# ShoppingPal Technical Specification

## CA326

Ben Kelly & Kyrylo Khaletskyy

## Table of Contents

# 1 Introduction

## 1.1 Overview

ShoppingPal is an Android application which helps users with restrictions to buy groceries safely from a local store. Whether the user has allergies or intolerances to certain ingredients found in food items, pharmaceutical products or toiletries. ShoppingPal also aims to help athletes with strict dietary and pharmaceutical restrictions buy items in a store without the worry of failing a drug test. Users will register with an email and password and proceed to set up their profile. A user's profile contains his/her restrictions at a given time. The app will provide a user with a wide variety of common food allergens, as well as an accurate approved WADA (World Anti Doping Agency) specified banned substances taken directly from a list provided by WADA. Users who have specific allergies or restrictions not found in the common list provided have the option to set up their own profiles containing ingredients manually selected/inputted by the user. Upon successful registry and profile set up, the user will be given the option to proceed to scan a barcode. Once the barcode is scanned it will fetch that particular item in the dataset and compare the list of ingredients found in that particular item to the list of ingredients found in the user's profile. The app will tell the user if they can buy this particular product and give the list of ingredients in the product as well as the ingredients that match the user's profile if the item contains restricted ingredients. The app will notify the user through a "Traffic Light" (Red & Green) system whether the user can consume the product or not. ShoppingPal is mostly aimed at people with allergies, intolerances and general restrictions but can be used by any user who wishes to avoid certain ingredients from their diet/use as well as people who have trouble reading labels in a store. It is estimated by the WAO (World Allergy Organisation) that allergy prevalence of the whole population by country ranges between 10-40% which in Ireland would total to at least 500,000 people. It is difficult to estimate the number of professional athletes in the world but according to WADA there have been 303,369 samples of drugs tests taken in 2015. ShoppingPal can therefore be used by a very large group of people.

## 1.2 Glossary

- **WADA** - World Anti Doping Agency
- **WAO** - World Allergy Organisation
- **XML** - Extensible Markup Language
- **API** - Application Programming Interface
- **JSON** - Javascript Object Notation, Data interchange format
- **Android Mobile SDK** - Software Development Kit for Android Application
- **HTTP** - Hypertext Transfer Protocol
- **EAN-13** - A type of barcode used by many stores.
- **ZXing** - A library used to scan a barcode and retrieve the corresponding numbers displayed under the barcode of an item.

# 2 Development

Shoppingpal has remained very close to the functional spec throughout its development, there haven't been any major changes in terms of functionality. All functionality mentioned in the functional spec can be seen in the final implementation. The development of ShoppingPal application was done in Android Studio, this is a free application provided by Google. It uses the Android Studio Mobile SDK for the application logic and XML for the layout and design of all the pages. The server used for this project is Google Firebase, it provides a plethora of features such as real-time database, account analytics, ability to expand and mainly it is free and reliable as it is hosted by Google. ShoppingPal is designed for any android device with Android 5.0.1 Lollipop or higher (API level 21+) we believe that this API was the best compromise in terms of modern SDK (features and performance) and for the app to work on the widest range of devices. ShoppingPal can adapt to screens of various sizes (as shown in testing) but performs best on phone displays as we believe this is what users will be bringing with them to stores. All screen images and graphics used in the application (XML) were created in Adobe Photoshop CS6 by us. During the lifetime of this project we have performed the following tests:

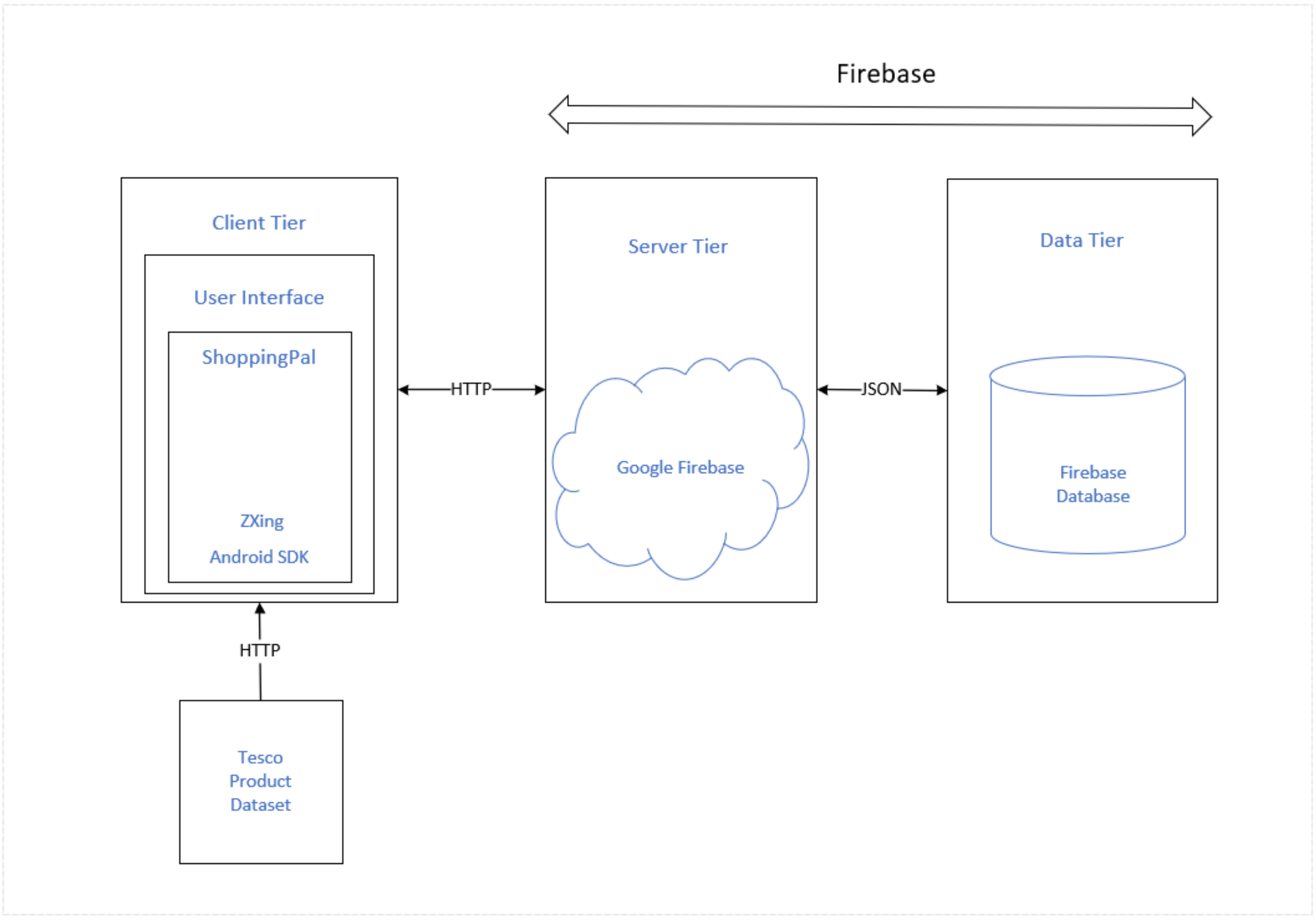
- Unit testing,
- Interface testing,
- User testing,
- Heuristic Analysis,
- Android studio logs analysis,
- Shneiderman's 8 golden rules.

Information on testing throughout the development of this app can be found on the Blog and the Testing document/files.

