Healthy Leaves

Sprint 3 Assignment

Presented to:

Professor Wahab Hamou-Lhaj

[Project GitHub Link](https://github.com/ItsKarlito/engineering-team-design-project)

|  |  |
| --- | --- |
| 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](#_Toc56369116)

[Table of Tables 3](#_Toc56369117)

[1. Introduction 5](#_Toc56369118)

[1.1 Product 5](#_Toc56369119)

[1.2 Functionality 5](#_Toc56369120)

[1.3 Benefits and Goals 5](#_Toc56369121)

[1.4 Potential Users 5](#_Toc56369122)

[2. Requirements 6](#_Toc56369123)

[2.1 Sprint Backlog 7](#_Toc56369124)

[3. Design Document 7](#_Toc56369125)

[3.1 Android Application Wireframes 8](#_Toc56369126)

[3.2 System Architecture 11](#_Toc56369127)

[3.3 Hardware Architecture 12](#_Toc56369128)

[3.4 Software Architecture 13](#_Toc56369129)

[3.5 Use Cases and Sequence Diagrams 16](#_Toc56369130)

[3.5.1 Use Case 1 17](#_Toc56369131)

[3.5.1 Use Case 2 18](#_Toc56369132)

[3.5.1 Use Case 3 19](#_Toc56369133)

[3.5.1 Use Case 4 20](#_Toc56369134)

[4. Testing 21](#_Toc56369135)

[4.1 Test Plan 1: Firebase Connection 21](#_Toc56369136)

[4.2 Test Plan 2: Plant Profile and Display 22](#_Toc56369137)

[4.3 Test Plan 3: Database 25](#_Toc56369138)

[4.4 Test Plan 4: Hardware 27](#_Toc56369139)

[4.5 Test Plan 5: Notifications 31](#_Toc56369140)

[5. Definition of Done 35](#_Toc56369141)

# Table of Figures

[*Figure 1: Healthy Leaves wireframe user experience.* 8](#_Toc56369148)

[Figure 2: Activity hierarchy for HealthyLeaves project. 10](#_Toc56369149)

[*Figure 3: System Architecture of Firebase authentication.* 11](#_Toc56369150)

[*Figure 4: Hardware architecture.* 12](#_Toc56369151)

[*Figure 5: Software architecture of Healthy Leaves project.* 13](#_Toc56369152)

[*Figure 6: Entity relationship diagram (ER-diagram) of Healthy Leaves project.* 14](#_Toc56369153)

[*Figure 7: Database diagram of Healthy Leaves project.* 16](#_Toc56369154)

[*Figure 8: Use case 1.* 17](#_Toc56369155)

[*Figure 9: Use case 2.* 18](#_Toc56369156)

[*Figure 10: Use case 3.* 19](#_Toc56369157)

[*Figure 11: Use case 4.* 20](#_Toc56369158)

# Table of Tables

[Table 1: Backlog Information. 6](#_Toc56370191)

[Table 2: S-10.1 Test Case. 21](#_Toc56370192)

[Table 3: S-10.2 Test Case. 21](#_Toc56370193)

[Table 4: S-10.3 Test Case. 22](#_Toc56370194)

[Table 5: S-10.7 Test Case. 22](#_Toc56370195)

[Table 6: S-14.1 Test Case. 23](#_Toc56370196)

[Table 7: S-14.2 Test Case. 23](#_Toc56370197)

[Table 8: S-14.3 Test Case. 23](#_Toc56370198)

[Table 9: S-14.4 Test Case. 24](#_Toc56370199)

[Table 10: S-16.1 Test Case. 25](#_Toc56370200)

[Table 11: S-13.2 Test Case. 25](#_Toc56370201)

[Table 12: S-16.3 Test Case. 26](#_Toc56370202)

[Table 13: S-16.4 Test Case. 26](#_Toc56370203)

[Table 14: S-16.5 Test Case. 27](#_Toc56370204)

[Table 15: C-1.1 Test Case. 28](#_Toc56370205)

[Table 16: C-1.2 Test Case. 28](#_Toc56370206)

[Table 17: C-1.3 Test Case. 28](#_Toc56370207)

[Table 18: C-1.4 Test Case. 29](#_Toc56370208)

