How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

GitHub Username: ben-8409

PlantBuddy

Description

PlantBuddy knows when your plants need watering and can remind you at the right time. It has a database of popular plants and their watering needs. You track the times you are watering the plants and PlantBuddy reminds you at the right time.

Intended User

This is an app for people who own some plants but don't know how to take care of them

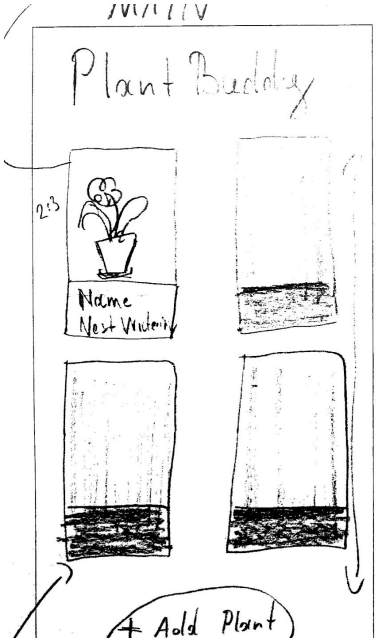
Features

List the main features of your app. For example:

- Has a list of plants and their watering needs (including a picture)
- Allows you to add plants from the db to your plant collection
- Allows you to track the watering
- Reminds you when it's time to water the plants

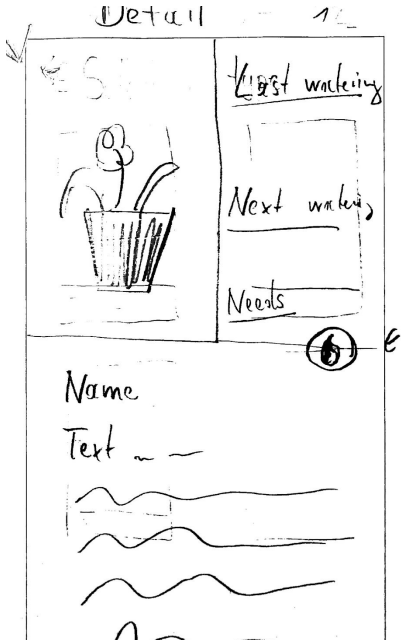
User Interface Mocks

Screen 1



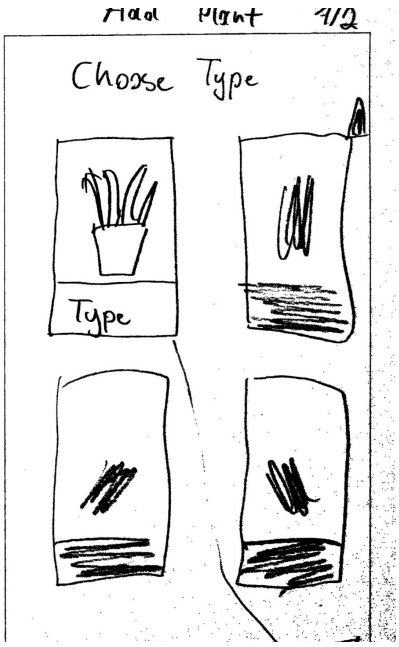
The main screen features a grid list of the plant the user tracks the watering for. Each plant is highlighted by a large image and small text below showing the name and nextg watering time. The watering times are color coded: greenish means this plant is good, orange means watering is due and red means overdue.
Tapping a plant, opens the plant overview (Screen 2). Tapping the Add Plant Fab opens the Add Plant View (Screen 3).

