PlantTracker

Jaremy Longley

# **Vision Statement**

For green thumbs young and old and any experience-level who want to easily track important details of their plants, the PlantTracker app is a mobile application designed to help users keep an eye on watering frequencies, soil conditions, specific requirements for plants, etc. PlantTracker gives the user piece-of-mind knowing that the routines for their plants remain consistent. PlantTracker will remind the user of upcoming waterings or any other specific needs that the user must remember. PlantTracker will also help the user determine how often to water their plants and what sort of climate the plants grow best in.

Identify Categories of Users

**Gardener –** These are users with basic app experience and gardening experience (or a desire to make gardening easier for themselves).

**User Stories**

* As a gardener I want an app that is easy to use that I can track my houseplants with.
* As a gardener I want an electronic way to keep track of all the plants I own and how often I water them, prune them, etc.

**Requirements**

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| Gardener | Have a list of plants they are taking care of |
|  | Keep track of the needs of their plants |
|  | Make a schedule of needs for their specific plant |
|  | Coordinate schedule of plants so they can align on similar days |
|  | Set reminders for the needs they must keep track of(such as watering, repotting, etc.) |
|  | Identify specific traits of the plants(such as shrub, flower, succulent, etc.) |
|  | Upload progress photos of their plants |
| System | Based on plant traits(such as shrub, flower, succulent, etc.) search json information for watering tips, potting tips, etc. |

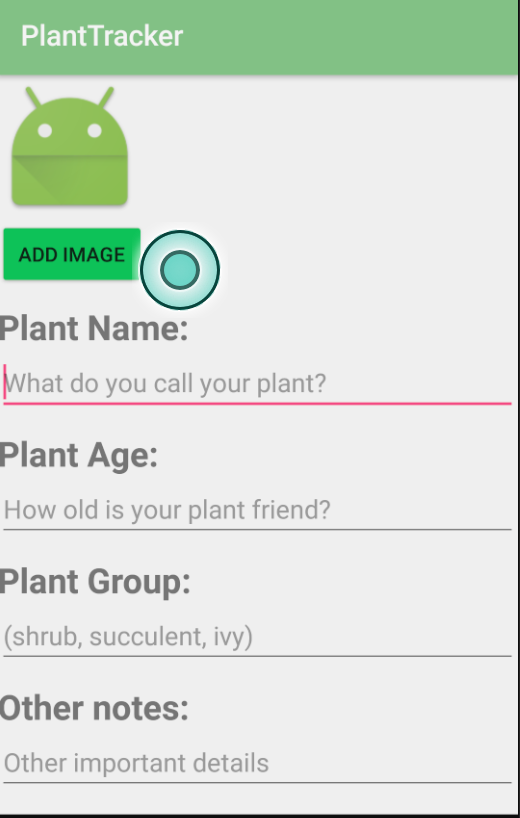
**Product Backlog**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Story ID** | **Story** | **Story Points** | **Priority** | **Status** |
| S1 | Allow user to create plant list and upload photos/plant information | 10 | 1 | Partially completed 9/23/2016 |
| S2 | As a user I would like to upload my photos and track my plant progress. I would also like to save my plant information that has been uploaded to the app. | 10 | 2 | Completed 10/8/17 |
| S3 | As a user I would like my plant’s progress to be displayed in a user-friendly manner so I can see how I have done with my plant. | 8 | 3 | Completed 10/22/17 |
| S4 | Sync the information to SQL lite in order to make it easier to transfer from activity to activity. | 15 | 4 | Completed  11/5/17 |
| S5 | As a user, I would like the apps User Interface to be very self-explanatory and easy to understand. | 10 | 5 | Completed  11/26/17 |
| S6 | From a design perspective, there needs to be checks to make sure the information is added correctly. | 15 | 6 |  |
| S7 | As a developer, I want the code to be clean and concise so that I can read through and update the app easily. | 10 | 8 |  |

**Sprint #1 Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S1 | Design plant table UI | 3 | 4 |
|  | Implement add button to add plant information | 1 | 1 |
|  | Code add button event handler | 2 | 1.5 |
|  | Implement add button to add photo | 1 | - |
|  | Code event handler for photo | 2 | - |
|  | Add uploaded information to database | 1 | - |

