
GitHub Username: **AndreaWolff**
https://github.com/AndreaWolff

Let There Be Life

Description

Let There Be Life was created for all your plant growing needs. Let There Be Life will assist you with keeping track of what plants you have growing and information about each plant. Let There Be Life is also a great app for an upcoming hydroponic plant growing enthusiast who needs to keep track of the plants they have planted.

Intended User

The intended users for Let There Be Life are plant growing enthusiasts, including hydroponics.

Features

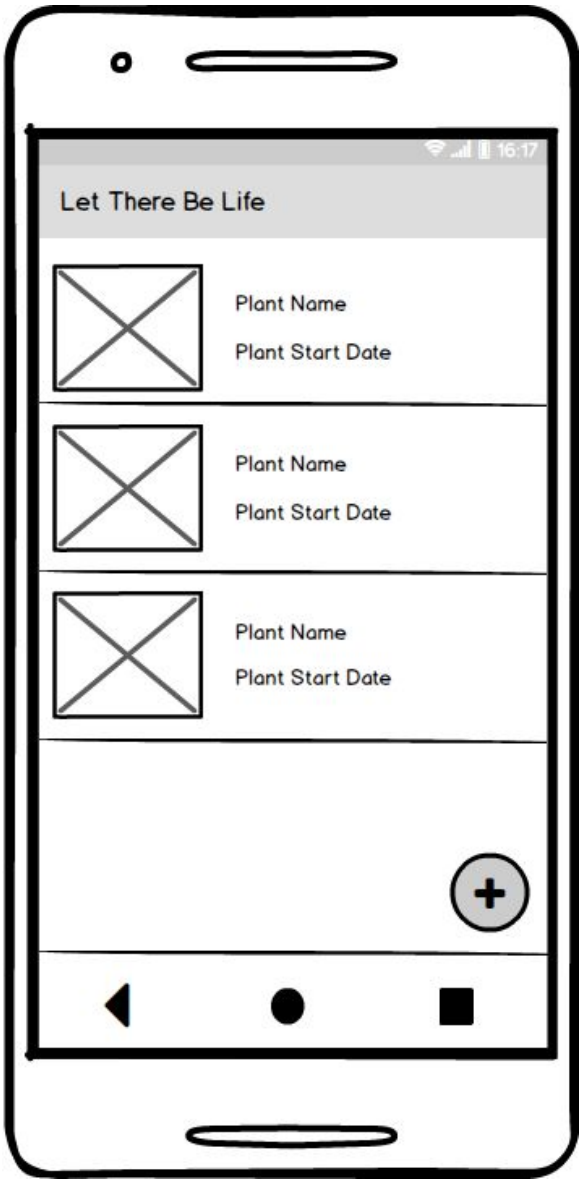
List the main features or Minimum Viable Product (MVP) of Let There Be Life app:

- Displays a list of Plants
- Saves new Plants
- Displays selected Plant details including:
    - Plant name
    - Plant seed date
    - Plant description
    - Plant image (optional)
- Allows the user to share Plant Details
- Queries online web service to retrieve Plant information (if the Plant name is on the server)
- Homescreen Widget displays your Plants
- Firebase Crashlytics to ensure quality, crash-free performance for the user
- Screen designs include:
    - Phones
    - Tablets (Optional)
        - Portrait
        - Landscape
- This app will be solely written in the Java Programming Language

User Interface Mocks

Below are the proposed mock design for Let There Be Life.

Screen 1 - Main Activity



The Main Activity is the launcher screen in the Let There Be Life app.

Main Activity functionality:

- Displays a list of Plants from a Firebase Real-Time Database
- Allows the user to create a New Plant via a floating action button
- Allows the user to select a Plant to view the Plant Details

Each item in the Plant list includes:

- Plant name
- Seed date
- Plant Image (Optional)

Each item is selectable, and when selected will navigate to the Plant Details screen.

The Floating Action Button on this screen, when selected navigates the user to the New Plant screen.

Screen 2 - New Plant Activity

The New Plant Activity is shown after being selected on the Main Activity screen.

New Plant Activity functionality includes:

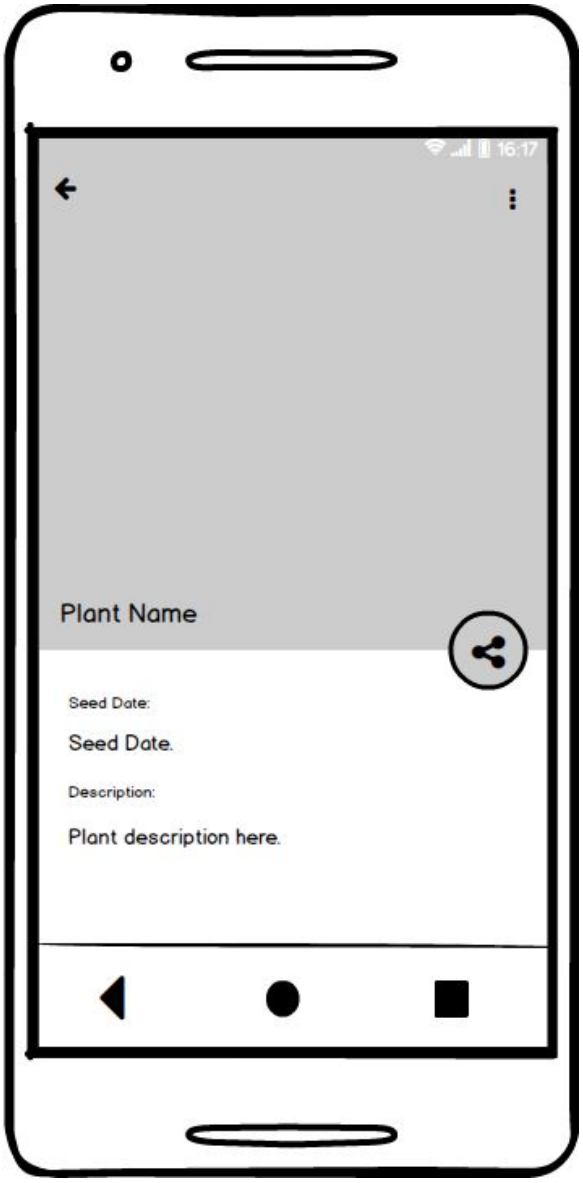
- Displays a form for the user to fill out to create a new plant
- Displays a Floating Action Button to take a picture of the new Plant
- Displays a SAVE button to save the newly created plant in the Firebase Real-Time Database
- Allows the user to navigate back to the previous screen

Each field in the form is an EditText that includes a Hint. When all these fields are filled in, the SAVE button will enable; which allows the user to save this newly created plant into the Firebase Real-Time Database.

The Floating Action Button allows the user to take a picture of their newly created plant to be saved in the database. This uses a camera intent service to launch the camera app.

The user is also able to use the Home Is Up button or the Android back button to navigate to the previous screen.

Screen 3 - Plant Details Activity



The Plant Details Activity is shown after selecting a Plant on the Main Activity screen.

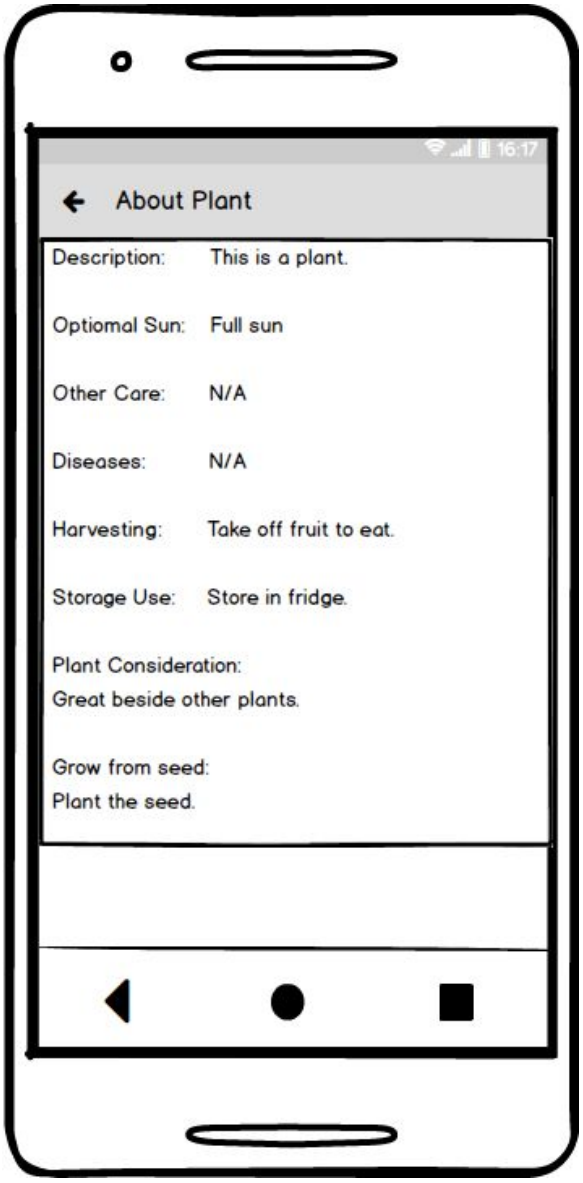
Plant Details Activity functionality:

- Displays the selected Plant Details
- Displays a Floating Action Button to share Plant Details
- Allows the user to navigate back to the previous screen

The Menu Item is only display for Plant names that match the plant information name from the web service. If there is no match, the Menu Item will not be visible.

