GLOBE Grocery Tracker

MOBILE APPLICATION

TECHNICAL DOCUMENTATION

Created By	GLOBE – Deakin Research Tech Translators
Document Purpose	Explanation of Project code and artefacts
Date of Release	
Project Supervisor	Rosemary Van Der Meer

Version#	**Author**	**Change Made**
v0.1	Joel Daniel	Base Build of document.
v0.2	Joel Daniel	Modifications to build and pictures in document.
v0.3	Amie Le	Explanation of IDE download and setting up application repository access.
v.04	Ryan Gallagher	Explanation of Firebase Database and utilizing it for the application.
v0.5	Faiz Shibly	Inclusion of repository structure and explanation of folders.
v1.0	Joel Daniel	Explained main functions of application. Polished the draft as first version of documentation.
v1.1	Ryan Gallagher	Step-by-step guide for Developer IDE Setup, establishing VCS, connecting to Bitbucket, general environment setup guide.
v1.2	Joel Daniel	Addition of Document Control and Changes
v1.3	Faiz Shibly	Receipts View and Dynamic Notification features explained
V1.4	Joel Daniel	Inclusion of method to connect to Google Sheets containing database data.

Contents

Introduction

The Globe Grocery Tracker mobile app provides an easy way for users to help Deakin University Globe Research team to research about population health and obesity trends through participants giving research data, by uploading the images of their grocery receipt and some personal information (e.g age, height, weight, gender).

Compatibility

Compatible with all Android Devices.

Developed and tested against API XX and

Software used

- Android Studio 4.1 Canary
- Java
- Android Virtual Device Emulator (comes with Android Studio, to be downloaded)

Bitbucket Repository = https://bitbucket-students.deakin.edu.au/scm/globe/globe---t319.git

Getting Started

The application codebase is hosted in a Bitbucket Repository and can be downloaded or pulled into the developer's local machine for development purposes.

Requirements:

- IDE – Android Studio 4.1 (Canary 6+ Release). Windows, MacOS and selected Linux operating systems, support the IDE. The relevant version can be downloaded from this link -> https://developer.android.com/studio/preview
- GIT account – To safely connect to the repository, enable version control for future development
- Access to the Bitbucket repository, the link which can be found above
- A Google Firebase account
- A Google Sheets account

Steps:

1. Downloading the Right IDE

All development works for the *Grocery Tracker* application have been undertaken using **Android Studio.** It is important to ensure you use the *Canary* build of Android Studio.

Important Notes:

- ***To experienced Java developers, Android's IDE includes the relevant Java Development Kit (JDK) when you download and set up the IDE for the first time. Java-based Android projects are usually developed on the JDK-8 release.***
- *This is not a tutorial for the Android Studio environment in of itself, rather a brief setup guide for Grocery Tracker developers. Additional information and online resources to getting familiar with Android Studio, and Android development in general, is provided in the appendix of this document.*
- *Android Studio Canary is an early preview build of the application. The application for Linux, Windows and Mac come in zip files, no installation required.*
- *To install Android Virtual Device Emulator, simply go to Tools after opening the application, select AVD and proceed to install the emulator (However, if you already had Android Virtual Device emulator installed already on another Android Studio application, then there is no need to install AVD through Canary).*
- *Android Studio Canary can be run alongside the stable or Beta version of the application on the same device.*

For Mac

a. Download the ZIP file and unzip its contents.
b. Drag the app file to the *Applications* folder.
c. You can create a shortcut link to the application executable file.
d. Now click and launch the application.

For Linux

a. Download the Linux application ZIP file and unzip it.
b. Rename the folder to *android-studio-preview* or *android-studio-canary*
c. Move the folder to the */usr/share/* or */usr/local/* folder location, where your Linux programs are usually installed.
d. Nagivate to the */bin* folder in the android studio canary folder through terminal, and run the *studio.sh* file**.**

For Windows

a. Download the IDE using this link. Once complete, you should see the .zip file in your *Downloads* folder. Select "*extract all*" and nominate a suitable file location.
b. As this is the *Canary* build, it might be easier for you to set your own shortcut to launch the application. Navigate to the download folder/android-studio/bin, and create a shortcut for the *studio64.exe* icon on your desktop.
c. To launch the application, simply double-click this icon to get started, and select the "*Do not import settings"* icon.
d. If you are prompted to download any updates, such as the included device emulator, please do so.

For more information on getting started with Android Studio 4.1 Canary, please refer to the link below: https://www.youtube.com/watch?v=7vvMltQtfxY

2. Connecting to the Bitbucket Repository, and Connecting to Android Studio

Now that you have installed and initialized the development environment, you may now initialize your connection with the *Grocery Tracker* Bitbucket repository.

a. Go to the *Grocery Tracker* repository here:

   https://bitbucket-students.deakin.edu.au/projects/GLOBE/repos/globe---t319/browse

b. Once you have logged in, navigate to the *Clone* icon, and copy the repository's URL suffixed with .git.
c. *Downloading the Bitbucket Plugin:* Navigate back to Android Studio, and at the top navigation bar, select Tools > SDK Manager > Plugins. Locate "*Bitbucket Linky by Atlassian Labs"* and Install.
d. Once the plugin installation is complete, restart the IDE.