**Review**

****

**Retrospective**

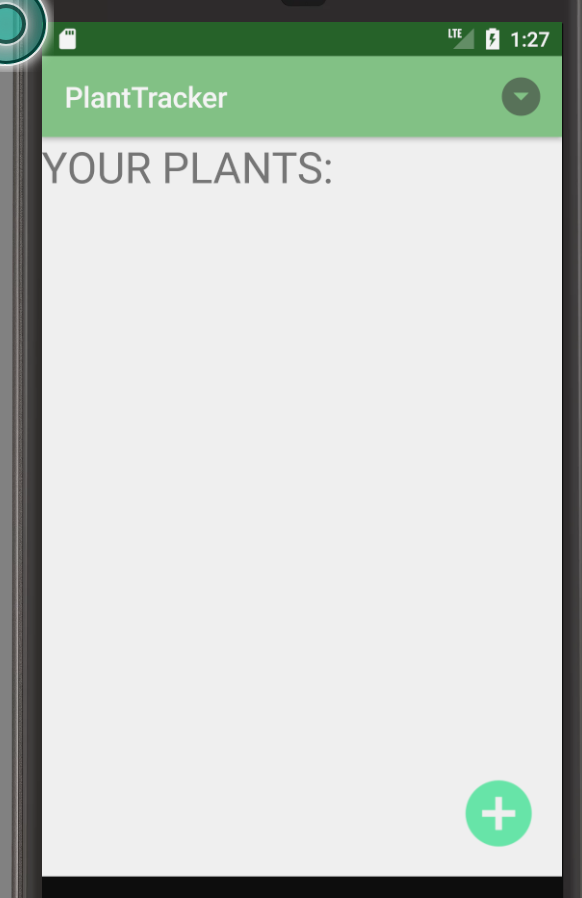
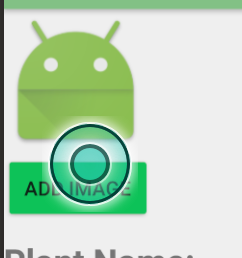
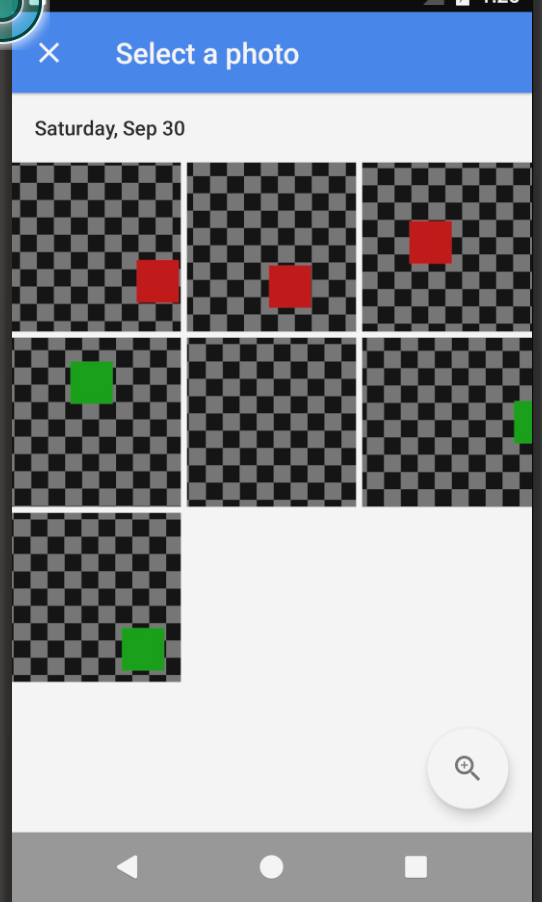
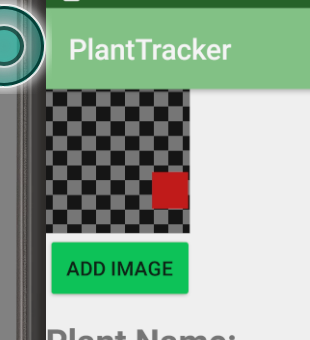
For this iteration, I planned to complete my first story-point. I am proud of the work I was able to accomplish, but was unable to finish all of the tasks that I set out to complete. Taking the time to understand Android Studio’s interface took a little bit longer than I previously thought. Also, I was unable to make time to figure out how to allow users to gain access to their Android camera and upload photos. I will push that task to the next iteration. I may scrap uploading information to a database unless I can find a solution that I can implement in the given period of time. I think for future stories, I may need to break them up into smaller stories because I lumped a lot of tasks into single stories.