[Table 19: C-1.5 Test Case. 29](#_Toc56370209)

[Table 20: C-1.6 Test Case. 30](#_Toc56370210)

[Table 21: C-1.7 Test Case. 30](#_Toc56370211)

[Table 22: C-1.8 Test Case. 31](#_Toc56370212)

[Table 23: S-1.1 Test Case. 32](#_Toc56370213)

[Table 24: S-1.2 Test Case. 32](#_Toc56370214)

[Table 25: S-1.3 Test Case. 33](#_Toc56370215)

[Table 26: S-1.4 Test Case. 33](#_Toc56370216)

[Table 27: S-1.5 Test Case. 34](#_Toc56370217)

[Table 28: S-1.6 Test Case. 34](#_Toc56370218)

[Table 29: Definition of Done for A-3. 35](#_Toc56370219)

[Table 30: Definition of Done for A-8. 35](#_Toc56370220)

[Table 31: Definition of Done for A-9. 36](#_Toc56370221)

[Table 32: Definition of Done for A-10. 36](#_Toc56370222)

[Table 33: Definition of Done for C-1. 37](#_Toc56370223)

[Table 34: Definition of Done for C-2. 38](#_Toc56370224)

[Table 35: Definition of Done for C-3. 38](#_Toc56370225)

[Table 36: Definition of Done for E-4. 38](#_Toc56370226)

[Table 37: Definition of Done for E-5. 39](#_Toc56370227)

[Table 38: Definition of Done for S-1. 39](#_Toc56370228)

[Table 39: Definition of Done for S-11. 40](#_Toc56370229)

[Table 40: Definition of Done for S-15. 40](#_Toc56370230)

[Table 41: Definition of Done for S-16. 41](#_Toc56370231)

[Table 42: Definition of Done for S-20. 42](#_Toc56370232)

# 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 Information.

## 2.1 Sprint Backlog

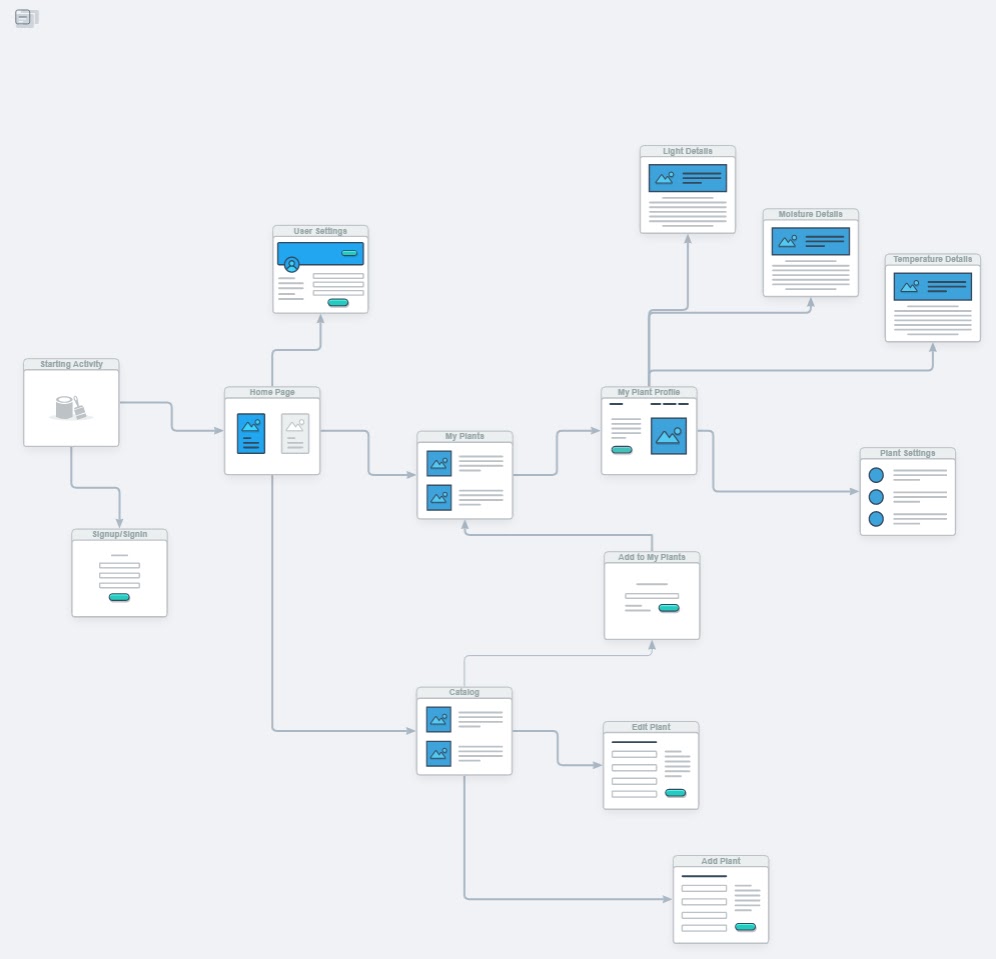
Our first sprint was not efficient due to midterm week; in addition, the goals we have set were too ambitious. After meeting with the whole team, we realized that we are not on the same page regarding how the project works and how it will look like. The situation was somewhat fixed, and we reworked the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

