COMP2161 – Final Project
Garden Planner App

Evan Hauk
T00023763
August 3rd, 2022

Functional description

The Garden Planner app was created for amateur gardeners hoping to learn how to grow and harvest basic vegetables, and have a space to manage their hobby gardens and harvests. The app is free to the users to ensure there is no hesitation to download and try out the app. The app is locked to portrait only orientation, and some features require an internet connection to work.

There are 3 main features of the app which are split into different tabs. The main screen (home) is the dashboard which gives metrics on the users harvests, current planted vegetables, watering, etc. The 'Planning' tab utilizes an online database of 45 vegetables and allows the user to browse vegetables and their attributes to plan for their planting. The final tab is the 'Planting' tab which allows users to add up to 5 garden boxes and add the desired vegetables to the garden box.

Upon entry of the app, the user must review and scroll to the bottom of the welcome / terms and conditions of the app to accept. There is also a settings icon in the top right hand side which gives contact information and allows the user to display the welcome message again.

Market survey and comparison with similar apps

There are several similar garden planning apps out there that offer similar features to my Garden Planner. 'Armitage's Great Garden Plants' app is only a garden dictionary and gives the user only vegetable information. 'Kitchen Garden Planner at Gardener's Supply' offers an all-encompassing research and planning experience, but it is only available on the web, which is not ideal for bringing the app into the garden. 'Garden Planner at Territorial Seed' has a great app with one downfall, it has too many features for the amateur gardener, and it requires an annual subscription.

My Garden Planner app fits nicely into the missing niche found in the garden planning space. The app is for beginner to intermediate gardeners who want to keep track of small hobby gardens. The app provides a limited number of garden beds for the user to plant in, displays 45 common vegetables and the information needed to garden these vegetables.

Based on my user analysis, people looking for a beginner app for gardening won't want to pay for a professional version of the app. This allows my app to stand out as it is a free alternative for those looking to get into hobby gardening.

A simple pay structure that can be implemented in the future is a pro version with account verification and private payment method. As this app is being released to the google play store as a free app, it will have to remain free. However, in app purchases such as a pro

upgrade are possible. The pro upgrade would allow the use of unlimited garden beds, additional vegetables, and additional tracking features.

Code description

The code format for this app is to utilize a MainActivity that holds calls from a navigation menu and has a navigation view. Upon selection of navigation buttons, we use getSupportFragmentManager to replace and commit the new fragment. As we are using fragments for our main tabs, shared preferences are used to store critical information between tabs, and that should be stored to the user's phone.

An example item, and example adaptor classes are used to create the custom list objects displayed in the recycler view. This list holds custom example items as garden boxes. It will hold an image, the title, and list of vegetables. It also has a button for removing the item.

There are classes for a vegetableDataService and VegetableListReport which handle API calls and formatting and receiving of data from the online JSON database called by the app from the API. We use a MySingleton to only allow one call to the API at once as an AsyncTask.

Additional code is HTML drawable, layout and string files which are fairly straightforward.

Spec-sheet (technologies used)

- Bottom Button Navigation Bar with 3 Fragment Activities
- Volley 1.2.1 - for access to internet and making API requests
- Harvest Helper API database access
- JSON data conversion from database
- Bitmap graphics for Image download / display
- AsyncTask for querying API database in the background
- Volley Response Listener to ensure only one call to the API is allowed at once
- Shared Preferences to save user data on device and pass data between fragments

- RecyclerView with custom list items
- GSON 2.8.7 - JSON data conversion for passing custom list items to shared preferences

List of features

Welcome Message and Terms & Conditions

When the user enters the app for the first time, they are prompted by the terms and conditions of the app. These are in a scroll view and at the bottom, the user must agree to terms and click 'got it' before they will disappear. This does not allow users to use the dashboard until agreeing to conditions.