3. Enabling VCS (Version Control)

Your IDE is nearly ready to pull the code repository from Bitbucket. You now need to enable a Git repository locally.

a. Go to VCS > Enable Version Control Integration
b. Select "Git" as your preferred VCS and select OK.

4. **Cloning Environment**

   Now you can pull the *Grocery Tracker* codebase into your local machine.

   a. Navigate to VCS > “Get From Version Control” > “Repository URL”.
   b. From the bitbucket URL you copied earlier, paste this into the “URL” form.
   c. Select “Clone”, and place in your *bitbucket-students* username and password.
   d. After a short wait, the application will begin to download. As the application codebase (including build files) in unbuilt form is approx. 100mb, this may take some time for slower internet connections.

   Note: It may also take an extended period of time to configure the application for the first time. Should Android Studio prompt you to download updated dependencies, please select OK.

   e. Once this is complete, the *Build Inspector* in your IDE will provide a message that states “CONFIGURE SUCCESSFUL”.

The repository has been successfully cloned into your local machine from the repository.

5. **Test Commit**

   After a successful clone, you may now test the connection between your local *Grocery Tracker* environment and the Bitbucket repository.

   a. Navigate to VCS>Commit
   b. Type in “My first commit” or another message that specifies no changes to the codebase has been made, and this is simply a test commit.
   c. Select *Commit*
   d. Now the changes you have made have been staged to push. Navigate to VCS>Git>Push. Your recent commit to this branch will appear. Select the commits you wish to push in to the branch repository, and select *Push.*
   e. When you log back in to Bitbucket and navigate to the relevant feature branch, you will see the most recent commit, with the message specified in step 5 (b).

Outcome:

The application code-base should be pulled from the repo, and have built successfully, allowing you to continue development from your local machine.

Additional Items:

I. Connecting to Firebase Realtime Database

Firebase™ is a realtime storage, database and analytics platform powered by Google. It is commonly used with Android applications, as it allows app developers to initialize a centralized database that synchronizes with individual app users in real time. For the purposes of *Grocery tracker,* Firebase's realtime database capabilities are used to capture individual app user's information and grocery shopping receipts.

As at the most recent *Grocery Tracker* build, it will not be necessary to initialize a new firebase Realtime database, as the current development is linked to the existing database and account set up for the GLOBE client.

II. Connecting Firebase to Google Sheets

Google sheets is also used in the *Grocery Tracker* application as a means for GLOBE to access user and purchasing information within an easy to use interface. In the first instance, you will not be required to

To access the Google sheet storing user and receipt data, proceed to the following link and provide these credentials:

- Link = https://docs.google.com/spreadsheets/d/107JElJ7SOh8YgFaq7vViZn-70mjCDPY5C28wRaERjMs/edit#gid=0
- Email = **globedeakin@gmail.com**
- Password = **Globe1234**

The document contains two sheets, one for Receipt data, and the other for User data. A button **"Synchronize Data"** synchronizes the document with the Firebase Database, enabling the analyst/team to view user provided data.

III. Common problems / troubleshooting:

a. Gradle Build versions
b. OS-specific issues.
c. checkout from the correct branches
d. Build problems – Accepting SDK build tools licenses.

