Healthy Leaves

Sprint 3 Assignment

Presented to:

Professor Wahab Hamou-Lhaj

Project GitHub Link

Daniel Savin	40010051
Karl Noory	40059592
Thomas Tran	40095654
Vicentiu-Cristian Badea	40027683
Jun Young Kim	40063176

Abstract **-** Owning houseplants has become a ubiquitous practice particularly for the millennial demographic. However, one of the problems that arises from this practice is poor handling of houseplants typically because of lack of knowledge and dedication. To further the dilemma of lack of knowledge, different plants require different care routines which may seem overwhelming for some who possess many houseplants.

To circumvent this common issue, we are developing an android application that will centralize all this data and have it readily available for houseplant owners. The application will be connected to a sensor via Wi-Fi that can measure the temperature, moisture, and light exposure of the specific plant. This information would then be sent to the user via notification on the application where it will then prompt the user into various activities (ex: watering, changing plant location, etc.) to reinvigorate the health of said plant.

Table of Contents

Table of Figures 3

Table of Tables 3

1. Introduction 6

1.1 Product 6

1.2 Functionality 6

1.3 Benefits and Goals 6

1.4 Potential Users 6

2. Requirements 7

2.1 Sprint Backlog 9

3. Design Document 18

3.1 Android Application Wireframes 18

3.2 System Architecture 21

3.3 Hardware Architecture 22

3.4 Software Architecture 26

3.5 Use Cases and Sequence Diagrams 29

3.5.1 Use Case 1 30

3.5.1 Use Case 2 31

3.5.1 Use Case 3 32

3.5.1 Use Case 4 33

4. Testing 34

4.1 Test Plan 1: Firebase Connection 34

4.2 Test Plan 2: Plant Profile and Display 35

4.3 Test Plan 3: Database 38

4.4 Test Plan 4: Hardware 43

4.5 Test Plan 5: Notifications 47

5. Definition of Done 51

Table of Figures

Figure 1: Healthy Leaves wireframe user experience. 18
Figure 2: Activity hierarchy for HealthyLeaves project. 20
Figure 3: System Architecture of Firebase authentication. 21
Figure 4: Hardware architecture. 22
Figure 5: External Hardware Design. 23
Figure 6: Outer Shell. 24
Figure 7: Main MC - SparkFun ESP32 Thing. 24
Figure 8: Sensor shield for microcontroller. 25
Figure 9: 3.7V, 1Ah LiPo Battery (DTP603450). 25
Figure 10: Software architecture of Healthy Leaves project. 26
Figure 11: Entity relationship diagram (ER-diagram) of Healthy Leaves project. 27
Figure 12: Database diagram of Healthy Leaves project. 29
Figure 13: Use case 1. 30
Figure 14: Use case 2. 31
Figure 15: Use case 3. 32
Figure 16: Use case 4. 33

Table of Tables

Table 1: Backlog 7
Table 2: Sprint 1. 10
Table 3: Sprint 2. 13
Table 4: Sprint 3. 16
Table 5: S-10.1 Test Case 34
Table 6: S-10.2 Test Case 34

Table 7: S-10.3 Test Case. 35
Table 8: S-10.7 Test Case. 35
Table 9: S-14.1 Test Case. 36
Table 10: S-14.2 Test Case. 36
Table 11: S-14.3 Test Case. 36
Table 12: S-14.4 Test Case. 37
Table 13: S-16.1 Test Case. 38
Table 14: S-13.2 Test Case. 39
Table 15: S-16.3 Test Case. 39
Table 16: S-16.4 Test Case. 40
Table 17: S-16.5 Test Case. 40
Table 18: S-16.6 Test Case. 41
Table 19: S-16.7 Test Case. 41
Table 20: S-16.8 Test Case. 42
Table 21: S-16.9 Test Case. 42
Table 22: S-16.10 Test Case. 43
Table 23: C-1.1 Test Case. 43
Table 24: C-1.2 Test Case. 44
Table 25: C-1.3 Test Case. 44
Table 26: C-1.4 Test Case. 45
Table 27: C-1.5 Test Case. 45
Table 28: C-1.6 Test Case. 46
Table 29: C-1.7 Test Case. 46
Table 30: C-1.8 Test Case. 47
Table 31: S-1.1 Test Case 48
Table 32: S-1.2 Test Case 48
Table 33: S-1.3 Test Case 49
Table 34: S-1.4 Test Case 49
Table 35: S-1.5 Test Case 50
Table 36: S-1.6 Test Case 50
Table 37: Definition of Done for A-3. 51

Table 38: Definition of Done for A-8. 51
Table 39: Definition of Done for A-9. 52
Table 40: Definition of Done for A-10. 52
Table 41: Definition of Done for C-1. 53
Table 42: Definition of Done for C-2. 54
Table 43: Definition of Done for C-3. 54
Table 44: Definition of Done for E-4. 55
Table 45: Definition of Done for E-5. 55
Table 46: Definition of Done for S-1. 56
Table 47: Definition of Done for S-11. 56
Table 48: Definition of Done for S-15. 57
Table 49: Definition of Done for S-16. 58
Table 50: Definition of Done for S-20. 59

1. Introduction

1.1 Product

Soil-insertable, battery powered device that collects data parameters consisting of light exposure (% from full sunlight), moisture (referenced between air and water) , and temperature (in Celsius) and sends it to users' Android phone, providing notifications when said parameters are out of comfortable range for the plant.

1.2 Functionality

Collect information on plants and represent the data in an elegant fashion to the user, as well as provide useful plant care notifications. User may either set preferences from the database of plants online or add their own preferences to the database.

1.3 Benefits and Goals

The benefits of owning our product is that users can optimize the time they spend with their plants by providing optimal care. The goal is to get more people interested in plant ownership by making it fun and easy for everyone.

1.4 Potential Users

The potential users are mainly millennials as they are more likely to be interested in house plant ownership. Additionally, any person who owns plants may be interested in our solution.

2. Requirements

Our project requirements are presented in our backlog. The legend for abbreviations and color code is attached. The backlog is sorted by sprints.

Table 1: Backlog.

Legend	
A	Admin
C	Computer
E	Electrical
S	Software
Done	
WIP	

Stor	Story Title	Card	Poi	Spr	Priority	Status	Conversation	Confirmation	Notes
S-10	Firebase Connection	As an engineer I want to set up connection so that I can receive data streams via WiFi on my phone.	3	1	Must	Done	We should be able to connect to Firebase from an Android plantform to get data that is logged in there.	1. Can we authenticate in the Firebase? 2. Is the app connected to Firebase?	
S-14	Plant Profiles	As a user I want to create and access various plant profiles so that I can differentiate between different plants.	4	1	Must	Done	User should be able to differentiate their plants through different profiles.	1. Can the user create a plant profile? 2. Can the user save a name in a plant profile? 3. Can the user access plant profiles? 4. Can the user the the list of plant profiles?	
A-1	Sprint 2 Plan	As a project owner I want to create Sprint 2 Plan so that the delivrables are documented and everyone got their task.	4	1	Must	Done	We are graded on the Sprint 2 plan so this task is pretty obvious. To make sure project goes forward, everyone must know what to do.	1. Does each team member knows what to do in Sprint 2? 2. Is the sprint 2 document complete?	
A-2	Obtain Module	As an engineer i want to have a microcontroller module with WiFi so that I can design a working prototype.	1	1	Must	Done	We need a microcontroller to perform data analysis of sensor data as well as a WiFi module to send that data to Firebase.	1. Do we have a microcontroller that is working? 2. Do we have WiFi module with Firebase libraries available?	
A-4	Sprint 1 Testing Doc	As an engineer I want to have a testing document so that the requirement satisfaction is verified and the possible bugs are detected.	5	1	Must	Done	We need to make sure that every feature that is planned and executed is working well. If something doesn't work, it is not complete. If the app crashed something is wrong. Therefore we try to break the app and its features so we can fix them. This is to make sure that the customer gets the best experience.	1. Can we verify all the requirements? 2. Can we break down the app somehow?	
A-5	Sprint 1 Design Doc	As a project owner I want to have a design document so that the project description is available in a readable format.	5	1	Must	Done	The design document is an important description of the whole project. It has all the relevant information in a readable format (compared to backlog, for example).	1. Does the design document contain all the infomation about the project? 2. Is it submitted?	
C-1	Moisture Level	As an engineer I want to know the plant moisture levels (i.e. open Firebase and see values) so that I can tell the user when to water the plant.	4	2	Must	Done	Moisture level of the soil is one of the most important things for a plant. Too much moisture leads to mold and rot development; too little leads to withering.	1. Can the information from the moisture sensor be read? 2. Is the information sensible? 3. Is the information sent to Firebase?	
S-1	Moisture Notification	As a User I want to receive a notification when the soil is too dry/wet so that I water/don't water my plant to keep it healthy.	4	2	Must	Done	To make sure that the plant stays healthy, the user must water it properly. When the moisture is too low, the user shall be notified.	1. Does the user get a notification to water a plant when the moisture is too low? 2. Does the user get a notification when the soil is too moist?	
S-15	Plant Research	As a Engineer I want to research plants so that I can create a database with reference values.	4	2	Must	Done	We need to reseach various plant types and their needs to create a database. In addition we should look into making our own database or take data from someone else.	1. Do we have a knowledge of most popular potted plants? 2. Do we have quantifiable knowledge of moisture, light levels, and temperature range for each type?	
A-3	Obtain Moisture Sensor	As a Engineer I want to have a moisture sensor so that I can obtain moisture data for the prototype.	1	2	Must	Done	We need to obtain a moisture sensor to get soil moisture data.	1. Do we have a sensor that is compatible with the microcontroller? 2. Can we obtain sensible data from it?	
A-8	Sprint 3 Plan	As a Project Owner I want to create Sprint 3 Plan so that the delivrables are documented and everyone got their task.	4	2	Must	Done	We are graded on the Sprint 3 plan so this task is pretty obvious. To make sure project goes forward, everyone must know what to do.	1. Does each team member knows what to do in Sprint 3? 2. Is the sprint 3 document complete?	
A-9	Sprint 2 Testing Doc	As a Project Owner I want to have a testing document so that the requirement satisfaction is verified and the possible bugs are detected.	5	2	Must	Done	We need to make sure that every feature that is planned and executed is working well. If something doesn't work, it is not complete. If the app crashed something is wrong. Therefore we try to break the app and its features so we can fix them. This is to make sure that the customer gets the best experience.	1. Can we verify all the requirements? 2. Can we break down the app somehow?	
A-10	Sprint 2 Design Doc	As a Project Owner I want to have a design document so that the project description is available in a readable format.	5	2	Must	Done	The design document is an important description of the whole project. It has all the relevant information in a readable format (compared to backlog, for example).	1. Does the design document contain all the infomation about the project? 2. Is it submitted?	
S-20	Main Folder Refactoring	As a Engineer I want to have proper naming of the project folders and classes so that the project doesn't bug and my teammates can read the code	2	2	Must	Done	Refactoring project directory, because the naming convention is confusing and causes bugs and app crashes.	1. Is the project naming okay and the app compile? 2. Can team members understand the purpose of all classes?	