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.

# 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.

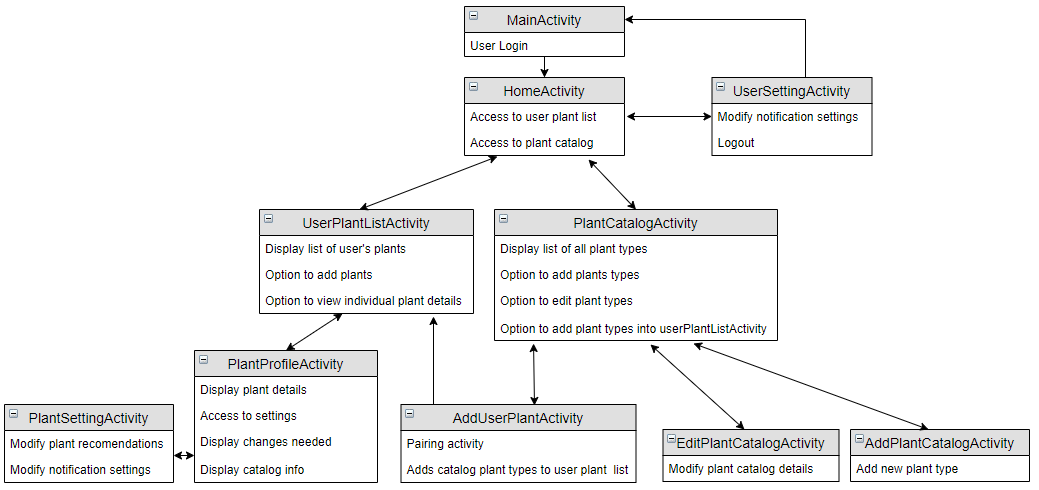
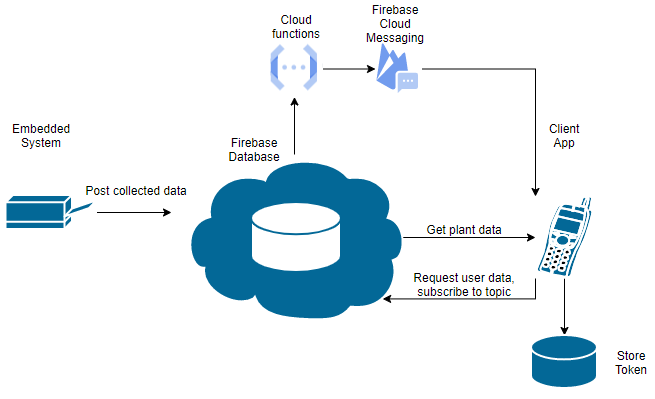


Figure 2: Activity hierarchy for HealthyLeaves project.

