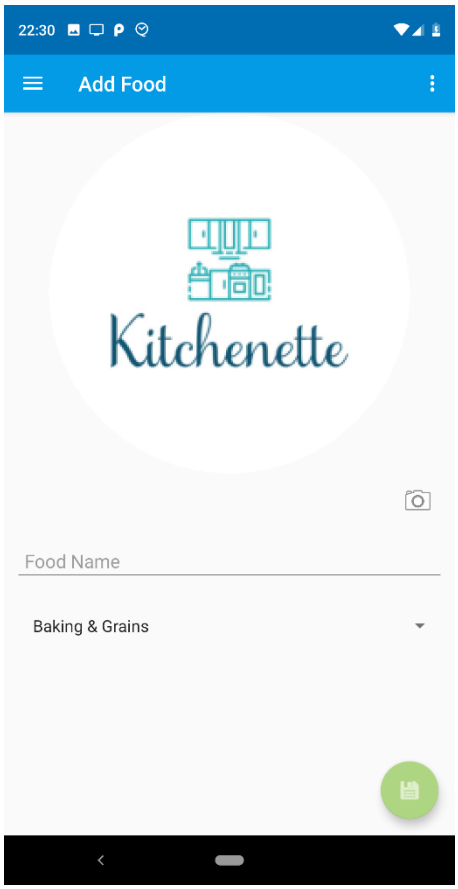
Figure 3 search food by category



When a user begins typing in the search bar a list is generated with items that contain the same or similar string as what the user has entered. The search bar is located on many screens in the application. Depending on which screen it appears on, the list it can generated predictions from differs. These generated lists that appear are as follows:

- Search All Food – all food in the database can be searched through
- Cupboard – only food currently in the users "Cupboard" will appear in the generated list
- Cookbook – The generated list will contain all recipes in the database

Figure 4 using the search bar



If the food the user is searching for is missing from the database, the user can create a new custom food item. Here the user may set the name of thee food, select its category, and upload a photo of the food. Once the user selects the save button in the bottom right corner, the custom food ill be created and added to the database for future use. All food items and recipes, custom or otherwise may be edited also. If a user selects edit on the View Food page, the custom food page is filled with the existing food details and the user may make changes as desired.

Figure 5 Add a new custom food item

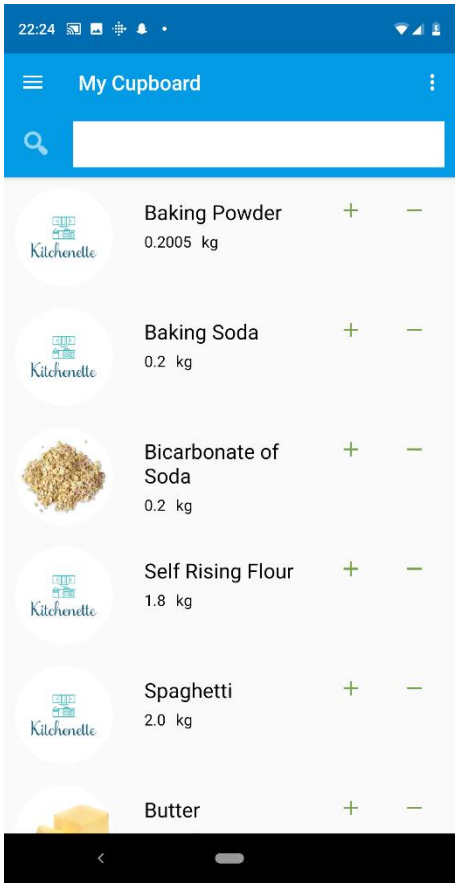


Figure 6 cupboard layout

The layout of the cupboard functionality is almost identical to that of the search food page. It differs in that, underneath each food the quantity the user currently has in stock is listed beneath so the user may ascertain how much they have at a glance. There are also two buttons to the right of each food item, for adding or removing a quantity from each item. When selected, a pop up appears, and the user may enter how much to add or remove from the database for that product. As mentioned earlier, the search bar on this page is populated with only items which are already in the cupboard.

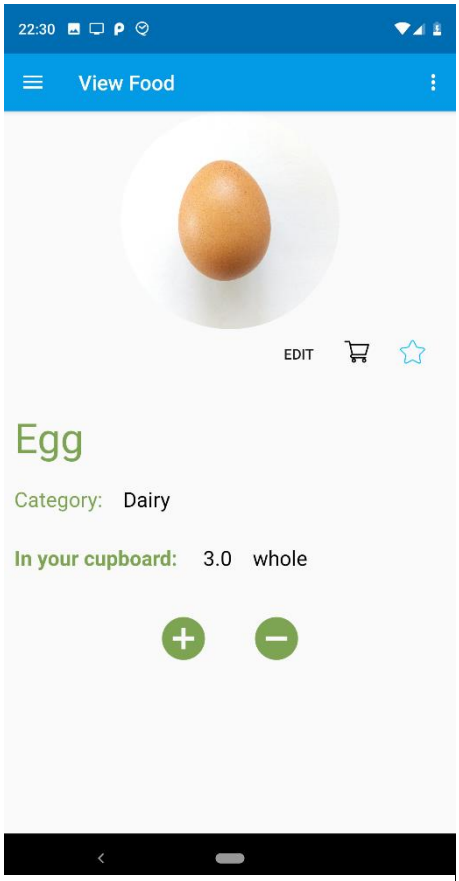


Figure 7 Food item display

By clicking on any food item on the cupboard page, or indeed on any of the search food pages, the user will be redirected to a display page for that item. On this page the user may view any details about the food item, as well as perform some functionality. The user may add or remove a quantity of this food product to the cupboard by selecting the plus or minus buttons, respectively. The user may add the food item to the shopping list or add the item to their favourites. In either case if the user has been added to the hopping list or favourites collection, their respective icons of the shopping cart or star will be blue when the page is loaded. If unselected, they will appear black again. If the user wishes, they can edit any details about the food items here.