E-4	Device Size	As a User I want to have a device that does not occupy more place than a normal plant pot so that the overall design of my living space is not affected.	2	2	Must	Done	The device should be as small as possible so the aesthetics of the plant are put forward.	1. Does the device contribute to the aesthetics of the plant-pot enseble in a positive way?	
C-2	Light Level	As a Engineer I want to know the plant light levels so that I can tell the user whether the plant has enough light.	4	2	Should	Done	Light exposure is also an important metric for plants, since it is required for photosynthesis. However, too much light can kill certain plants.	1. Can the information from the light sensor be read? 2. Is the information sensible? 3. Is the information sent to Firebase?	
C-3	Temperature Level	As a Engineer I want to know the plant temperature levels so that I can tell the user whether the environment is too cold or hot for the plant.	4	2	Should	Done	Temperature levels are important, because different plants grow in different regions. Generally plants live in a certain temperature range, depending on their genus.	1. Can the information from the temperature sensor be read? 2. Is the information sensible? 3. Is the information sent to Firebase?	
E-2	Charging	As a User I want to charge my device so that it keeps working.	5	2	Must	To do	The system uses power to work, and therefore should be recharged. The user should be able to charge the device.	1. Can the user plug in a charging cable in the device? 2. Does the battery get charged?	Charger built in the SparkFun ESP32 Thing.
S-11	Firebase: Read Moisture	As a Engineer I want to read data concerning moisture sensor so that I can verify if the value is too low or too high.	2	3	Must	WIP	Once connected to Firebase, the app should obtain data values for moisture.	1. Can the app obtain the data values for moisture form the Firebase? 2. Can the data values be logged in the app?	
S-16	Database	As a Engineer I want to create multiple tables of various plant types so that I can have references for notifications.	10	3	Must	WIP	Once data is known, we should decide how to implement the database: 1. make out own 2. use someone elses.	1. Do we have an accessible database of various plants with their quantifiable information? 2. Can we access the database?	
S-17	Moisture Interface	As a User I want to see the current and the past moisture levels so that I can adjust my watering routine.	5	3	Must	To do	The user should be able to see the moisture levels to either water the plant preventively or to see how he or she can modify their routine. While notifications are important, it is better not to reach a point when the soil is too dry or moist.	1. Can the user access current level of moisture on the app? 2. Can the user see previous moisture levels? 3. Can the user see the graphical representation of moisture levels over time?	
E-1	Power	As a User I want to turn on/off the device so that I can choose when to use its battery.	5	3	Must	To do	To save power when the system is not needed, the user should have control on the system's state. For safety, the user should also turn the system off when working with the pot.	1. Can the user easily access a switch that turns on/off the system? 2. Can the user see the on/off state of the system?	
A-11	Sprint 3 Testing Doc	As a Engineer I want to have a testing document so that the requirement satisfaction is verified and the possible bugs are detected.	5	3	Must	To do	We need to make sure that every feature that is planned and executed is working well. If something doesn't work, it is not complete. If the app crashed something is wrong. Therefore we try to break the app and its features so we can fix them. This is to make sure that the customer gets the best experience.	1. Can we verify all the requirements? 2. Can we break down the app somehow?	
A-12	Sprint 3 Design Doc	As a Project Owner I want to have a design document so that the project description is available in a readable format.	5	3	Must	To do	The design document is an important description of the whole project. It has all the relevant information in a readable format (compared to backlog, for example).	1. Does the design document contain all the infomation about the project? 2. Is it submitted?	
S-21	Sprint 3 Software Testing	As an Engineer I want to make sure that everything is tested so that I don't have to return and rework past stories.	5	3	Must	To do	Testing is an integral part of development process. It allows us to test our code to see whether it can work properly. If it doesn't, bugs will appear in further iterations.	1. Are there ways to break the software from the user side? 2. Are all aspects of code tested?	
C-4	Sprint 4 Hardware Testing	As an Engineer I want to make sure that everything is tested so that I don't have to return and rework past stories.	5	3	Must	To do	Hardware testing is even more important than the software testing since software can be updated distantly. If something breaks or stops working, we will have to recall the product and replace it with it with a new one or a monetary compensation. This should not happen.	1. Are there ways to break the hardware from the user side? 2. What can go wrong with hardware?	
A-13	Financial Analysis	As a Product Owner I want to know at what price the product should be sold so that I can analyze whether it is gonna sell or not.	4	3	Must	To do	It is important to understand what is the cost of our product. Given the sunk and variable costs a price can be established. Given that price, we can see whether users will buy the product, given interview results.	1. What is the final cost of hardware? 2. What are the R&D costs? 3. What price will the device be? 4. Is the device at a price people will buy it?	
S-2	Light Notification	As a User I want to receive a notification when the light levels are too low/high so that I move my plant to a better place in the house.	4	4	Should	To do	To make sure the plant gets proper light exposure, the user must choose a good place for the plant pot. When there is not enough or too much light the user shall be notified.	1. Does the user get a notification when the light levels are not proper?	
S-3	Temperature Notification	As a User I want to receive a notification when the temperature levels are too low/high so that I change the temperature of the room, or move it to another room.	4	4	Should	To do	To make sure the plant exists in proper temperature range, the user must keep proper temperature in the area. When the temperature is not appropriate, the user shall be notified.	1. Does the user get a notification to when the temperature is too low? 2. Does the user get a notification to when the temperature is too high?	

E-5	Environmental Resistance	As a User I want to have a device working in various environmental conditions so that I can install the device for outside plants.	2	4	Could	To do	Plant owners may grow plants outdoors, thus it is preferrable that the device does not break down when face with outside envrionment.	1. Does the device keep working when put outside? 2. How long does it take to break down given that it constantly rains?	
S-4	Recommendations	As a User I want to receive recommendation on healthier practices for the plant so that my plant stays healthy.	5	4	Could	To do	There are many hints & tips out there for plant owners; it's always good to give users some ideas.	1. Does the user get tips inside the app? 2. Do the tips change?	
S-6	Plant Identification	As a User I want to identify my plant with a picture so that I can understand it better.	10	4	Could	To do	Some plant owners don't remember the name of their plant or don't know it. It is nice to have a possibility to for the user to determine what is the plant, so he/she can link it to the database. Using a google lense API.	1. Can the user take a picture form the app? 2. Does the app determine the plants' name based on the picture?	
E-3	Rugged Design	As a User I want to have a device that it low maintenance so that I don't have to spend my time repairing it.	5	4	Should	To do	The device must be sturdy and not breakdown easily.	1. Can the device survive 1 meter fall? 2. Can the device be inserted in soil without breaking?	
S-7	Notification Recurrency	As a User I want to configure how often I receive the notifications so that the app is more adapted to my lifestyle.	2	4	Should	To do	The user should be able to choose how often the app notifies them.	1. Can the user go to options and change notification time range? 2. Does notification frequency change based on user preferences?	
S-8	Leaves and Pot Notification	As a User I want to receive reminders to cut leaves and repot the plant so that the plant growth is controlled.	3	4	Could	To do	Some plants require pruning, and most plants require repotting. If a plant is not repotted, the root system may overgrow, killing the plant.	1. Is there a notification to cut leaves, based on the amount of time passed since last pruning? 2. Is there a notification to repot the plant since last repotting?	Merged in S-5: Pot Notification.
S-9	Plant Database Link	As a User I want to access plant database where I can link my plant to existing plant models so that I can get specific information on how to take care of my plant.	3	4	Should	To do	Each users plant should be linked to a plant in a database so that the app has reference to the desired levels of moisture, light exposure and temperature range.	1. Can the user access a dropdown list of various plants in the plant profile? 2. Can the user choose a plant type? 3. Does the profile get reference levels for all three metrics based on plant type chosen?	
S-12	Firebase: Read Light	As a Engineer I want to read data concerning light sensor so that I can verify if the value is too low or too high.	1	4	Should	To do	Once connected to Firebase, the app should obtain data values for light.	1. Can the app obtain the data values for light form the Firebase? 2. Can the data values be logged in the app?	
S-13	Firebase: Read Temperature	As a Engineer I want to read data concerning temperature sensor so that I can verify if the value is too low or too high.	1	4	Should	To do	Once connected to Firebase, the app should obtain data values for temperature.	1. Can the app obtain the data values for temperature form the Firebase? 2. Can the data values be logged in the app?	
A-6	Obtain Light Sensor	As a Engineer I want to have a light sensor so that I can obtain light exposure data for the prototype.	1	4	Could	To do	We need to obtain a light exposure sensor to get soil moisture data.	1. Do we have a sensor that is compatible with the microcontroller? 2. Can we obtain sensible data from it?	
A-7	Obtain Temperature Sensor	As a Engineer I want to have a temperature sensor so that I can obtain temperature data for the prototype.	1	4	Could	To do	We need to obtain a temperature sensor to get soil moisture data.	1. Do we have a sensor that is compatible with the microcontroller? 2. Can we obtain sensible data from it?	
S-18	Light Interface	As a User I want to see the current and the past light exposure levels so that I can see what spots are best in my home.	5	4	Should	To do	The user should be able to see the light levels to see how much sunlight the plant gets during the day.	1. Can the user access current light levels on the app? 2. Can the user see previous light levels? 3. Can the user see the graphical representation of the light levels over time?	
S-19	Temperature Interface	As a User I want to see the current and the past temperature levels so that I can adjust the temperature so the plants grow better.	5	4	Should	To do	The user should be able to see the temeprature levels over time, so they can adjust it to optimal levels.	1. Can the user access current temperature on the app? 2. Can the user see previous temeperature levels? 3. Can the user see the graphical representation of temperature levels over time?	

2.1 Sprint Backlog

Our first sprint was not efficient due to midterm week; in addition, the goals we have set were too ambitious. After meeting with the whole team, we realized that we are not on the same page regarding how the project works and how it will look like. The situation was somewhat fixed, and we reworked the Sprint 1 based off work that was done, with the goal of creating local and remote database of plants.

Table 2: Sprint 1.

Goal: Create a local and remote database of plants, that the user can interact with in order to organize a list of plants, and post data with embedded system to the remote server.							
Story ID	**Task Id**	**Title**	**Task Description**	**Time**	**Status**	**By**	**Notes**
S-10	S-10.1	Notification	Create notification in the app.	2	Done	Vily	The application allows in app notifications for testing purposes
	S-10.2	Create Firebase Link	Create a connection to firebase database	2	Done	Vily	The application now contains all prerequisite connections to the Firebase database
	S-10.3	Push notification	Create push notification for specific users, given token	2	Done	Vily	Push notifications were created and tested on specified users
	S-10.4	Authentication	Create Firebase authentication service.	2	Done	Vily	Users can register with an email and password, saving it to the Firebase database
	S-10.5	Link App to Firebase	Link the app project to the Firebase database.	2	Done	Vily	The app can communicate with the Firebase database to retrieve and save user token
S-14	S-14.1	Profile	Create a plant profile containing name, watering interval, light exposure, temperature and growth.	5	Done	Jun	
	S-14.2	Display Watering	Display recommended moisture levels.	1	Done	Jun	
	S-14.3	Display Light	Display recommended light levels.	1	Done	Jun	
	S-14.4	Display Temperature	Display recommended temperature.	1	Done	Jun	
	S-14.5	Display Growth	Display how often to prune and to repot.	1	WIP	Jun	
	S-14.6	Plant ListView	Create a a list of all plant profiles with an option to create a new one.	5	Done	Jun	

C-1	C-1.1	Setup	Setup a microcontroller with the WiFi module.	1	Done	Karl	Note: mock data was used. Moisture sensor will be verified once we get it.
	C-1.2	Translate Data	Analyze incoming data and translate it to sensible values (0-100% moisture).	1	WIP	Karl	
	C-1.3	Send Data	Send the resulting data to the Firebase.	5	Done	Karl	
	C-1.4	Receive Data	Get data from the Firebase.	5	Done	Karl	Dummy data was used.
A-1	A-1.1	Backlog Grooming	Remove redundant stories, remove useless stories, merge some stories, clean up the rest.	5	Done	Daniel	Given that the project started during midterms, the original backlog was rushed and had to be extensively groomed.
	A-1.2	Sprint 1 Cleanup	Cleanup sprint 1 tasks.	2	Done	Daniel	Sprint 1 was unrealistic given exam period, and tasks were disjointed.
	A-1.3	Sprint 1 and 2 Tasks	Meetup with team members to discuss the direction of the project and tasks completed as well as the tasks to do.	3	Done	Daniel	Karl - 0.5 hrs Vily - 0.5 hrs Jun - 0.5 hrs Thomas - 0.5 hrs
A-2	A-2.1	Find and Get	Choose a module from the ones given by Concordia and get them.	1	Done	Daniel	
A-4	A-4.1	Software	Create test cases for software.	5	Done		Vily, Thomas and Jun
	A-4.2	Hardware	Create test cases for hardware.	5	Done		Karl and Daniel
A-5	A-5.1	Wireframe	Create the Android wireframe for the project.	2	Done	Vily	
	A-5.2	System Architecture	Write down system architecture for the project.	2	Done	Vily	
	A-5.3	Hardware Architecture	Write down hardware architecture for the project.	2	Done		Daniel, Karl
	A-5.4	Software Architecture	Write down software architecture for the project.	2	Done		Vily, Thomas
	A-5.5	Use Cases	Write down the possible use cases.	2	Done	Vily	
	A-5.6	Testing	Write down the test plan.	2	Done		Vily, Jun
	A-5.7	Appendix	Put additional information to the appendix.	2	Done		Daniel

S-16	S-16.1	Relational Database	Design a database with all its relations and attributes.	3	Done	Thomas	Create diagrams: ER and Database to illustrate the relationships of the database.
	S-16.2	Firebase Database	Create template for team members by implementing first tuples for each table through Firebase console.	2	Done	Thomas	
	S-16.3	Plant Container Class	Implement Plant class.	2	Done	Thomas	Attributes: strings: name, description ints: moisture, light, temperature
	S-16.4	Plant as Database Input	Implement an activity allowing users to add plant entities/tuples to the Plant table from the Firebase database.	2	Done	Thomas	
	S-16.5	Plant Write Firebase	Create cloud based database for the plant entities.	5	Done	Thomas	When given Plant object to the function, it is then added onto the Firebase database.
	S-16.6	RecyclerView for Plant Tuples	Implement an activity that will display a RecyclerView list that will be populated by all the Plant tuples.	5	Done	Thomas	Use handler MyAdapter class that will populate the PlantDatabaseActivity.
	S-16.7	Plant Read Firebase	On start of PlantDatabaseActivity RecyclerView gets populated with data read from Firebase database.	5	Done	Thomas	Every single tuple is read from the database and added to the Plant linkedlist. List sent as argument to myAdapter.
	S-16.8	Merge	Merge App to Team Branch	3	Done	Thomas	Merge plant database features to the team branch.