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 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. 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

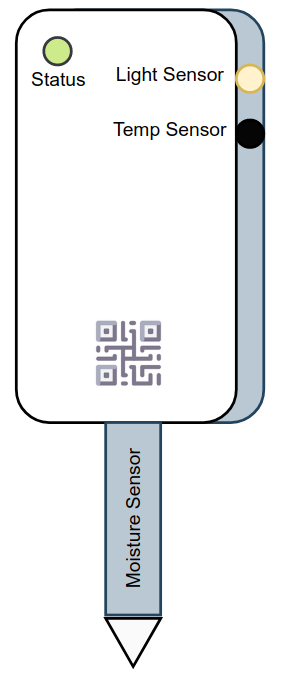
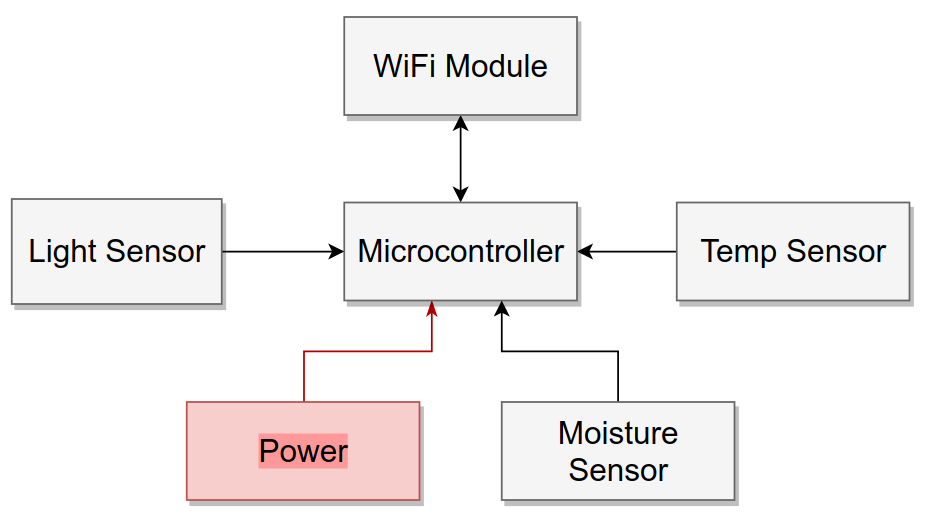


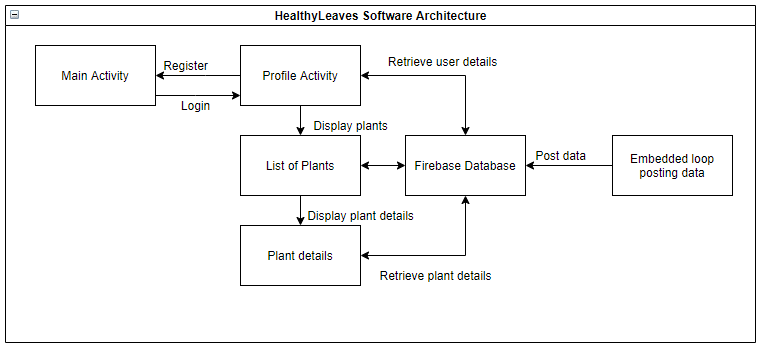
Figure 4: Sensor location on device.