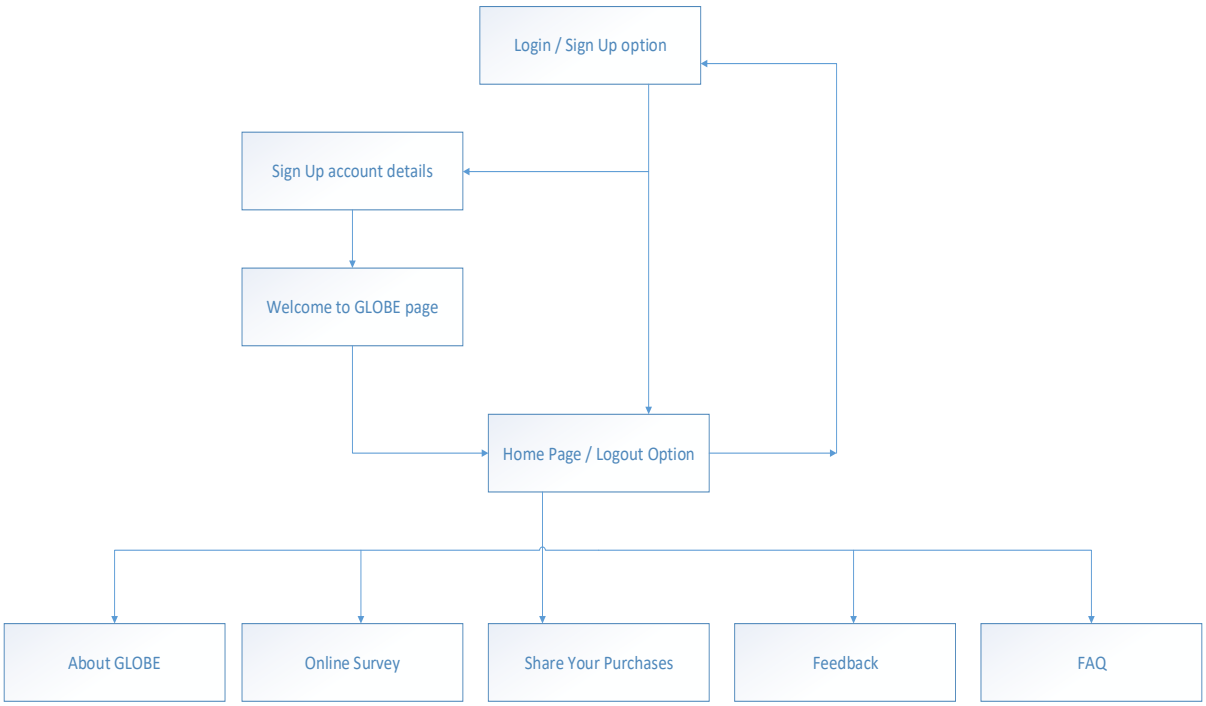
   Oftentimes, you may try to Build the project and the process will fail, with a warning similar to *"Failed to Install the following Android SDK Packages"…*

   This often occurs because when you set up your development environment, you may have not had the chance to install and/or accept licenses to specific build packages that are relevant to *this* project.

   - This can be resolved by navigating to Tools > SDK Tools > Android SDK.

- If the specified package that inhibited the build has not been included as an installed package, you need to locate the relevant package and install it. Be sure to accept agreements.
- Once this is complete, retry the project build, and in the absence of other build issues, it should build successfully.

Application Roadmap



Application File Contents

The **main** directory contains the .java, .xml and .webp files that the application will utilize. Resource files such as icons, layouts, strings etc… will be found in **src/main/res,** while the files containing application functions and modules can be found in **src/main/java/com/stevecrossin/grocerytracker.**

Application Repositories

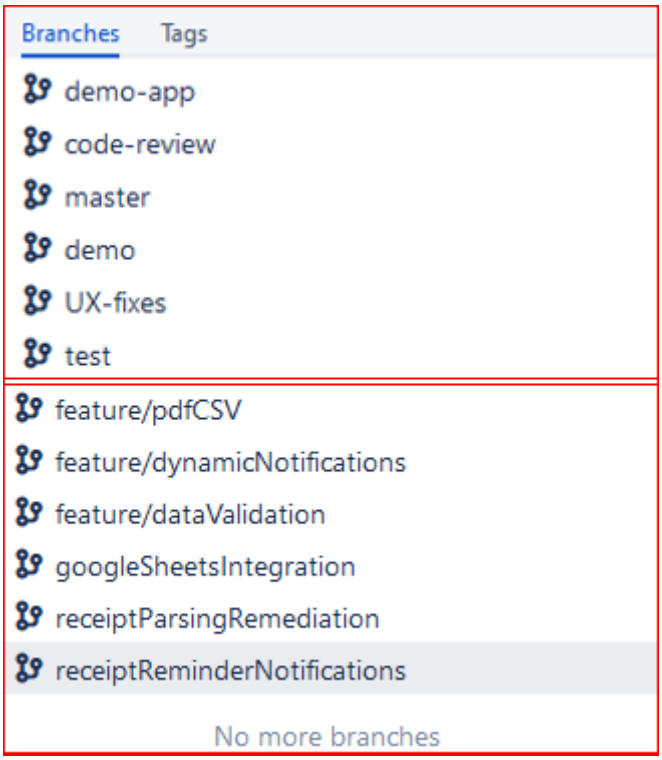


Figure 1: Repository Branches

master = main branch, only Steve(project leader)/authorized member has authority to change and modify this branch as the app runs from this.

test/UnitTesting = branch for testing application modifications and integrations.

Other branches = for separate testing and development of features, bugs etc…

Bitbucket directory structure (master branch)

App folder – contains all source file of the app. Further broken down, it includes:

/res folder - contains all layout XML files for the app, colors, strings, styles and draw able objects. Also contains a raw directory which holds all read only text files.

/Java files:

Adapter - contains the code for all the elements that make adapters and view holders that are contained in the application function

Database - contains the data repositories for the application and entities that exist in that database

Entities - contains the entities for each database table in the application, and the SQL operations for each database

Models – Contains source codes for parsing Woolworths and Coles integrations.

Screens - contains the files that provides how each activity in the application functions and the operations of those activities when the user interacts with the UI.