Sprint 2 was planned based on tasks we did in Sprint 1; the goal was to fully integrate hardware and software through Firebase: we wanted to see moisture data from hardware on our app. In addition, the moisture data from the hardware was to be appropriately stored on the app. Tasks that carried over from Sprint 1 were further broken down and completed as well.

Table 3: Sprint 2.

Sprint 2 Goal: Fully integrate hardware and software through Firebase: we want to see moisture data from hardware on our app. To appropiatly store the moisture measurement data from the hardware to the app, user owned plants can be stored in the database.							
Story ID	**Task Id**	**Title**	**Task Description**	**Time**	**Status**	**By**	**Notes**
C-1	C-1.1	Setup	Setup a microcontroller with the WiFi module and current time.	5	Done	Karl	Issues for Daniel: old router. [Karl and Daniel] Using UNIX time.
	C-1.2	Calibrate	Analyze incoming data and translate it to sensible values (0-100% moisture).	1	Done	Daniel	0% = air 100% = water
	C-1.3	getMoisture()	Set up hardware and write a function reading and returning integer describing the % moisture.	1	Done	Daniel	
	C-1.4	getEpochTime()	Set up a function that takes time from a server and translates it to UNIX format at the time of sensor reading.	2	Done	Karl	
	C-1.5	Send Data	Send the resulting data with time to the Firebase.	5	Done	Karl	Impemented in Sprint 1.
	C-1.6	Confirm	Get confirmation from Firebase.	5	Done	Karl	Implemented in Sprint 1.
	C-1.7	Refactoring	Refactor writeToDatabase()	2	Done	Karl	
C-2	C-2.1	Calibrate	Analyze incoming data and translate it to sensisble values (0-100& light).	1	Done	Karl	0% = covered 100% = full sun
	C-2.2	getLight()	Set up hardware and write a function reading and returning integer describing the % light.	1	Done	Karl	
	C-2.3	Send Data	Send the resulting data to the Firebase.	1	Done	Karl	
	C-2.4	Refactoring	Refactor the function.	1	Done	Daniel	
C-3	C-1.2	Calibrate	Analyze incoming data and translate it to sensible values (0-100% moisture).	1	Done	Daniel	Karl - initial calibration: 0.5 Daniel - final calibration: 2
	C-1.3	getTemperature()	Set up hardware and write a function reading and returning integer describing the % moisture.	1	Done	Karl	
	C-1.4	Send Data	Send the resulting data to the Firebase.	1	Done	Karl	
	C-1.5	Refactoring	Refactor the function.	2	Done	Daniel	SparkFun has different architecture from Arduino. "int" and "float" must be cast.
S-11	S-11.1	Read Data	Read and record data from the firebase.	4	Done	Thomas	
	S-11.2	Low Values	Verify if the values are low or high, given recommendations.	1	WIP	Thomas	
S-1	S-1.1	Push Notification	Create a push notification for moisure level	5	Done	Vily	Frequency to set.
	S-1.2	Cloud Function	Define cloud functions for Firebase.	5	Done	Vily	
	S-1.3	Trigger	Trigger defined cloud functions when user values are too low.	5	Done	Vily	Triggers on write.
	S-1.4	Logout	Allow user to log out from his account.	5	Done	Vily	
	S-1.5	Sign-in	Allow the user to sign in with different credentials after opening the app.	5	Done	Vily	
S-15	S-15.1	Plant Types	Research various plant types.	5	Done	Jun	
	S-15.2	Plant Needs	Based on previous task, find what each type of plant needs.	5	Done	Jun	
	S-15.3	Quantification	Quantify the plant needs to numbers that we can work with.	5	Done	Jun	

S-16	S-16.9	Fill	Fill the database with found and/or calculated values.	5	Done	Jun	
	S-16.10	Edit Database	Implement a frontend edit function to database entries to modify their attributes.	3	Done	Thomas	
	S-16.11	Edit Tuples	Given a plantID, edit Plant tuples from the database.	5	Done	Thomas	editPlantDatabaseActivity receives an Intent to know on which tuple the edit should be done.
	S-16.12	add userPlant	Implement an activity that lets user add a plant from the catalog to their own	3	Done	Thomas	
	S-16.13	store userPlants in database	Create OwnsA relationships in the database through the client application	5	Done	Thomas	
	S-16.14	display user's plants in a list	Display all user plants in a list and each item has an onClick that directs to a detailed page on the plant	5	Done	Thomas	
	S-16.15	Get userPlant Profile	Using the token generated by authentication, the userPlant object gets all the attributes of the plant.	2	Done	Thomas	
	S-16.16	Get Matching Plant Profile	Using the same token, attributes of the plant that matches the plant owned by the user can be stored in a Plant object.	2	Done	Thomas	
	S-16.17	Compare Attributes	Compare userPlan data to the database plant recommendations.	2	WIP	Thomas	Redundant to S-11.2; once S-11.2 is done, this one should be done.
	S-16.18	Send Notification	Send notification if the difference is present.	3	To do		
	S-16.19	Store Pics	Upload pictures to database	5	To do		Sep to diff story
	S-16.20	OwnsA Relationship	The user can post picture of its plant to the general database.	5	To do		

A-3	A-3.1	Research	Find a moisture sensor that is within 10$ and that can read soil moisture.	0.5	Done	Daniel	Got one from Concordia.
	A-3.2	Buy	Buy it.	0.5	Done	Daniel	
	A-3.3	Test	Test the sensor.	1	Done	Daniel	
	A-3.4	Data Translation	Translate data from the sensor to some sensible value, confirming C-1.3.	1	Done	Daniel	Test with the results of C-1 story tasks and setup.
E-4	E-4.1	Components	Choose internal components (MC, battery, etc.)	5	To do	Karl	
	E-4.2	PCB	Design PCB for given components.	5	Done	Karl	L,M,T -> P36,37,38
	E-4.3	External	Design the external look of the device (a box) that will contain the internal components.	5	WIP	Karl	
E-5	E-5.1	Material Choice	Research various materials that can withstand the outside environment.	2	To do		
	E-5.2	Apply to Design	Apply the selected material to the external hardware design, defined in E-4.3.	1	To do		
S-20	S-20.1	Refactoring	Refactor the main folder, since naming conventions are creating bugs.	4	Done	Thomas	
	S-20.2	Debug Sprint 1	Debug code that failed acceptance test.	3	Done	Thomas	
A-8	A-8.1	Sprint 2 Scrum	Get feedback on tasks, and update the Sprint 2.	2	To do	Daniel	[Everyone]
	A-8.2	Sprint 3 Plan	Meet with teammates and discuss what features will be implemented and what bugs should be fixed in Sprint 3.	3	To do	Daniel	[Everyone]
	A-8.3	Backlog Grooming	Clean up and reorganize the backlog based on previous tasks.	4	To do	Daniel	

A-9	A-9.1	Hardware	Write tests for hardware (new sensors mostly).	3	To do	Daniel	
	A-9.2	Test	Test the hardware. Light test, Moisture test, Temperature test	3	Done	Karl	Sprint 2 is critical; hardware should be working flawlessly, since software is the issue.
	A-9.3	Software	Write tests for software.	5	To do		Jun, Vily, Thomas
	A-9.4	DoD	Write Definition of Done for tasks.	5	To do	Jun	
A-10	A-10.1	Wireframe	Update the Android wireframe for the project.	1	To do	Thomas	
	A-10.2	System Architecture	Update system architecture for the project.	1	To do	Vily	
	A-10.3	Hardware Architecture	Update hardware architecture for the project.	1	To do	Daniel	
	A-10.4	Software Architecture	Update software architecture for the project.	4	Done	Jun	Introduced parent-child relationship and organized hierarchy between activities.
	A-10.5	Use Cases	Update the possible use cases.	5	To do		
	A-10.6	Testing	Write down the test plan.	3	To do		
	A-10.7	Finance	Determine the price of components.	5	To do		
	A-10.8	Appendix	Put additional information to the appendix.	2	To do		
	A-10.9	Introduction	Write down the intro.	1	Done	Daniel	

Sprint 3 will focus on making things right; that is bug fixing as well as a better graphic design.

Table 4: Sprint 3.

Sprint 3: The goal of this sprint is to implement temperature and light readings, as well as to graphically demonstrate all of the values over time period. Finally if time permits, graphical interface could use some love.

Story ID	Task Id	Title	Task Description	Time	Status	By	Notes
S-16	S-16.17	Compare Attributes	Compare userPlan data to the database plant recommendations.	2	WIP	Thomas	Redundant to S-11.2; once S-11.2 is done, this one should be done ASAP.
	S-16.18	Send Notification	Send notification if the difference is present.	3	To do		
	S-16.19	Store Pics	Upload pictures to database	5	To do		
	S-16.20	OwnsA Relationship	The user can post picture of its plant to the general database.	5	To do		
S-11	S-11.2	Low Values	Verify if the values are low or high, given recommendations.	1	WIP	Thomas	
E-5	E-5.1	Material Choice	Research various materials that can withstand the outside environment.	2	To do		
	E-5.2	Apply to Design	Apply the selected material to the external hardware design, defined in E-4.3.	1	To do		
S-17	S-17.1	Recent Value	Get most recent value of moisture.	1	To do		
	S-17.2	Click	Make recent value clickable, transferring me to the list of previous data.	2	To do		
	S-17.3	Update Data	Get all data that is new since last time it was checked.	5	To do		
	S-17.4	Clean Up	Delete old data periodically.	5	To do		
	S-17.5	Graph	Display a graph of the logged data.	5	To do		
S-21	S-21.1	Firebase Connection	Run and fix test cases from requirement ID S-10	5	To do		
	S-21.2	Plant Profile and Display	Run and fix test cases from requirement ID S-14	5	To do		
	S-21.3	Database	Run and fix test cases from requirement ID S-16	5	To do		
	S-21.4	Notifications	Run and fix test cases from requirement ID S-1	5	To do		

E-1	E-1.1	Switch	Implement a hardware switch.	2	To do		
	E-1.2	PCB	Add switch to PCB.	1	To do		
	E-1.3	Drawing	Add switch to graphical design.	1	To do		
A-11	A-11.1	Hardware	Write tests for hardware (new sensors mostly).	3	To do		
	A-11.2	Test	Test the hardware. Light test, Moisture test, Temperature test	3	To do		
	A-11.3	Software	Write tests for software.	5	To do		
	A-11.4	DoD	Write Definition of Done for tasks.	5	To do		
A-12	A-12.1	Wireframe	Update the Android wireframe for the project.	1	To do		
	A-12.2	System Architecture	Update system architecture for the project.	1	To do		
	A-12.3	Hardware Architecture	Update hardware architecture for the project.	1	To do		
	A-12.4	Software Architecture	Update software architecture for the project.	4	To do		
	A-12.5	Use Cases	Update the possible use cases.	5	To do		
	A-12.6	Testing	Write down the test plan.	3	To do		
	A-12.7	Finance	Determine the price of components.	5	To do		
	A-12.8	Appendix	Put additional information to the appendix.	2	To do		
	A-12.9	Introduction	Write down the intro.	1	To do		
C-4	C-4.1	Hardware	Run and fix test cases from requirement ID C-1	5	To do		
A-13	A-13.1	Hardware Cost	Determine the cost of hardware.	3	To do		
	C-4.3	R&D Cost	Determine the cost of maintenance and development.	3	To do		
	C-4.4	Price	Find the optimal price based on previous interviews.	4	To do		
	C-4.5	Market	Research market.	5	To do		

3. Design Document

The current design document explores various views of the project, as well as the latest iteration of features included.