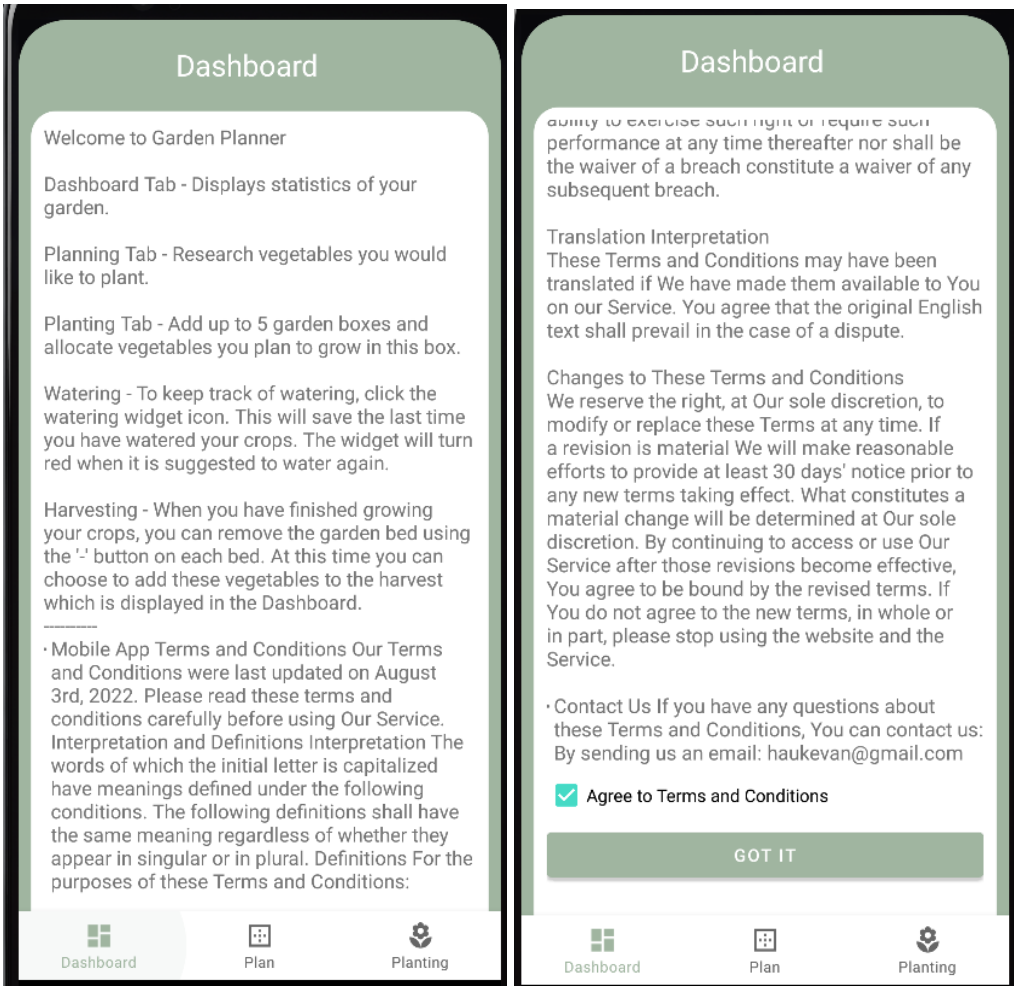

Figure 1: user agreement of terms and conditions

Dashboard

The Dashboard displays helpful garden metrics about number of crops, type of crops, watering intervals, favorite crop, and harvest information.