The Floating Action Button allows the user to share Plant Details via an intent.

Screen 4 - Plant Information Activity

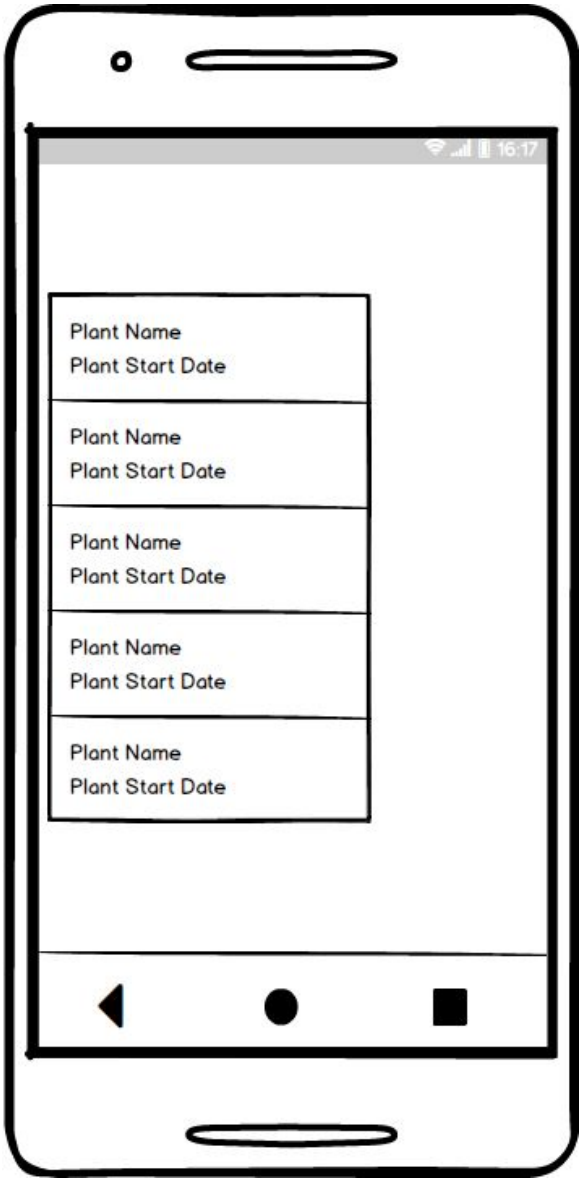


The Plant Information Activity is shown after the Menu Item is selected on the Plant Details Activity screen.

Plant Information Activity functionality:

- Displays Plant Information retrieve from a Room Database via a ViewModel
- Allows the user to navigate back to the previous screen

Screen 5 - Homescreen Plant Widget



The Homescreen Plant Widget can be used on the main screen of your device.

Plant Widget functionality:

- Displays the Plant name and Seed date
- On selection on a Plant item it will take the user to the Plant List screen

Key Considerations

How will your app handle data persistence?

- Firebase Real-Time Database
    - This database contains all the Plants the user has created, including:
        - Plant name
        - Plant seed date
        - Plant description
        - Plant image (Optional)
    - This Plant information is inserted in the New Plant Activity, and is retrieved in the Main Activity to display a list of Plants
    - The information is inserted and queried using the Firebase Real-Time Database framework and configurations
- Room Database
    - This database contains the contents of an online web service, found at *http://harvesthelper.herokuapp.com/api/v1/plants?api_key=YourApiKey*
    - A free API Key can be obtained via *http://harvesthelper.herokuapp.com/developers*
    - The information is inserted and queried using the Room frameworks and configurations as well as RxJava 2 for Room
- Share Preferences
    - This is used to retrieve a list of Plants and display on the Homescreen Widget

Describe any edge or corner cases in the UX.

- Let There Be Life app navigation:
    - Main Activity
        - Floating Action Button
            - Launches the New Plant Activity
        - Plant list item
            - On selecting a list item, launches the Plant Details Activity
    - New Plant Activity
        - Home Is Up or Android back button
            - Navigates back to Main Activity
        - Floating Action Button
            - Adds Plant image via a camera intent
        - SAVE menu item
            - Saves plant details into Firebase Real-Time Database
    - Plant Details Activity
        - Home Is Up or Android back button

                - Navigates back to Main Activity
            - Floating Action Button
                - Shares Plant Details
            - Collapsed menu (Optional visibility)
                - About Plant, launches Plant Information Activity
    - Plant Information Activity
        - Home Is Up or Android back button
            - Navigates back to Plant Details Activity