Utils – Contains source code for validating inputs, notification reminder feature

Other files and folders in the master branch:

/googleSheetsIntegration folder: Contains source code for integration between Firebase and Google sheet, user tables with google sheet and instructions to connect google sheets with firebase

Bugs.txt: A list of active and resolved bugs in the application can be found here

Changelog.txt- list of changes made in the application

Todo.txt- A list of tasks that are needed to be done.

Application UIs and Functionality

Application Layout files can be found in **app\src\main\res\layout** (the **layout(land)** has the landscape file for the Home/Main page).

Login

File = activity_login.xml

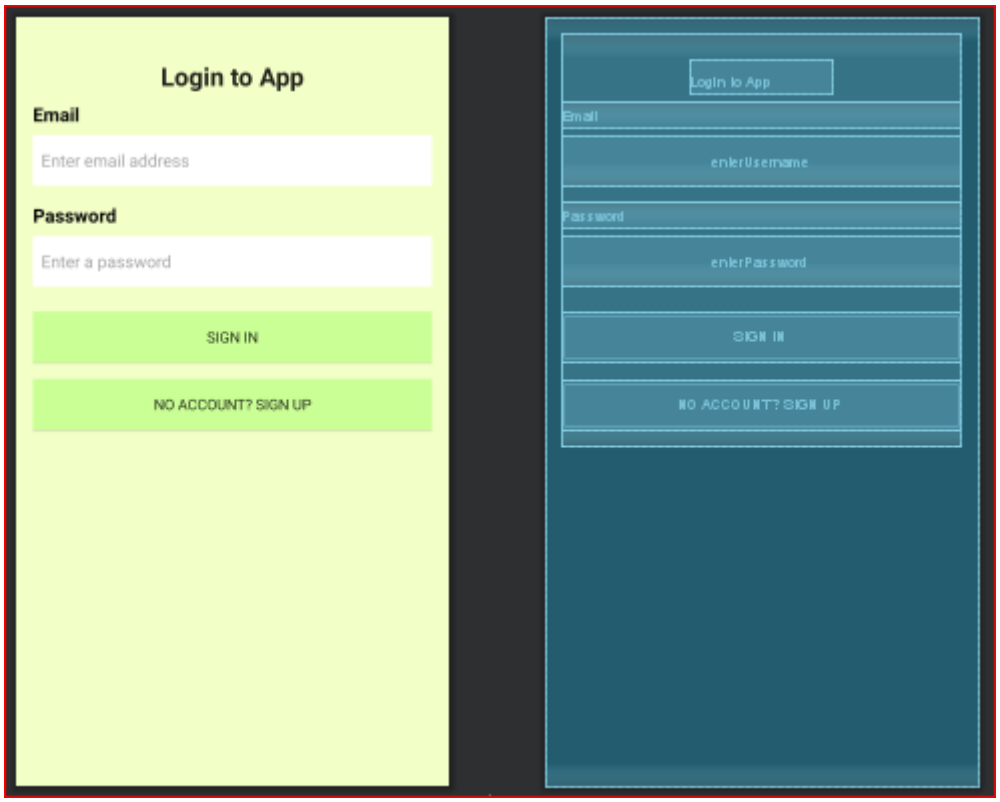


Figure 2: Login UI and Blueprint

The user is required to provide the email with which they created their account as well as the password which they provided. After providing the email and password, the user has to click on the “Sign In” button to authenticate themselves. If the credentials don’t match or exist, an error stating incorrect username/password will appear, requiring the user to re-enter their credentials.

If it is a new applicant, they can access the **Sign Up** page using the “No Account? Sign Up” button.