## 3 System Architecture

The System Architecture is made up of 3 main parts:

- Client
- Database
- Product Data



### 3.1 Client

ShoppingPal consists of 6 classes (Activities) which correspond to 6 user pages. The layout for each page (Activity) is defined by its corresponding "layout.xml" file, here the layout of each visible object, color and layout is defined using XML. The build.gradle is the file that contains all the library and external functions imported to make the project function. Here the firebase and ZXing library dependencies are added, the ZXing library is what scans the barcode and returns the corresponding string which contains a 13 digit barcode in the EAN-13 format. Along with the build.gradle file the system contains a vast amount of class, layout build and properties files which form the ShoppingPal system. In the drawables folder there are a number of icons, images and backgrounds all designed by us which bring the application to life and make it more attractive. The built-in profiles are stored in the strings.xml file and can be easily modified, and new profiles can be easily added.

## 3.2 Database

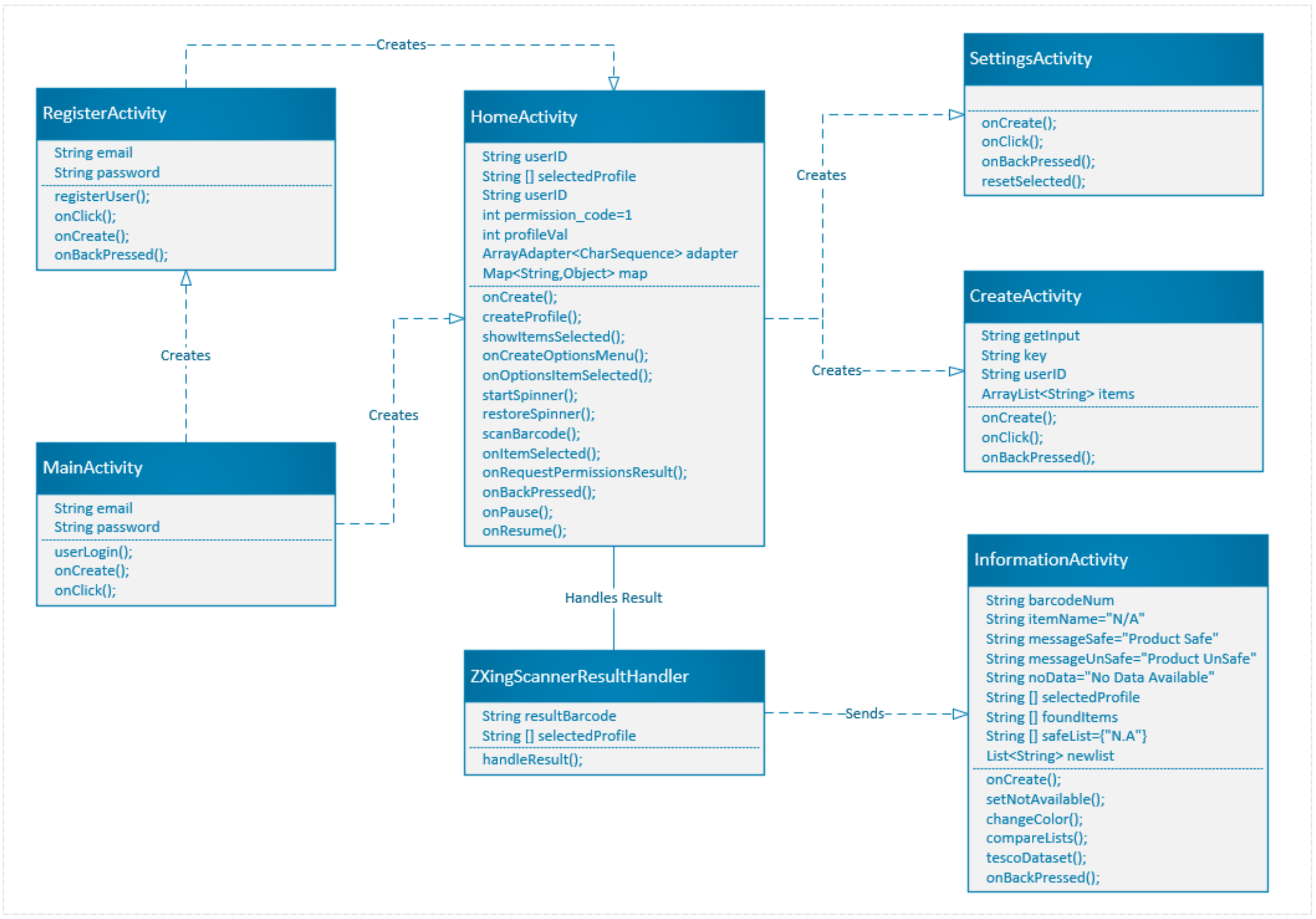
Our database of choice is Firebase, the client stores and accesses data on the NoSQL cloud Database. The client connects to the server through HTTP and linked through the build.gradle file along with the google-services.json file found in the /code of this repository. Data is Stored as JSON and synchronized in realtime to every connected client. Here all account information is stored such as email and password. In the Database the data is formed as a JSON tree, more information on the structure can be seen at 4.6.

## 3.3 Product Data

Without product data, our application is useless since you need to know the ingredients of a product to compare them to your profile. Throughout the lifetime of this project we have had difficulties with providers (outlined in problems and resolutions) but in the end we have found a viable solution. We are using the Tesco Dataset to retrieve product data through HTTP Get Request. We implemented HttpUrlConnection rather than using their API. A HTTP Get Request is sent along with the barcode number (String) and a subscription key provided by Tesco. A JSON object returned containing all the product data, and now the ingredients are parsed using java regex (the reason why we have to use regex rather than extracting "ingredients" child outlined in problems/resolutions also).