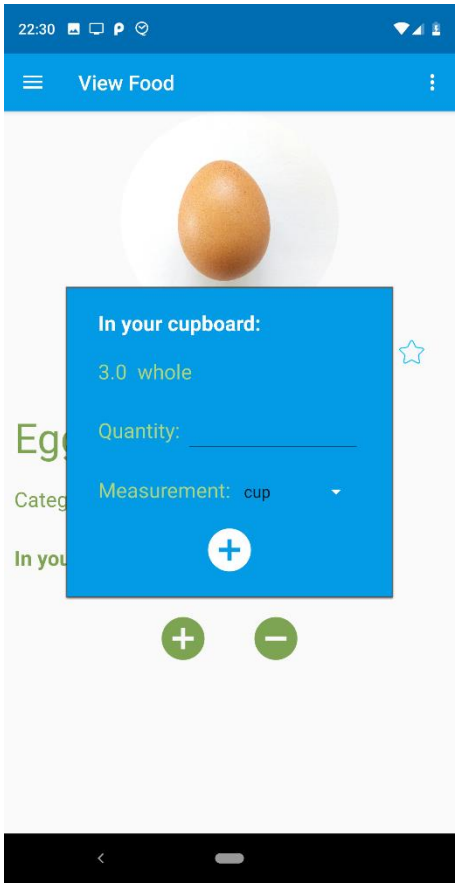


Figure 8 add/remove quantity popup

If the user selects the plus or minus button, a small popup appears. This popup will display how much of the item is in the cupboard at present, and he user may then add or remove quantities as desired. The display will differ only in that on the remove quantity screen, a minus is displayed instead of a plus. The user will enter the numeric quantity, sorted as a Double internally, and they will select the measurement they like also. The system will convert this quantity, based on the selected measurement, into grams inside the system. The amount will be added or subtracted from the current stored quantity and converted back into kg or litres for the users display. The user can select from:

- cup
- dessertspoon
- fl. Oz
- grams
- kg
- litres
- ml
- oz
- pint
- tbsp
- tsp or
- whole

2.1.2 – Navigation

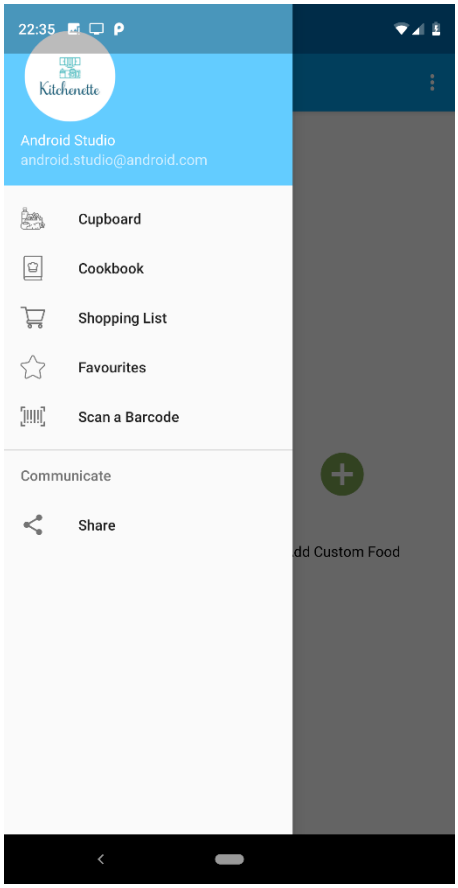


Figure 9 navigation drawer

Navigation throughout the app is done mainly through a slide out navigation drawer. This navigation drawer can be accessed from every screen in the application. It is accessed by using a sliding motion out from the left-hand side of the screen on a touch screen device, or by selecting the drawer button in the top left-hand corner of each screen.

2.1.3 – Recipes

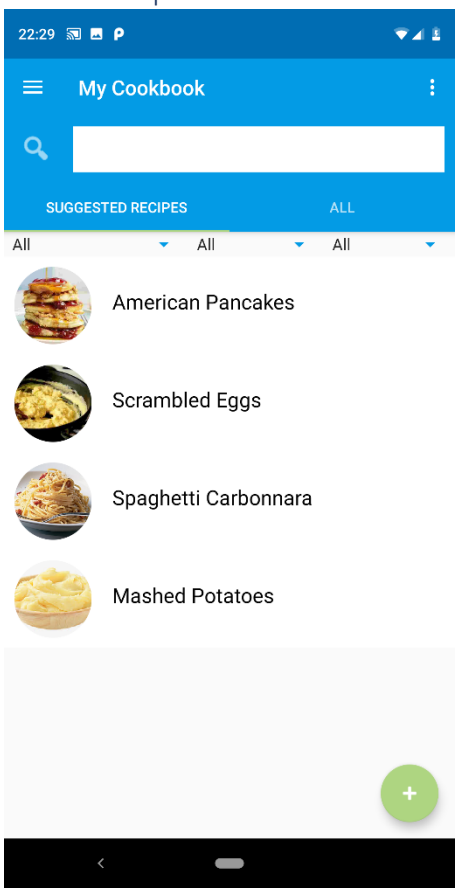


Figure 10 suggested recipes

When the user selects the cookbook activity from the navigation drawer, they land firstly on the suggested recipes screen. The recipes on this screen are those of which the user has the correct type and quantity of each ingredient the recipe requires in their cupboard at present. Essentially, this screen display recipes the user can make now.

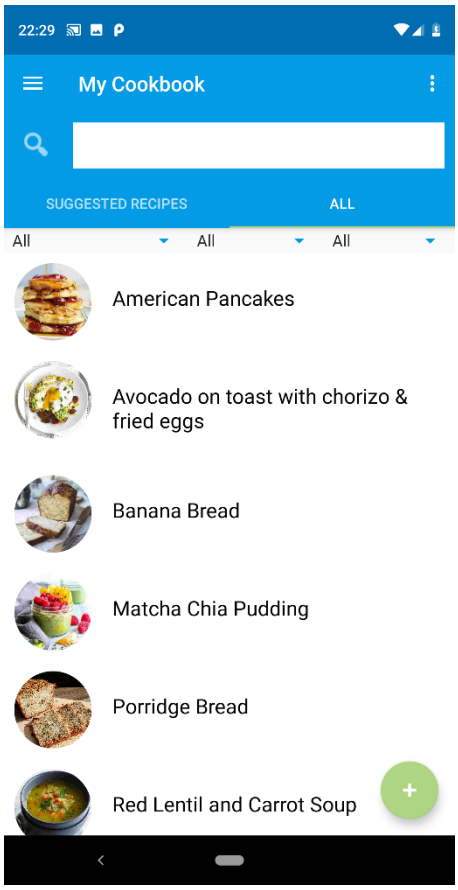


Figure 11 all recipes

The user can also swipe across to the next tab on the right of the cookbook display, and view or search through all recipes in the database. Recipes shown can be filtered by meal type (Breakfast, Dinner, Lunch, Desserts and Snacks, Other), Cuisines (such as Italian, Japanese or Spanish), or diet requirements (such as Vegan, Vegetarian, Gluten Free etc).If the user cannot find the specific recipe they want, the user can choose to create a new custom recipe by selecting the plus button in the bottom right corner of the display.