Sign Up

File = activity_signup.xml

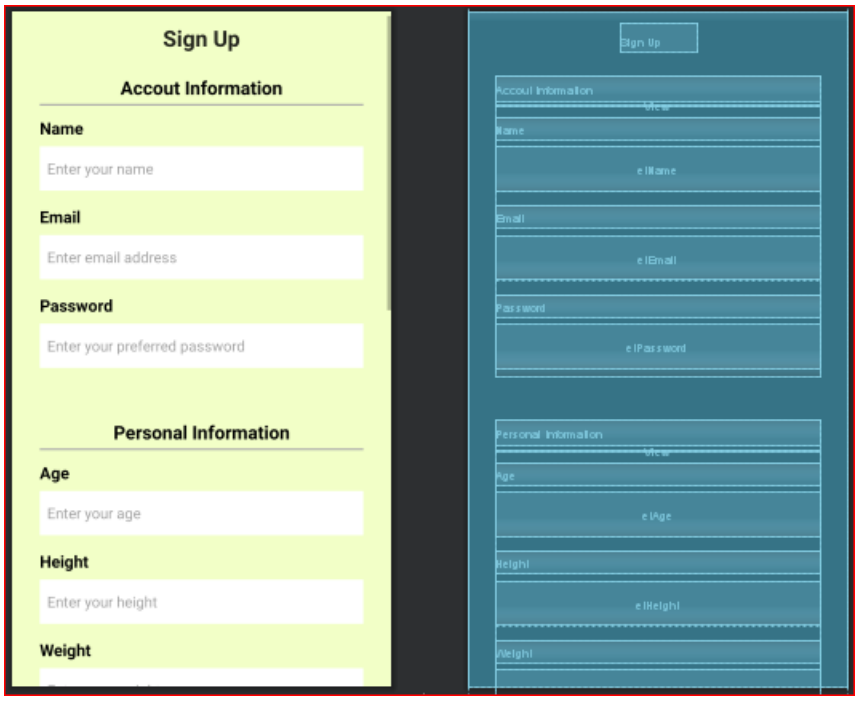


Figure 3: Sign Up UI and Blueprint (Account and Personal Information)

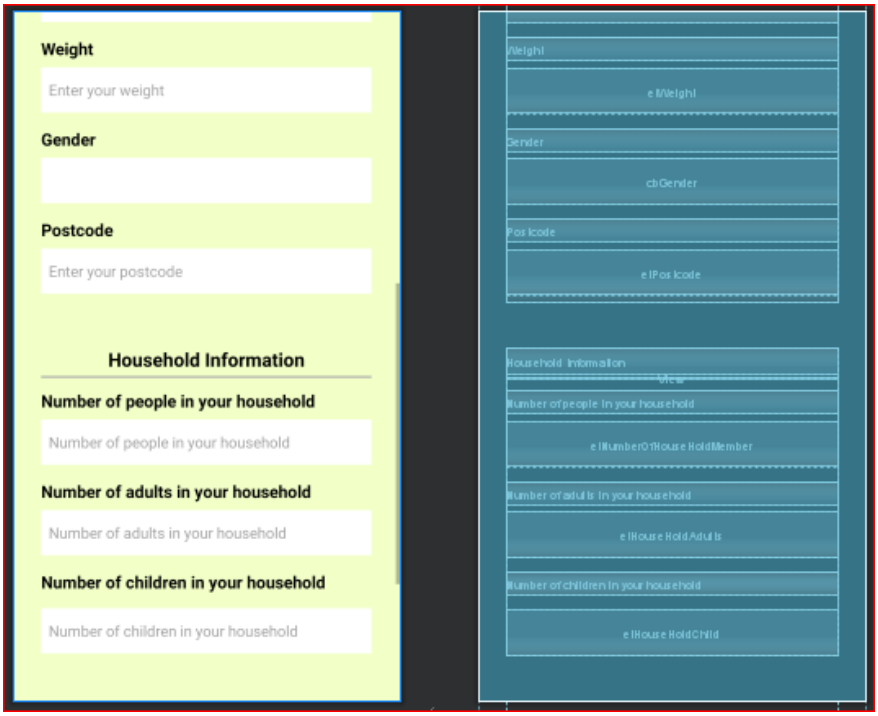


Figure 4: Sign Up UI and Blueprint (Personal and Household Information)

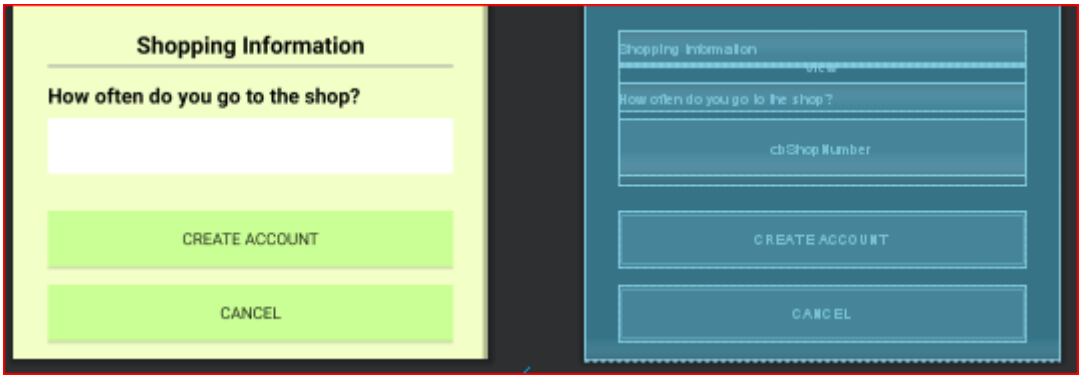


Figure 5: Sign Up UI and Blueprint (Shopping Information)

Below table shows a list of the Sign Up information required, their format and validation rules implemented:

Field	Input Type (Front-end)	Validation rule (Back-end)	Implemented
Name	Any	Required	Yes
		Characters only	Yes
		No numeric data	Yes
		Allows hyphens (-) and apostrophes (‘)	Yes
Email	textEmailAddress	Required	Yes
		Follow Email Pattern	Yes
Password	Any	Required	Yes
		6-150 characters	Yes
Age	Number	Required	Yes
		From 18 – 150	Yes
Postcode	Number	Required	Yes
		Exactly 4 characters	Yes
Gender	Dropdown Box	Required	Yes
Weight (kg)	Number	Required	Yes
		Maximum 3 characters	Yes
Height (cm)	Number	Required	Yes
		Maximum 3 characters	Yes

Household Number	Number	Required	Yes
		Match with Adult and Children Number	Yes
Adult Number	Number	Required	Yes
		At least 1	Yes
Children Number	Number	Required	Yes
Shopping Frequency	Number	Required	Yes

Welcome

File = activity_welcome.xml

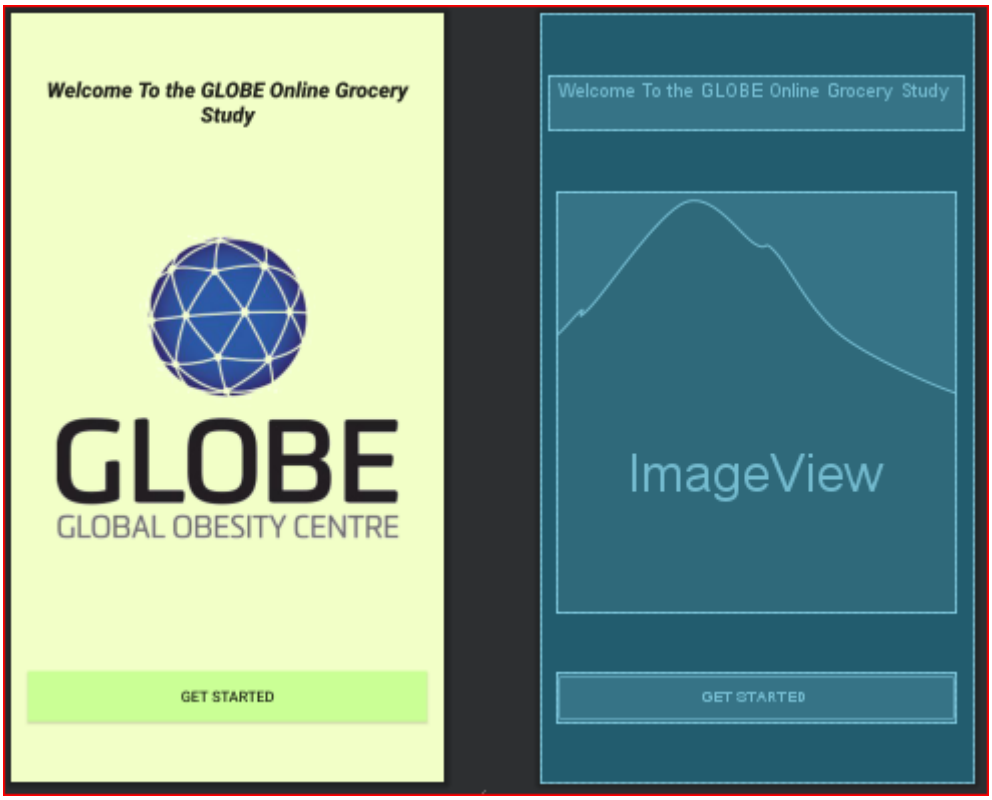


Figure 6: Welcome Page UI and Blueprint

Once the user has successfully created an account through **Sign Up**, this UI pops up, where the "Get Started" button direct the user to the **Home** page. This UI only pops up when a new user is created.

Home (Main)

File = activity_main.xml , activity_main.xml (land)

Figure 7: Home/Main Page UI and Blueprint (Portrait)

Figure 8: Home/Main Page UI and Blueprint (Landscape)

After successfully validating the login, the user is taken to this page. The **Home** page allows the user to access various facilities such as **Share your Purchases, Online Survey, About GLOBE, Feedback, FAQ** and **Receipts.** The user must be in the **Home** page in order to successfully logout should he/she decide to, by clicking the "Logout" button.