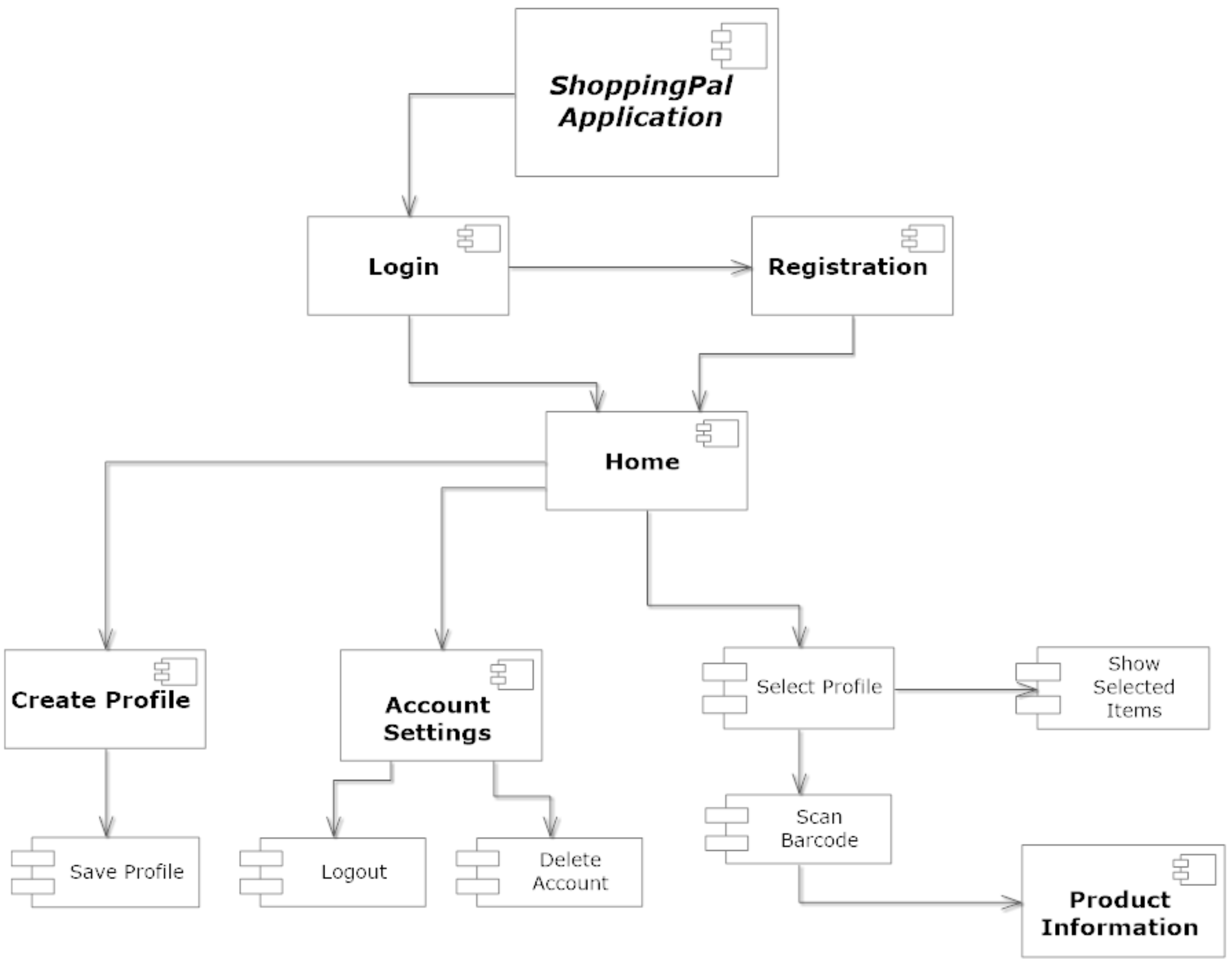
# 4 High Level Design

## 4.1 Class Diagram



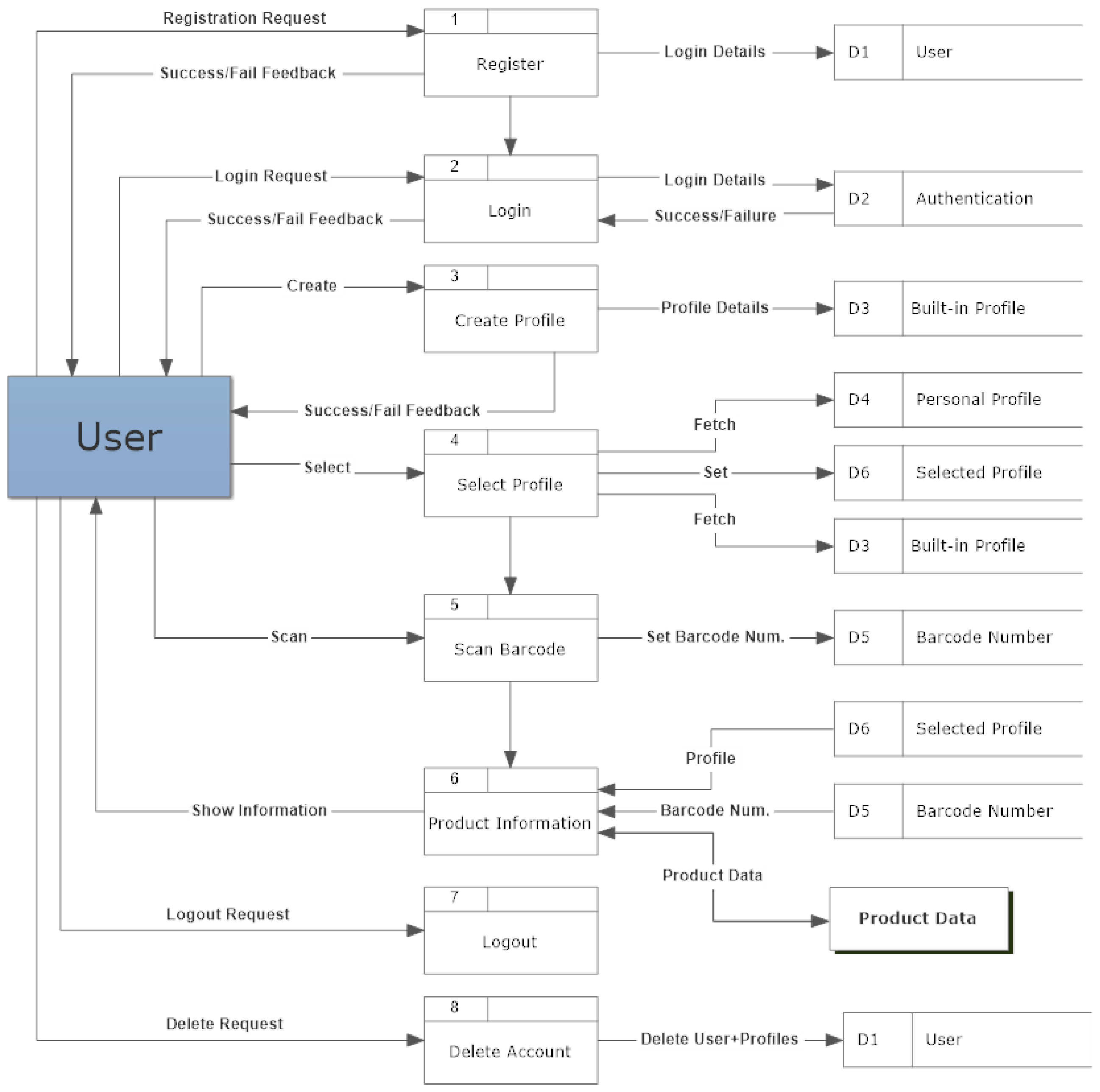
This is a class diagram of the ShoppingPal System. A class diagram is a type of static structure diagram that describes the structure of a system. In this case, it is showing ShoppingPal's classes, attributes, methods and the relationship between all the classes. ShoppingPal has 7 classes in total, 1 class for each "Activity" (Screen) and within each class lies its methods and attributes with makes up functionality contained within each screen. Another class called ZXingScannerResultHandler handles the result of the scan performed by the ZXing Library and sends the result to the "InformationActivity", which can later be used for product data retrieval.