*Figure 5: Hardware architecture.*

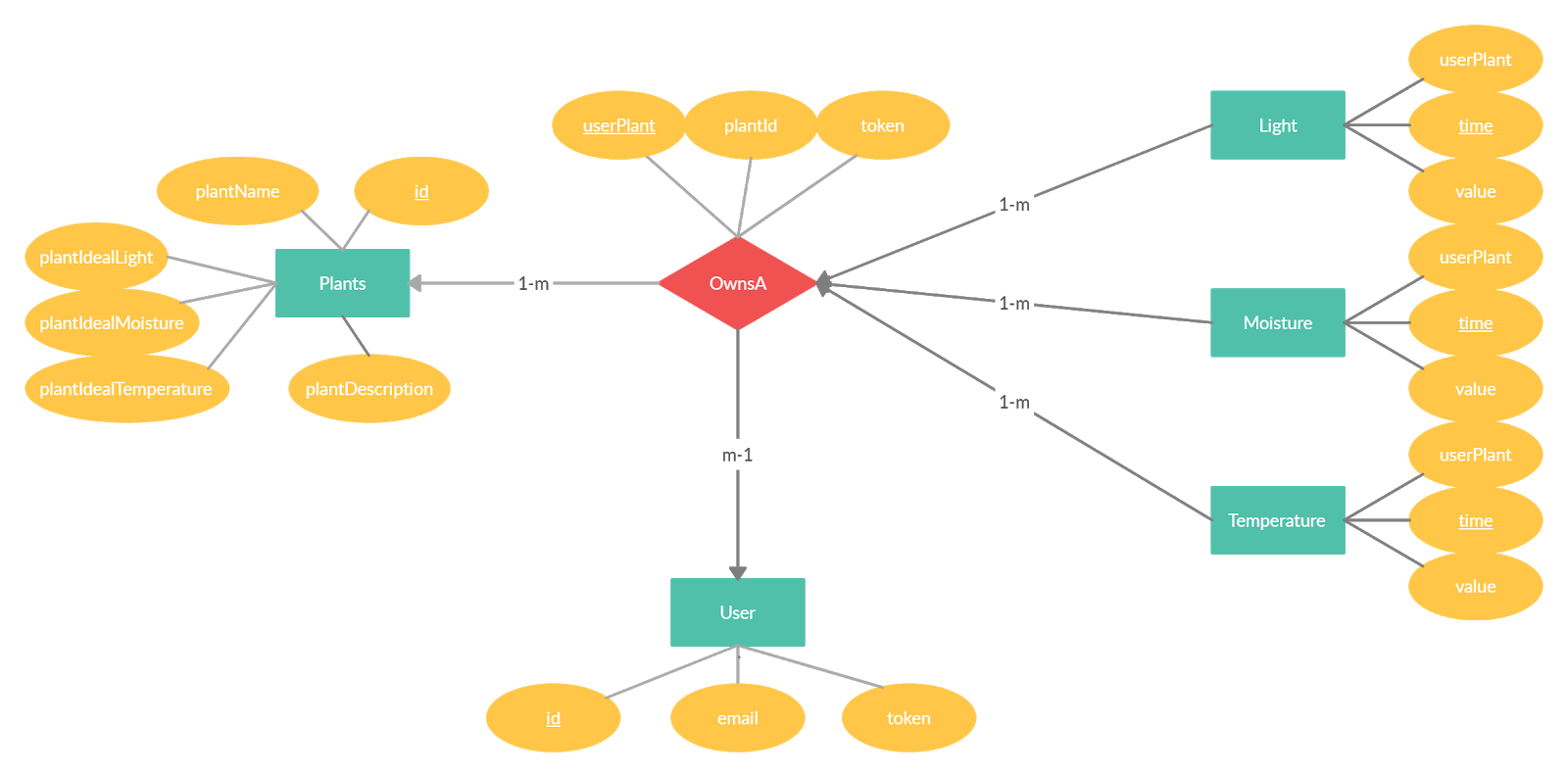
*Figure 3* shows the hardware architecture of the Healthy Leaves project. The device consists of three sensors connected to the microcontroller, SparkFun ESP32 Thing, through analog pins. The data is read and sent to Firebase through an onboard Wi-Fi chipset – ESP8266. Currently, for prototyping purposes, 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 6: 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.