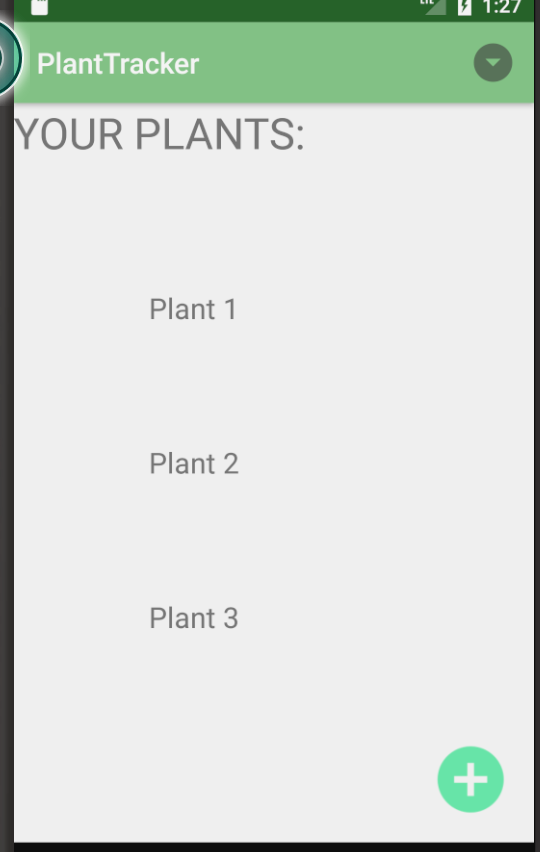
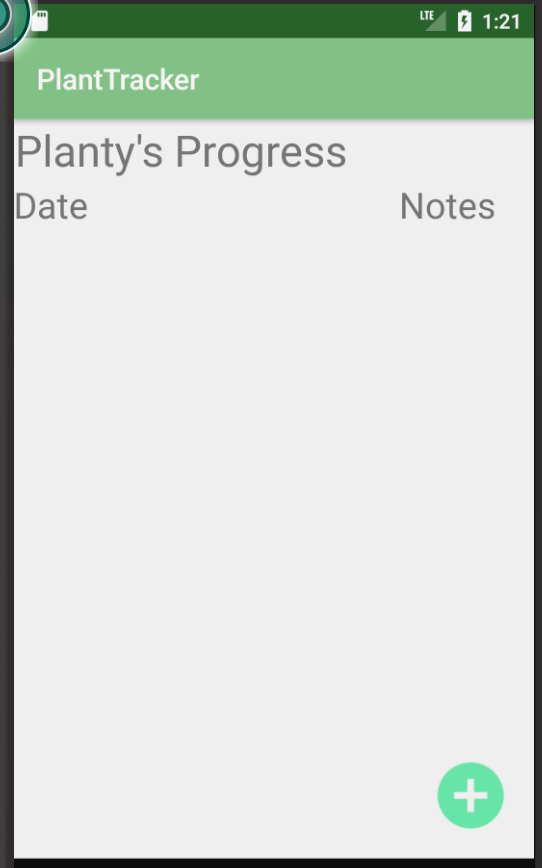
**Sprint Velocity: 10**

**Sprint #2 Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S1 (finish it) | Implement add button to add photo | 1 | 1 |
|  | Code event handler for photo | 1 | 1.25 |
| S2 | Save information user has uploaded | 2 | 1.5 |
|  | Pass the saved information back to the previous Activity | 2 | 2 |
|  | Display the passed information in a user-friendly interface | 2 | .25 |
|  | Create General UI for plant-progress tracker | 2 | 2.25 |
|  | When plant is clicked in main Activity, open up progress tracker | 1.5 | 1 |
|  |  |  |  |

**Review**

****

****

**Retrospective**

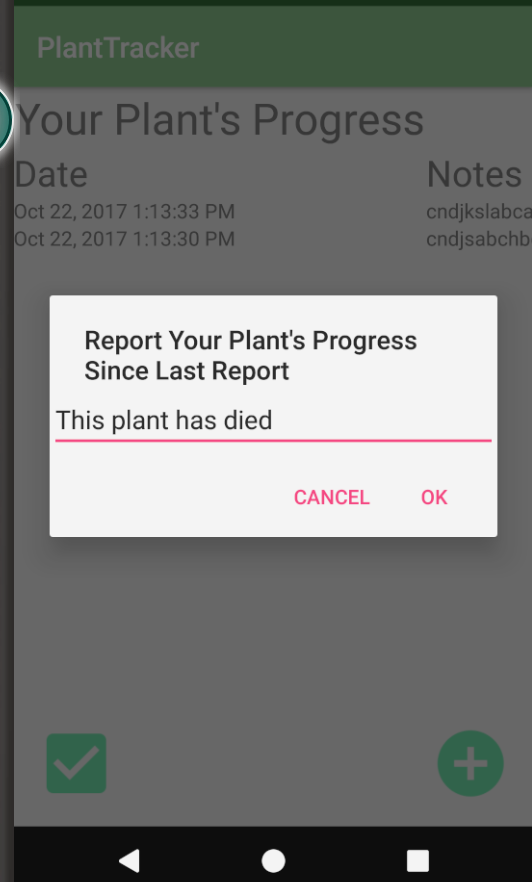
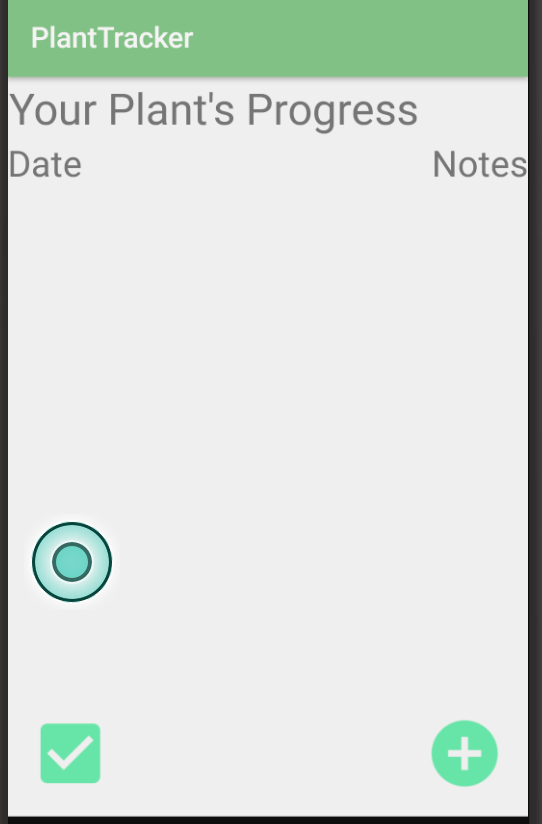
For this sprint I worked on finishing up Story Point 1 and working on story-point 2. There were some complications I ran into, but for the most part I was able to do everything I wanted to. I think I end up planning to do a lot more than I am actually able to accomplish, so I end up having to move some of the story points around. It is a lot harder to develop a UI for each individual activity than I think it is. I am also having trouble passing some of the information between activities (especially with passing the images between the different activities). One thing I was able to accomplish that isn’t on my plan is that I found a really easy way to utilize a simple database, so I will probably implement that later on. I have most of my UI’s done now, except for one, so I can really start displaying information and refactoring some of my code. I have come to find out that designing a good UI is really difficult to do, so I may have to create basic UI’s now and finishing adding more components later on when I get some UI advice.