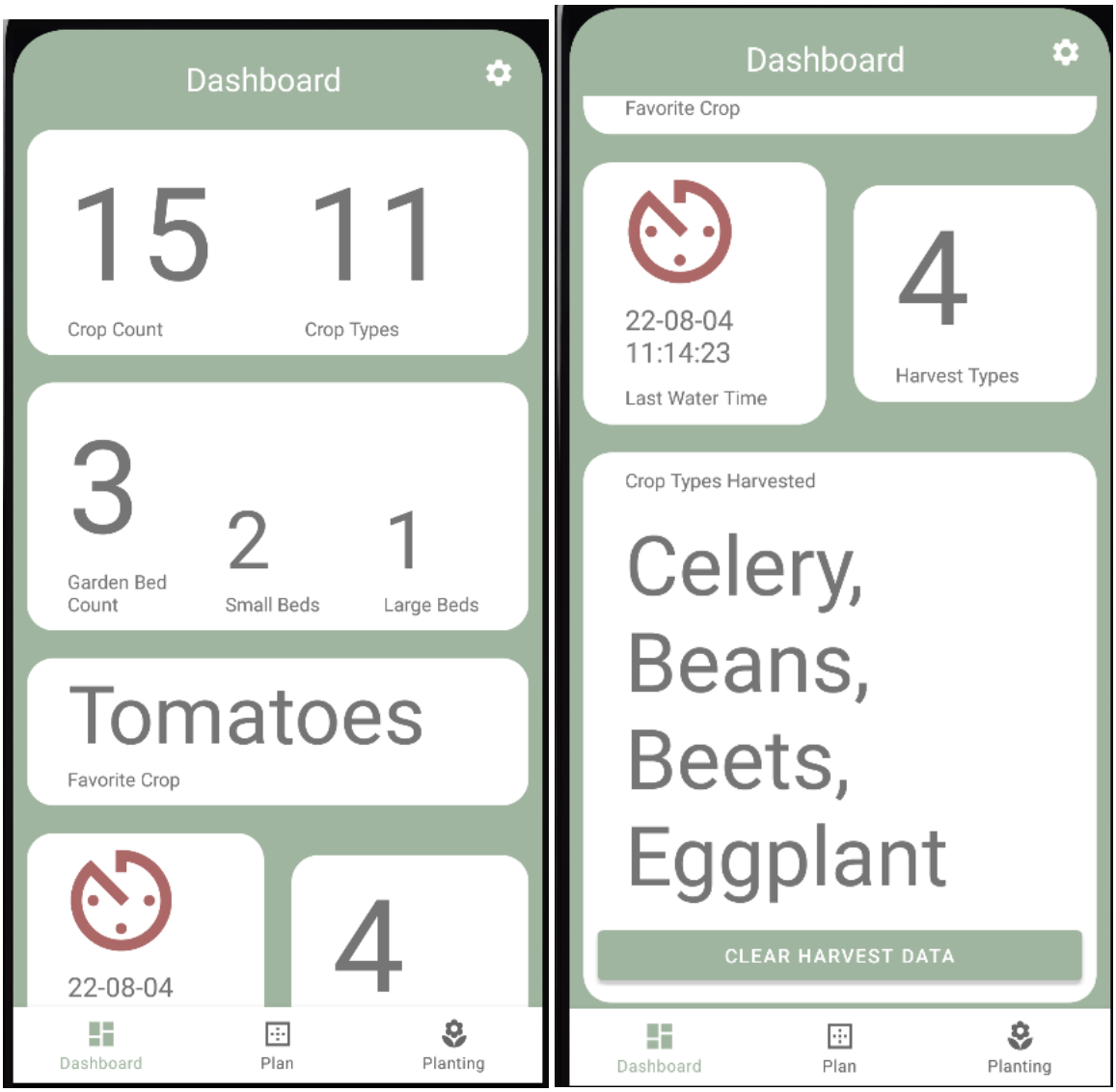

Figure 2: Dashboard View

Settings

Setting Screen simply allows the user to show the welcome and terms view again if they would like to review. The settings popup also gives contact information for bug fixes.

NOTE: The settings icon is only available to be clicked after the user initially agrees to the terms and conditions. This ensures the user cannot use the dashboard until they check and agree.