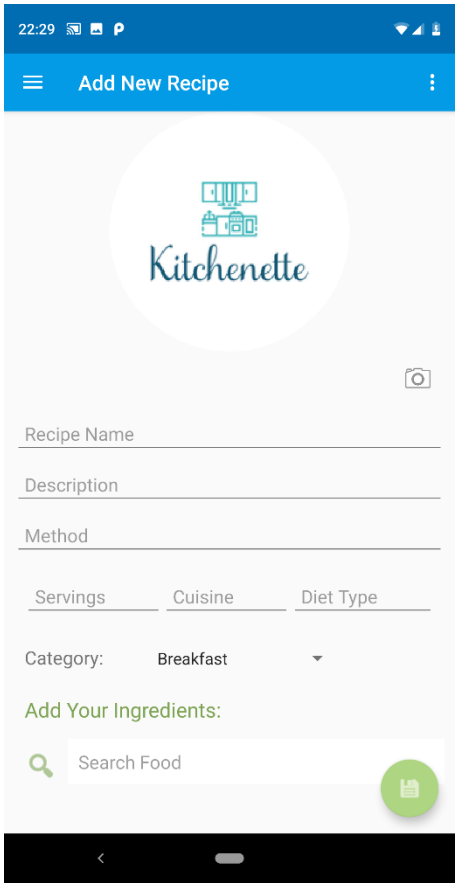


Figure 12 Add a new custom recipe

This screen presents the user with the option to create a new, custom recipe. Here the user may enter the details of a new recipe. The user will use the search bar at the bottom of the screen to add ingredients to the recipe. Once an ingredient is selected, the add quantity popup will appear so the user may enter how much of the ingredient is needed for the recipe. Once the user selects the save button in the bottom right corner, the view recipe page will load with the newly created recipe. The recipe will be saved in the database for all future use. Alternatively, the user may choose to select edit on an existing recipe. The same layout will be populated with the existing recipes details and the user may make whatever changes to the recipe that best suits themselves.

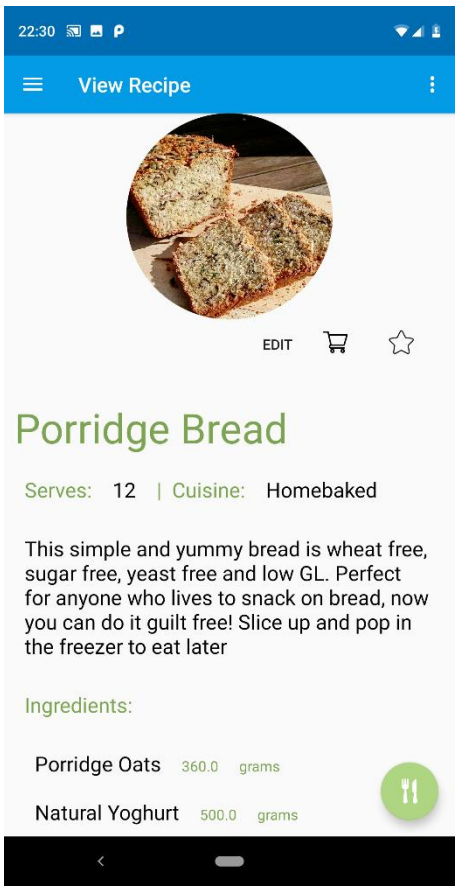


Figure 13 view recipe

When a user selects a recipe item on any page which it is displayed in list form, the view recipe page is loaded. Here the user can view all the details of a recipe, including its ingredients and methods. The display for this page can be quite long, depending on the chosen recipe, so not all of it may be seen at one time. The View Recipe page is a scrollable view page so the user may scroll to see the entire recipe. From this page, a user may choose to edit the recipe, favourite it, add all or some of its ingredients to the shopping list or, by selecting the button in the bottom right corner of the display, “Make This” recipe.

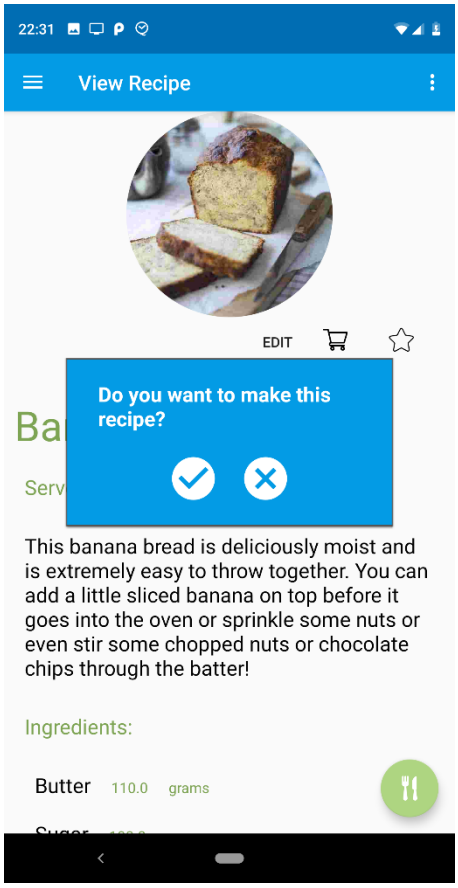


Figure 14 make this recipe

Once the user selects the "Make This" button in the bottom right corner of the screen, this popup display will be shown. The user can choose to make the recipe or cancel, making no changes. If the user selects to make the recipe, the application will remove the exact quantity of each ingredient from its respective amount in the cupboard. The cupboard will update, as well the suggested recipes page. The user will then be given an option to add any of the ingredients used to the shopping list, shown in the display below.