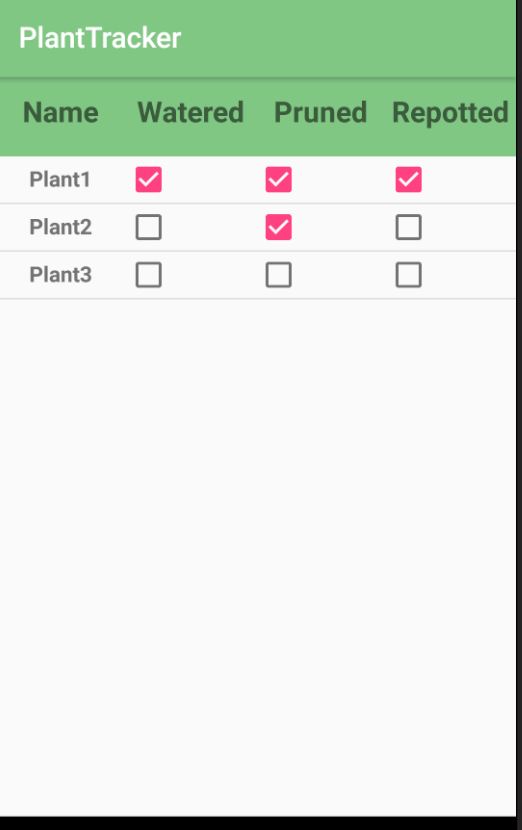
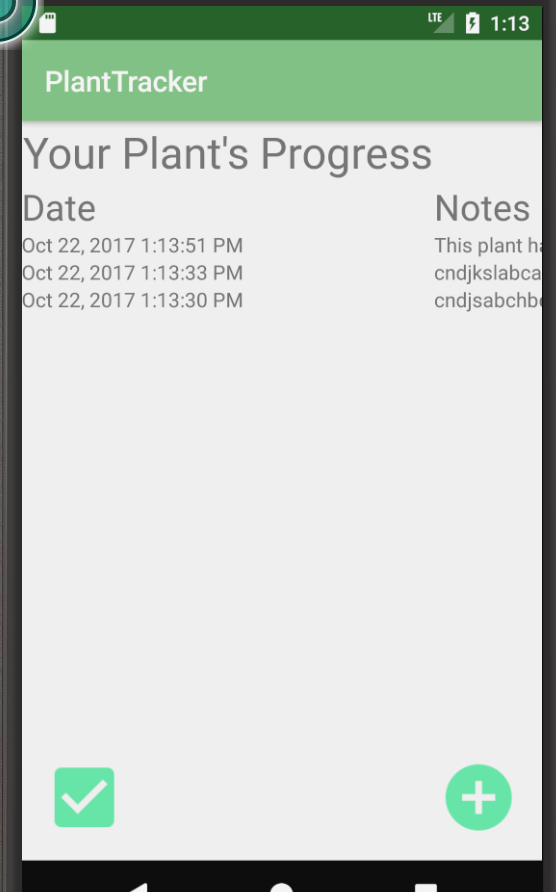
**Sprint Velocity: 10**

**Sprint #3 Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S3 | Display user’s plants(from main activity) in a user-friendly manner | 1 | 1.5 |
|  | Add another activity for making a checklist of last watered, last pruned, etc. | 1.5 | 1.25 |
|  | Give user ability to add progress photos and a brief description to the progress-tracker UI | 2 | 2.5 |
|  | Create UI for allowing user to mark when last-watered, last-pruned, etc. | 2 | 3 |
|  | Refactor Navigation | .5 | .5 |

**Review**

****

****

**Retrospective**

This iteration started off to a really good start. I quickly added a better User interface for displaying the plant’s in the main activity, easily implemented the ability for users to enter progress for each of their plants and was able to refactor some of the activity navigation. However, I ran into a snag when trying to implement the Checklist class and it’s associated activity. I had to find some online help to display information in a table-column style layout. I was running into a problem navigating to that activity and ended up deleting and rebuilding the entire class, only to find out that my problem was just simply not adding the activity to the Manifest.