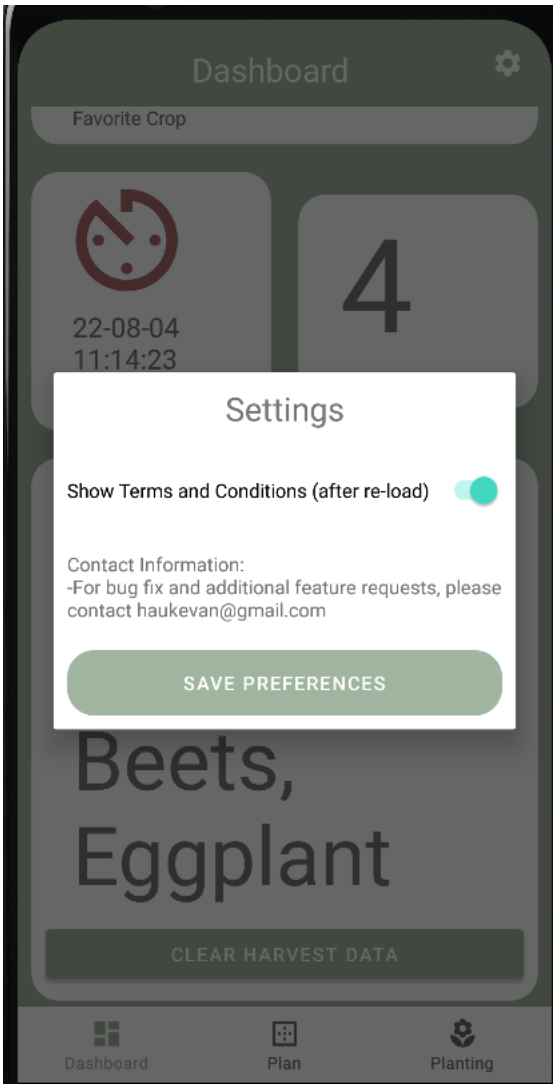

Figure 3: Settings Popup

Watering

An additional feature of the dashboard is it allows the user to set the time that they have last watered. The first watering occurs when the user enters the app for the first time. Upon clicking the timer symbol, the time will update to the current, and the symbol will turn green for some feedback on the press. If the user returns the dashboard prior to 24 hours since the last watering, the symbol will be grey. If the time is longer than 24 hours since last water, the symbol will turn red.

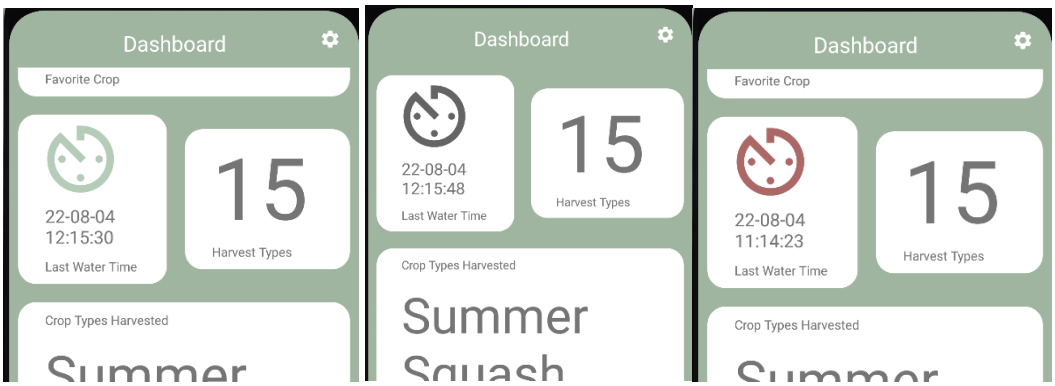

Figure 4: Crop Watering

Color Scheme

The colors used represent the outdoors and a green space. The app is light with white and a light faded green as the accent color. In the Planting tab, the user gets a gradient starting from a deep green (grass) and fading into a pale blue (sky). Each garden box added has a different color which allows them to stand out. They are all different shades of green.

```
<color name="card_color_one">#0B5748</color>
<color name="card_color_two">#197E5A</color>
<color name="card_color_three">#08673D</color>
<color name="card_color_four">#177138</color>
<color name="card_color_five">#2E6724</color>

<color name="app_green">#9EB5A1</color>
<color name="app_blue">#517988</color>
```

Figure 5: App Colors

Crop Planning Tab

This tab allows the user to get all the information they need on any crop. The crop list however, is limited to 45 crops from the current API updated database. The user is prompted with a list of available crops in the top left which can be scrolled through. Once selects a crop, the user will be displayed a photo of the vegetable, and a scroll view of the crop information.