Share your Purchases

File = activity_add_receipt.xml

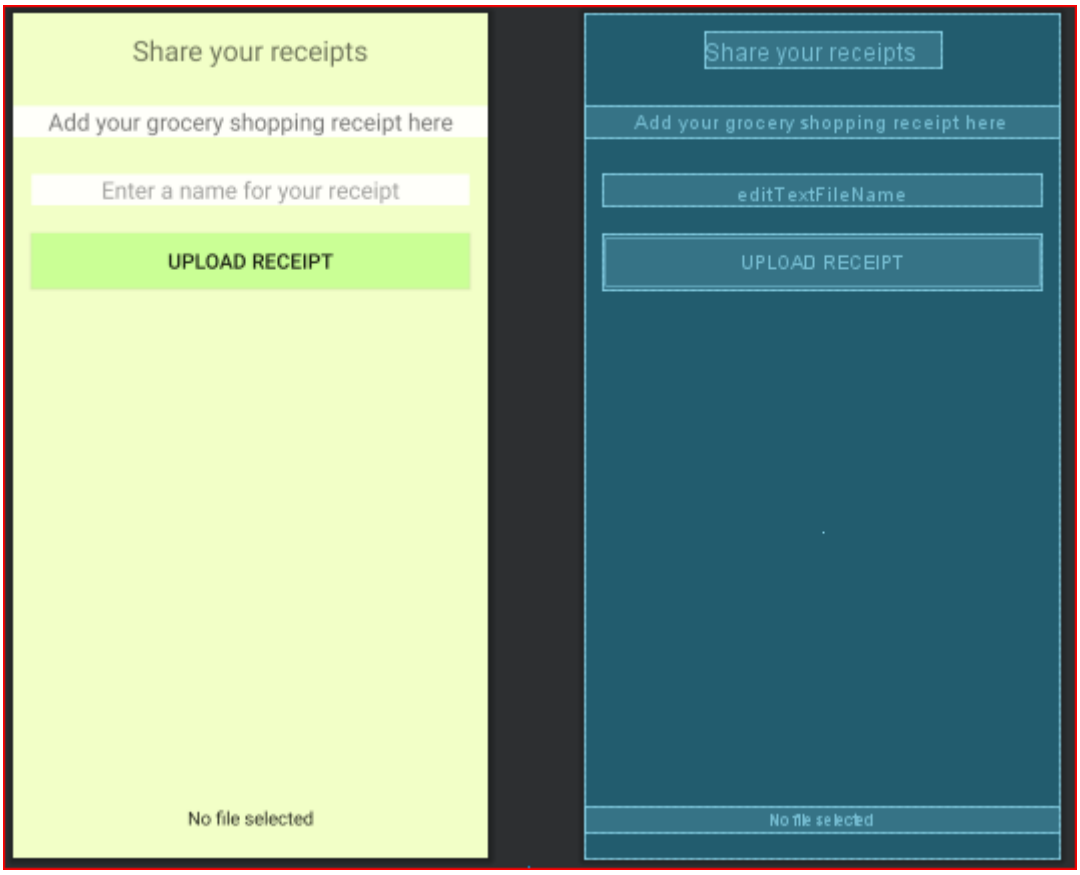


Figure 9: Share your Purchases Page UI and Blueprint

After giving an appropriate name for the receipt to be uploaded in the "Enter a name for your receipt" textbox, the user is required to click on the "Upload Receipt" button, which will take him/her to the Android Recents location. From there the user can navigate to where their receipt is and select it, which would then get uploaded automatically.

Online Survey

```
URL = https://globalobesity.com.au
```

Figure 10: GLOBE Website

User can view and learn about GLOBE and its functions, as well as take part in an online survey.