**Sprint Velocity: 8**

**Sprint #4 Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S4 | Design a SQL lite database helper class | 1.5 | 2 |
|  | Add this helper class to my other classes | .5 | .5 |
|  | Add all of the plant information to the database in order to persist the information across all of the app and after app has closed | 2 | 5 |

**Review**

Most of this intent was backend, so no pictures for that. I created a SQL database class and pushed all of the plant info to the persistent database.

**Retrospective**

Writing the database helper class and figuring out how to retrieve database information took way longer than planned. However, I finally got it working but I had to push some of my stuff to the next Sprint. The data can now be added to the database class via a data access object Database Helper.

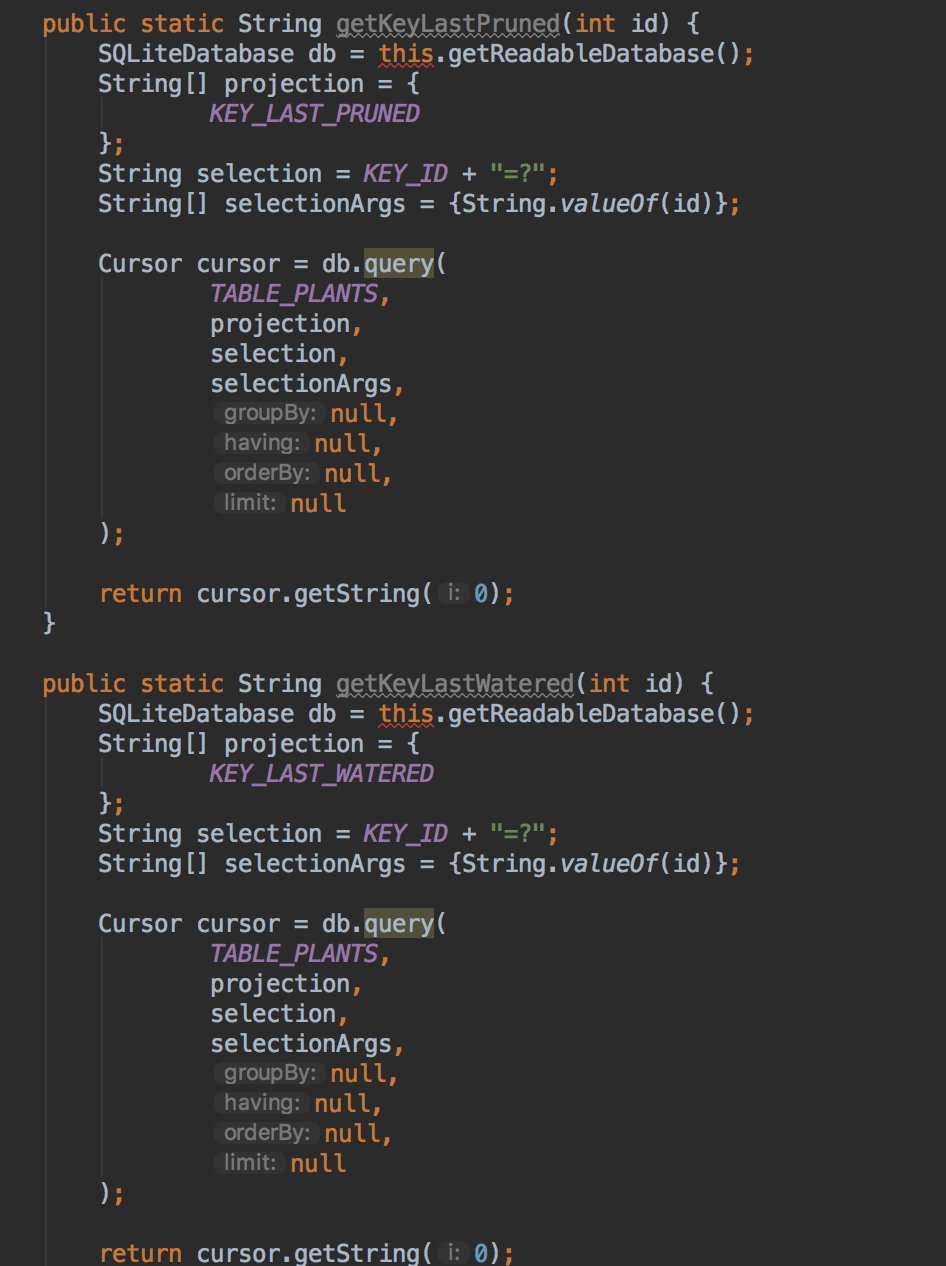
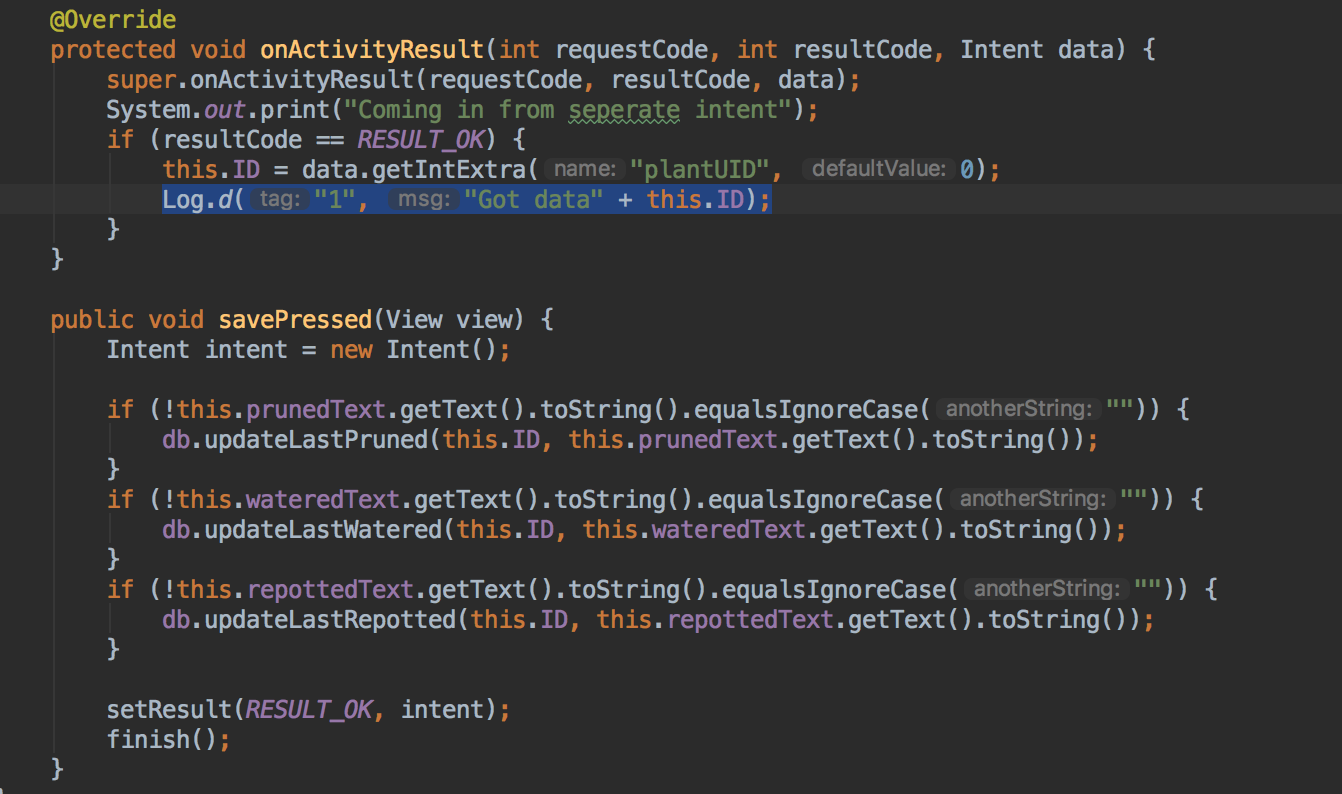
**Sprint Velocity: 15**

