

Healthy Leaves

Sprint 2 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.....	5
1.1 Product	5
1.2 Functionality.....	5
1.3 Benefits and Goals	5
1.4 Potential Users.....	5
2. Requirements.....	6
2.1 Sprint Backlog	9
3. Design Document.....	13
3.1 Android Application Wireframes.....	13
3.2 System Architecture	14
3.3 Hardware Architecture	15
3.4 Software Architecture	16
3.5 Use Cases and Sequence Diagrams.....	19
3.5.1 Use Case 1	20
3.5.1 Use Case 2	21
3.5.1 Use Case 3	22
3.5.1 Use Case 4	23
4. Testing.....	24
4.1 Test Plan 1: Firebase Connection.....	24
4.2 Test Plan 2: Plant Profile and Display	25
4.3 Test Plan 3: Database.....	28
4.2 Test Plan 4: SparkFun ESP32 MCU	31

5. Definition of Done.....	33
----------------------------	----

Table of Figures

<i>Figure 1: Healthy Leaves wireframe user experience.....</i>	<i>13</i>
<i>Figure 2: System Architecture of Firebase authentication.</i>	<i>14</i>
<i>Figure 3: Hardware architecture.....</i>	<i>15</i>
<i>Figure 4: Software architecture of Healthy Leaves project.....</i>	<i>16</i>
<i>Figure 5: Entity relationship diagram (ER-diagram) of Healthy Leaves project.....</i>	<i>17</i>
<i>Figure 6: Database diagram of Healthy Leaves project.....</i>	<i>19</i>
<i>Figure 7: Use case 1.....</i>	<i>20</i>
<i>Figure 8: Use case 2.....</i>	<i>21</i>
<i>Figure 9: Use case 3.....</i>	<i>22</i>
<i>Figure 10: Use case 4.....</i>	<i>23</i>

Table of Tables

Table 1: Backlog Information.....	6
Table 2: Sprint 1.	9
Table 3: Sprint 2.	11
Table 4: S-10.1 Test Case.....	24
Table 5: S-10.2 Test Case.....	24
Table 6: S-10.3 Test Case.....	25
Table 7: S-10.7 Test Case.....	25
Table 8: S-14.1 Test Case.....	26
Table 9: S-14.2 Test Case.....	26
Table 10: S-14.3 Test Case.....	27
Table 11: S-14.4 Test Case.....	27
Table 12: S-16.1 Test Case.....	28
Table 13: S-13.2 Test Case.....	29

Table 14: S-16.3 Test Case.....	29
Table 15: S-16.4 Test Case.....	30
Table 16: S-16.5 Test Case.....	30
Table 17: C-1.1 Test Case.	31
Table 18: C-1.2 Test Case.	31
Table 19: C-1.3 Test Case.	32
Table 20: DoD Breakdown.....	33

1. Introduction

1.1 Product

A device that collects data (moisture of soil, light levels, and temperature) and sends it to users' Android phone, providing relevant information and notifications on plants care.

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.

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 Information.