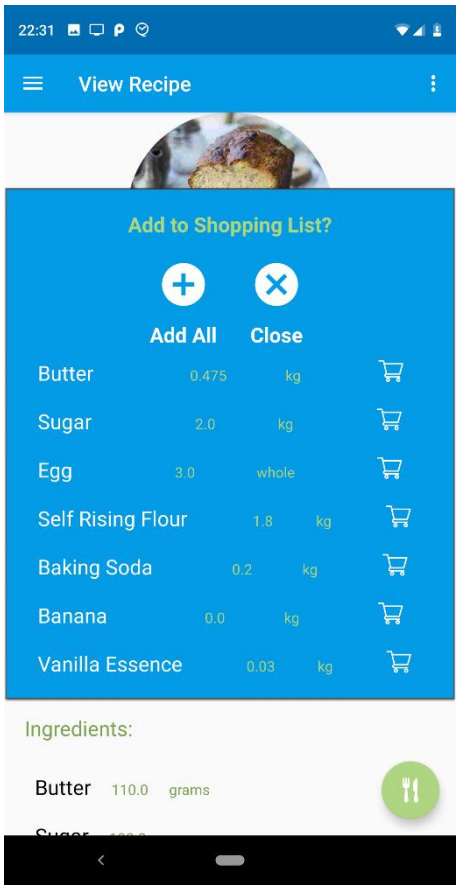
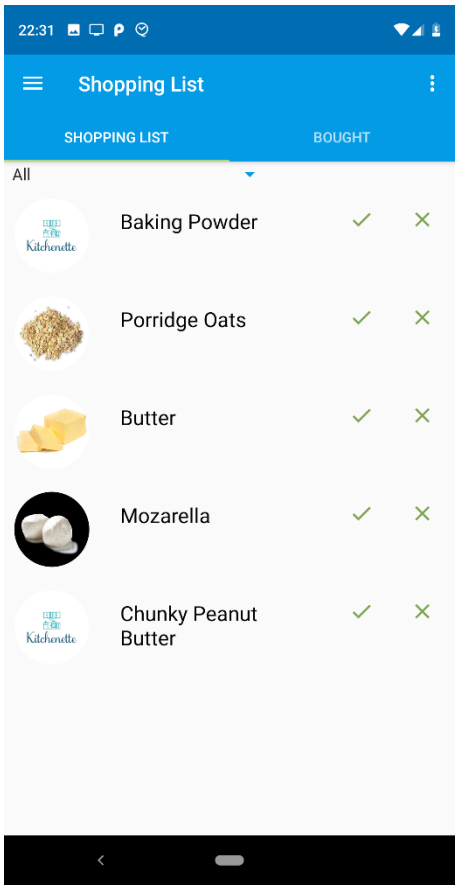


Figure 15 add ingredients to the shopping list

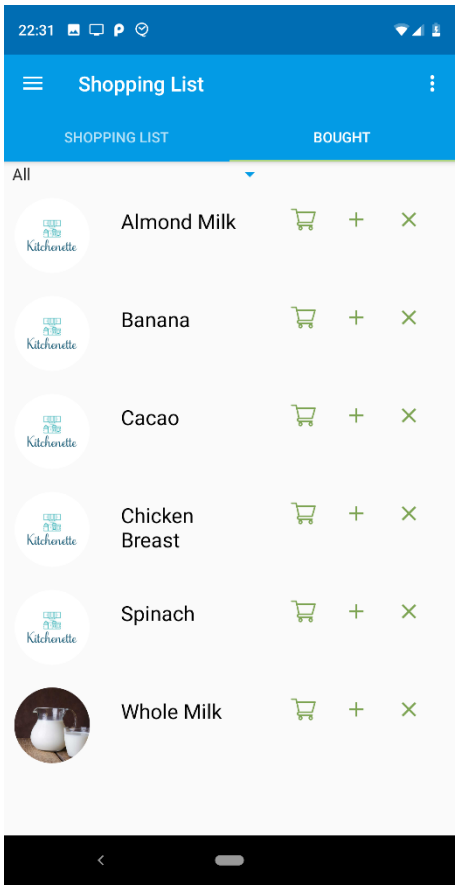
This popup will be shown when a user selects that they have made this recipe, or when the select the shopping cart button. For each ingredient in the list, the quantity of that ingredient contained in the cupboard is displayed. The user can select the shopping cart beside any singular ingredient and that item will be added to the shopping list and removed from the popup display. Alternatively, the user may choose to add all ingredients displayed to the shopping list or close the popup and add no ingredients to the shopping list. Any ingredients added singularly to the shopping list in this popup will not be affected by closing the popup. Only the remaining ingredients will not be added.

2.1.4 – Shopping List



On selecting the shopping list activity, the user is presented with a tabbed screen with the main "shopping list" and a "bought" screen. Any food items added to the shopping list will be displayed here. The user may choose to filter this screen by food categories as with the cupboard and search food lists. From this list, the user can tap the select button, which will add the item to the bought list, or remove the item from the shopping list altogether. Once an item is "bought" or removed, it disappears from the shopping list and the display is automatically updated to reflect this change.