3.1 Android Application Wireframes

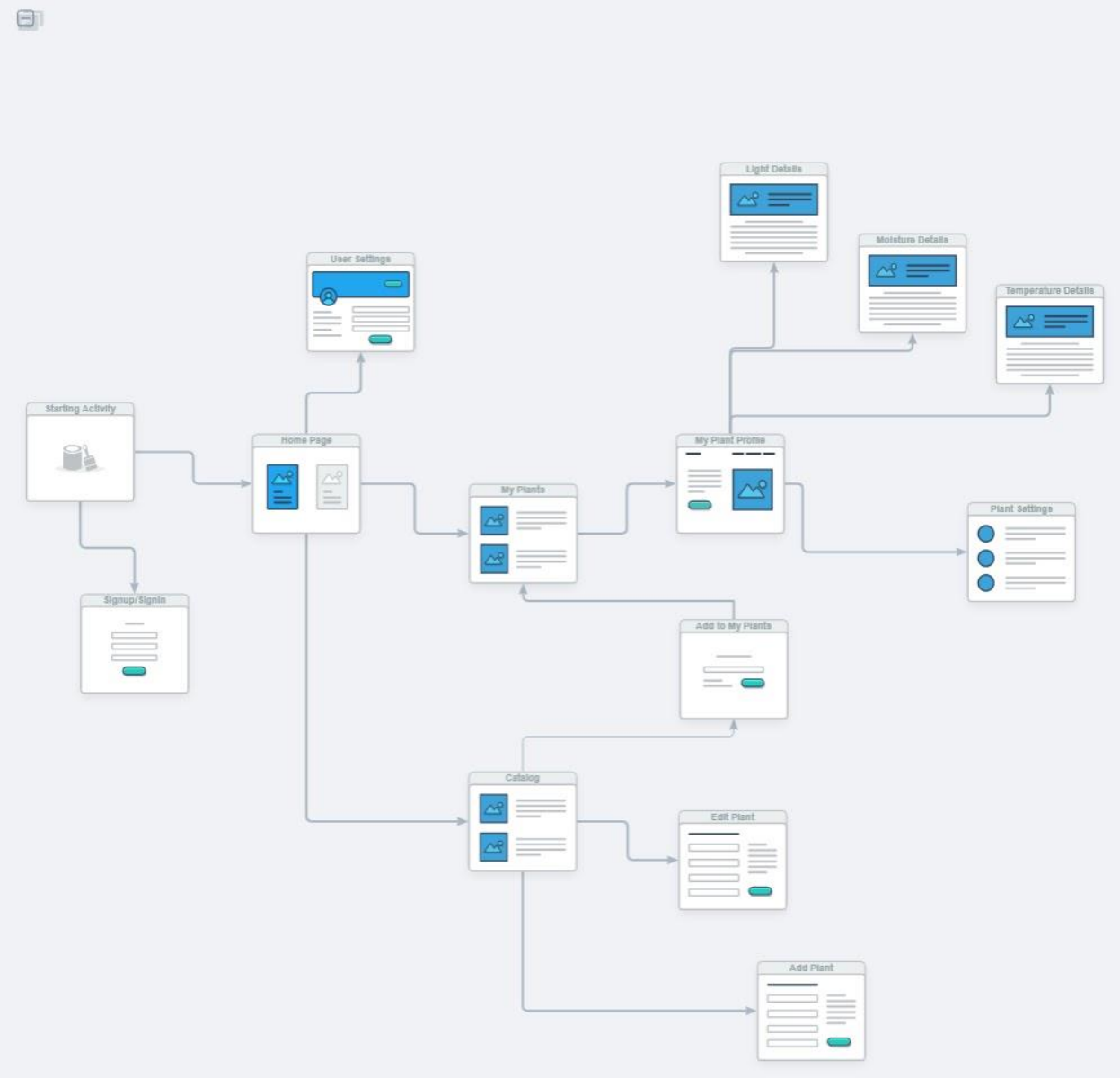

Figure 1: Healthy Leaves wireframe user experience.

The diagram in *Figure 1* describes the user experience through the Android application client. This diagram uses web wireframes although ideally it should be done with

android displays, however no free wire flow tools were available to us at the moment. Assume each box represents a distinct android activity to be displayed to the user. On the application start, by default, the application goes to the Starting activity and checks in the phone's cache if the user has already previously signed up or signed in. If so, the user is directed directly to the home page. Otherwise, the user is directed to the sign in/signup page where their email and password can be entered to create a new user profile within our firebase database. After successfully entering or creating their password, they will also be directed to the home page going back to the starting activity first. From the home page, the user has many buttons to his/her disposition. They can access their user settings page, log themselves out on the specific device. A first button lets the user access to a catalog, a common database of plants and their specified ideal values for light, moisture, and temperature levels. This catalog is a Wikipedia style page where any other user also has the power to modify what are the best values for each plant and add new plants if they were not documented yet. Therefore, the user has a button that takes them to another page where they can add an entry to the catalog of plants and for each plant row (each previous plant entry) each user has the option to edit the plant entry's information or add that plant type to one of the plants they own themselves. The other option from the home page is to see instead a list of the plants they own themselves. This list is called My Plants and displays each unique plant they own (created from the addUserPlantActvity() from the Catalog side). These plants have their own name that was given to them by their owner and their plant type. Each of these rows can be clicked on which takes the user to that specific plant's profile. On the plant profile, the unique plant's real time measurements of light, moisture and temperature levels are displayed and compared to the ideal same values. A small overview of the changes that should be made to increase the plant's chance for healthiness is displayed. Each measurement (aka light, moisture, and temperature) can be clicked on which takes the user to more detailed information about each of these variables. In these detailed pages, a log of the measurements is shown, and graphs are displayed. From the plant profile, there is also the option to go to the specific plant's settings so personalize the information that is given also.

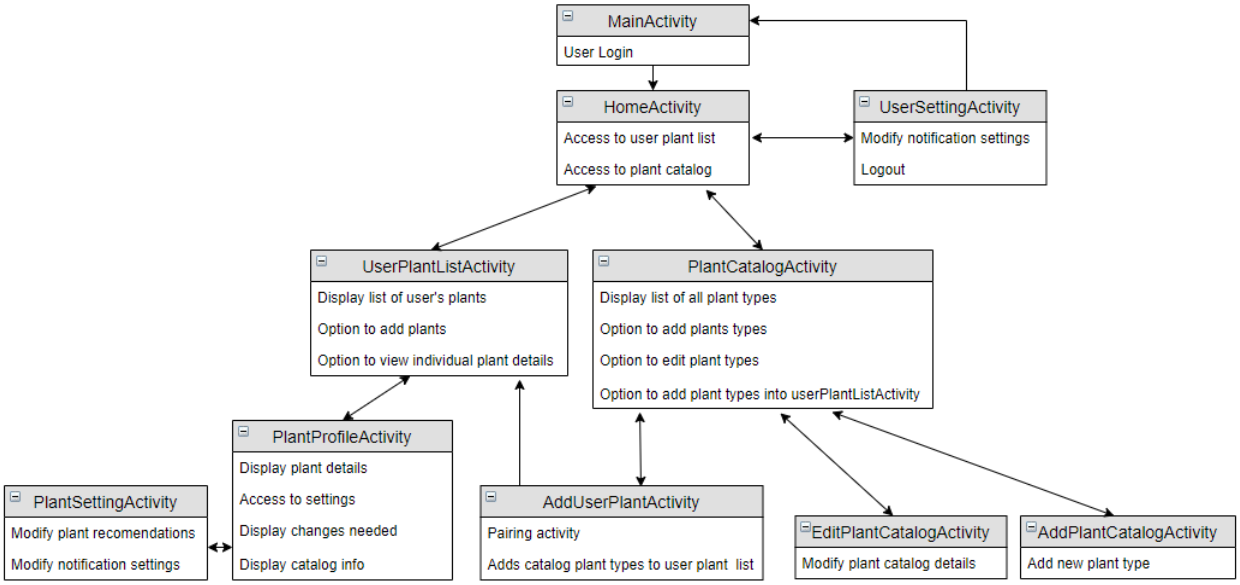

Figure 2: Activity hierarchy for HealthyLeaves project.

The HomeActivity is the activity that the user will first open when the application starts running. If the user has never logged in before, he/she will be taken to the MainActivity, which will prompt the user to register with an account email and password. These credentials will then be saved into the Firebase database, and the user will be given a token. In HomeActivity, the user will be given the chance to select between UserPlantListActivity (the user's personal plant list), and the PlantCatalogActivity (the shared community plant list with ideal plant care information). They will also have access to a UserSettingActivity where they can alter notification settings and logout to return to the MainActivity. In UserPlantListActivity, the user can click on a specific plant on the list and obtain the plants information in PlantProfileActivity. From here, the user can click on plant settings to modify plant information or their notification settings for the plant. In PlantCatalogActivity, the user can view the shared plant database with their ideal thriving settings. From here the user can choose to edit a specific plant in the catalog with EditPlantCatalogActivity, or they can add a new entry into the database with AddCatalogActivity. Furthermore, if the user is interested in a particular plant in the catalog, they may choose to click on the add button featured next to the plant name. This will redirect the user into the AddUserPlantActivity, where the specific catalog plant will be given a new name and be added to the userPlantListActivity.

3.2 System Architecture

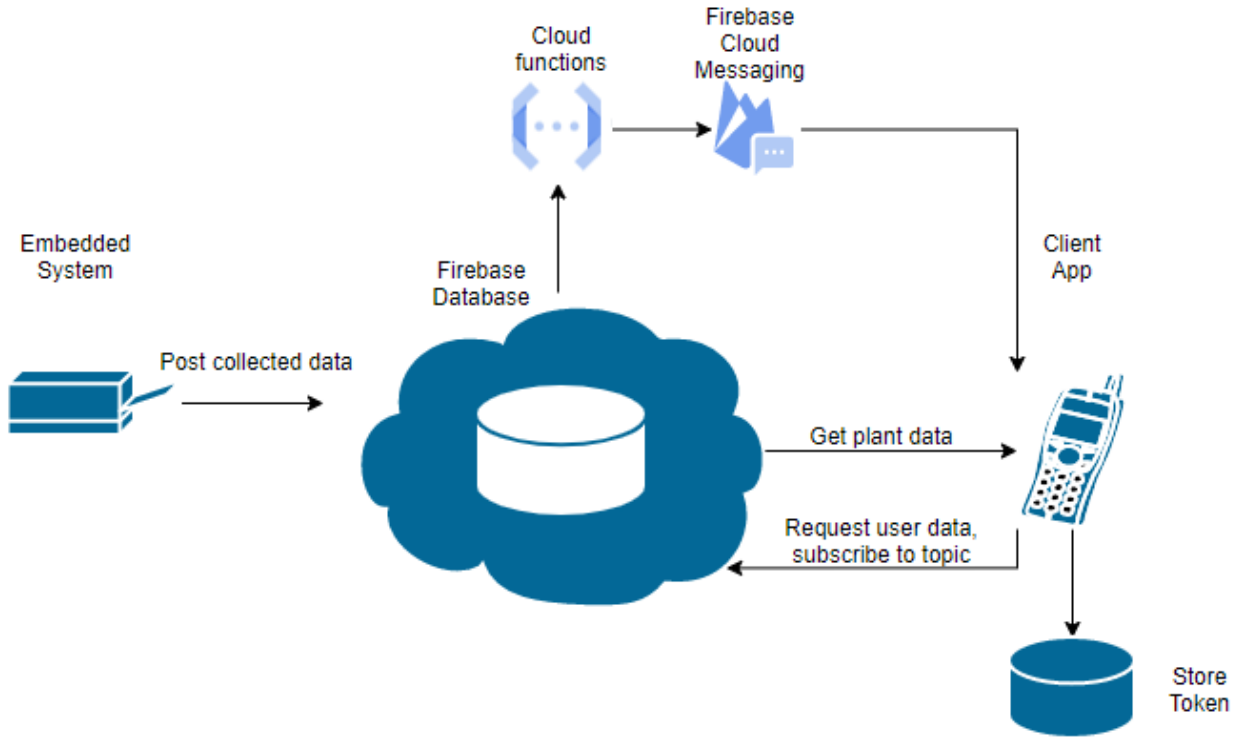

Figure 3: System Architecture of Firebase authentication.

The *Figure 3* given above describes the system architecture of firebase authentication. The Firebase database is connected to two main types of devices: the embedded systems on the plant and its pot (Arduino) and the android client application. On the Arduino side, only posts/write will be performed to the Firebase database as it measures the plant's environment. On the client app, both read and write operations will be performed and will get authenticated by an encrypted Token (using HTTPS) stored on the android phone client side which is used to determine if the database read and write operations requested by the android client are done by a signed-in user on the server. Therefore, the token is a security measure used to verify the integrity and authenticity of the request and the server can retrieve the userID from it. Functions written for the cloud functions of Firebase will be deployed to trigger notifications on the user side, according to data posted by the embedded system. The functions are intended to fire when the measured and posted temperature, moisture or light data is considered inadequate for the growth of the clients monitored plant. These alerts have the intent to warn the user when the plants need care.