Legend											
A	Admin										
C	Computer										
E	Electrical										
S	Software										
Done											
WIP											
Story ID	Story Title	As a/an...	I want to...	so that...	Point	Sprint	Priority	Status	Conversation	Confirmation	Notes
S-10	Firebase Connection	Engineer	set up a connection to Firebase	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 platform 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	User	create and access various plant profiles	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 use the list of plant profiles?	
A-1	Sprint 2 Plan	Project Owner	create Sprint 2 Plan	the deliverables 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	Engineer	have a microcontroller module with WiFi	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	Engineer	have a testing document	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	Project Owner	have a design document	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 information about the project? 2. Is it submitted?	
C-1	Moisture Level	Engineer	know the plant moisture levels	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	User	receive a notification when the soil is too dry/wet	I water/don't water my plant to keep it healthy.	4	2	Must	WIP	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-11	Firebase: Read Moisture	Engineer	read data concerning moisture sensor	I can verify if the value is too low or too high.	1	2	Must	WIP	Once connected to Firebase, the app should obtain data values for moisture.	1. Can the app obtain the data values for moisture from the Firebase? 2. Can the data values be logged in the app?
S-15	Plant Research	Engineer	research plants	I can create a database with reference values.	4	2	Must	WIP	We need to research 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?
S-16	Database	Engineer	create multiple tables of various plant types	I can have references for notifications.	10	2	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?
A-3	Obtain Moisture Sensor	Engineer	have a moisture sensor	I can obtain moisture data for the prototype.	1	2	Must	WIP	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	Project Owner	create Sprint 3 Plan	the deliverables are documented and everyone got their task.	4	2	Must	WIP	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	Project Owner	have a testing document	the requirement satisfaction is verified and the possible bugs are detected.	5	2	Must	WIP	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	Project Owner	have a design document	the project description is available in a readable format.	5	2	Must	WIP	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 information about the project? 2. Is it submitted?
S-20	Main Folder Refactoring	Engineer	have proper naming of the project folders and classes	the project doesn't bug and my teammates can read the code	2	2	Must	To do	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	User	have a device that does not occupy more place than a normal plant pot	the overall design of my living space is not affected.	2	2	Must	To do	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 ensemble in a positive way?
E-5	Environmental Resistance	User	have a device working in various environmental conditions	I can install the device for outside plants.	2	2	Could	To do	Plant owners may grow plants outdoors, thus it is preferable that the device does not break down when face with outside environment.	1. Does the device keep working when put outside? 2. How long does it take to break down given that it constantly rains?
S-17	Moisture Interface	User	see the current and the past moisture levels	I can adjust my watering routine	5	3	Must	WIP	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	User	turn on/off the device	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?
E-2	Charging	User	charge my device	It keeps working.	5	3	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?
C-2	Light Level	Engineer	know the plant light levels	I can tell the user whether the plant has enough light.	4	3	Should	To do	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	Engineer	know the plant temperature levels	I can tell the user whether the environment is too cold or hot for the plant.	4	3	Should	To do	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?
S-2	Light Notification	User	receive a notification when the light levels are too low/high	I move my plant to a better place in the house.	4	3	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	User	receive a notification when the temperature levels are too low/high	I change the temperature of the room, or move it to another room.	4	3	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?
S-4	Recommendations	User	receive recommendation on healthier practices for the plant	my plant stays healthy.	5	3	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	User	identify my plant with a picture	I can understand it better.	10	3	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.	1. Can the user take a picture from the app? 2. Does the app determine the plants' name based on the picture?
E-3	Rugged Design	User	have a device that it low maintenance	I don't have to spend my time repairing it.	5	3	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	User	configure how often I receive the notifications	the app is more adapted to my lifestyle.	2	3	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	User	receive reminders to cut leaves and repot the plant	the plant growth is controlled.	3	3	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	User	access plant database where I can link my plant to existing plant models	I can get specific information on how to take care of my plant.	3	3	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	Engineer	read data concerning light sensor	I can verify if the value is too low or too high.	1	3	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	Engineer	read data concerning temperature sensor	I can verify if the value is too low or too high.	1	3	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	Engineer	have a light sensor	I can obtain light exposure data for the prototype.	1	3	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	Engineer	have a temperature sensor	I can obtain temperature data for the prototype.	1	3	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	User	see the current and the past light exposure levels	I can see what spots are best in my home.	5	3	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	User	see the current and the past temperature levels	I can adjust the temperature so the plants grow better.	5	3	Should	To do	The user should be able to see the temperature 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 temperature levels? 3. Can the user see the graphical representation of temperature levels over time?	
A-11	Sprint 3 Testing Doc	Engineer	have a testing document	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	Project Owner	have a design document	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 information about the project? 2. Is it submitted?	

2.1 Sprint Backlog

Our first sprint was a disaster 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.

Sprint 2 was planned based on tasks we did in sprint 1; the goal is to make a working product, so that we can do bug fixes in Sprint 3. Some tasks carried over from Sprint 1 to Sprint 2, notably the database implementation, S-16.

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	To do ▾	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.

Table 3: Sprint 2.

