Healthy Leaves

Sprint 2 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](#_heading=h.gjdgxs)

[Table of Tables 3](#_heading=h.30j0zll)

[1. Introduction 5](#_heading=h.1fob9te)

[1.1 Product 5](#_heading=h.3znysh7)

[1.2 Functionality 5](#_heading=h.2et92p0)

[1.3 Benefits and Goals 5](#_heading=h.tyjcwt)

[1.4 Potential Users 5](#_heading=h.3dy6vkm)

[2. Requirements 6](#_heading=h.1t3h5sf)

[2.1 Sprint Backlog 9](#_heading=h.2s8eyo1)

[3. Design Document 13](#_heading=h.26in1rg)

[3.1 Android Application Wireframes 13](#_heading=h.lnxbz9)

[3.2 System Architecture 14](#_heading=h.1ksv4uv)

[3.3 Hardware Architecture 15](#_heading=h.2jxsxqh)

[3.4 Software Architecture 16](#_heading=h.3j2qqm3)

[3.5 Use Cases and Sequence Diagrams 19](#_heading=h.1ci93xb)

[3.5.1 Use Case 1 20](#_heading=h.3whwml4)

[3.5.1 Use Case 2 21](#_heading=h.qsh70q)

[3.5.1 Use Case 3 22](#_heading=h.1pxezwc)

[3.5.1 Use Case 4 23](#_heading=h.2p2csry)

[4. Testing 24](#_heading=h.3o7alnk)

[4.1 Test Plan 1: Firebase Connection 24](#_heading=h.23ckvvd)

[4.2 Test Plan 2: Plant Profile and Display 25](#_heading=h.2grqrue)

[4.3 Test Plan 3: Database 28](#_heading=h.2u6wntf)

[4.2 Test Plan 4: SparkFun ESP32 MCU 31](#_heading=h.1mrcu09)

[5. Definition of Done 33](#_heading=h.3l18frh)

# Table of Figures

[*Figure 1: Healthy Leaves wireframe user experience.* 13](#_heading=h.35nkun2)

[*Figure 2: System Architecture of Firebase authentication.* 14](#_heading=h.44sinio)

[*Figure 3: Hardware architecture.* 15](#_heading=h.z337ya)

[*Figure 4: Software architecture of Healthy Leaves project.* 16](#_heading=h.1y810tw)

[*Figure 5: Entity relationship diagram (ER-diagram) of Healthy Leaves project.* 17](#_heading=h.4i7ojhp)

[*Figure 6: Database diagram of Healthy Leaves project.* 19](#_heading=h.2xcytpi)

[*Figure 7: Use case 1.* 20](#_heading=h.2bn6wsx)

[*Figure 8: Use case 2.* 21](#_heading=h.3as4poj)

[*Figure 9: Use case 3.* 22](#_heading=h.49x2ik5)

[*Figure 10: Use case 4.* 23](#_heading=h.147n2zr)

# Table of Tables

[Table 1: Backlog Information. 6](#_heading=h.4d34og8)

[Table 2: Sprint 1. 9](#_heading=h.17dp8vu)

[Table 3: Sprint 2. 11](#_heading=h.3rdcrjn)

[Table 4: S-10.1 Test Case. 24](#_heading=h.ihv636)

[Table 5: S-10.2 Test Case. 24](#_heading=h.32hioqz)

[Table 6: S-10.3 Test Case. 25](#_heading=h.1hmsyys)

[Table 7: S-10.7 Test Case. 25](#_heading=h.41mghml)

[Table 8: S-14.1 Test Case. 26](#_heading=h.vx1227)

[Table 9: S-14.2 Test Case. 26](#_heading=h.3fwokq0)

[Table 10: S-14.3 Test Case. 27](#_heading=h.1v1yuxt)

[Table 11: S-14.4 Test Case. 27](#_heading=h.4f1mdlm)

[Table 12: S-16.1 Test Case. 28](#_heading=h.19c6y18)

[Table 13: S-13.2 Test Case. 29](#_heading=h.3tbugp1)

[Table 14: S-16.3 Test Case. 29](#_heading=h.28h4qwu)

[Table 15: S-16.4 Test Case. 30](#_heading=h.nmf14n)

[Table 16: S-16.5 Test Case. 30](#_heading=h.37m2jsg)

[Table 17: C-1.1 Test Case. 31](#_heading=h.46r0co2)

[Table 18: C-1.2 Test Case. 31](#_heading=h.2lwamvv)

[Table 19: C-1.3 Test Case. 32](#_heading=h.111kx3o)

[Table 20: DoD Breakdown. 33](#_heading=h.206ipza)

# 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.

# 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.Graphical user interface, table, Excel

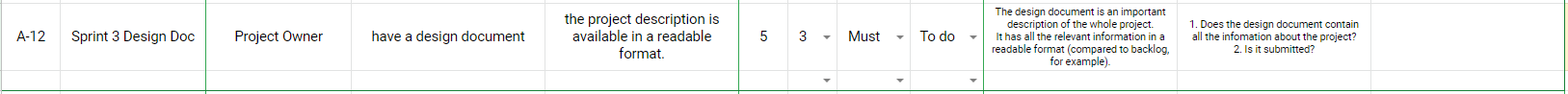
Description automatically generated

Calendar

Description automatically generatedCalendar

Description automatically generated

Calendar

Description automatically generated

## 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.

Graphical user interface, application, table

Description automatically generated

Table

Description automatically generatedCalendar

Description automatically generated

Table 3: Sprint 2.