## 4.2 Component Diagram



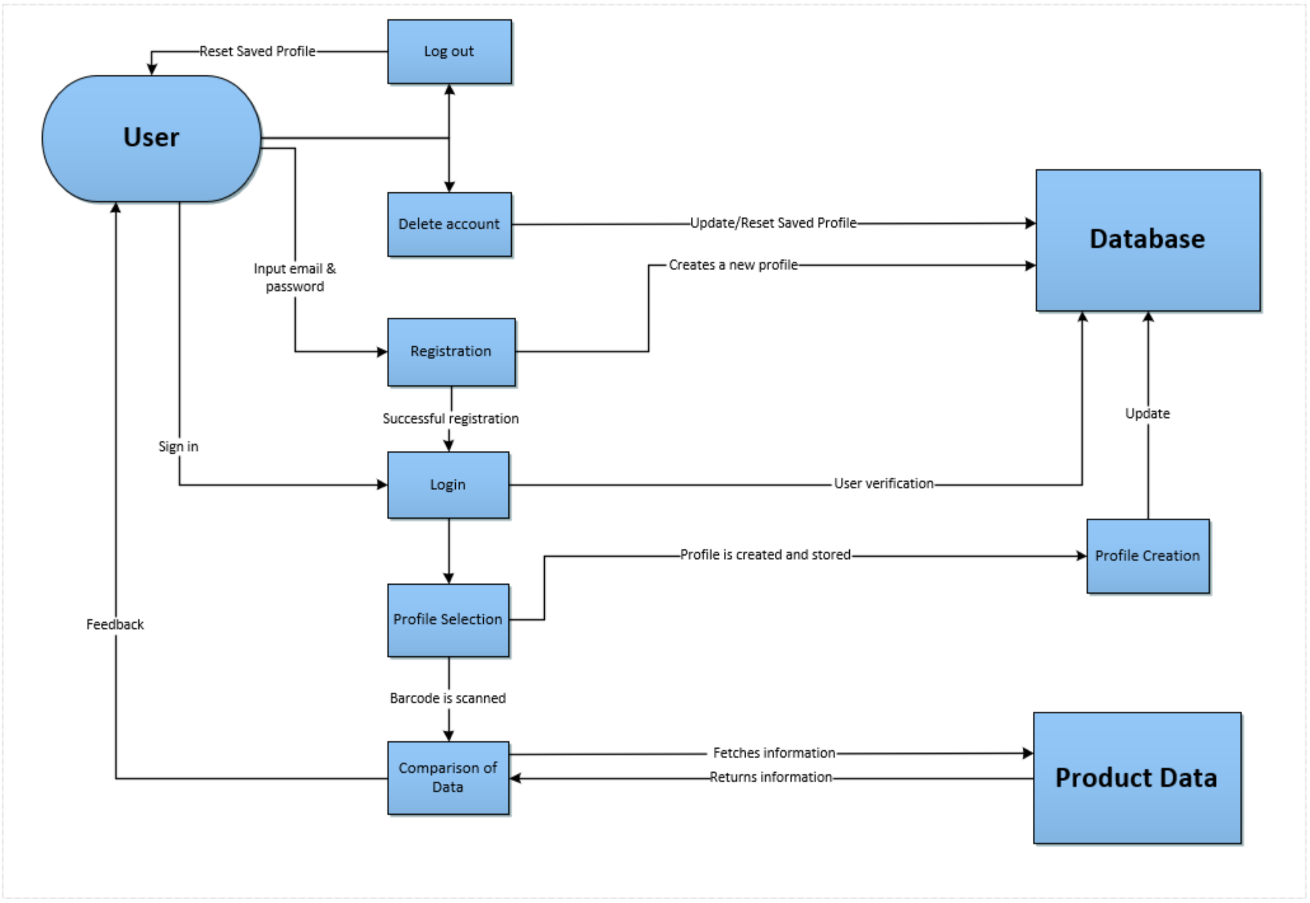
This is a modified version of the component diagram. It shows each screen and the functionality that is contained within them. It also shows the order that these screens and functions are executed in. For example, when you are on the "Home" page you can enter the "Account Settings" page, within "Account Settings" lies 2 functions "Delete Account" and "Logout". Likewise, when you are on the "Home" page you cannot perform the "Scan" function until you perform the "Select Profile" function as one relies on the other.

## 4.3 Data Flow Diagram



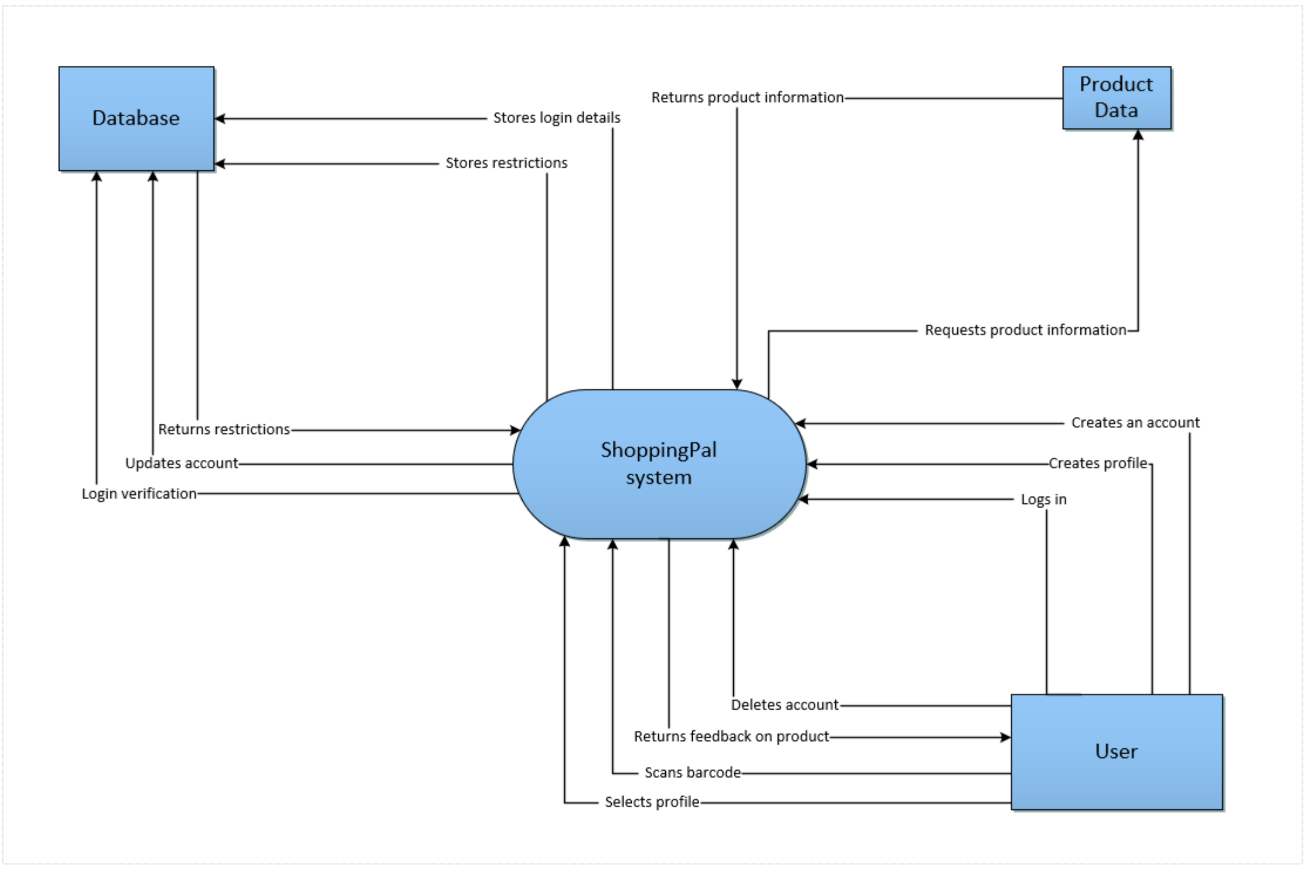
This is a data flow diagram. It is a representation of how the data advances through the ShoppingPal system and the relationship between the data and the processes.