Sprint 2 Goal: Fully integrate hardware and software through Firebase.							
Story ID	Task Id	Title	Task Description	Time	Status	By	Notes
S-11	S-11.1	Read Data	Read and record data from the Firebase.	4	WIP ▾	Vily ▾	
	S-11.2	Low Values	Verify if the values are low or high, given recommendations	1	WIP ▾	Vily ▾	
S-1	S-1.1	Push Notification	Create a push notification for water level.	5	To do ▾	Vily ▾	
	S-1.2	Cloud Function	Define cloud functions for Firebase.	5	To do ▾	Vily ▾	
	S-1.3	Trigger	Trigger defined cloud functions when user values are too low.	5	To do ▾	Vily ▾	
S-15	S-15.1	Plant Types	Research various plant types.	5	WIP ▾	Jun ▾	
	S-15.2	Plant Needs	Based on previous task, find what each type of plant needs.	5	To do ▾	Jun ▾	
	S-15.3	Quantification	Quantify the plant needs to numbers that we can work with.	5	To do ▾	Jun ▾	
S-16	S-16.9	Fill	Fill the database with found and/or calculated values.	5	To do ▾	Jun ▾	Preset Plant class objects to the Firebase database.
	S-16.10	Edit Database	Implement an edit function to database entries to modify their attributes.	3	WIP ▾	Thomas ▾	
		Edit Tuples	Given a plantID, edit Plant tuples from the database.	5	WIP ▾	Thomas ▾	editPlantDatabaseActivity receives an Intent to know on which tuple the edit should be done.
		Get userPlant Profile	Using the token generated by authentication, the userPlant object gets all the attributes of the plant.	4	To do ▾	Thomas ▾	
		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	To do ▾	Thomas ▾	
		Compare Attributes	Compare userPlan data to the database plant recommendations.	2	To do ▾	Thomas ▾	
		Send Notification	Send notification if the difference is present.	3	To do ▾	Thomas ▾	
		Store Pics	Upload pictures to database	5	To do ▾	Thomas ▾	
		OwnsA Relationship	The user can post picture of its plant to the general database.	5	To do ▾	Thomas ▾	
A-3	A-3.1	Research	Find a moisture sensor that is within 10\$ and that can read soil moisture.	0.5	Done ▾	Karl ▾	
	A-3.2	Buy	Buy it.	0.5	WIP ▾	Karl ▾	
	A-3.3	Test	Test the sensor.		▾	▾	
	A-3.4	Data Translation	Translate data from the sensor to some sensible value, confirming C-1.3.	1	WIP ▾	Karl ▾	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	To do ▾	Karl ▾	
	E-4.3	External	Design the external look of the device (a box) that will contain the internal components.	5	To do ▾	Karl ▾	
E-5	E-5.1	Material Choice	Research various materials that can withstand the outside environment.	2	To do ▾	Karl ▾	
	E-5.2	Apply to Design	Apply the selected material to the external hardware design, defined in E-4.3.	1	To do ▾	Karl ▾	
S-20	S-20.1	Refactoring	Refactor the main folder, since naming conventions are creating bugs.	5	To do ▾	Jun ▾	
	S-20.2	Testing	Test the refactored folders.	5	To do ▾	Jun ▾	
A-8	A-8.1	Sprint 2 Scrum	Get feedback on tasks, and update the Sprint 2.	2	To do ▾	Daniel ▾	1 Hour for everyone else.
	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.	2	To do ▾	Daniel ▾	
	A-8.3	Backlog Grooming	Clean up and reorganize the backlog based on previous tasks.	3	To do ▾	Daniel ▾	
A-9	A-9.1	Hardware	Write tests for hardware (new sensors mostly).	3	To do ▾	Daniel ▾	
	A-9.2	Test	Perform tests on hardware.	3	To do ▾	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 ▾		
A-10	A-10.1	Wireframe	Update the Android wireframe for the project.	1	To do ▾	Daniel ▾	
	A-10.2	System Architecture	Update system architecture for the project.	1	To do ▾	Daniel ▾	
	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.	2	To do ▾	Daniel ▾	
	A-10.5	Use Cases	Update the possible use cases.	5	To do ▾	Daniel ▾	
	A-10.6	Testing	Write down the test plan.	3	To do ▾	Daniel ▾	
	A-10.7	Appendix	Put additional information to the appendix.	2	To do ▾	Daniel ▾	
	A-10.8	Introduction	Write down the intro.	1	To do ▾	Daniel ▾	

3. Design Document

The current design document explores the Sprint 1 iteration of features included, giving the user the ability to register an account to the firebase database, create a plant profile, view its details and be able to receive notifications.