3.3 Hardware Architecture

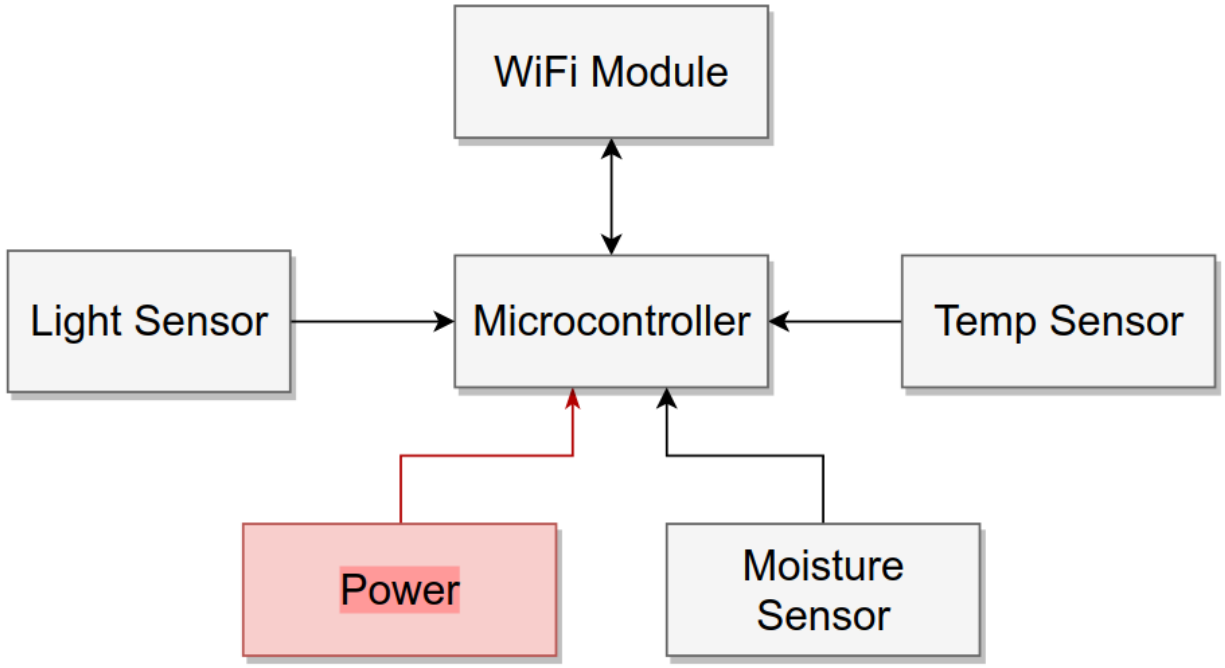

Figure 4: Hardware architecture.

Figure 4 shows the hardware architecture of the Healthy Leaves project. The device consists of three sensors connected to the microcontroller, SparkFun ESP32 Thing, through analog pins. The data is read and sent to Firebase through an onboard Wi-Fi chipset – ESP8266. Currently, for prototyping purposes, the device is powered through the USB port, however the board has connector for a LiPo battery, so it can work wirelessly.

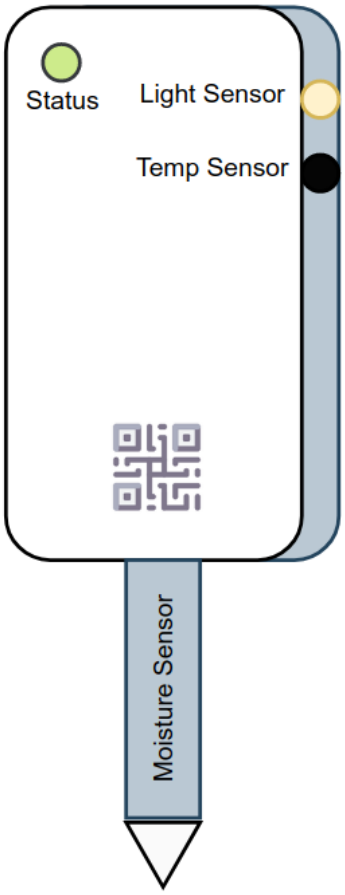

Figure 5: External Hardware Design.

The final product will have close resemblance to the figure above. The internal components will be enclosed in a plastic box with dedicated holes for all the sensors. The box will also have a QR code so the user can link to hardware to their mobile device. The moisture sensor protrudes from the bottom allowing the user to stick the device in the soil of the plant the want to monitor.

Figure 6: Outer Shell.

Figure 7: Main MC - SparkFun ESP32 Thing.

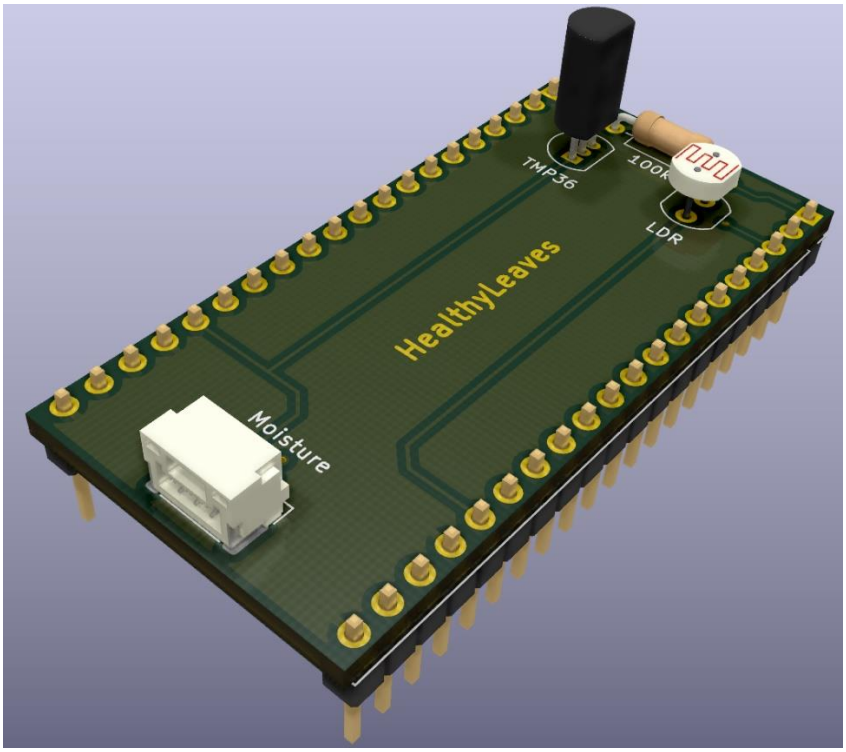

Figure 8: Sensor shield for microcontroller.

The *Figure 8* above shows the visual 3-dimensional rendition of what the circuit board containing the sensors looks like. The temperature and light sensors are located at the top of the circuit board, while the moisture sensor has a dedicated connector located at the bottom of board. It is important to note that the plant monitoring hardware is wireless and requires the power of a small battery to function. The chosen battery is shown in *Figure 9*.

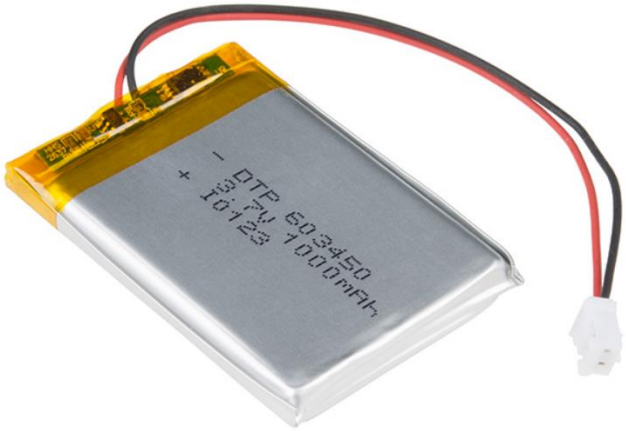

Figure 9: 3.7V, 1Ah LiPo Battery (DTP603450).

3.4 Software Architecture

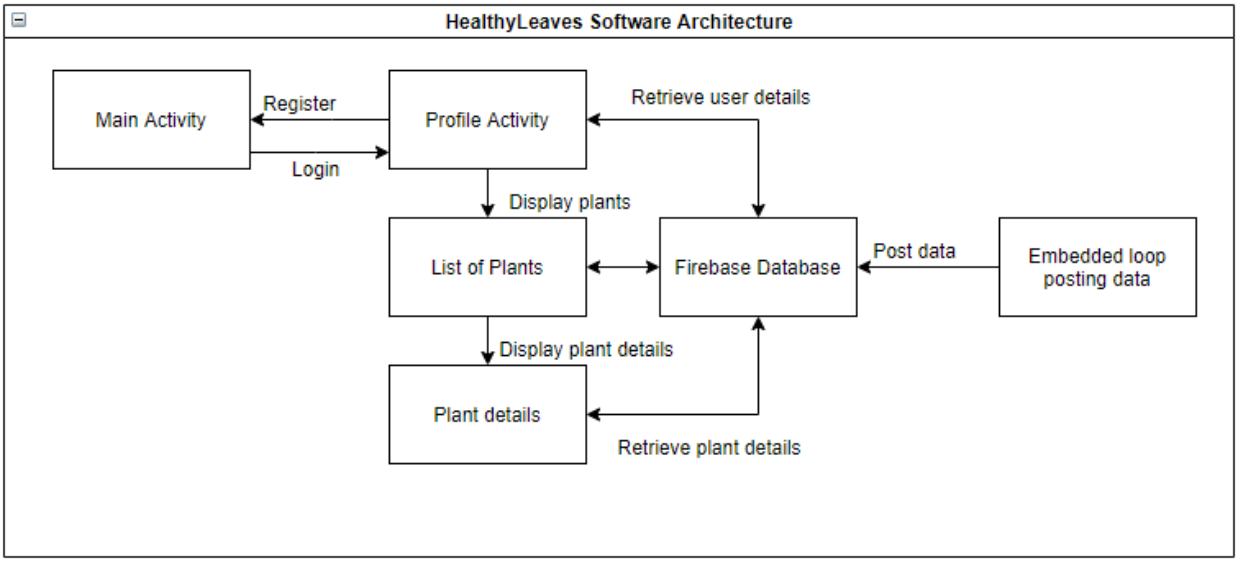

Figure 10: Software architecture of Healthy Leaves project.

The ProfileActivity is the activity that the user will first open when the application starts running. If the user has never logged in before, he/she will be taken to the MainActivity, which will prompt the user to register with an account email and password. These credentials will then be saved into the Firebase database, and the user will be given a token. The MainActivity will then redirect the user back to the ProfileActivity, where it will verify if the user has a token once again. Given that a token is now saved for that user, he/she will be redirected back to ListOfPlants. The ListOfPlants contains the list of plants that the user has already added to his account. If the Addition button is pressed in the bottom right corner of the screen, he/she will be redirected to the PlantActivity screen, where the user will be prompted to fill in the name, growth, light exposure, temperature and watering interval fields to add that respective plant. The information above is currently being hard coded and will eventually be pulled from a database of plants that contains all the relevant information for a plant name. If the user clicks on a plant name from the ListOfPlants, the user will be redirected to the PlantProfile, where he/she will be able to see the relevant information to the plant that was clicked on. Given the user's token ID and plant details, a relevant cloud function will send a notification to the respective user, advising on how to

manage his/her plant. In addition, the user will be able to add data to the Plant catalog, which is a table containing all plants, independent of users. Future sprints will allow users to input their local plants according to the plants available in the database.

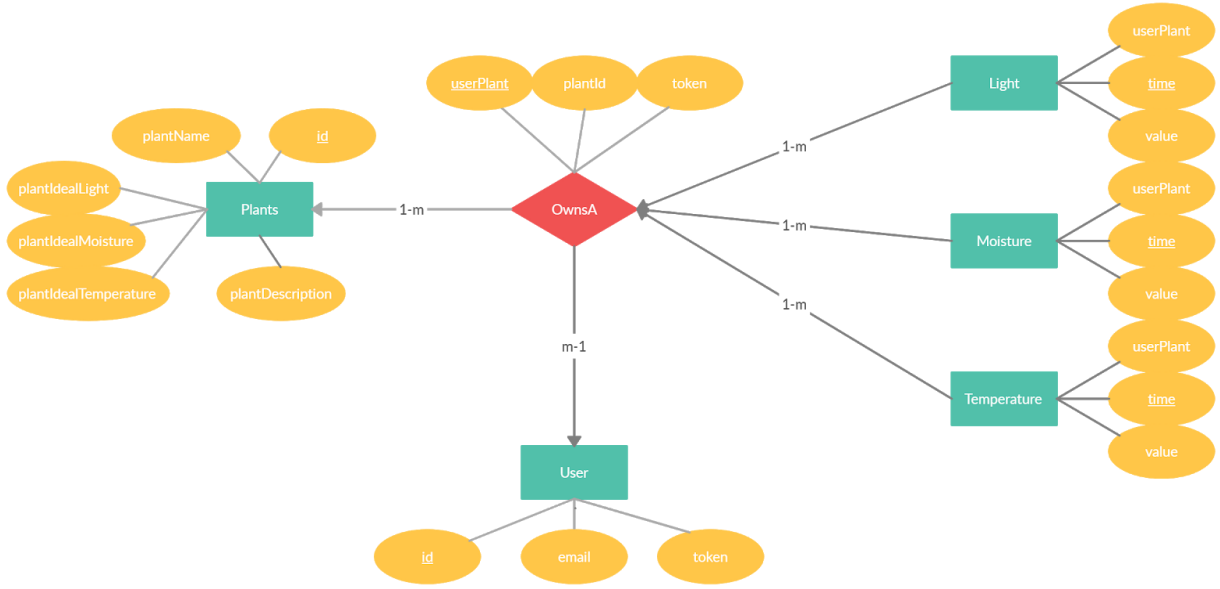

Figure 11: Entity relationship diagram (ER-diagram) of Healthy Leaves project.