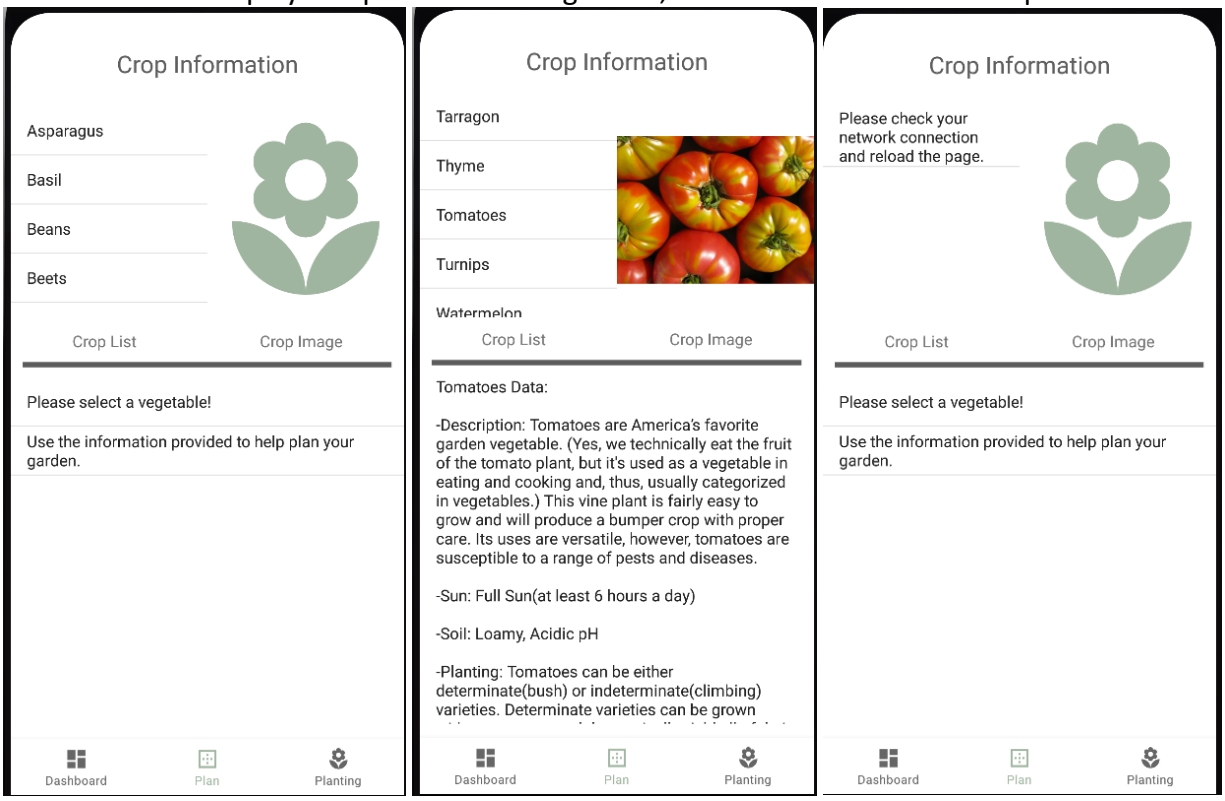

Figure 6: Planning Tab - Entry Screen, Vegetable Selected, No Network Connection Error

Planting Tab

This tab allows the user to add new garden boxes to the garden up to a maximum of 5. If the user tries to add more than 5, they will be prompted to upgrade to PRO which may be made available in the future.

A popup will display once the user clicks the '+' button to add a new box. The user must enter the name of the box to be allowed to add it. They can now set the box size and add crops. To add crops they must click an addition '+' button displaying a nested popup and a list of available crops. Once the user enters the information and saves the bed, it will be added to the list.