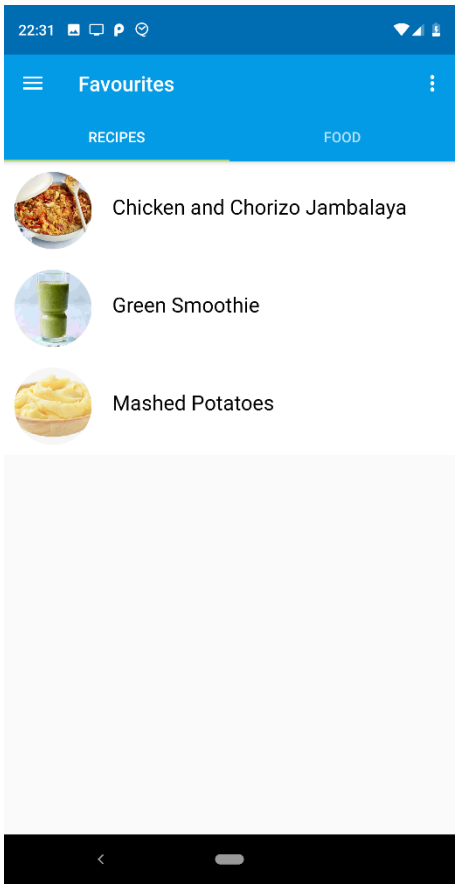
Figure 16 shopping lists



The idea behind the bought list is to contain items that the user has marked as bought, but which they have not updated the cupboard to reflect this change yet. From the bought screen the user can add the item back to the shopping list, remove it from the list altogether, or add it to the cupboard. By tapping the plus button, the add quantity popup will appear for the user to add the item and how much they bought to their cupboard. The cupboard will either update its existing quantity to include the new one, or if the product is not currently in the cupboard, it will add it with the new quantity. All affected screens will be updated automatically to reflect this change.

Figure 17 the bought list

2.1.5 – Favourites



The favourites screen is a simple scrollable list screen which displays all recipes and food that the user has marked as "favourites." There is very little functionality involved with these screens. They allow the user to quickly access food and recipes that they may frequently use, rather than searching for them each time. This presents the user a higher ease of access functionality.

Figure 18 favourite recipes

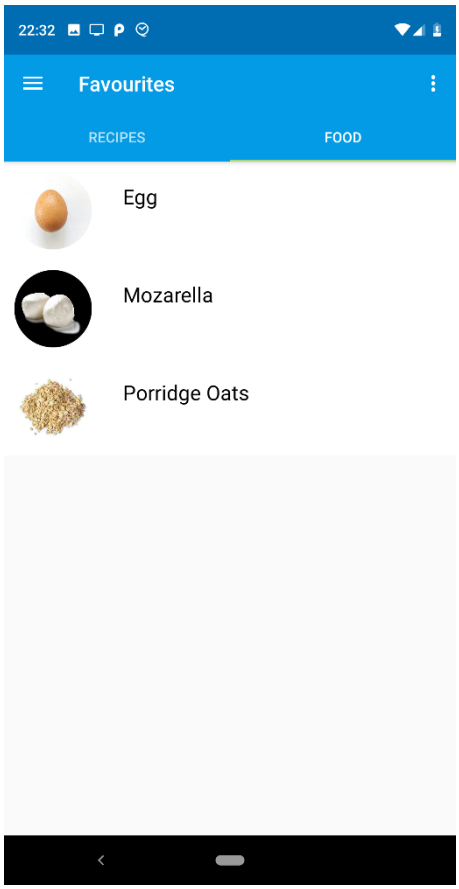
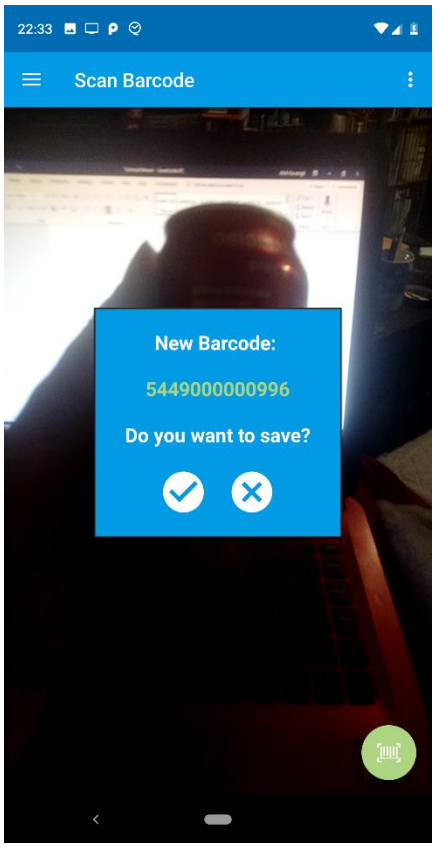


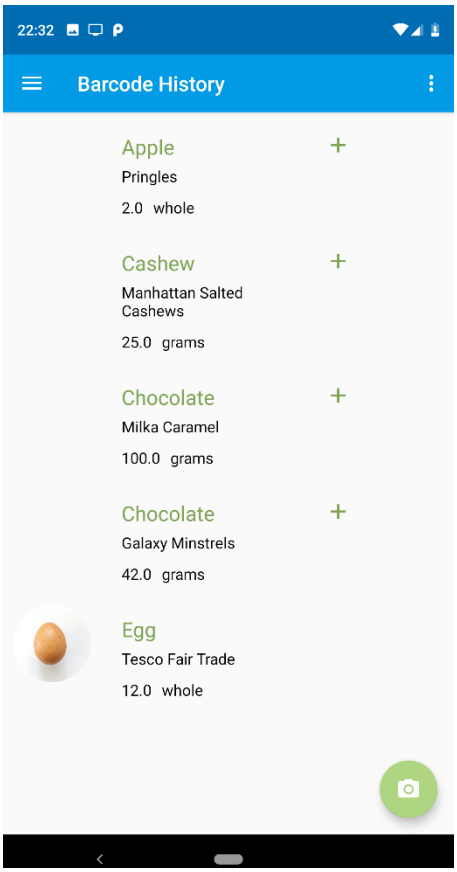
Figure 189 favourite food

2.1.6 – Barcode Scanning



The application allows the user the opportunity to scan in a barcode off a food product. If the barcode scanned is a not recognised, or has never been scanned by the application before, a popup message will allow the user to save the barcode or ignore it. If the user selects to save it, it will appear in the barcode's history page. The barcode will then be recognised in the future when scanned. The barcode history page can be accessed by selecting the button in the bottom right corner of the screen.

Figure 20 new barcode



The Barcodes history tab shows the user what barcodes they have scanned in the past. It is sorted by most recent scanned barcode. If the user has saved a new barcode but not attached it to a food product, its barcode identification number will appear in this list instead of the food name and description. From this screen, if a barcode is attached to a food item, the user may add it to the cupboard. The exact amount of the product will be added to the cupboard and the item will be removed, if present, from the shopping or bought list.