3.1 Android Application Wireframes

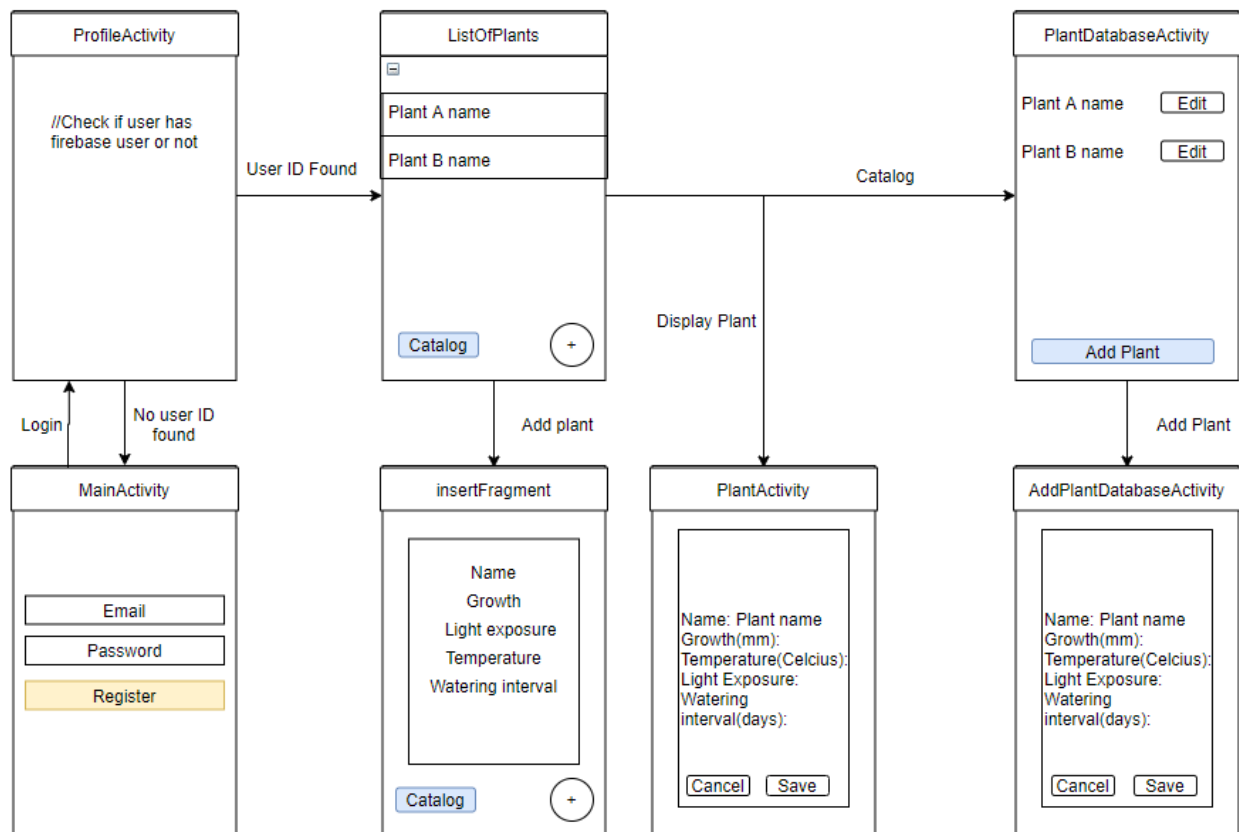


Figure 1: Healthy Leaves wireframe user experience.

The application wireframe can be described with *Figure 1*. As a user opens and starts the Healthy Leaves android application it will by default start the MainActivity due to Android Manifest declarations, it will directly go to the ProfileActivity which checks whether or not a userID can be found for this specific android device. If no userID was stored, the activity will be reverted to MainActivity where the user can create a user profile with the signup page. Else, if the userID is found, the activity ListOfPlants that displays a list of plants owned by the user is displayed (local database that will be changed in future sprints). It also gives access to expand the specific plant's information (and graphs in the future) and the

user can hit the plus circle button to add a new tuple to the local database of plants. The button catalog takes the user to PlantDatabaseActivity where a cloud-based Firebase database of all plants entered by any user on any device is displayed with their information too. A button add plant can be pressed to be taken to AddPlantDatabaseActivity where the user can write to the firebase database to store new types of plants with their information.

3.2 System Architecture

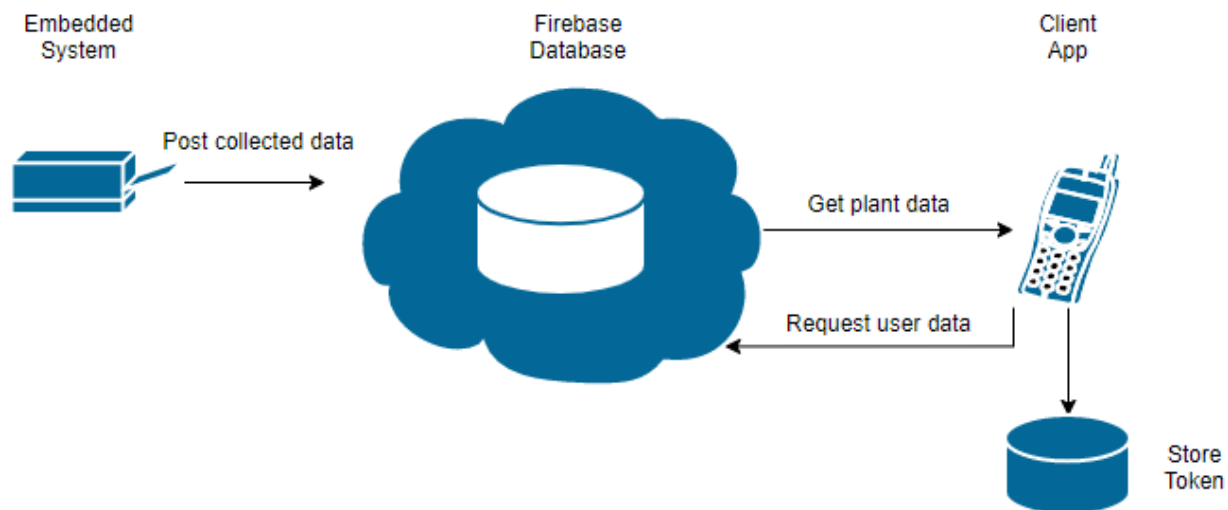


Figure 2: System Architecture of Firebase authentication.