Screen 2



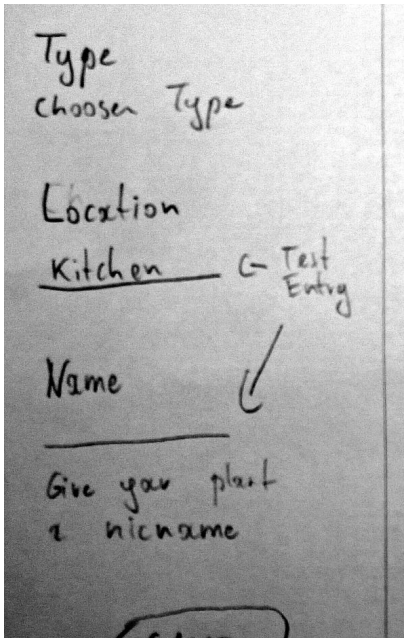
The plant detail screen can be divided in three areas: The upper left area holds a large image in ⅔ format. To the right of the image, there is a overview of the following times: last watering, next watering and general watering needs (weekly, monthly, daily). Below this area, to the right of a screen, a fab is placed. The fab is used to track a watering.
Below that, is a area reserved to information about the plant, which is sourced from Wikipedia.

Screen 3



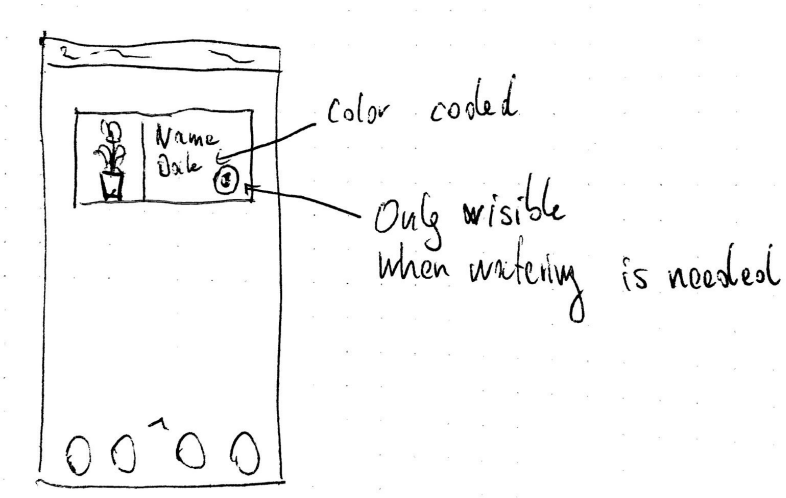
The add plant view contains a list of plants that can be added as tracked plants. The scrollable grid list features large cards that contain an image of the plant and some text. A plant is selected by tapping it, which leads the user to screen four.

Screen 4



The second screen of the add plant process allows to user to register a location and give a name for the newly added plant. When pressing the big save button, the plant is added to the tracked plants and the user returns to the main screen.

Widget



The app provides a widget. The widget displays the time until the next watering of a plant is necessary. As soon as the watering day is reached, a quick button to water the plant is displayed. Watering the plant updates the widget immediately.

Key Considerations

How will your app handle data persistence?

My app will handle data persistence using android architecture components, specifically Room, to save the data in a sqlite database on the device.

Describe any edge or corner cases in the UX.

When the user has no plants tracked, an empty screen is displayed nudging the user to add a plant.
Plants available to add are loaded from from the internet and cached. If there are no plants available, an error screen is displayed requesting the user to go online.

Describe any libraries you'll be using and share your reasoning for including them.

I plan to handle the Room Persistence Library for persistence since it is the new Google endorsed way to store data and handle the data persistence with a on device database.
I plan to use LiveData and ViewModel from Architecture Components to to handle the view data and simplify data lifecycle.
I plan to use Picasso for image loading to facilitate image loading since I have good experiences with this library from the course. I also plan to use okhttp for loading a JSON file with the plant database.
I will use the support library for AppCompatAcitivty and RecyclerViews.

Describe how you will implement Google Play Services or other external services.

I will use Google Mobile Maps to display a Banner ad in the MainActivity. I will use Google Location and Activity Services to get the location when adding a plant and to set up geofences.