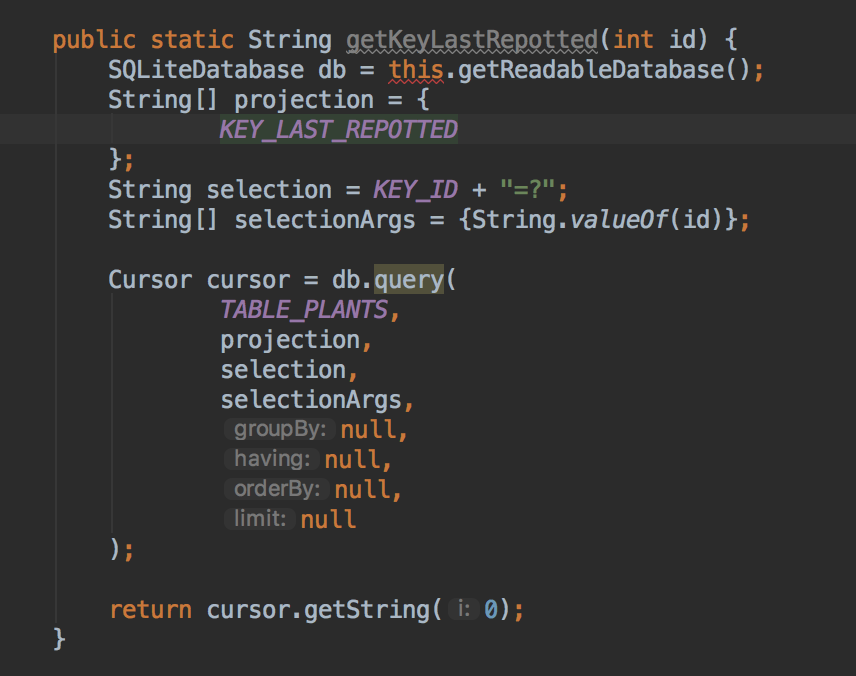
**Sprint #5 Plan**

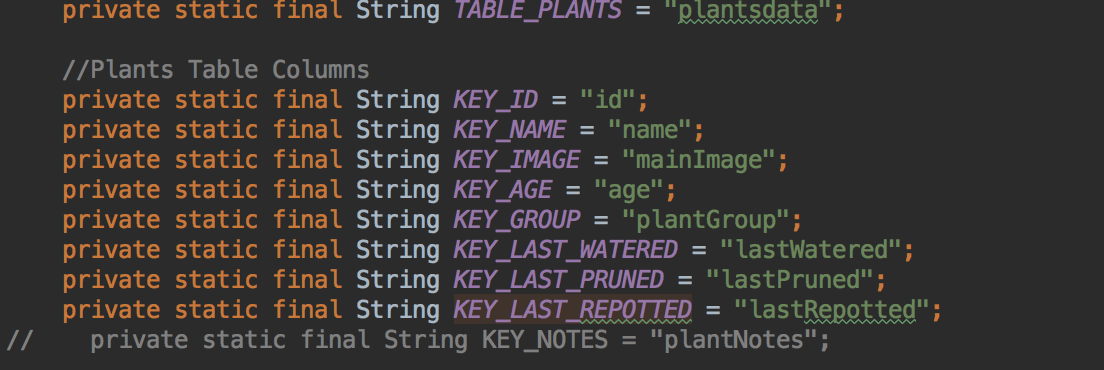
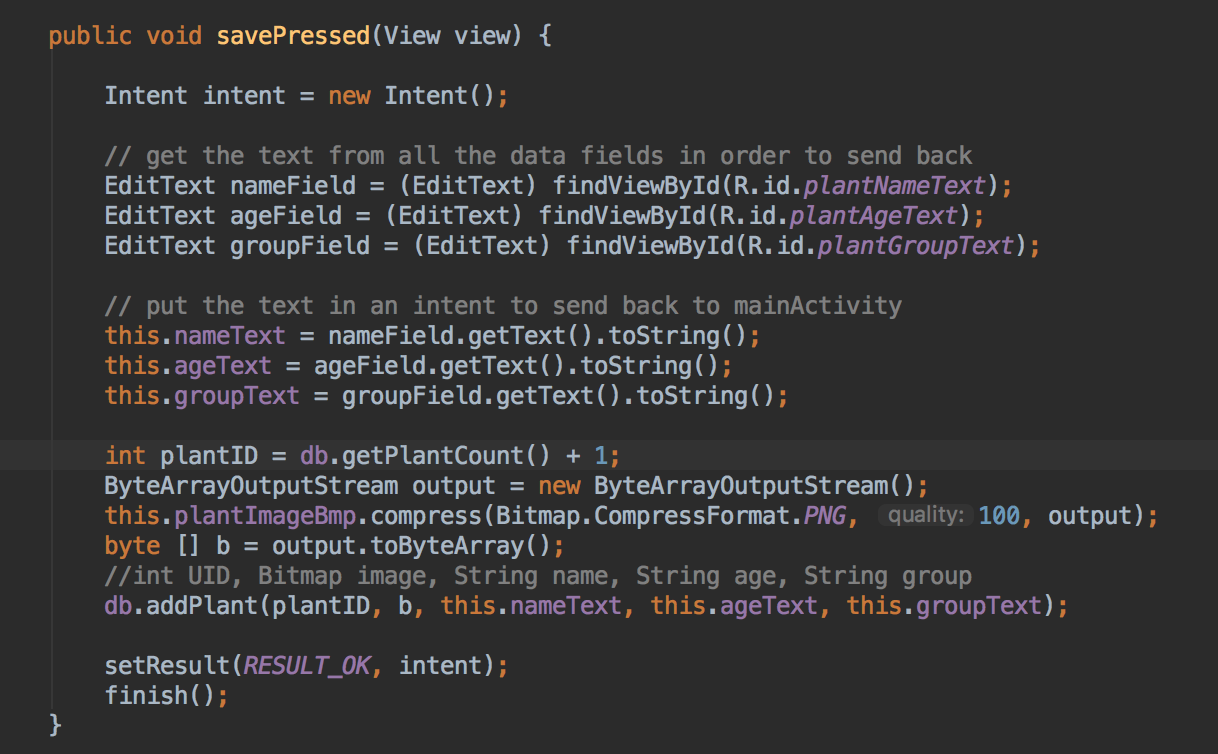
|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S5 | Consult outside sources for ease of app use | 1 | 1 |
|  | Add checklist to database | 1 | 1 |
|  | Add image into database and intents based on bitmap | 1 | 2 |
|  | Update app UI based on information from outside sources | 2 | 1 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