The *Figure 2* 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.

3.3 Hardware Architecture

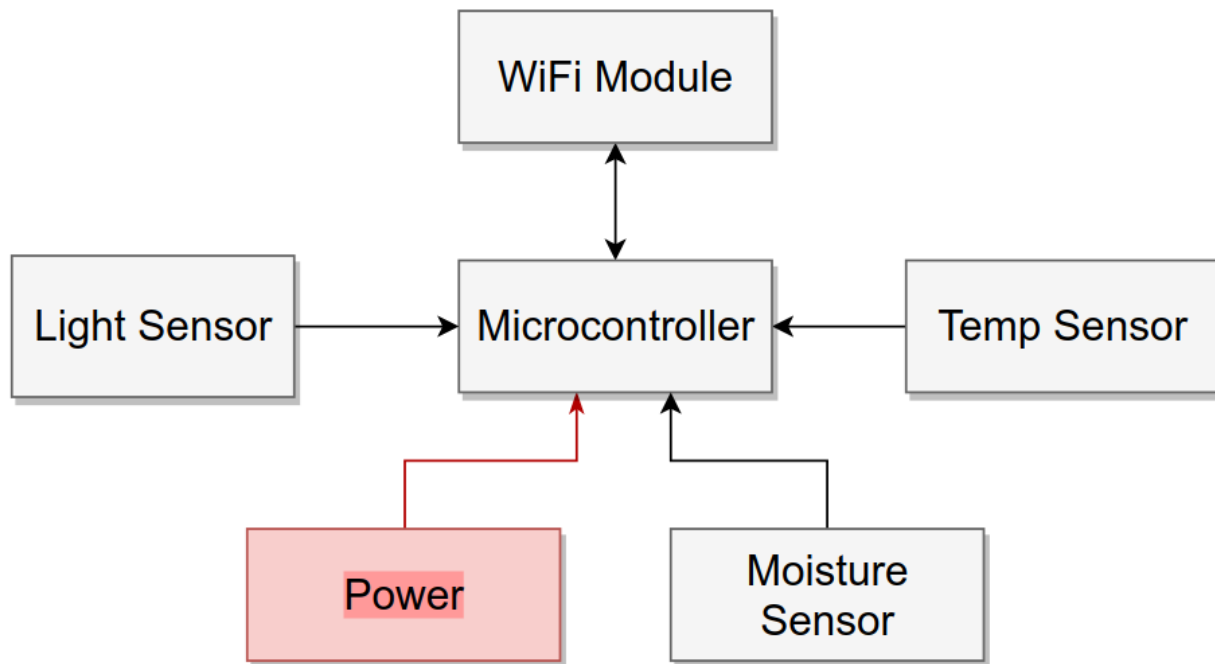


Figure 3: Hardware architecture.

Figure 3 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.