Figure 21 barcode history

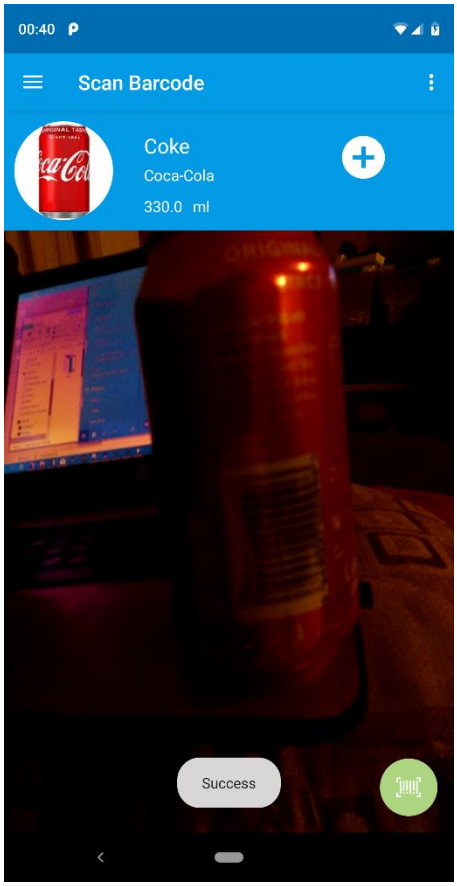


Figure 19 Scanning a recognised barcode

If the barcode is recognised or was saved and attached to a food item by the user, the application will automatically identify it upon detection. In this case a popup will appear at the top of the screen identifying the product. The user can select the add button on this screen to automatically update this item to the cupboard and remove it from the shopping or bough lists.

2.2 – Development Challenges

2.2.1 – Kotlin

Since Kotlin is a relatively new language, this comes with a myriad of benefits and problems. Learning any new language can seem a daunting challenge. Learning Kotlin is not merely learning another useful programming language, rather instead learning a language that is essentially new. Originally created in 2011 and integrated into Android Studio as an officially recognised language for Android development by Google in 2017, Kotlin is a young language, still in its early days.

For developing in many languages, the support of an online community can be invaluable. When difficulties arise in code during development, oftentimes the first thing most developers will do is turn to an online community for support and advice. Throughout the development of this project one of the greatest challenges was the scarcity of online support and documentation for Kotlin Android development. Kotlin's own documentation gave an impression of developing with Kotlin as a general language, not specifically with Android. Developing with Android can sometimes require methods and features to be implemented in a different way to how they would be in a desktop application.

Android development also contains a lot of features and protocols specific to, but necessary for, Android development. Despite Kotlin's recent rise in popularity, Java remains dominantly as the top language for Android development. The effect of this was that most online support forums, communities and documentation focused heavily on implementing features primarily in Java. Few of these, if any, offered support for doing the same thing in Kotlin. This makes learning Android development from scratch in conjunction with Kotlin a difficult process. Devising solutions in Kotlin relied heavily, at times, on trial and error methods of building and testing code, as the Java code would need to be translated to a working Kotlin version.

2.2.2 – Barcode Scanning

One main feature of the app was to give the users the ability to scan a barcode off a product they had just bought, identify the product or food item, and add the product and its quantity to the cupboard. This feature would allow a greater ease of use for any users of the Kitchenette application, in comparison to asking a user to manually enter this data for every product. There are many barcode detectors and scanners online. Very few of these offer the ability to identify the product by its barcode, but merely save the barcode identification number. There were several API's online that could identify the barcode, but at a high cost. One such scanner did have an accompanying, free to use website, barcodelookup.com.

A python script can be used to access the database with the barcode pulled from the scanner implemented into Android. The script would have the website search for that barcode and return the result. The database for the Kitchenette application was initially designed in the Design Manual. In general, this design was adhered to throughout the entire development process. This led to the main problem with implementing a barcode scanner, barcodes recognise a specific product rather than the general food item. For example, recognising Brennan's Bread instead of simply "sliced bread" or "Babybel" instead of "soft cheese." It is unknown in this case whether the use of the paid API version could have alleviated this problem. Restructuring the database slightly may have also alleviated this issue. Unfortunately, this issue was discovered late in development and to change the database structure may have caused issues in other parts of the application.

It was always intended that the user could save a new, unrecognised barcode into the database and attach it to a food item for future use. Due to the issue discussed above, this feature became the main functionality for identifying and updating barcodes.

2.2.3 – Hardware Restrictions

The laptop used to develop this project is several years old, with an intel core i3 processor, 8gb RAM and a 2tb hard drive. Though these are still decent specifications to have, the laptop struggled to run any virtual machine tasks. This includes the Android Virtual Machine (AVM) in Android Studio. Whenever the AVM was run on the laptop, it caused the laptop to stop responding, and in some cases shut down. This was an unfortunate occurrence despite any workarounds that were implemented, this issue proceeded.

All building and debugging for the Kitchenette application had to be done through an Android phone every time. This did cause issues with slowing down the debugging process. At many stages through development the Kitchenette application crashed or had an error. This not only caused the Kitchenette application to run slow, but also caused the phone to have several problems later.

2.2.4 – Database Entry

The database structure for this project was laid out in the Design Manual and followed closely throughout development of the Kitchenette application. The database structure and its handlers were the first part of the code created and the rest of the application was built around this. Initially, the database was created on the first loading of the app using SQL Open Helper libraries, as is normal for SQLite databases in Android applications. Later, the SQLite database was upgraded using SQLite Asset Helper open source library. This allowed a pre-loaded database to be download with the application on install.

Working with a database that was static throughout the development despite it being constantly pushed and pulled in opposite directions, did present some difficulties. The database structural design allowed the food and recipes to be stored in a manner that was idea. This frigidly unchanging database structure, though compatible with the Kitchenette application, makes it difficult to combine with other API’s or applications.