## 4.4 Process Diagram



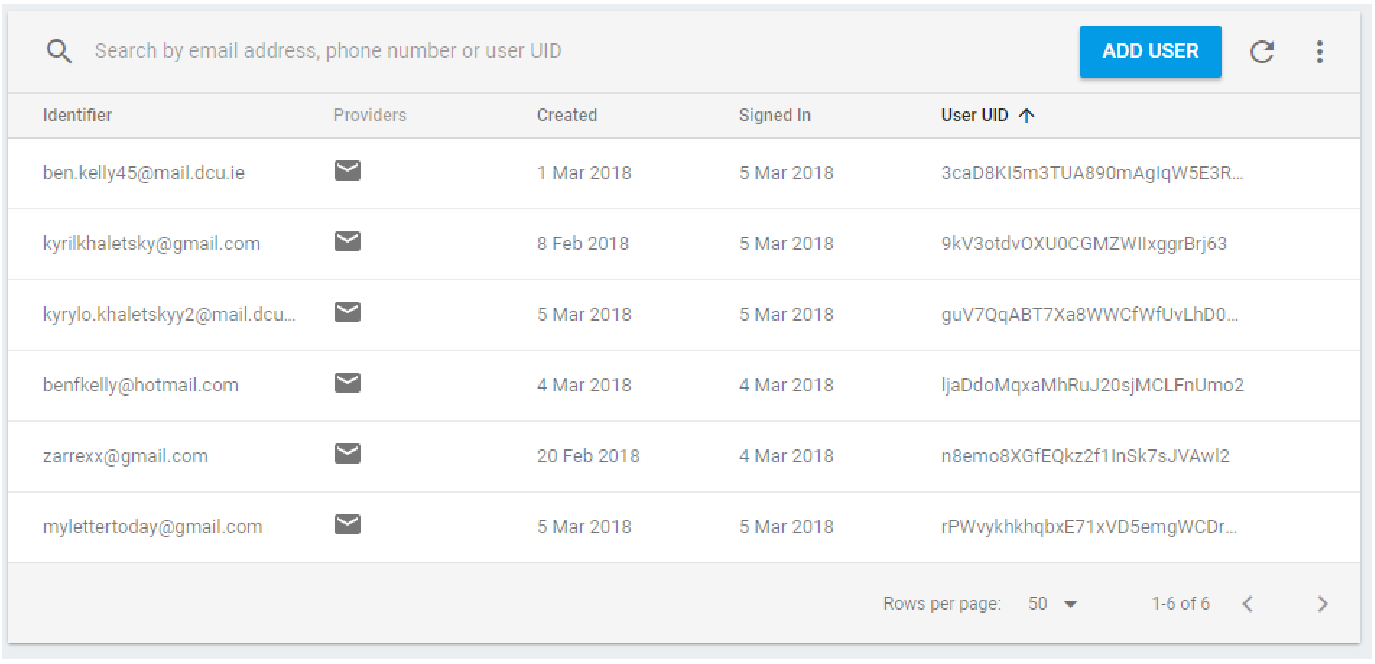
The process diagram has not seen much change compared to our functional spec. The process diagram above is a graphical representation of the movement of processes through our system. It essentially shows how each process works at step by step intervals. For instance, a user cannot directly interact with the Database or our product data but through the app in various ways the User's details can be changed to reflect an apparent change to each respective process.

## 4.5 Use Case Diagram



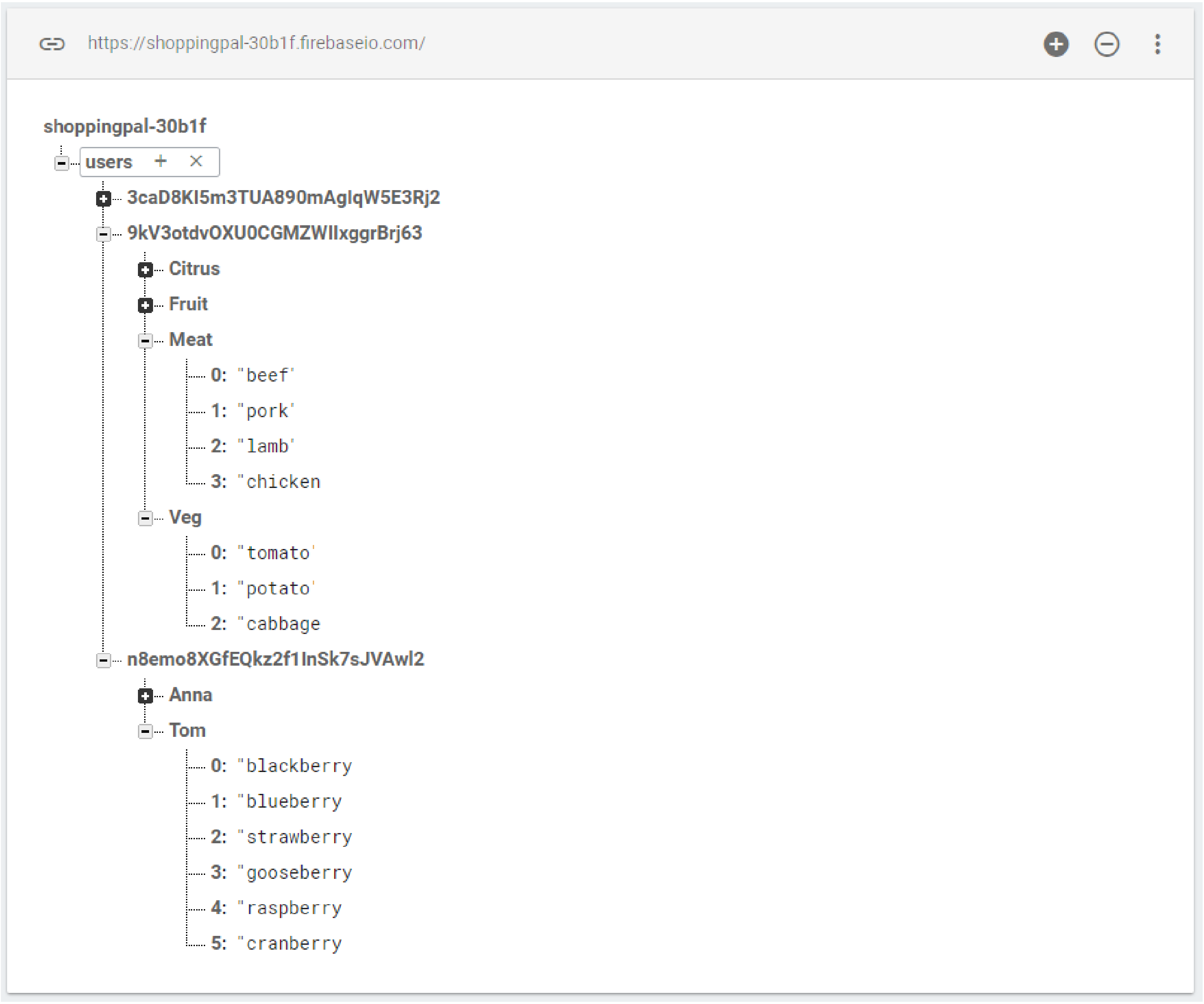
The process diagram has not seen much change compared to our functional spec also. We have stuck to our formula and implemented all the functionality stated previously. This Use Case diagram is displaying the set of actions that the ShoppingPal system performs in collaboration with the external user and product dataset and Firebase Database.