The *Figure 11* above illustrates the entity relationship diagram that will structure the firebase database of the HealthyLeaves project. These database entity relationships will be used both on the android and arduino as the whole information being written and read by both hardware entities will partition the collected data in these uniform relationship structures. In fact, the Plants, OwnsA and Users tuples will be created and written from the android side of the project while the Light, Moisture, and Temperature will be created and written from the Arduino side of the project.

The Plants will have a unique primary key given by their id and hold general information on the specific plant such as its name, its ideal light, moisture, and temperature level, and a short description of the plant. The Users will have a unique primary key given by their id and hold authentication attributes such as their email and token. In android studio

firebase implementation, the token can be used to get the current user logged in profile the application is currently running on.

OwnsA is a relationship that describes a User tuple owns a plant. A user can own multiple times the same plant type from Plants, therefore plantID and token cannot form a primary key. Therefore, OwnsA has its own unique non null primary key userPlantID. Note the many to one relationship where a user can own multiple plants and a plant can be owned by multiple users. However, the relationship OwnsA can only describe the relationship between a single user and a single plant.

Finally, the light, moisture, and temperature entities describe each data measured by the Arduino module and its sensors. Each measurement by the different 3 types of measurements will collect an integer type measurement and will note the time at which it was taken at. Because no two measurements by the same sensor can be done at the same time (using epoch time, how many seconds have elapsed since January 1st 1970), the attribute time can be used as the primary key of the light, moisture and temperature primary keys. They also hold a foreigh key of the OwnsA userPlantID to know to which specific plant owned by a user those measurements are referring to. Every measurement is only associated to one OwnsA relationship while an OwnsA relationship will have multiple measurements. In terms of optimal database design, these light, moisture, and temperature tables will be extremely large and may lead to long response time.

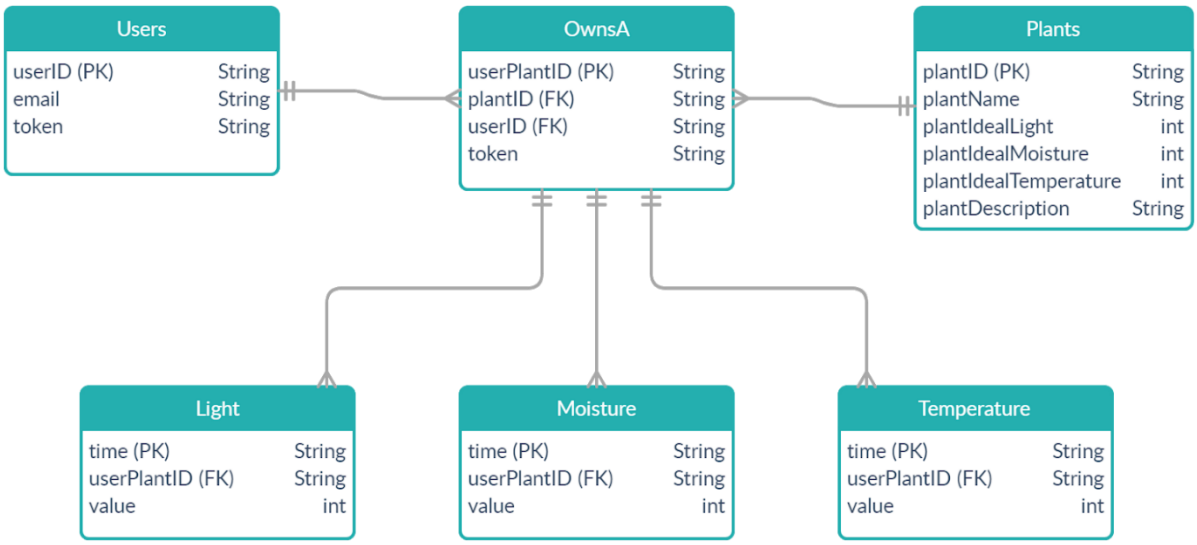

Figure 12: Database diagram of Healthy Leaves project.

The figure above illustrates the database diagram that will be used to implement the firebase database of the Healthy Leaves project. It follows the logical relationships designed in the ER-diagram of *Figure 11*.

3.5 Use Cases and Sequence Diagrams

The following Use Cases, given the current iteration of the product, will be tested:

1. The user registers to the app for the first time.
2. The user is already registered and logs in.
3. The user adds a plant.
4. The user opens plant details.

3.5.1 Use Case 1

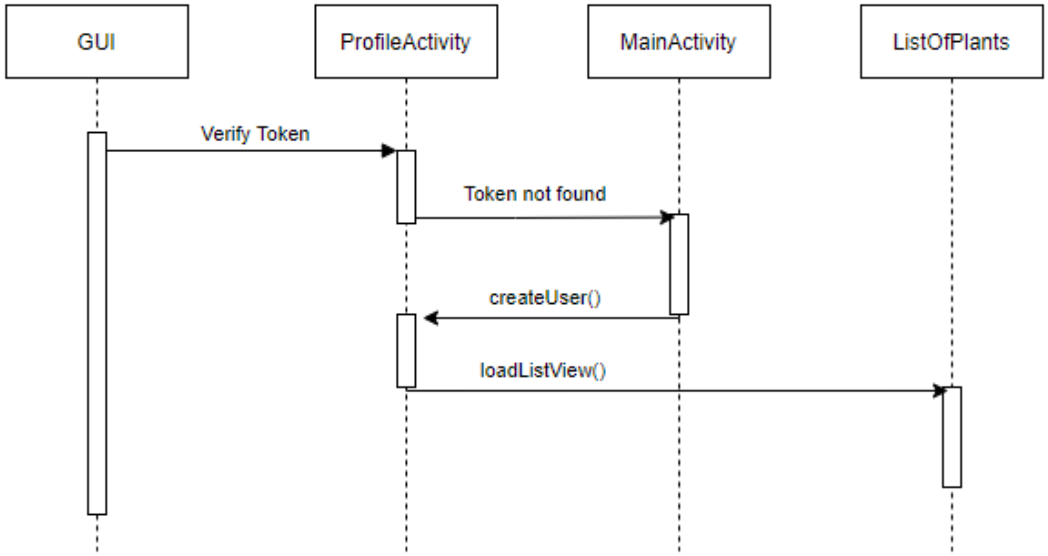

Figure 13: Use case 1.

The ProfileActivity will begin by verifying the current token and determining if it is null or not. Given that the user is registering for the first time, it will be. The token is then considered not found and prompts the user to the MainActivity. The MainActivity will force the user to create an account given an email address and password. The account creation will create a token for the user, save it locally and redirect the user to the ProfileActivity. From there, the token will be found, and redirect the user to the ListOfPlants, where his respective plants will be displayed.

3.5.1 Use Case 2

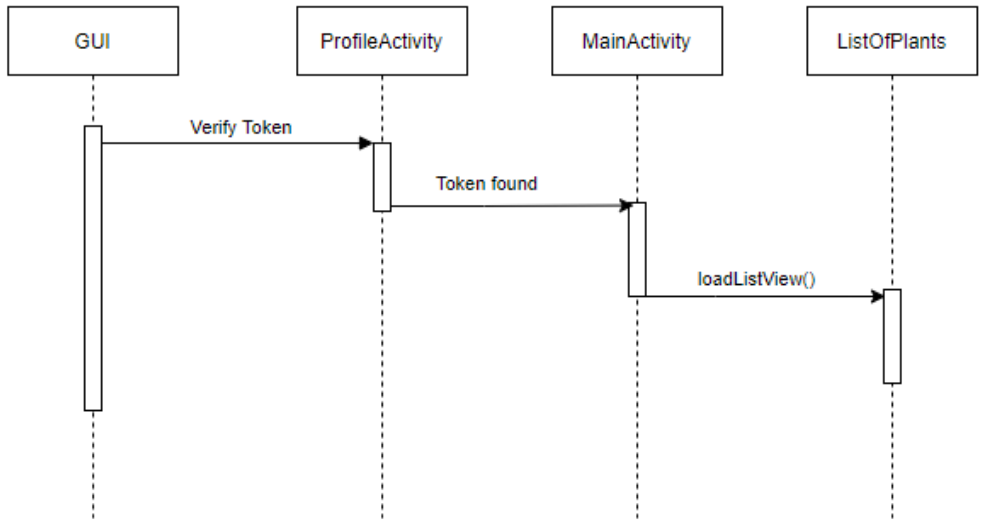

Figure 14: Use case 2.

The second use case assumes that the user is already registered and has a token saved locally. Upon opening the app, it will find the respective token, and immediately prompt him/her to the ListOfPlants.

3.5.1 Use Case 3

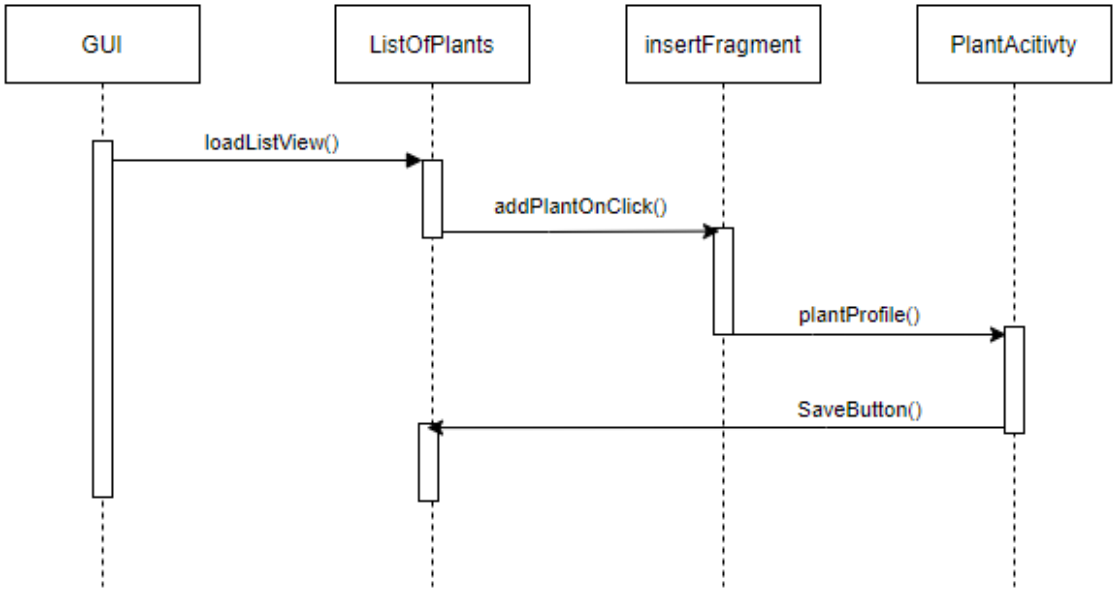

Figure 15: Use case 3.

Given that the user is logged in, upon pressing the button on the bottom right corner of the screen, it will open a fragment, prompting the user to input text fields relating to the plant creation process. As of sprint 1, the plant creation process is done manually by the user. Upon completion, the PlantAcitivty will redirect the user to the ListOfPlants if the button “Save” is clicked.