A screenshot of a computer screen

Description automatically generatedA picture containing calendar

Description automatically generatedA picture containing calendar

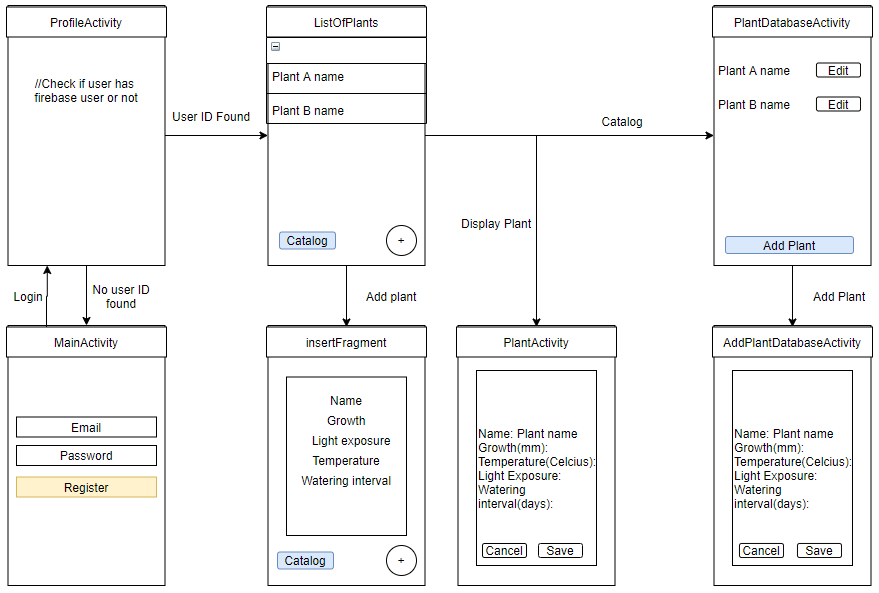
Description automatically generatedA picture containing graphical user interface

Description automatically generated

# 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.

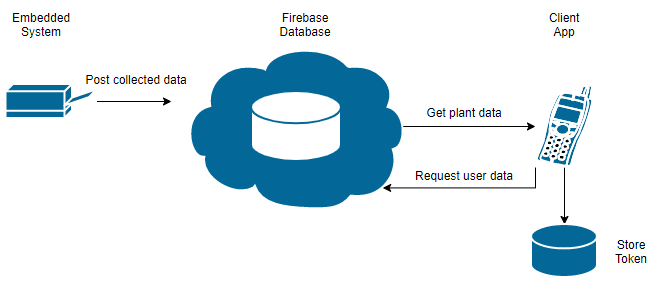
## 3.1 Android Application Wireframes



*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.

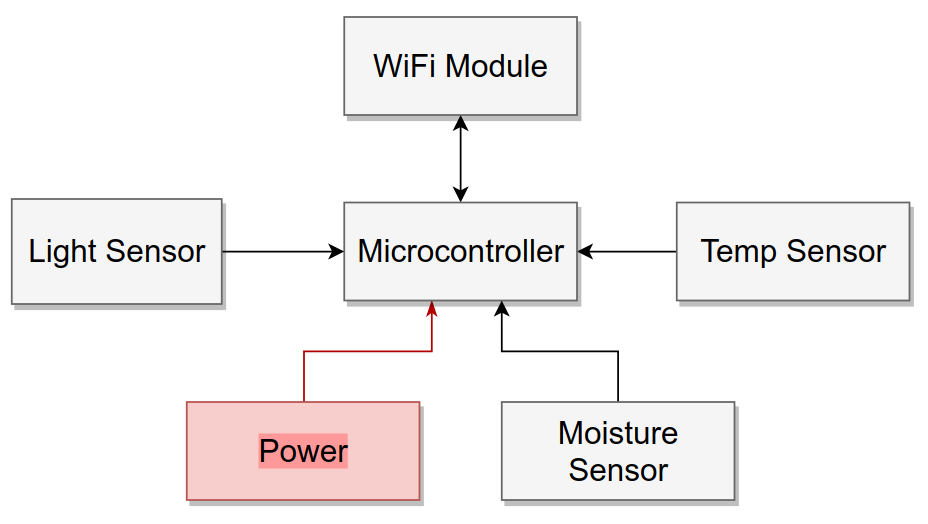
## 3.2 System Architecture



*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.