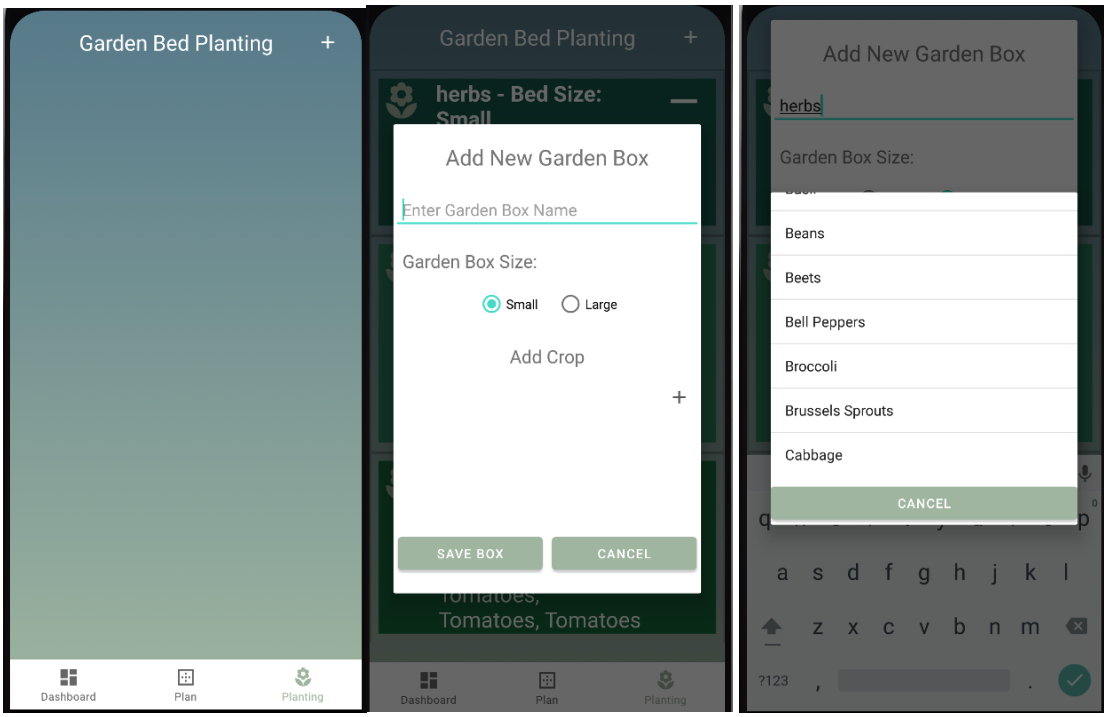

Figure 7: Planning Tab, Add New Box, Add Crops to Box

Once the user has finished growing their crop and are ready to harvest, the user may select to remove the box. Once they do so, they can remove the box, or harvest crops (which will also remove the box). The can also cancel at this point.