3.5.1 Use Case 4

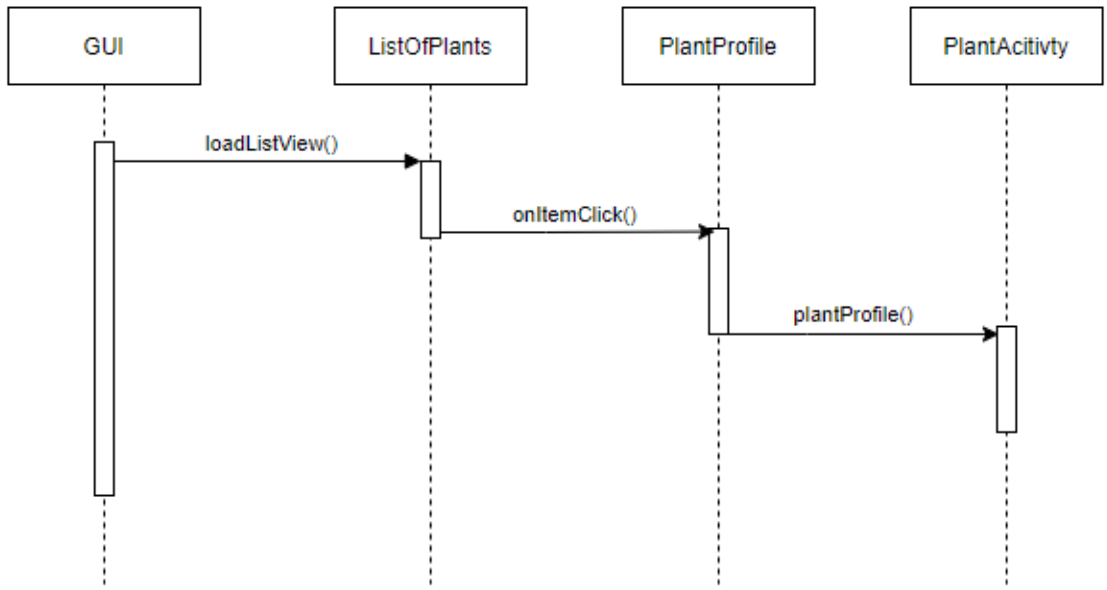

Figure 16: Use case 4.

The use case where the user is viewing the plant details assumes the user is already logged in and viewing his own plants. Upon clicking a plant, the application will redirect the user to PlantActivity, where the relevant data attributed to the plant will be displayed.

4. Testing

4.1 Test Plan 1: Firebase Connection

Requirement ID: S-10

In this section, we are trying to verify the connection with the Firebase Database, in order to ensure that the user can connect to the database and properly receive notifications, regardless of platform or application activity (on or off).

Table 5: S-10.1 Test Case.

Test Case S-10.1		
Pre-Condition: Notifying user with application closed		
Steps	Expected Results	Actual Results
1. Close the phone application	The phone application should close	The phone application closes
2. Turn on Wifi	Wifi should turn off	Wifi turns off
3. Wait for notification	Notification should still arrive	Notification does arrive
Result: Pass		

Table 6: S-10.2 Test Case.

Test Case S-10.2		
Pre-Condition: Notifying user application on		
Steps	Expected Results	Actual Results
1. Open application	The phone application should open	The phone application opens
2. View List of Plants	The list of plants should appear	The list of plants does appear
3. Wait for notification	Notification should still arrive	Notification does arrive
Result: Pass		

Table 7: S-10.3 Test Case.

Test Case S-10.3		
Pre-Condition: User tries to avoid registering		
Steps	Expected Results	Actual Results
1. Open application	The phone application should open	The phone application opens
2. Press back multiple times	The application should redirect to ProfileActivity, and MainActivity again	The application does redirect the user to MainActivity
Result: Pass		

Table 8: S-10.7 Test Case.

Test Case S-10.4		
Pre-Condition: User logs in to different phone with same credentials		
Steps	Expected Results	Actual Results
1. Open application	The phone application should open	The phone application opens
2. Open new phone and enters credentials	The application should not register the same account again, it allow the user to log in	The application does allow the user to log in without register the same account again
3. View List of Plants	The application should redirect the user to his list of plants	The application does redirect the user to his list of plants
Result: Pass		

4.2 Test Plan 2: Plant Profile and Display

Requirement ID: S-14

In this section, we are creating a listview display in the main activity to view all the plants in the database. Furthermore, we want to ensure that users can add additional plants with their corresponding information in the listview.

Table 9: S-14.1 Test Case.

Test Case S-14.1		
Pre-Condition: Adding new plants		
Steps	Expected Results	Actual Results
1. User opens dialog fragment in main activity by pressing floating action button	Dialog fragment opens to allow user to add new plant and corresponding details	Dialog fragment opens to allow user to add new plant and corresponding details
2. Entering plant profile information in the dialog fragment	Text boxes are filled with user inputs	Text boxes are filled with user inputs
3. Pressing save button	Once filled correctly, user input is saved and the dialog fragment closes. New plant added to listview in main activity.	Once filled correctly, user input is saved and the dialog fragment closes. New plant added to listview in main activity.
Result: Pass		

Table 10: S-14.2 Test Case.

Test Case S-14.2		
Pre-Condition: Leaving blanks in dialog fragment		
Steps	Expected Results	Actual Results
1. User opens dialog fragment in main activity by pressing floating action button	Dialog fragment opens to allow user to add new plant and corresponding details	Dialog fragment opens to allow user to new plant and corresponding details
2. Filling text boxes with numbers and letters, as well as leaving them blank	Text boxes are filled with user inputs	Text boxes are filled with user inputs
3. Pressing save button	If any textboxes are left blank, the app prompts the user to fill them out. Cannot proceed to save before doing so. Once filled, user input is saved and the dialog fragment closes. New plant added to listview in main activity.	Application crashes
Result: Pass		

Table 11: S-14.3 Test Case.

Test Case S-14.3		
Pre-Condition: Entering incorrect information in fragment		
Steps	Expected Results	Actual Results
1. User opens dialog fragment in main activity by pressing floating action button	Dialog fragment opens to allow user to add new plant and corresponding details	Dialog fragment opens to allow user to add new plant and corresponding details
Filling plant description textboxes with invalid inputs (ex: watering interval to 0, etc)	Text boxes are filled with user inputs	Text boxes are filled with user inputs
3. Pressing save button	Upon clicking the save button, the application warns the user about invalid inputs in the fragment and prompts them to change them. Cannot proceed to save before doing so. Once filled correctly, user input is saved and the dialog fragment closes. New plant added to listview in main activity.	No warnings given to users. Invalid inputs are saved in the plant profile
Result: Pass		

Table 12: S-14.4 Test Case.

Test Case S-14.4		
Pre-Condition: Entering already used plant name		
Steps	Expected Results	Actual Results
1. User opens dialog fragment in main activity by pressing floating action button	Dialog fragment opens to allow user to add new plant and corresponding details	Dialog fragment opens to allow user to add new plant and corresponding details
Entering name for plant that is already used by another plant in the database	Text boxes are filled with user inputs	Text boxes are filled with user inputs
3. Pressing save button	Upon clicking the save button, the application warns the user that the plant name is already being used by another plant in the database and prompts the user to change the name. Cannot proceed before doing so. Once filled correctly, user input is saved and the dialog fragment closes. New plant added to listview in main activity.	plant name is saved and there are now multiple plants in the database with identical names
Result: Pass		

4.3 Test Plan 3: Database

Requirement ID: S-16

In this section, we are creating a connection to the firebase database and saving plant profiles to it. The user should be able to add plants and retrieve from the firebase database. In addition, we are verifying if the information displayed in the plant catalog is also present in the real-time firebase database. Furthermore, we would also like to test the edition/adding feature in the plant catalog activity works as it should. Finally, we would also like to test if we can successfully add a user plant from the catalog.

Table 13: S-16.1 Test Case.

Test Case S-16.1		
Pre-Condition: User inputs all empty fields		
Steps	Expected Results	Actual Results
1. Open application	The phone application should open	The phone application opens
2. Click on Catalog	The phone application should redirect the user to the catalog list of plants	The phone application does redirect the user to the catalog list of plants
3. Click on add plant	The phone application should redirect the user to the add plant to database page, where the user can fill in the inputs	The phone application does redirect the user to the add plant to database
4. Leave all inputs blank and press save	The phone application should alert the user and focus on empty inputs to fill out	App alerts the user and focuses on inputs.
Result: Pass		

Table 14: S-13.2 Test Case.

Test Case S-16.2		
Pre-Condition: User inputs some empty fields		
Steps	Expected Results	Actual Results
1. Open application	The phone application should open	The phone application opens
2. Click on Catalog	The phone application should redirect the user to the catalog list of plants	The phone application does redirect the user to the catalog list of plants
3. Click on add plant	The phone application should redirect the user to the add plant to database page, where the user can fill in the inputs	The phone application does redirect the user to the add plant to database
4. Leave some inputs blank and press save	The phone application should alert the user and focus on empty inputs to fill out	App alerts the user and focuses on inputs.
Result: Pass		

Table 15: S-16.3 Test Case.

Test Case S-16.3		
Pre-Condition: User inputs fields in every input		
Steps	Expected Results	Actual Results
1. Open application	The phone application should open	The phone application opens
2. Click on Catalog	The phone application should redirect the user to the catalog list of plants	The phone application does redirect the user to the catalog list of plants
3. Click on add plant	The phone application should redirect the user to the add plant to database page, where the user can fill in the inputs	The phone application does redirect the user to the add plant to database
4. Fill out all fields	The phone application should save a plant to the database	The phone application does save a plant to the database
Result: Pass		

Table 16: S-16.4 Test Case.

Test Case S-16.4		
Pre-Condition: Plant database loads when going to plant database		
Steps	Expected Results	Actual Results
1. Open application	The phone application should open	The phone application opens
2. Click on Catalog	The phone application should redirect the user to the catalog list of plants and display the plants	The phone application displays the plants
Result: Pass		

Table 17: S-16.5 Test Case.

Test Case S-16.5		
Pre-Condition: Plant database loads when going to plant database after adding plant		
Steps	Expected Results	Actual Results
1. Open application	The phone application should open	The phone application opens
2. Click on Catalog	The phone application should redirect the user to the catalog list of plants and display the plants	The phone application does not display the plants
3. Add plant	The phone application should add a plant given the user input	The phone application does add a plant
4. View plants	The phone application should retrieve plants and display newly added plant at the bottom of the recyclerview	The phone application does display the recyclerview and the newly added plant
Result: Pass		

Table 18: S-16.6 Test Case.

Test Case S-16.6		
Pre-condition: Adding new plants into firebase database		
Steps	Expected Results	Actual Results
1. User wants to add new plant to the catalog	Upon clicking the add button, the application redirects to a new activity to fill out new plant info.	Upon clicking the add button, the application redirects to a new activity to fill out new plant info.
2. User fills out text boxes of new plant details	Text boxes are filled in by the user and show up in the catalog once the save button is pressed.	Text boxes are filled in by the user and show up in the catalog once the save button is pressed.
3. Plant entry is now in firebase database	Saved plant entry is found in the catalog and also displayed in the firebase database.	Saved plant entry is found in the catalog and also displayed in the firebase database.
Result: Pass		

Table 19: S-16.7 Test Case.

Test Case S-16.7		
Pre-condition: Editing plant information in catalog		
Steps	Expected Results	Actual Results
1. User wants to edit a specific plant detail in the catalog	Upon clicking the edit button, the application redirects to a new activity to replace old plant info.	.Upon clicking the edit button, the application redirects to a new activity to replace old plant info.
2. User fills out text boxes to update plant detail	Text boxes are filled in by the user and the updated information shows up in the catalog once the save button is pressed.	Text boxes are filled in by the user and the updated information shows up in the catalog once the save button is pressed.
3. Edited information is present in firebase	Saved plant entry is found in the catalog and also displayed in the firebase database.	Saved plant entry is found in the catalog and also displayed in the firebase database.
Result: Pass		

Table 20: S-16.8 Test Case.

Test Case S-16.8		
Pre-condition: Adding a catalog plant into user's own plant list		
Steps	Expected Results	Actual Results
1. User wants to add a plant in the catalog into their own plant list	Upon clicking the add button next to the specific plant, the application redirects to a new activity where the user gives the new plant a name.	Upon clicking the add button next to the specific plant, the application redirects to a new activity where the user gives the new plant a name.
2. User fills out name text box	The new activity displays the plant type and its associated information. User inputs a new name for the plant to be added.	The new activity displays the plant type and its associated information. User inputs a new name for the plant to be added.
3. User clicks add button	.User is redirected to their own plant list where they can view their newly added plant.	.User is redirected to their own plant list where they can view their newly added plant.
Result: Pass		

Table 21: S-16.9 Test Case.

Test Case S-16.9		
Pre-condition: Display user plant list		
Steps	Expected Results	Actual Results
1. In the homepage, user clicks on userPlantListActivity.java	User is redirected to a new activity where they're plant list is displayed.	User is redirected to a new activity where they're plant list is displayed.
Result: Pass		

Table 22: S-16.10 Test Case.

Test Case S-16.10		
Pre-condition: Display plant profile		
Steps	Expected Results	Actual Results
1. User clicks on specific plant in their own plant list	Application redirects the user to new activity where the information for that specific plant is displayed.	Application redirects the user to new activity where the information for that specific plant is displayed.
Result: Pass		

4.4 Test Plan 4: Hardware

Requirement ID: C-1

In this section, we are creating a connection between the microcontroller and the firebase database. In addition, we are verifying whether the microcontroller can read the information from the moisture sensor, send it to Firebase and confirm it.

Table 23: C-1.1 Test Case.