3.4 Software Architecture

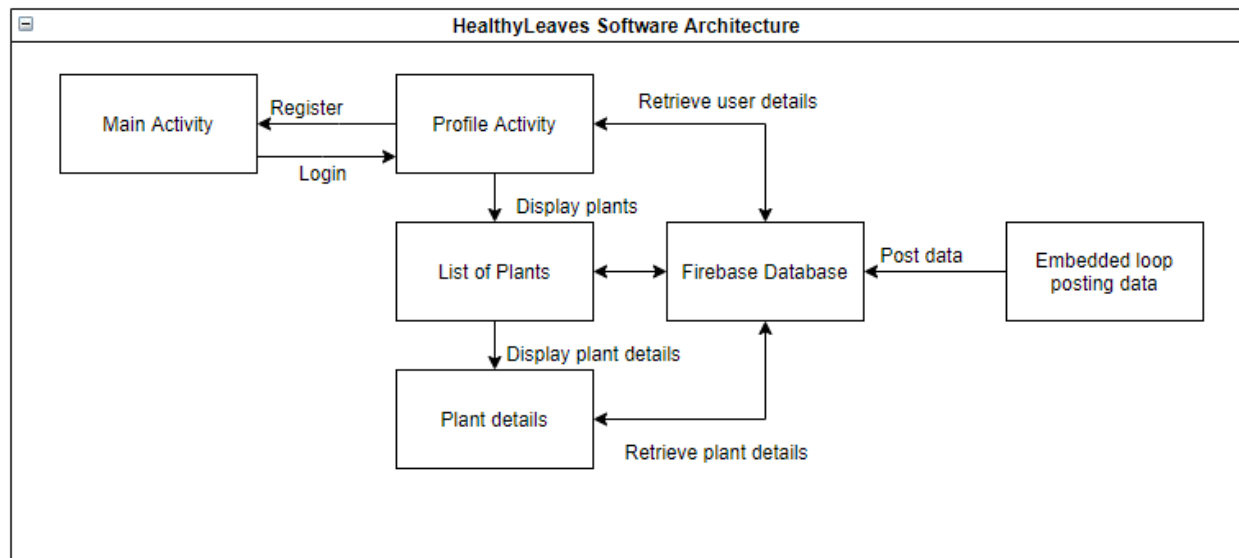


Figure 4: Software architecture of Healthy Leaves project.

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

The *Figure 5* 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.

OwensA 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, OwensA 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 OwensA 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 foreign key of the OwensA userPlantID to know to which specific plant owned by a user those measurements are referring to. Every measurement is only associated to one OwensA relationship while an OwensA 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.

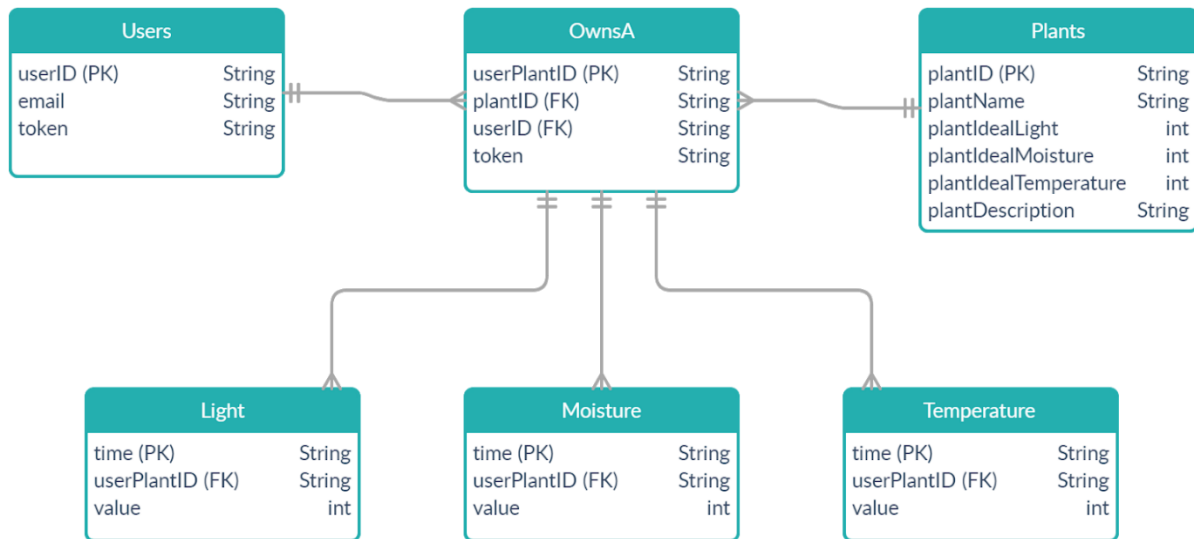


Figure 6: Database diagram of Healthy Leaves project.

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 5*.