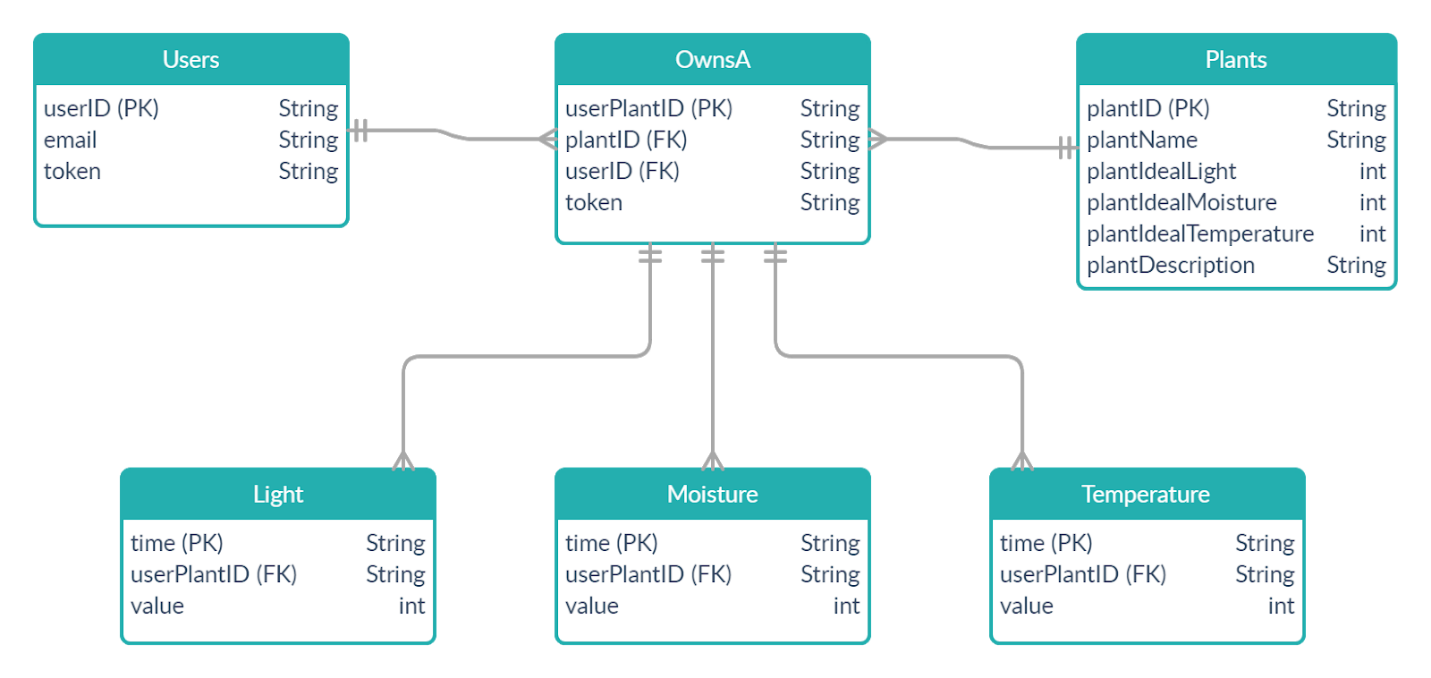
*Figure 7: Entity relationship diagram (ER-diagram) of Healthy Leaves project.*

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.

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 8: Database diagram of Healthy Leaves project.*

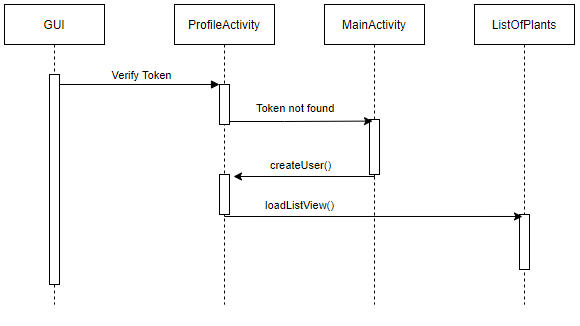
The figure above illustrates the database diagram that will be used to implement the firebase database of the Healthy Leaves project. It follows the logical relationships designed in the ER-diagram of *Figure 5*.

## 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.

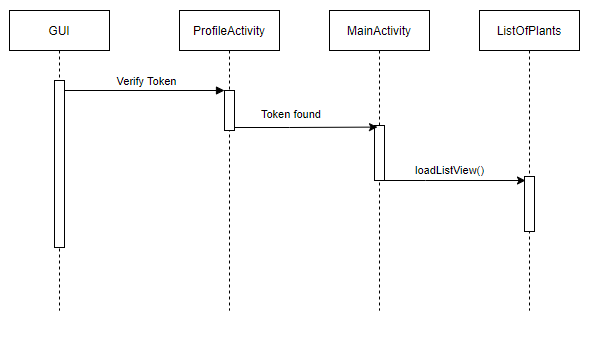
### 3.5.1 Use Case 1



*Figure 9: 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.

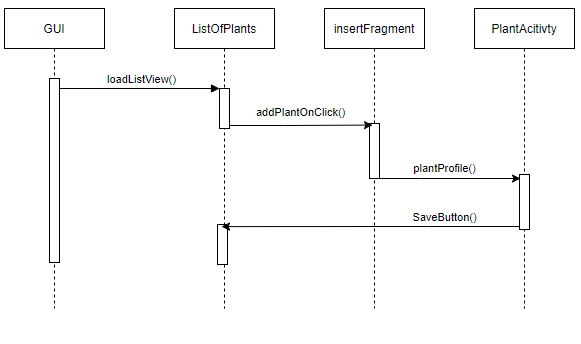
### 3.5.1 Use Case 2



*Figure 10: 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.