**Each screens restores after a configuration change.*

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

- Android Support Libraries
    - This library gives us access to such things as:
        - RecyclerViews - used to display a Plant list
        - Constraint Layouts - used to create new xml layouts
        - Card View - used to display a Plant list
        - Design - used for Floating Action Buttons
- Butterknife
    - This library reducing the amount of boilerplate code within an Activity
- Dagger 2
    - This library is used for app dependency injection, which is useful for code testability
- Retrofit, OkHttp, RxJava2 and Gson
    - This library is used to help with network calls to a web service
    - The RxJava2 Library allows us to make network calls on the background thread and observe it on the main thread. This allows us to build functionality to inform the user when a call has failed
- Glide
    - This library is used to configure and display images in Let There Be Life
- Stetho
    - This library is used to be observe network traffic during the use of the app. This allows us to view our calls to the server, what information we are passing in, and what information it retrieved
    - This library is great for debugging
- Firebase
    - Firebase Real-Time Database
        - This library is used to store a Plant, including its:
            - Name
            - Seed date
            - Description
            - Image (Optional)

    - Firebase Crashlytics
        - This library allows us to observe crashes that have occured while using Let There Be Life, and be able to look into fixes for these crashes.
        - This library uses the Firebase Console and shows a lot of good information about each user session
- Android Architecture
    - Room Database and RxJava 2
        - This library is used as an on device database which contains all the Plant Information which was pulled from a web service using RxJava 2
    - LiveData and ViewModel
        - This library is used to access the frameworks for ViewModels and LiveData
- Mockito (Optional)
    - This library is used to mock objects for testability purposes

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

Describe which Services you will use and how.

- Firebase
    - Realtime database
        - This service is used to store a Plant including its:
            - Name
            - Seed date
            - Description
            - Image (Optional)
        - This information is then inserted and queried using the Firebase Real-Time Database framework
    - Crashlytics
        - This service is used to observe crashes that occur when using Let There Be Life
        - This allows us to understand the crash a bit, and come up with a fix to ensure that the user has a crash free experience

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

- Create ‘Let There Be Life’ Android project
    - API 21 - Android 5.0 - Lollipop
- Create ‘Let There Be Life’ repository on GitHub
    - This repository will not include the google-service.json folder required for Firebase or the web service API key (this will be included in a zip folder for the Udacity reviewer)

Task 2: Setup Firebase

- Create a Firebase Project, including:
    - Real Time Database
    - Crashlytics
- Configuring google-services.json file

Task 3: Configure gradle dependencies

- Configure 3rd party libraries that will be used in the ‘Let There Be Life’
    - There can all be found under the Library section of this proposal

Task 4: App Setup

- Set up Dagger 2
- Create skeleton, basic navigation and mock data to display on the screen, for:
    - Main Activity
    - Details Activity
    - New Plant Activity
    - Plant Information Activity
- Implement Model-View-Presenter (MVP) framework and Model-View-ViewModel (MVVM)

Task 5: Implement UI for Main Activity

- Create Main Fragment
- Create Fragment Layout
- Move Activity code into Fragment

Task 6: Implement data for Main Activity

- Create Plant POJO
- Create Adapter for RecyclerView

- Connect Firebase Real-Time Database to RecyclerView

Task 7: Implement Room Database

- Create Room Database
- Create DAO
- Create Plant Information POJO

Task 8: Implement Job Intent Service

- Create Job Intent Service in Manifest
- Create Job Intent Service class
- Connect Plant Information web service
- Bulk insert Plant Information into Room Database

Task 9: Implement UI for Details Activity

- Create Details Fragment
- Create Fragment Layout
- Move Activity code into Fragment

Task 10: Implement data for Details Activity

- Create Menu (only to display if there is Plant Information)
- Display Plant Details
- Query data from Room Database to check whether Plant Information will be shown

Task 11: Implement UI for New Plant Activity

- Create New Plant Fragment
- Create Fragment Layout
- Move Activity code into Fragment

Task 12: Implement data for New Plant Activity

- Connect Firebase Real-Time Database to save data

Task 11: Implement UI for Plant Information Activity

- Setup Plant Information View Model/Factory
- Query Plant Information by id from the Room Database
- Display Plant Information

Task 12: Implement UI for Plant Widget

- Create Widget via the Widget Wizard
- Populate Widget with Plant name and Start date (from the Plant menu Firebase Real-Time Database)

Task 13: Implement Error Handling

- Implement error handling for:
    - RxJava 2 failure

    - Firebase Real-Time Database failures
    - Room Database RxJava 2 failures

Task 14: Clean up UI

- Implement transitions
- Clean up all layout designs

Task 15: Clean up code

- Clean up all String resources
- Clean up all content descriptions
- Clean up all dimensions

Task 16: Implement Tablet layouts (Optional)

- Implements tablet layouts for:
    - Main Activity
    - New Plant Activity
    - Plant Details Activity
    - Plant Information Activity

Task 17: Automation (Espresso) Testing (Optional)

- Create a Customer journey through the app

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"