General Technical Specification

The App is written solely in the Java Programming Language. Of all libraries, the current public stable version is used. At the time of writing, for the Google Support Libraries, that is version 27.1.0. Furthermore, the app is build with the current stable version of Android Studio (3.1) and the Gradle Plugin (3.1.3).
When using user visible strings, the strings.xml file is always used to keep the app translatable. The includes notifications and the widget.

Accessibility

The app implements the usual accessibility best practices, including d-pad support und content description. There is no audio only content. Make sure the app supports RTL layouts as well as LTR layouts (use START/END margins instead of LEFT/RIGHT, for example)

Version table

Adhere to the following application and library versions.

Library/Software	Version	Purpose
Android Studio	3.1.3	
Gradle/Gradle Plugin	4.4/**3.1.3**	Build tools
Compile/Min/Target SDK	27/21/27	Android SDK Version
Android Support Libs	27.1.1	AppCompat, RecyclerView, Support Libs
ConstrainedLayout	1.1.2	Responsive Layouts
JUnit	4.12	Unit Testing
Espresso / Espresso Runner	3.0.2/1.0.2	Integration Testing
Google Location and Activity Services	15.0.1	Geofencing
Google Mobile Ads	15.0.1	Ad Revenue with Banner Ad
Android Lifecycle Architecture Components – ViewModel and LiveData Room	1.1.1	Persistence, Lifecycle Aware Data binding

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

Task 1: Project Setup

Start a new Android App using the Android Studio Wizard.

- Use Lollipop (API Level 20) as min SDK Level. Use Android Support Libraries to get the AppCompactActivity.
- Add the Android Architecture Component libs for the Room persistence. https://developer.android.com/topic/libraries/architecture/adding-components
- Add the Google AdMobs and Firebase Messaging Libraries
- Draft the persistence database table

Task 2: Implement UI for Each Activity and Fragment

Implement the ui for the described views

- Implement the MainActivity
- Add a RecyclerView the MainActivity and choose a GridLayoutManager to display the tracked plants
- Tapping a plant should open the PlantDetailActivity
- Add a FAB to the MainActivity add a plant to the list of tracked plants. The FAB should open the ListPlantActivity
- Add the ListPlantActivity
- Add a RecyclerView to the ListPlantActivity and use the GridLayoutManager to display the list plant types.
- Tapping a plant type should open the AddPlantActivity
- Implement the AddPlantActivity
- Add two text inputs to allow the user to give the plant a location and a name.
- Add a button to save the plant. On saving, the plant should be persisted to the database and the user should move to the MainActivity.
- Show a toast when successfully saving the plant.

Task 3: Draft the data structures

Two types of data need to be persisted: Tabular data (sqlite) for plant types and tracked plants, images for the plants.

- Sketch out the table layouts for the tabular data. Use Room for ORM.
- Research a good way to save the pictures on device.

Task 4: Gather the data, prepare a JSON file

Plant types, names, pictures and watering needs need to available for the app to download and update. Prepare a JSON file including all plants and links to image resources.

- Research latin name, a royality free picture, a short description and the watering needs for 10 popular plants.
- Add the data to a JSON file you host on the internet.

- Document the JSON structure

Task 5: Download the data and parse the JSON

Use an adequate library to download the data from a server and parse the JSON. Generate Room entities (defined above) and save them on device.

- Implement a background task to download JSON and images
- Implement a parser to turn the JSON into objects
- Persist the plant types library

Task 6: Put it together

- Implement a background task to download plant types, parse the json and persist the data.
- Make sure the data is downloaded when empty

Task 7: Implement notifications

Use best practices to schedule and display a notification when a plant needs watering. Use geofencing to only notify when the user is at home

- Set up GeoFencing
- Schedule the notifications
- Implement three actions on the notification
    - Open the plants detail view
    - Water the plant
    - Ignore

Task 8: Keep it up to date

Whenever a push notification is received, the plant types database is updated. Additionally a JobDispatcher or a SyncAdapter is used to update the app data in background at regular intervals.