## 4.6 Firebase Layout



| Identifier | Providers | Created | Signed In | User UID ↑ |
|---|---|---|---|---|
| ben.kelly45@mail.dcu.ie | ✉ | 1 Mar 2018 | 5 Mar 2018 | 3caD8KI5m3TUA890mAgIqW5E3R... |
| kyrilkhaletsky@gmail.com | ✉ | 8 Feb 2018 | 5 Mar 2018 | 9kV3otdvOXU0CGMZWIlxggrBrj63 |
| kyrylo.khaletskyy2@mail.dcu... | ✉ | 5 Mar 2018 | 5 Mar 2018 | guV7QqABT7Xa8WWCfWfUvLhD0... |
| benfkelly@hotmail.com | ✉ | 4 Mar 2018 | 4 Mar 2018 | ljaDdoMqxaMhRuJ20sjMCLFnUmo2 |
| zarrexx@gmail.com | ✉ | 20 Feb 2018 | 4 Mar 2018 | n8emo8XGfEQkz2f1InSk7sJVAwl2 |
| mylettertoday@gmail.com | ✉ | 5 Mar 2018 | 5 Mar 2018 | rPWvykhkhqbxE71xVD5emgWCDr... |

Rows per page: 50 1-6 of 6



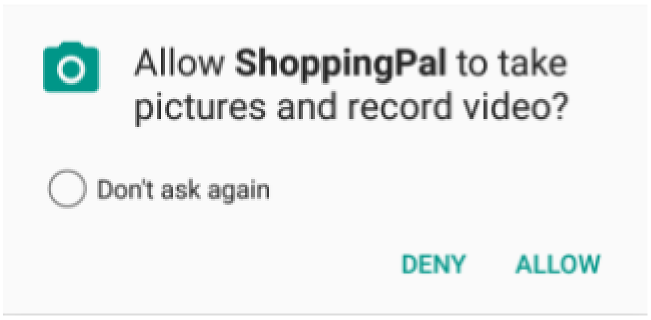
```
https://shoppingpal-30b1f.firebaseio.com/

shoppingpal-30b1f
  users
    3caD8KI5m3TUA890mAglqW5E3Rj2
    9kV3otdvOXU0CGMZWIlxggrBrj63
      Citrus
      Fruit
      Meat
        0: "beef"
        1: "pork"
        2: "lamb"
        3: "chicken
      Veg
        0: "tomato"
        1: "potato"
        2: "cabbage
    n8emo8XGfEQkz2f1InSk7sJVAwl2
      Anna
      Tom
        0: "blackberry
        1: "blueberry
        2: "strawberry
        3: "gooseberry
        4: "raspberry
        5: "cranberry
```

The images above show the layout of the Firebase Server and Database. The first image is the list of users currently signed up (our personal and DCU accounts). Firebase also shows other useful information such as when the user last signed in and when the account was created. On the right hand side, you can see the corresponding User UID which is later used to define a user within the Firebase Database. The Database is comprised of a JSON format and uses a tree system to locate users and their profiles. As shown in the second image above there is a users child which contains each user (userID) that created a profile. Only users that created a personal profile are entered into the database, this reduces the number of users which need to be added increasing overall performance and storage. Each userID has a number of children named after the profile name which they have created previously, each profile name contains a list of ingredients added by the user when they were creating a profile.

## 5 Problems and Resolution

### Camera permissions


Allow **ShoppingPal** to take pictures and record video?

Don't ask again

DENY ALLOW

We ran into a major issue that when you try to access the camera for the first time you need to allow permissions deep in Android settings. The problem here is that the user demographic for this app is not expected to be “tech savvy” and therefore we needed to implement this solution so that it asks directly within the app when the camera is launched the first time. We wanted to implement an attractive popup dialog (shown above) which would prompt the user to grant permissions. To solve this problem, we have implemented a method (code shown in the blog) which requested Camera permissions directly within the app if the user has not accepted them prior, if they have it will continue as normal, then it checks if the permissions have been granted every time the user presses the scan button, in case the user has reset their phone to factory settings.

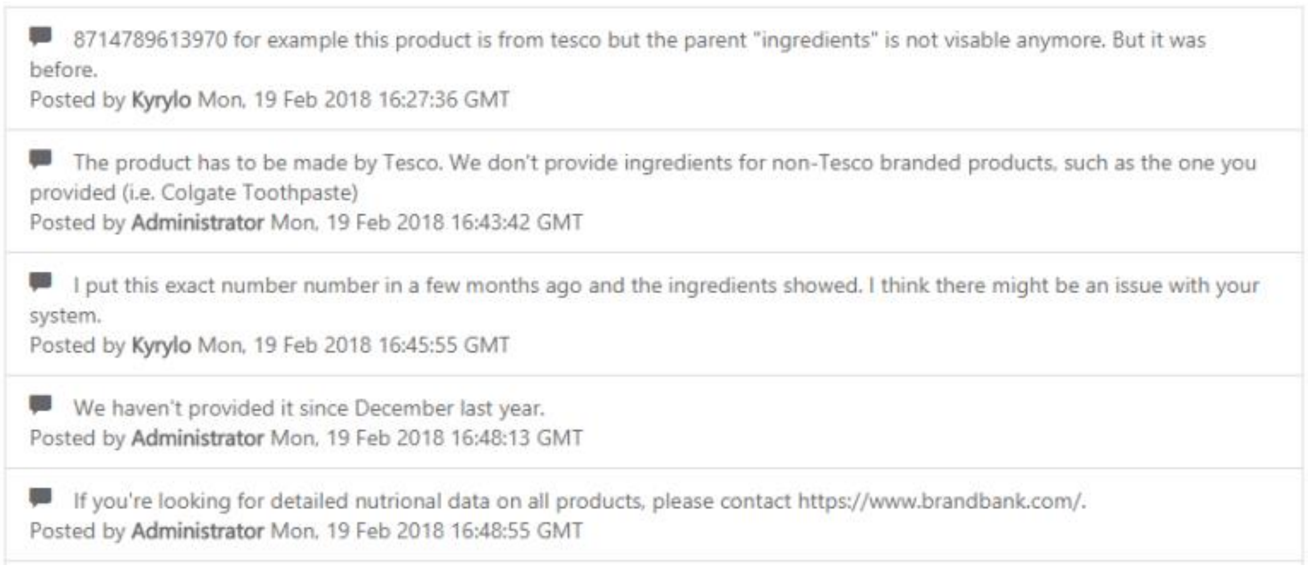
### Drop-down menu resetting its value/state

During the implementation of the drop-down menu, we have found that the value of the menu (profile) does not stay as the value selected by the user when the user exits the app or minimizes the app or even when the user presses back (since the back button starts the Activity again). With the previous implementation, the user had to manually select the profile they would like to use every time they scan an item. The source of this problem is that the spinner was initialized in the onCreate() method. To solve this issue we created 2 methods, startSpinner() and restoreSpinner() which were initialized any time the application was paused or restarted. We also stored the selected value in sharedPreferences which is Androids built-in preferences, this saved the position of the drop-down menu so the user no longer needs to re-select the profile they intend on using.

### Apache library depreciated

Originally we were planning on using the API provided by Tesco which would return product data. But as we were implementing this functionality we found out that Androids HTTP Apache library has depreciated. To combat this we implemented our own version of Tesco's API using HttpUrlConnection which performed in the same way but now avoiding the depreciated HTTP Apache library.