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

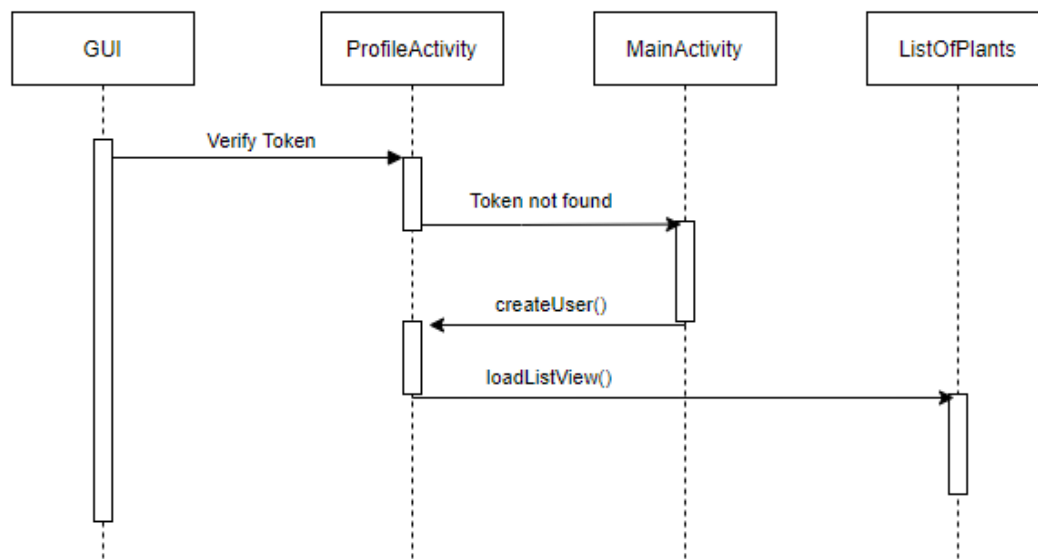


Figure 7: Use case 1.

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

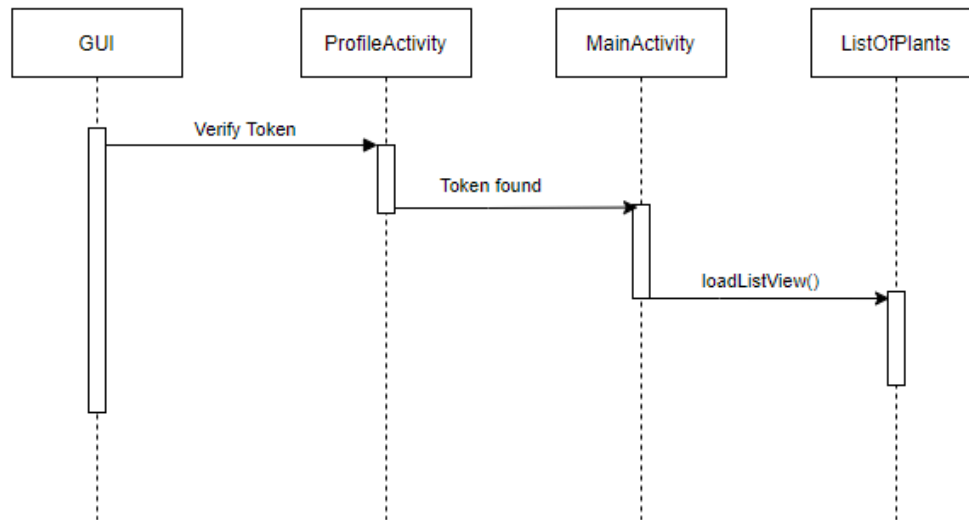


Figure 8: Use case 2.

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

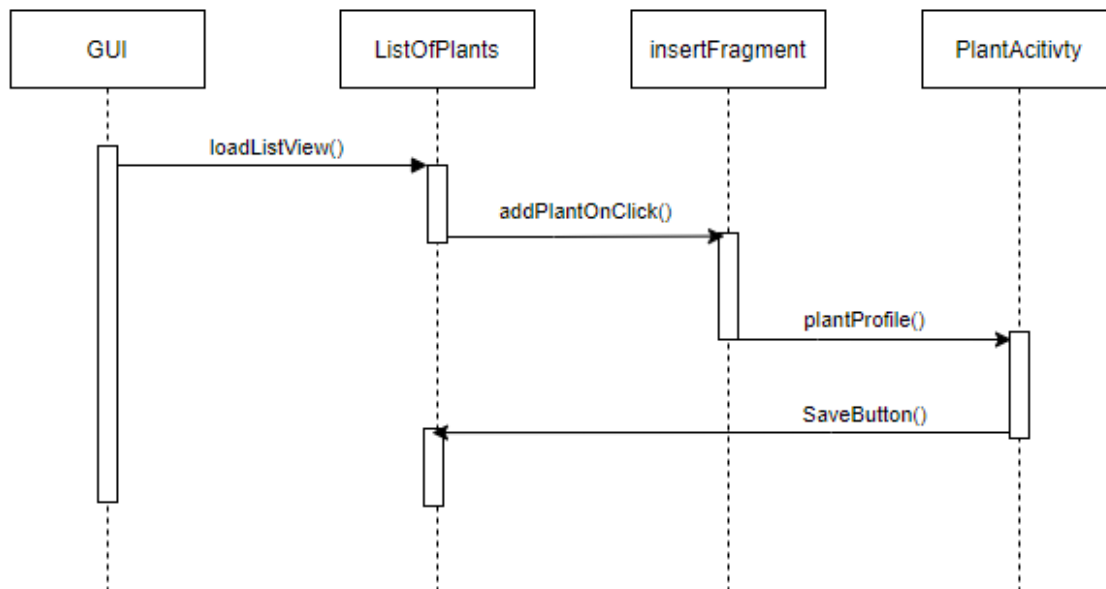


Figure 9: Use case 3.