Test Case C-1.1		
Pre-Condition: Microcontroller is powered		
Steps	Expected Results	Actual Results
1. Setup ESP32 library in ArduinoIDE	Be able to successfully compile code to the Sparkfun ESP32	Successfully compiled code to the Sparkfun ESP32
Result: Pass		

Table 24: C-1.2 Test Case.

Test Case C-1.2		
Pre-Condition: Microcontroller is powered		
Steps	Expected Results	Actual Results
1. Connect microcontroller to WiFi network	The microcontroller successfully connects to WiFi and return no error code	The microcontroller does successfully connect to WiFi with no errors
2. Send data to Firebase	The microcontroller must successfully stores data on Firebase	The microcontroller does successfully store data on Firebase
Result: Pass		

Table 25: C-1.3 Test Case.

Test Case C-1.3		
Pre-Condition: Microcontroller is powered		
Steps	Expected Results	Actual Results
1. Connect microcontroller to WiFi network	The microcontroller successfully connects to WiFi and return no error code	The microcontroller does successfully connect to WiFi with no errors
2. Receive data from Firebase	The microcontroller must successfully fetch data on Firebase	The microcontroller does successfully fetch data from Firebase
Result: Pass		

Table 26: C-1.4 Test Case.

Test Case C-1.4		
Pre-condition: WiFi available		
Steps	Expected Results	Actual Results
1. Enter SSID and password for WiFi in sketch and upload it to the SparkFun ESP32 Thing.	Upload successful.	Upload successful.
2. Open serial to see WiFi connection.	Click on serial console in Arduino IDE; message saying WiFi connection is established should appear with the IP.	Connection successful and IP is shown.
Result: Pass		

Table 27: C-1.5 Test Case.

Test Case C-1.5		
Pre-condition: PC and hardware wired		
Steps	Expected Results	Actual Results
1. Plug sensor to dry soil and upload the code.	"Upload successful" message.	"Upload successful" message.
2. Open serial to observe the values.	Serial prints values below 30.	Values ranging in between 15 and 25 for various plants with dry soil.
Result: Pass		

Table 28: C-1.6 Test Case.

Test Case C-1.6		
Pre-condition: PC and hardware wired		
Steps	Expected Results	Actual Results
1. Plug sensor to soil that was just watered and upload the code.	"Upload successful" message.	"Upload successful" message.
2. Open serial to observe the values.	Serial prints values above 50.	Values ranging in between 50 and 90 for various plants with wet soil.
Result: Pass		

Table 29: C-1.7 Test Case.

Test Case C-1.7		
Pre-condition: PC and hardware wired		
Steps	Expected Results	Actual Results
1. Leave the sensor in soil that was watered overnight with MC working (>8 hours).	Soil dries up a bit, original values above 50.	Value after watering was 80 and soil dried up a bit after 12 hours.
2. Open serial to observe the values.	Serial prints values that are below the previously.	New value was 40, which is below 80.
Result: Pass		

Table 30: C-1.8 Test Case.

Test Case C-1.8		
Pre-condition: PC and hardware wired, WiFi available		
Steps	Expected Results	Actual Results
1. Plug sensor to soil and upload the code with valid WiFi SSID and password.	Upload and connection successful.	Upload successful, connection established.
2. Open serial to observe the values.	Serial prints values that make sense given soil moisture.	Serial prints 40, which makes sense for plants soil moisture level.
3. Check if the message for Firebase upload is PASSED or FAILED.	Serial prints the value of sensor and UNIX time with PASSED message.	Serial prints 40 and current time (as of Greenwich) along with "Passed" and data ID.
4. Check Firebase database in browser and compare the value, time and ID.	All data is identical.	All data is identical.
Result: Pass		

4.5 Test Plan 5: Notifications

Requirement ID: S-1

In this section, we are verifying the functionality of the push notifications triggered by the cloud functions in Firebase. The notifications should alert the user of low or high temperature, moisture or light data posted to the database.

Table 31: S-1.1 Test Case.

Test Case S-1.1		
Post temperature data with app open		
Steps	Expected Results	Actual Results
1. Open app	Application should open	Application opens
2. Sign in	Sign in automatically	Sign in automatically
3. Post data	Data should post	Data posts
4. Receive notification	A notification should alert the user	A notification alerts the user that data was posted
Result: Pass		

Table 32: S-1.2 Test Case.

Test Case S-1.2		
Post temperature data with app minimized		
Steps	Expected Results	Actual Results
1. Open app	App should open	App opens
2. Sign in	Sign in automatically	Signs in automatically
3. Minimize app	App should minimize	App minimizes
4. Post data	Data should be posted	Data posts
5. Receive notification	Notification should be received by the user	Notification is received by the user
Result: Pass		

Table 33: S-1.3 Test Case.

Test Case S-1.3		
Post temperature data with app closed		
Steps	Expected Results	Actual Results
1. Keep app closed	App should be closed	App is closed
2. Post data	Data should be posted	Data gets posted
3. Receive notification	Notification should alert user that data was posted	Notification does NOT alert user
Result: Fail		

Table 34: S-1.4 Test Case.

Test Case S-1.4		
Post low temperature data		
Steps	Expected Results	Actual Results
1. Open app	App should open	App opens
2. Sign in	Sign in automatically	Signs in automatically
3. Post low temperature data	Data should post	Data posts
4. Receive notification saying that the temperature is low	Notification should alert the user that the data is low	Notification is not able to determine the value of the posted data yet
Result: Fail		

Table 35: S-1.5 Test Case.

Test Case S-1.5		
Post high temperature data		
Steps	Expected Results	Actual Results
1. Open app	App should open	App opens
2. Sign in	Sign in automatically	Signs in automatically
3. Post high temperature data	Data should post	Data posts
4. Receive notification saying that the temperature is high	Notification should alert the user that the data is high	Notification is not able to determine the value of the posted data yet
Result: Fail		

Table 36: S-1.6 Test Case.

Test Case S-1.6		
Post adequate temperature data		
Steps	Expected Results	Actual Results
1. Open app	App should open	App opens
2. Sign in	Sign in automatically	Signs in automatically
3. Post adequate temperature data	Data should post	Data posts
4. Do not receive notification	Notification should not be received	User receives notification anyway
Result: Fail		

5. Definition of Done

Our definition of done includes a table for each user story as well as its corresponding checklist. When a sprint task within the PBI passes each requirement in the checklist, they're statuses are marked as complete. Conversely, if a particular task is unfinished/does not check all requirements, they are either listed as WIP or incomplete.

Table 37: Definition of Done for A-3.

Story ID: A-3	
User Story: Obtain Moisture Sensor	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Unit test performed and passed - Test document updated - Sign off by product owner	Status
1. Find a moisture sensor that is within 10$ and that can read soil moisture.	Not applicable
2. Buy motion sensor	Not applicable
3. Test moisture sensor	Complete
4. Translate incoming sensor data to sensible, usable information	Complete

Table 38: Definition of Done for A-8.

Story ID: A-8	
User Story: Sprint 3 planning	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Design document updated - Test document updated - Project Backlog updated - Sprint 3 Backlog updated - Sprint velocity calculated - Sign off by product owner	Status
1. Get feedback on tasks, and update the Sprint 2.	Complete
2. Meet with teammates and discuss what features will be implemented and what bugs should be fixed in Sprint 3.	Complete
3. Clean up and reorganize the backlog based on previous tasks.	Complete

Table 39: Definition of Done for A-9.

Story ID: A-9	
User Story: Testing documentation	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Test document updated - Sign off by product owner	Status
1. Write tests for hardware (new sensors mostly).	Complete
2. Test the hardware. Light test, Moisture test, Temperature test	Complete
3. Write tests for software.	Complete
4. Write Definition of Done for tasks.	Complete

Table 40: Definition of Done for A-10.

Story ID: A-10	
User Story: Design documentation	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Design document updated - Sign off by product owner	Status
1. Update the Android wireframe for the project.	Complete
2. Update system architecture for the project.	Complete
3. Update hardware architecture for the project.	Complete
4. Update software architecture for the project.	Complete
5. Update the possible use cases.	Complete
6. Write down the test plan.	Complete
7. Determine the price of components.	WIP
8. Put additional information to the appendix.	Complete
9. Write down the intro.	Complete

Table 41: Definition of Done for C-1.

Story ID: C-1	
User Story: Moisture level	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Unit test performed and passed - Integration test performed and passed - Design document updated - Test document updated - Code review complete - Code build with no error - Sign off by product owner	Status
1. Setup microcontroller with wifi module and current time	Complete
2. Calibrate incoming sensor data and translate to sensible, usable information	Complete
3. Setup hardware and write function in software capable of reading and returning moisture levels	Complete
4. Setup function that takes time from server and translates it into UNIX format at the time of sensor reading	Complete
5. Send resulting data to firebase	Complete
6. Get confirmation from Firebase.	Complete
7. Refactor function writeToDatabase()	Complete

Table 42: Definition of Done for C-2.

Story ID: C-2	
User Story: Light level	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Unit test performed and passed - Integration test performed and passed - Design document updated - Test document updated - Code review complete - Code build with no error - Sign off by product owner	Status
1. Calibrate incoming sensor data and translate to sensible, usable information	Complete
2. Setup hardware and write function in software capable of reading and returning light levels	Complete
3. Send resulting data to firebase	Complete
4. Refactor function writeToDatabase()	Complete

Table 43: Definition of Done for C-3.

Story ID: C-3	
User Story: Temperature level	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Unit test performed and passed - Integration test performed and passed - Design document updated - Test document updated - Code review complete - Code build with no error - Sign off by product owner	Status
1. Calibrate incoming sensor data and translate to sensible, usable information	Complete
2. Setup hardware and write function in software capable of reading and returning temperature levels	Complete
3. Send resulting data to firebase	Complete
4. Refactor function writeToDatabase()	Complete

Table 44: Definition of Done for E-4.

Story ID: E-4	
User Story: Device size	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Design document updated - Test document updated - Sign off by product owner	Status
1. Choose internal components (MC, battery, etc.)	Incomplete
2. Design PCB	Complete
3. Design external look of device to contain internal components	WIP

Table 45: Definition of Done for E-5.

Story ID: E-5	
User Story: Environmental resistance	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Design document updated - Test document updated - Research data peer reviewed - Sign off by product owner	Status
1. Research various materials that can withstand the outside environment	Incomplete
2. Apply the selected material to the external hardware design	Incomplete

Table 46: Definition of Done for S-1.

Story ID: S-1	
User Story: Moisture Notification	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Unit test performed and passed - Integration test performed and passed - Design document updated - Test document updated - Code review complete - Code build with no error - Sign off by product owner	Status
1. Create a push notification for moisture level	Complete
2. Define cloud functions for Firebase	Complete
3. Trigger defined cloud functions when user values are too low	Complete
4. Allow user to log out from his account	Complete
5. Allow the user to sign in with different credentials after opening the app.	Complete

Table 47: Definition of Done for S-11.

Story ID: S-11	
User Story: Firebase - Read moisture	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Unit test performed and passed - Integration test performed and passed - Design document updated - Test document updated - Sign off by product owner	Status
1. Read and record data from the firebase database	WIP
2. Verify if incoming data for user plants are too high/low compared to ideal plant data in the firebase database	WIP

Table 48: Definition of Done for S-15.

Story ID: S-15	
User Story: Plant research	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Researched data peer reviewed - Sign off by product owner	Status
1. Research done on various types of houseplant	Complete
2. Obtain specific information relating to the ideal growth environment for each specific houseplant	Complete
3. Translate researched information into sensible, usable data	Complete

Table 49: Definition of Done for S-16.

Story ID: S-16	
User Story: Database	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Unit test performed and passed - Integration test performed and passed - Design document updated - Test document updated - Code review complete - Code build with no error - Sign off by product owner	Status
1. Filling the realtime firebase database with plant types	Complete
2. Edit button to modify plant entries in catalog/firebase database	Complete
3. Edit tuples given plant ID	Complete
4. Implement activity that allows user to add a plant from the catalog into their own list	Complete
5. Implement list for user's plants with relationship in the database through client	Complete
6. Display user's plants in a list	Complete
7. Display individual plant profiles of the user's list	Complete
8. Getting matching plant profiles	Complete
9. Compare the data related to user plants with those in the catalog	WIP
10. Send notification to user if discrepancies are present between catalog plant info and user plant info	Incomplete
11. Upload pictures of plants to user's own database	Incomplete
12. Upload pictures of plants to general community database	Incomplete

Table 50: Definition of Done for S-20.

Story ID: S-20	
User Story: Main folder refactoring	
DoD Checklist for this PBI: - Requirements mentioned in PBI are satisfied - Design document updated - Test document updated - Code review complete - Code build with no error - Sign off by product owner	Status
1. Refactor the main folder, since naming conventions are creating bugs	Complete
2. Debug code that failed acceptance test.	Complete