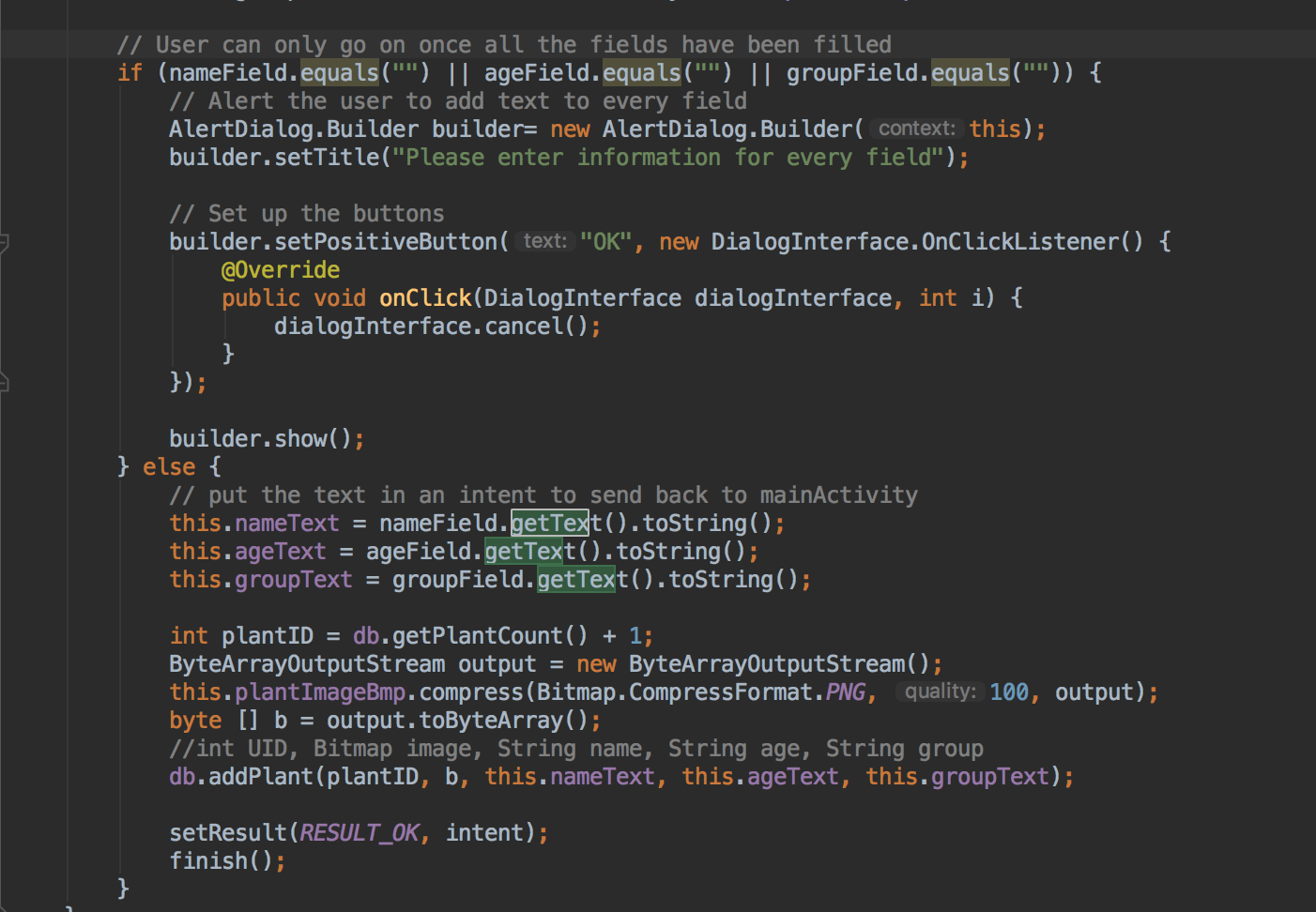
**Review**

I unfortunately wasn’t able to screenshot most of the app itself as my emulator is running extremely sluggish (so slow that I cannot even use it). Trying to diagnose that now. Here are some of the code changes that I made.

****

****

****

****

**Retrospective**

Still had to overcome some troubles with SQLite in Android (it is extremely confusing to use). I updated some of the database to be able to include bitmap representations of all the plant images so that I don’t have to pass around a plant URI path anymore. I’m in the process of changing the user interface for how the progress tracker and the checklist are set up. I made the checklist a simple text field because the multi-column checklist was getting a bit convoluted. I also cleaned up the code a bit. Eventually, the user will click on a plant from their main plant’s table, this will show a short page of all the information and then from there they can update the progress and some other information.

**Sprint Velocity: 10**

**Sprint #6 Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S6 | Add exceptions for false user input | 1 | 1 |
|  | Make sure that when data is entered, it is verified as correct | 1 | 1 |
|  | When user clicks on plant in main intent, display notes and information about the plant from the checklist | 2 |  |
|  | Throw errors and exceptions where necessary | 1 | 1 |