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 PlantAcitivy will redirect the user to the ListOfPlants if the button "Save" is clicked.

3.5.1 Use Case 4

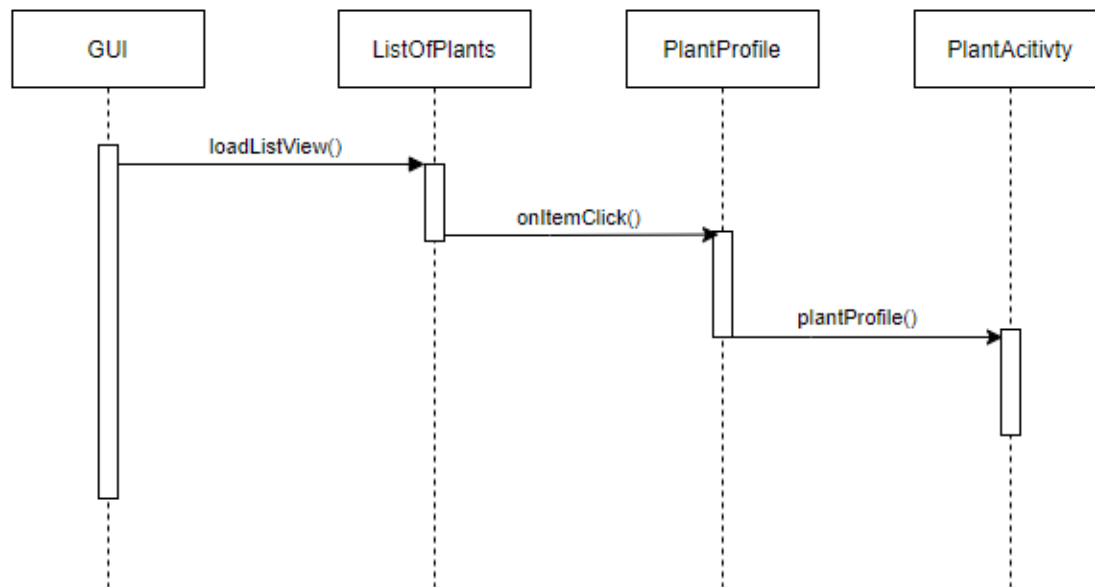


Figure 10: Use case 4.

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 4: 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 5: 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 6: 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 7: 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 8: 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 9: 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: Fail		

Table 10: 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: Fail		

Table 11: 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: Fail		

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.

Table 12: 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	The phone application crashes
Result: Fail		

Table 13: 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	The phone application crashes
Result: Fail 		

Table 14: 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 15: 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 does not display the plants
Result: Fail		

Table 16: 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		

4.2 Test Plan 4: SparkFun ESP32 MCU

Requirement ID: C-1

In this section, we are creating a connection between the microcontroller and the firebase database. This enables sending data to and from the firebase database.


Table 17: 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 18: 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 19: 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		

5. Definition of Done

Our definition of done include story confirmation (stakeholder side) as well as all other stakeholders' confirmations. That is, if a user can access and use something per backlog confirmation, it does not mean that it is well done. We decided that we will have different levels of confirmation:

1. Had a code review (naming conventions, comments, etc. correct).
2. Have been tested (refer to test plan).
3. Deployable (version compiles, runs and is merged into master/release).

Current situation is captured in Table 20.

Table 20: DoD Breakdown.

Story ID	DoD	Confirmation	Validation
S-10	Tested	Can the microcontroller connect to the Firebase Database? Can the user register to the Firebase Database? Can the application retrieve the user token?	Complete
S-16	Tested	Do we have an accessible database of various plants with their quantifiable information? Do we have an accessible database of the user owned plants and their quantifiable accessible data log? Can we access the database? Can we post to the database?	Incomplete, the database doesn't establish relationships between plants and users and therefore also don't hold quantifiable data log.
S-14	Tested	Can the plant profile be saved locally? Can the plant profile be saved remotely?	Complete
A-1	Deployable	Does each team member knows what to do in Sprint 2? Is the sprint 2 document complete?	Complete
A-2	Deployable	Can my microcontroller connect to the WIFI given this module?	Complete
A-4	Deployable	Can we verify all the requirements? Can we break down the app somehow?	Complete
A-5	Deployable	Does the design document contain all the information about the project? Is it submitted?	Complete
C-1	Have had a code review	Can the microcontroller read moisture levels? Is the data saved to the microcontroller?	Complete