## Product data limited


8714789613970 for example this product is from tesco but the parent "ingredients" is not visable anymore. But it was before.
Posted by **Kyrylo** Mon, 19 Feb 2018 16:27:36 GMT

The product has to be made by Tesco. We don't provide ingredients for non-Tesco branded products, such as the one you provided (i.e. Colgate Toothpaste)
Posted by **Administrator** Mon, 19 Feb 2018 16:43:42 GMT

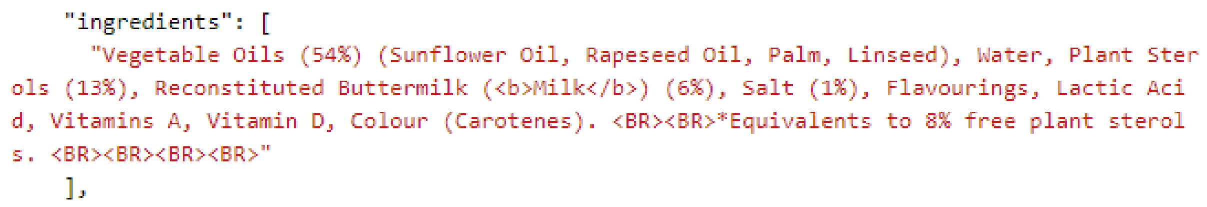
I put this exact number number in a few months ago and the ingredients showed. I think there might be an issue with your system.
Posted by **Kyrylo** Mon, 19 Feb 2018 16:45:55 GMT

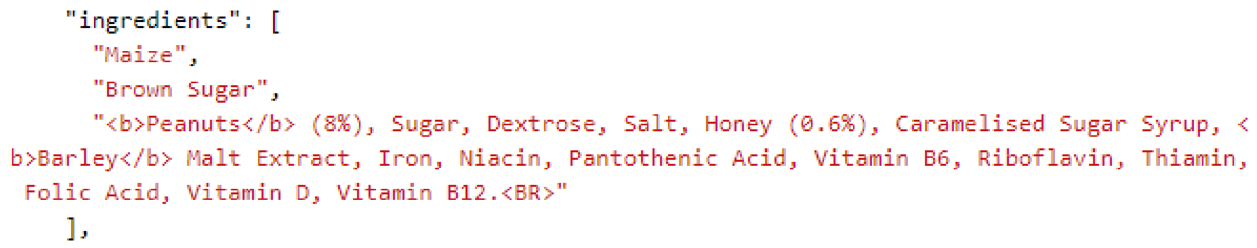
We haven't provided it since December last year.
Posted by **Administrator** Mon, 19 Feb 2018 16:48:13 GMT

If you're looking for detailed nutrional data on all products, please contact https://www.brandbank.com/.
Posted by **Administrator** Mon, 19 Feb 2018 16:48:55 GMT

During our research and functional spec phase, we have performed multiple tests with the Tesco API to see how it works, and we found that it returned the data for all products found in Tesco as well as products found in other shops (with exception Aldi and Lidl). This was a great collection of products and worked well at the time of initial testing. But, after we came back to begin our implementation after semester 1 exams we found that some products which returned product data before are not working now. After talking to a representative from Tesco Support (as shown above) we found out that Tesco no longer provides product data on products not manufactured by them. This means that only Tesco own brand products were contained in the product dataset. We found it that the problem for this is because Tesco's system used to use Nielsen's data which Tesco no longer have permission to give out to us. We contacted Nielsen but we got no reply. Although only for Tesco products our application still runs perfectly.

## Tesco Dataset structure flawed


```
    "ingredients": [
      "Vegetable Oils (54%) (Sunflower Oil, Rapeseed Oil, Palm, Linseed), Water, Plant Ster
ols (13%), Reconstituted Buttermilk (<b>Milk</b>) (6%), Salt (1%), Flavourings, Lactic Aci
d, Vitamins A, Vitamin D, Colour (Carotenes). <BR><BR>*Equivalents to 8% free plant sterol
s. <BR><BR><BR><BR>"
    ],
```


```
    "ingredients": [
      "Maize",
      "Brown Sugar",
      "<b>Peanuts</b> (8%), Sugar, Dextrose, Salt, Honey (0.6%), Caramelised Sugar Syrup, <
b>Barley</b> Malt Extract, Iron, Niacin, Pantothenic Acid, Vitamin B6, Riboflavin, Thiamin,
 Folic Acid, Vitamin D, Vitamin B12.<BR>"
    ],
```

Another issue we ran into which was out of our hands was that the format which the Tesco Dataset returns its ingredients is flawed and inconsistent. When querying for products the format of the JSON object is not correct in the ingredients child. Screenshots of the flaw can be seen in the first 2 images above. The problem is that some products produce a correct JSON Object but others have the wrong output as outlined above. It seems that the quotation marks are in the wrong place which only registers total of 1 ingredient in the first image and total of 3 ingredients for the second image. We contacted support reporting this issue but it has still not been fixed. Due to this inconsistency which is out of our control we are forced to parse the entire JSON Object as a String and perform complex String Regex to retrieve the ingredients necessary for our app to function. It was important to remove the breaks from the String as well as some special characters, whitespaces and percentages, which are unnecessary for the functionality of our app. This unfortunately leads to our code being more "messy" but on the bright side, this means that in the future this code can be used to parse the ingredients from any database whether it is a JSON Object or not as this can be used to parse any String.

### Other

Other smaller issues regarding code and implementation of functionality we have found throughout the implementation of ShoppingPal are outlined in our blog (https://kellyb45ca326blog.wordpress.com) where we go through how we solved and tested them throughout every step of our build process. We have also included more detailed solutions to problems mentioned above along with screenshots and code samples.

## 6 Installation Guide

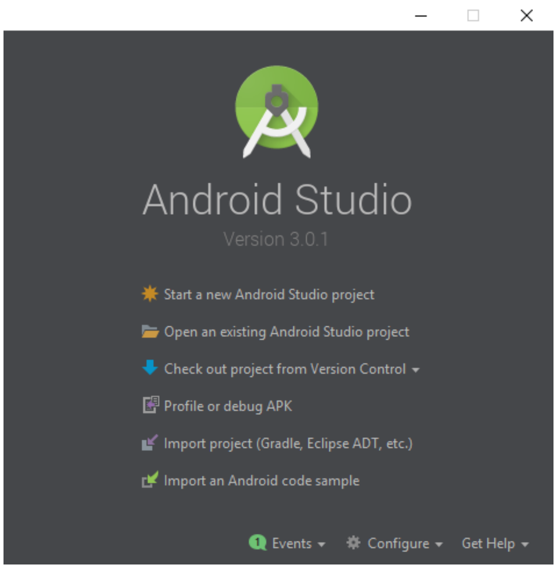
*Due to the nature of our Application you cannot scan barcodes from an emulator and must install the app on a real device to test scanning function (other functionality is fine on an emulator)*

### Required Software

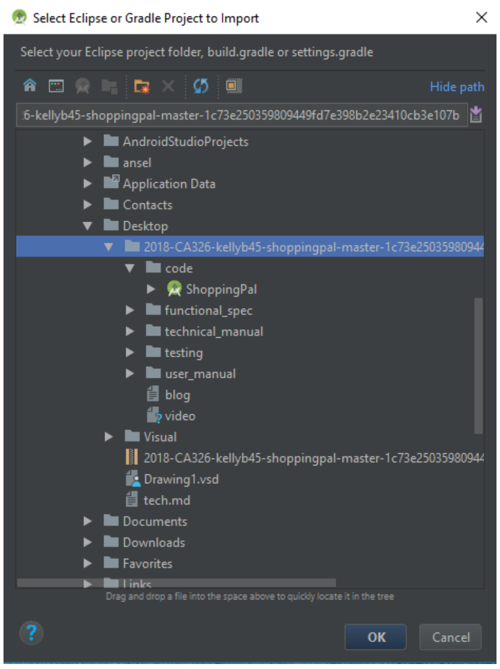
- Android Studio 3.0.1
- Java 8
- Android Device (5.0.1 Lollipop or higher)