Feedback, FAQs and About GLOBE

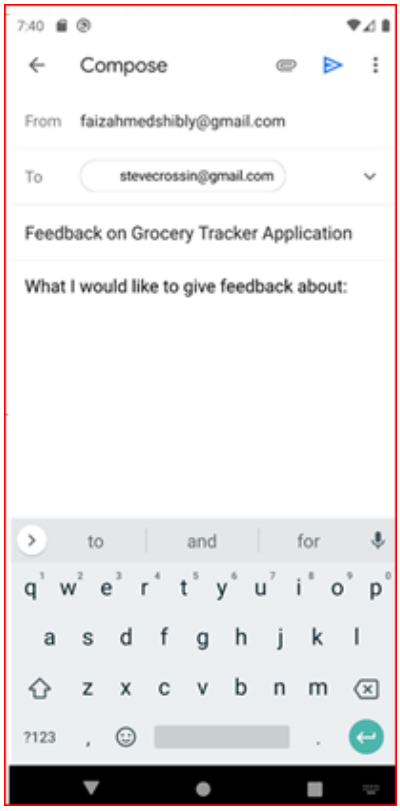


Figure 11: Provide Feedback Template

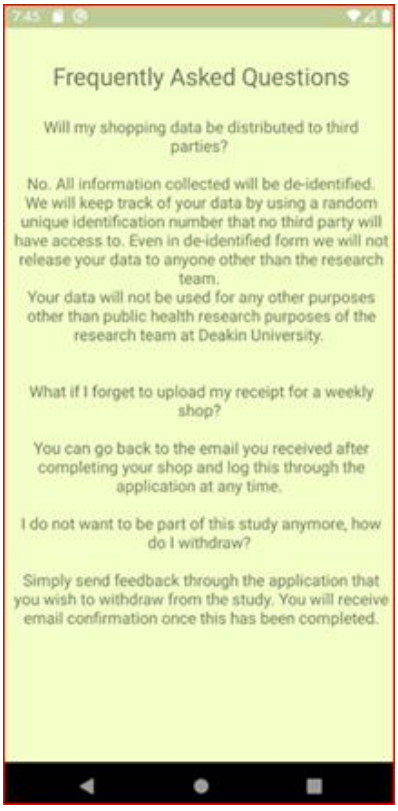


Figure 9: FAQs

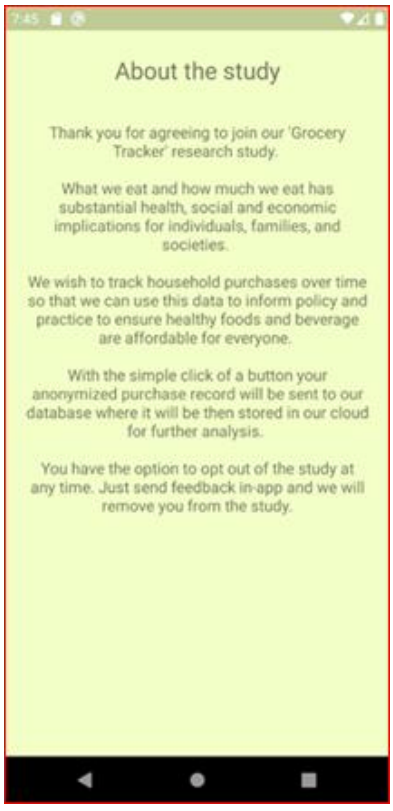


Figure 10: About GLOBE

The **Feedback** enables the user to provide their remarks and queried to the application development team.

FAQs deals with some common questions users may have about the application and its usage.

About GLOBE provides some insight on the study and details of data handling.

Application Java Classes

File = MainActivity.java

Imported Libraries and Modules ::

```
import android.annotation.SuppressLint;
import android.content.Intent;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.analytics.FirebaseAnalytics;
import com.google.firebase.inappmessaging.FirebaseInAppMessaging;
import com.stevecrossin.grocerytracker.R;
import com.stevecrossin.grocerytracker.database.AppDataRepo;
import com.stevecrossin.grocerytracker.entities.User;
```

The file proceeds to create a **Firebase** connection, the database with which the application communicates and stores data. At the same time, it renders the **activity_main.xml** layout file.

GoToReceipts() enables the user to access the **Share your Purchases** page, where they can upload their receipt.

GoToSurvey() is activated after the user clicks on the **Online Survey** button. This function opens the URL leading to the GLOBE website through the device's default browser, or after the user selects which installed browser to use.

GoToAbout() works through the **About** button. This function renders the **About** page layout.

GoToFAQ() works through the **FAQ** button. This function renders the **FAQ** page layout.

The **Receipts** button runs the **receipts()** function, which loads the **Receipts** layout, allowing the user to see the receipts he/she has loaded so far.

sendFeedbackMail() function is hosted on the **Feedback** button. This function opens the mail screen, filling details automatically with:

- receipient = **globedeakin@gmail.com**
- subject = **Feedback on Grocery Tracker Application**
- body = **What I would like to give feedback about** (user can add their comments into the body after this line)

isSignedIn() function is run by default whenever the user opens the application. As long as the user hasn't logged out of their application's previous session through the logout button, and the firebase database recognizes the user login status as *True*, even if the application is closed and restarted, the user will not have to login again.

navSignUp() function takes the user to the Sign Up page from the Login page. This is hosted on the **No account? Sign Up** button.

doInBackground() and **onPostExecute()** functions execute the manipulation of user provided email and password input to authenticate themselves. Initially the functions run validation functions to ensure proper input is provided, after which it connects to the database and retrieves the email provided. Once it confirms the email is present in the database, it proceed to use a ***PasswordScrambler*** class created to hash the password output and compare it with the hash matched to the username. Upon confirmation of the password, the account is authenticated, the database sets the user login status to *True*, and the user can proceed to the main page.

tryLogin() is the series of value validation that is run during the login process. This is hosted on the **Sign In** button.

submitSignUp() function is hosted on the **Create Account** button. This button runs a series on validation to confirm that proper data with types, data length and logical boundaries are provided. Once these validations are passed, the firebase database connection is established and the new user is added with the provided email as the identifier. However, should it detect a similar email is already present in the database, it will return an error, requiring the user to provide another email.

logout() function is hosted on the **Logout** button. The function updates the user value in the database as **False**, representing the user logging out of his session. It then takes the user to the **Login** page.

In addition, various validation functions to validate Sign Up and Login field entry details are are utilized to ensure that proper data is provided for the application to process and operate. Further details to explain the functions will be found in comments placed in the project code files.

Additional Functionality

Receipt View

Once the **onCreate()** function get executed the data from receipts and whenever the user gets to the receipt page, the function gets triggered and it fetches the data from firebase and passes the data to **Receipts View,** which shows the receipts the user has uploaded so far into the app.

Dynamic Notifications

Creates a background running notification channel which gets the context and the data from the notifications and it gets passed into the notification channel.