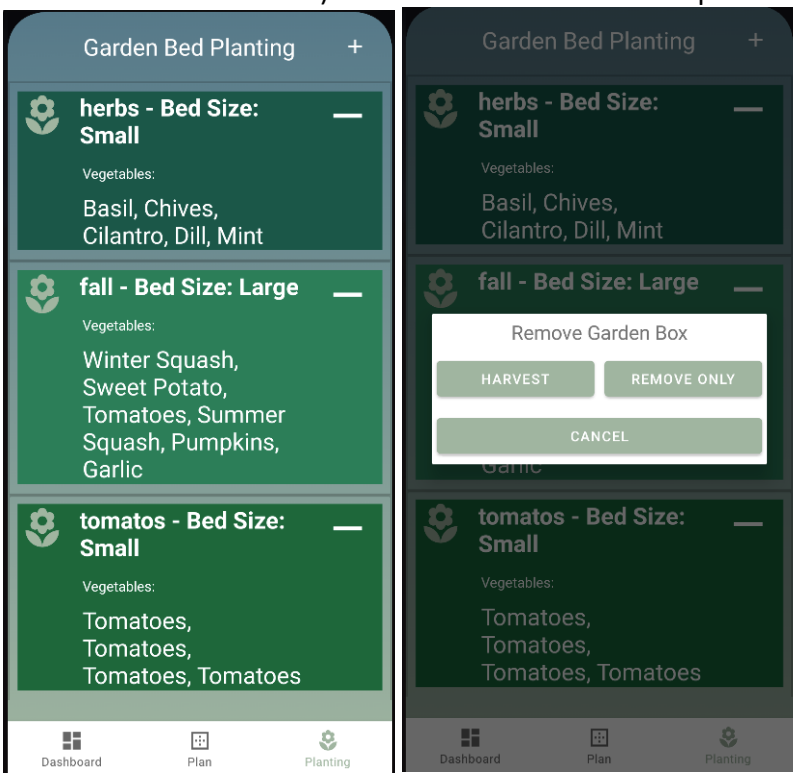

Figure 8: Planning Tab - Example Garden Boses, Prompt to remove or harvest crop box

Test cases

- Internet Connection Tests
    - If the user is not connected to the internet, they will get an message to check network connection when trying to access a list supplied by the API
    - There will also be a Toast notification saying 'something went wrong'.
- Shared Preferences
    - Data is saved in the onPause and onDestroy methods of the fragments. It was needed in the onPause to allow saving when the user switches fragments.
    - All data that should be accessed by other fragments or saved by the user is saved into shared preferences.
    - This test was completed by making all possible additions to data, closing the app, and re-opening to check state continuity.
- Add and remove buttons have been tested for edge cases.
    - Add buttons will only allow a maximum number of garden beds, and an unlimited number of vegetables per box.
        - This could be an issue, but for the time it takes for a user to add, it is unlikely that this would overload.
    - Removals have all been checked for attempting to try and remove a null item.
- Spam Button Clicks
    - All button clicks, specifically buttons which access data from API calls handle only single clicks. There is not way the user can request multiple calls at once with the user of the MySingleton Class which only allows one call to API at a time.
- Save buttons require minimum data entries to add data.
    - Crop names can be any characters so no issues if the user adds special characters or numbers
    - Only crop items selected from the list can be added, so there is no possibility for bad data being entered in the crop fields and used in dashboard calculations.
- User cannot select buttons beyond focus point of app, specifically when pop-ups occur. If the user selects outside of a popup, the popup up will close.

Performance evaluation

Upon review of the app, the performance of the worst class looks to only be O(n) with the use of a single For-loop. AsyncTask's are used to make API data calls so the forground of the app runs smoothly. The build of the app is continuously between 300 – 340ms on a virtual machine. No screen lag is detected while using. Depending on users internet access, the API calls are quite fast causing little to no lag to the user.

Justification of permissions

Terms and conditions of app use were taken from a boiler plate template found online. They were modified for Canada instead of the United States and additional user information was entered. My personal email address is currently used as the app company. This terms and conditions essentially covers all basis for the app. It was made sure to include a clause stating that the app may link external websites and the content pulled from those websites is not under the app's control.

Marketing strategy

The marketing strategy for this app was to create a simple gardening app for hobbyist individuals. The draw to this app versus the other apps on the market, is the simplicity of use of the app, the free price.

I believe this is a good structure for this app and as a userbase developed, an add space or pro version upgrade can supply revenue in the future.

Known bugs

1. Upon entering the app for the first time, and before agreeing to terms and conditions, the user can access the planning and planting tabs. They will have to agree to terms and conditions to use the dashboard.
2. Network Connection with virtual device changing WIFI:
    - App crashed while continuously trying to access the internet call to API when the app was opened on a virtual device, but the computer running the virtual device had changed wifi networks. This has something to do with the virtual device showing connection to network but the original network had changed.
    - Upon restarting Android Studio this problem fixes itself.
    - This is an unrealistic problem for users not working inside android studio.
    - When no internet is present, the call will return an error prompting the user to connect device to the internet.