To enter and upload the data to the database, a lot of data pre-processing was done. Food databases had to be sourced and cleaned firstly. It was found that most online food databases that were tried did not easily fit the structural integrity of the database. Once a suitable dataset of food was found, each row of data had to be manually added to the database in its correct format. Though this did create a substantial amount of extra work in building the project, it ensured the data was all entered in a manner that was compatible with all aspects of the application.

Section 3 - Conformance to Specification and Design

Feature wise, the Kitchenette application remained true to its design in many aspects. All the main features that were to be added into the application, were successfully added. These features include:

- Cupboard,
  - adding food in the home to the cupboard
- Cookbook,
  - viewing suggested recipes based on what you can currently make
- Favourites,
  - add a recipe to “favourites” list for easy finding later
- Shopping List
  - Add or remove food items to the shopping list.
- Barcode Scanning
  - can a barcode to add that product to the cupboard and remove it from any other lists.

There were several small changes made to specification and design of the project. As mentioned in Section 2.2.2 “Barcode Scanning”, the original design for the barcode scanning functionality had to be altered slightly. Originally scanning a barcode should have had one of two actions, scan a new, unrecognised barcode and be given the option to be saved and attached to a food item later, or identify and pull the barcodes from a remote server using an API. As discussed in Section 2.2.2 the second option could not be implemented. This activity deviated from its original functionality in this slight way yet still maintains its core purpose.

The graphical user interface (GUI) as designed in early documents has changed quite a lot over the course of this project. The original GUI design was just pencil sketches in the Functional Specification. When translated into XML code, that fitted with the backend Kotlin code, some GUI elements were lost, and some were drastically altered.

The database design created in the Design Manual did not alter throughout development. The database design became the core of the application, and any functionalities were built around it. The original database design did allow for the possibility of nutritional information to be added alongside other food or recipe details but was, unfortunately, left out in development. This was not a core feature though would have made a nice addition.

Every other aspect of functionality described in the original Design Manual and Functional Specification for the Cupboard, Cookbook, Favourites and Shopping List sections was implemented successfully. The core design of the application has remained true to its purpose, despite its appearance changing on the way.

Section 4 - Learning Outcomes

4.1 - Technical Learning

In the completion of this project, several new technologies and programming languages had to be learned as part of the project specification. Choosing to complete this project in unfamiliar technologies was an intentional choice to greater increase the learning outcomes of this project. The new technologies used to build the Kitchenette are discussed in further detail below.

4.1.1 – Kotlin

As mentioned previously, Kotlin is a relatively new language with a minimal amount of support online. Only one YouTube channel online predominantly focused on Kotlin and Android development in conjunction. This channel focused on the basics of Android development in Kotlin, which was a boosting starting point. It was difficult to find much other support online, whether in online communities or tutorials. As explored in Section 2.2.1, this did present many difficulties during development, especially in the beginning stages of building the Kitchenette application.

Indeed, many of the early stages of development may have progressed quicker if a more established and familiar language had been chosen such as Java. A lot of learning the Kotlin language was done by familiarising with the basic structure of Kotlin development, and then reading code in Java and translating it to Kotlin code. Translating from one language to another provided practice in an invaluable skill, which can be applied across multiple languages.

Once familiar with Kotlin and in in a position to translate from Java to Kotlin code, working with Kotlin became an enjoyable experience. It became evident that Kotlin was an ideal language for Android development. What may have taken 50 lines of Java code took only a fraction of that when written in Kotlin. Kotlin's ability to deal with null pointer exceptions without crashing the code became a boon during development and testing pages. Though the beginning stages of development were slowed down due to the difficulty of learning Kotlin, the later stages of development were certainly sped up and benefited from the use of Kotlin as the primary development language.

4.1.2 – SQLite

SQLite is similar in man ways to other relational database management systems such as MySQL. Its syntax is largely similar in performing most operations, like inserting, deleting, updating or searching the database. SQLite differs in that in order to make it more lightweight, it minimises the amount of data types it accepts. SQLite recognises only five data types, Integer, Text, Real Null or BLOB (Binary Large Object). It does not account for datetime variables, image files or Booleans. A datetime variable must be stored formatted into either an Integer or Text variable. Boolean datatypes must be converted into an Integer field with a 0 or 1 value. Image files must be converted into binary objects and stored in BLOB fields. All these datatypes were needed in the database for the Kitchenette application. There is extensive documentation online for SQLite so learning how to convert between datatypes was not an overly difficult task, but more another aspect of learning to use SQLite in the place of a more traditional relational database management system query language. Android's SQLite Open Helper and SQLite Asset Helper libraries both contain extensive support for SQLite. They help with integrating SQLite into an Android application and converting between data types for use in the application without much difficulty.

4.1.3 – Firebase

Using the Firebase platform can be difficult to learn in the beginning. Firebase contains an extensive range of products to meet many needs of different applications. The difficulty in learning Firebase rested mainly on learning about their various products and deducing which ones were applicable to the android application being built. For the Kitchenette application, the only Firebase product integrated into it for this project was its ability to store the product on the Google Cloud platform and store the pre-loaded SQLite database with the application for download together. Google Developer provided a lot of clear documentation and instructions on how to integrate Firebase into the application and to run it through Android Studio. On reflection, the Kitchenette application would have benefited greater if Firebase had been integrated into it earlier in its development. Familiarising with the Firebase platform and its products and learning how to do the initial install of Firebase will make the process much quicker for any future android developments.

4.1.4 – Android studio and Android Development

Up until beginning this project, Android Studio and Android development in general was relatively unlearned. The use of Android Studio had been covered briefly in third year Software Engineering classes, but not since. Beginning the development of the Kitchenette application meant learning to use the Android Studio development environment and its various plugins almost entirely from scratch. Developing the Kitchenette application could have been completed in other development environments, but Android Studio remains the leading platform for Android development at present. Learning Android development, its features and requirements, was a new experience. Doing so through Android Studio allowed for a smoother process and is an invaluable ability going forward.

4.2 - Personal Learning

This section will explore personal learning outcomes obtained during the development of the Kitchenette application. The author has deemed to write the following section in first person.