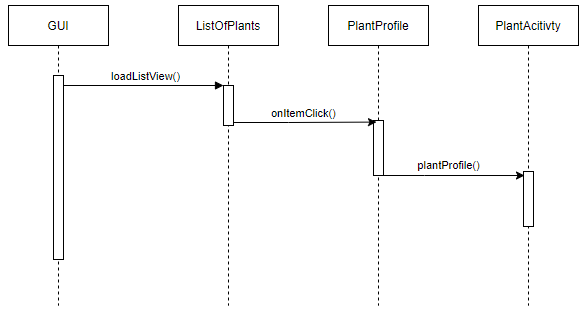
### 3.5.1 Use Case 3



*Figure 11: 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 12: 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 2: S-10.1 Test Case.

Table

Description automatically generated

Table 3: S-10.2 Test Case.

Table

Description automatically generated

Table 4: S-10.3 Test Case.

Table

Description automatically generated

Table 5: S-10.7 Test Case.

Table

Description automatically generated

## 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 6: S-14.1 Test Case.

Table

Description automatically generated

Table 7: S-14.2 Test Case.

A picture containing graphical user interface

Description automatically generated

Table 8: S-14.3 Test Case.

Table

Description automatically generated

Table 9: S-14.4 Test Case.

A picture containing graphical user interface, text

Description automatically generated

## 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 10: S-16.1 Test Case.

Table

Description automatically generated

Table 11: S-13.2 Test Case.

Table

Description automatically generated

Table 12: S-16.3 Test Case.

Table

Description automatically generated

Table 13: S-16.4 Test Case.

Table

Description automatically generated

Table 14: S-16.5 Test Case.

Table

Description automatically generated

## 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 15: C-1.1 Test Case.

Table

Description automatically generated

Table 16: C-1.2 Test Case.

Table

Description automatically generated

Table 17: C-1.3 Test Case.

Graphical user interface, application, table

Description automatically generated

Table 18: C-1.4 Test Case.

Table

Description automatically generated

Table 19: C-1.5 Test Case.

Table

Description automatically generated

Table 20: C-1.6 Test Case.

Table

Description automatically generated

Table 21: C-1.7 Test Case.

Table

Description automatically generated

Table 22: C-1.8 Test Case.

Table

Description automatically generated

## 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 23: S-1.1 Test Case.

Table

Description automatically generated

Table 24: S-1.2 Test Case.

Table

Description automatically generated

Table 25: S-1.3 Test Case.

Table

Description automatically generated

Table 26: S-1.4 Test Case.

Table

Description automatically generated

Table 27: S-1.5 Test Case.

Table

Description automatically generated

Table 28: S-1.6 Test Case.

Table

Description automatically generated

# 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 29: Definition of Done for A-3.

A picture containing timeline

Description automatically generated

Table 30: Definition of Done for A-8.

Table, treemap chart

Description automatically generated

Table 31: Definition of Done for A-9.

Timeline

Description automatically generated

Table 32: Definition of Done for A-10.

Table

Description automatically generated

Table 33: Definition of Done for C-1.

Text

Description automatically generated

Table 34: Definition of Done for C-2.

A picture containing table

Description automatically generated

Table 35: Definition of Done for C-3.

A picture containing table

Description automatically generated

Table 36: Definition of Done for E-4.

A picture containing graphical user interface

Description automatically generated

Table 37: Definition of Done for E-5.

A picture containing timeline

Description automatically generated

Table 38: Definition of Done for S-1.

A picture containing table

Description automatically generated

Table 39: Definition of Done for S-11.

A picture containing treemap chart

Description automatically generated

Table 40: Definition of Done for S-15.

Table

Description automatically generated

Table 41: Definition of Done for S-16.

A picture containing table

Description automatically generated

Table 42: Definition of Done for S-20.

Chart, timeline, treemap chart

Description automatically generated