### Android Studio Build Instructions

- Choose a directory to clone the Git repository
- Open a Git terminal and change to the chosen directory
- Run the following Git clone command: "git clone https://gitlab.computing.dcu.ie/kellyb45/2018-CA326-kellyb45-shoppingpal.git"
- Open Android Studio, you will be presented with the following screen:


Android Studio
Version 3.0.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Events
Configure
Get Help

- Choose "import project" and navigate to the 2018-CA326-kellyb45-shoppingpal/code/
- Select "ShoppingPal" as shown in the image below:


Select Eclipse or Gradle Project to Import

Select your Eclipse project folder, build.gradle or settings.gradle

6-kellyb45-shoppingpal-master-1c73e250359809449fd7e398b2e23410cb3e107b

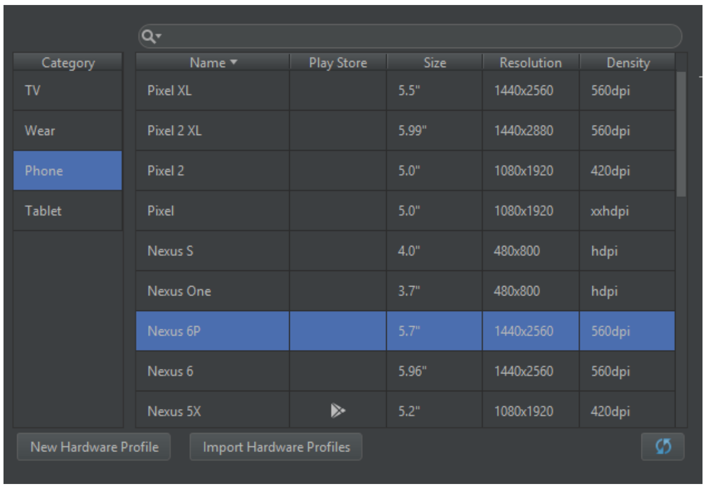
- AndroidStudioProjects
- ansel
- Application Data
- Contacts
- Desktop
  - 2018-CA326-kellyb45-shoppingpal-master-1c73e25035980944
    - code
      - ShoppingPal
    - functional_spec
    - technical_manual
    - testing
    - user_manual
    - blog
    - video
  - Visual
  - 2018-CA326-kellyb45-shoppingpal-master-1c73e25035980944
  - Drawing1.vsd
  - tech.md
- Documents
- Downloads
- Favorites
- Links

Drag and drop a file into the space above to quickly locate it in the tree

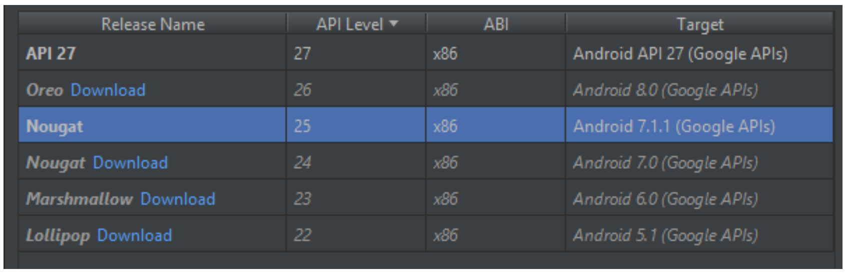
- Wait for the build to finish

## Emulator Instructions

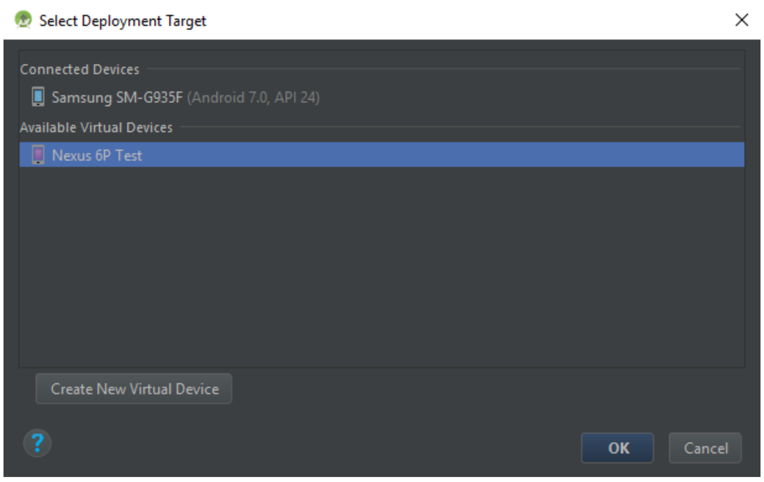
- Press on Tools>Android>AVD Manager found on the Toolbar
- Press on "Create Virtual Device"
- Choose Nexus 6P as shown below and press Next:


| Category | Name | Play Store | Size | Resolution | Density |
|---|---|---|---|---|---|
| TV | Pixel XL | | 5.5" | 1440x2560 | 560dpi |
| Wear | Pixel 2 XL | | 5.99" | 1440x2880 | 560dpi |
| Phone | Pixel 2 | | 5.0" | 1080x1920 | 420dpi |
| Tablet | Pixel | | 5.0" | 1080x1920 | xxhdpi |
| | Nexus S | | 4.0" | 480x800 | hdpi |
| | Nexus One | | 3.7" | 480x800 | hdpi |
| | Nexus 6P | | 5.7" | 1440x2560 | 560dpi |
| | Nexus 6 | | 5.96" | 1440x2560 | 560dpi |
| | Nexus 5X | ▶ | 5.2" | 1080x1920 | 420dpi |

- Choose "Nougat API level 25" shown below, Download and press Next


| Release Name | API Level | ABI | Target |
|---|---|---|---|
| API 27 | 27 | x86 | Android API 27 (Google APIs) |
| *Oreo* Download | 26 | x86 | Android 8.0 (Google APIs) |
| Nougat | 25 | x86 | Android 7.1.1 (Google APIs) |
| *Nougat* Download | 24 | x86 | Android 7.0 (Google APIs) |
| *Marshmallow* Download | 23 | x86 | Android 6.0 (Google APIs) |
| *Lollipop* Download | 22 | x86 | Android 5.1 (Google APIs) |

- Choose Portrait startup orientation and press Finish.
- Press "Run App" found in the toolbar
- Choose The Nexus 6P virtual device you just created and press ok as shown below:


Select Deployment Target
Connected Devices
Samsung SM-G935F (Android 7.0, API 24)
Available Virtual Devices
Nexus 6P Test
Create New Virtual Device
OK
Cancel

- The build will run and the application will start automatically
- Be sure to update google play services (if prompted) and enable internet connection

## Device Installation Instructions

- Find a file called shoppingpal.apk found at: 2018-CA326-kellyb45-shoppingpal/code/
- Copy this file to your device
- Open your device
- Allow installation of "Unknown Sources"
- Find where you copied the file on your device and press install
- App will install be sure to enable internet connection
- How to navigate and use the app can be found in the user_manual

## Device Installation Instructions through Android Studio

- You can install it by connecting your phone to your computer and enabling "ADB connection"
- Press "Run App" found in the toolbar
- Now select your device instead of an emulator (same as image above)
- Be sure to update google play services (if prompted) and enable internet connection