4.2.1 – Time Management and Organisational Skills

Time management became a skill that is invaluable both in working and personal life. For my college career, I have had to work part time will pursuing my studies. Though I have done both to a successful pace thus far, learning to balance the development of my fourth-year project, the Kitchenette application, with other college studies on top of having to work a minimum of 15 hours a week in part time employment was a challenging experience. Learning to optimise my time became a necessity more so than ever.

Essentially the weekends were, unfortunately, lost to me for college work and the development of the Kitchenette application while I was in work. Due to working late into the evening I had little time after work to work on projects. The small amounts of time I had in the evenings I used for attaining "learning goals." I would use this time to try to learn more about the technologies I needed to develop the Kitchenette application and deriving rough pseudocode estimations of what I needed going forward. In many cases, this time in the evening alone was not enough and I would pursue the same goals while on lunch breaks in work. This then meant balancing the time I had during the college week more efficiently to give enough time for development and college work.

This was a difficult task from start to finish. To optimise my time, I set out both daily and weekly goals. I broke down all tasks into the simplest of forms for daily goals to quickly fly through these lists and set more abstract weekly goals that were an accumulation of these daily tasks. This allowed me to keep track of my time and what it was being spent on. Once task schedules were laid out, they were categorised into order of expected difficulty (low, medium, hard). This balanced the tasks so that I could complete several "low" difficulty tasks as quickly as possible and did not try to attempt a large amount of "high" difficulty tasks on the same day. This also ensured that I was not spending too long or too short of a time on any one task. Of course, completely predicting tasks and how long they will take to complete is hard to do, these lists were in constant flux at the beginning. I found towards the end of the year I could crate my daily and weekly goals in a shorted amount of time, and they would be more accurately divided up.

Organising my time like this was invaluable and kept me on track for keeping up with the workload for developing the kitchenette application. Without organising and managing my time in such a manner I do not believe I would have been able to keep myself on track as much. I believe I did the best I could with the time I had available to me. And learning these skills is invaluable to productivity in work and personal life. Developing the Kitchenette application, for me, was a passion project start to finish that I enjoyed working on. The largest disappointment is still, of course, that I do believe if I had the time available to me on the weekends, I could have produced a much better version of the Kitchenette application.

Project Review

5.1 – Project Success and Achievements

Overall the Kitchenette project was successful when compare to the original goals, design and specification. All the main functionality was implemented, and the application is at a stage where it is usable. Though the application was not uploaded to the Google Play store for the end of this project, it requires no other functionality for it to be uploaded and made available to users.

As it stands, the application completes its original intent of enabling a user to add food to their cupboard and view suggested recipes that they can make now based on what they currently have in their home. The application accounts for the quantities of food added to the cupboard, and not merely the ingredients themselves when suggesting a recipe. After making a recipe, the application will notify the user if they are running low on a product and ask them if it should be added to the shopping list. The user can add food to a shopping list themselves and scan a barcode to add it back to the cupboard once purchased. The application also allows the user the opportunity to add custom food items and recipes for future use in the application. A user can also edit any existing recipes to allow full customisation.

The application is stored locally and managed with an SQLite database that runs on the Android device itself, making it lightweight. This means whatever changes a user makes to their own recipes and ingredients, it will not affect the changes made by other users, everything runs separately on a user to user basis.

Kotlin was a successful choice overall in developing this application. Kotlin allowed integration with SQLite open source libraries that could send a pre-loaded database to the user. Kotlin's syntax made it more concise and easier to maintain and manage, particularly at later stages of development, then its counterpart in Java would have been.

5.2 – Project Failures

Most of the development for the Kitchenette application was successful. The main failure in the development of the project was the inability to get the application and its database structure to work with barcode identification API's. If starting again, or while going forward, the database would need to be re structured to account for this. The API identifies the barcode by product rather than simply by food type. To integrate smoothly between the database and the API there would need to be some record of products associated with food type. This would require some further research to lay the database out in an optimum way.

The Kitchenette application would have benefited from having a greater presence on the Firebase platform. More cloud features would have improved the overall usability of the application including the ability to create an account and save the users data. This could have benefitted the applications barcode scanning problem also. With a cloud-based database for the barcodes, anytime a user saves a new barcode to the database t could be verified and saved to the cloud. This would essentially create a crowd sourcing aspect to identifying barcodes, and the application would improve this way over time.

The Kitchenette application would improve on having its database restructured slightly regardless of the barcode scanning. At present the application shows no details for food items other than their name and category. If updated to include nutritional value and breakdown for food and recipes this would increase the appeal of the application and its overall usability.

The application was also created originally without its pre-loaded database. The data at a later stage of development and the beginning stages was all custom added food and recipes. This shows in that on review of the application, many food items do not have an image icon attached to them. If starting again, the first thing that should be done is a fully functional and filled database.

5.3 – Future Development

There many features that can be added to the Kitchenette application that could be added in future development. As mentioned above, the application would benefit from having nutritional value for food items and recipes. This would take a bit of work to develop a class which could take the nutritional content of each ingredient and calculate the overall nutritional breakdown of the recipe. Coupling this feature with the ability for the user to track recipes they have made, and a meal breakdown chart, would allow a user to better watch their own personal intake of food. It would encourage the user to update their application more frequently if it had greater personal use for them.

The app would also benefit from having some community features. If multiple users living in the same home could share the Cupboard, recipes, and shopping list, this would make it easier for families to make use of the application. The app should be able to update across all user devices in a family home if a user has added to or removed from the cupboard. The app should be able to update to a family shopping list or keep a private one, depending on the users need. The accounts should still be separate so the user may choose what to share and what to keep private.

Acknowledgements

Many thanks to my supervisor Paul Barry, whose advice, expertise and guidance was invaluable on the development of this project.

Thank you to Vignesh from Code Android, his Kotlin tutorials on YouTube are the main reason any code in this project was completed. I'll eave the link to his channel just below.
https://www.youtube.com/channel/UCMvagHKkUlt3t_E4KxwQXXQ/featured

Plagiarism Declaration

Declaration

* I declare that all material in this submission e.g. thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.

* I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside.

* I have provided a complete bibliography of all works and sources used in the preparation of this submission.

* I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: Ailish Kavanagh

Student Number(s): C00206130

Signature(s): _____

Date: _____