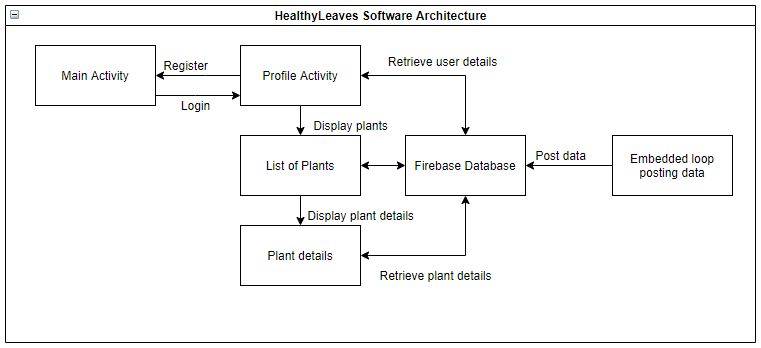
## 3.3 Hardware Architecture



*Figure 3: Hardware architecture.*

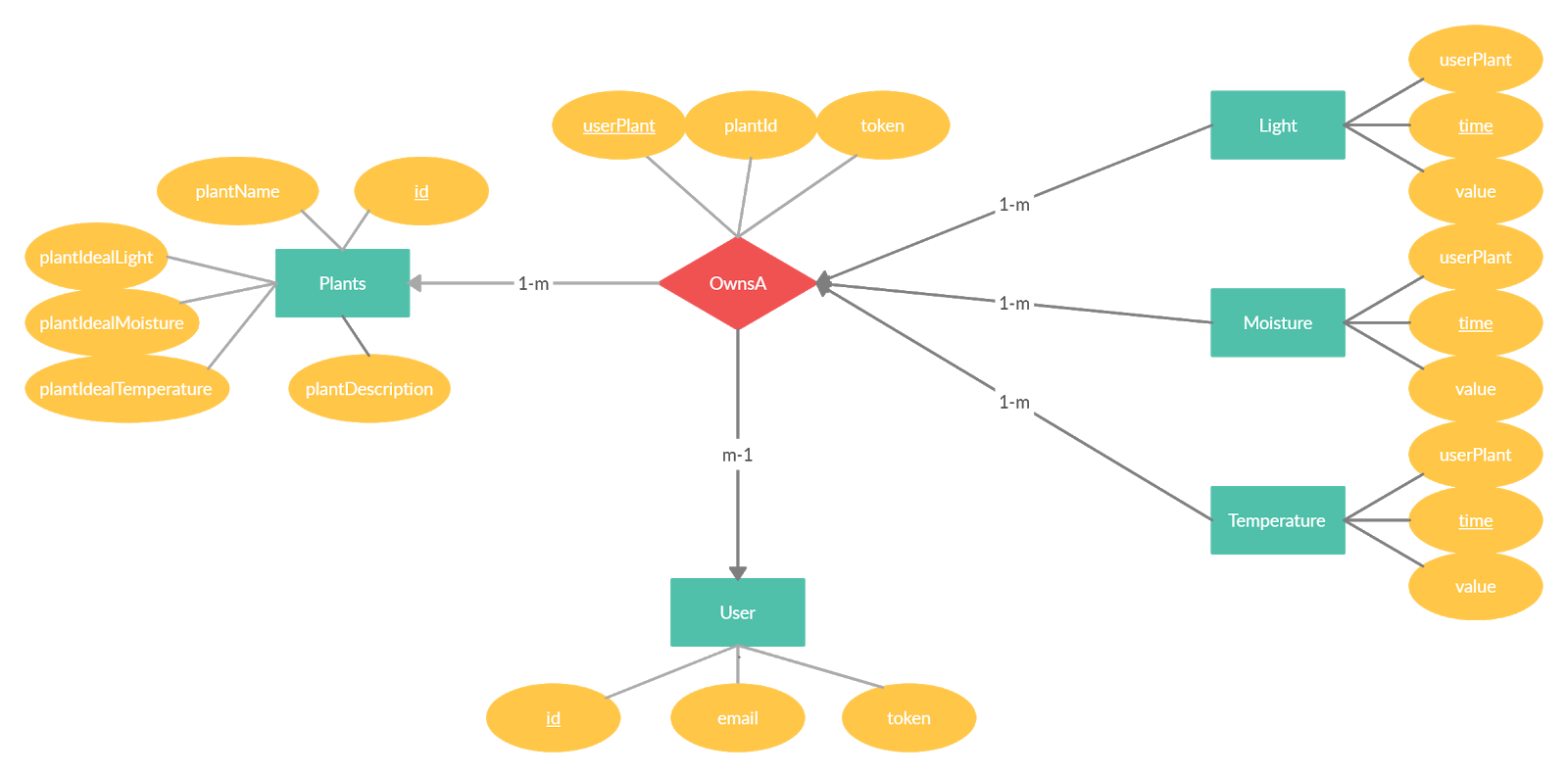
*Figure 3* shows the hardware architecture of the Healthy Leaves project. The device consists of three sensors connected to the microcontroller, SparkFun ESP32 Thing, through analog pins. The data is read and sent to Firebase through an onboard Wi-Fi chipset – ESP8266. Currently, for prototyping purposes,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



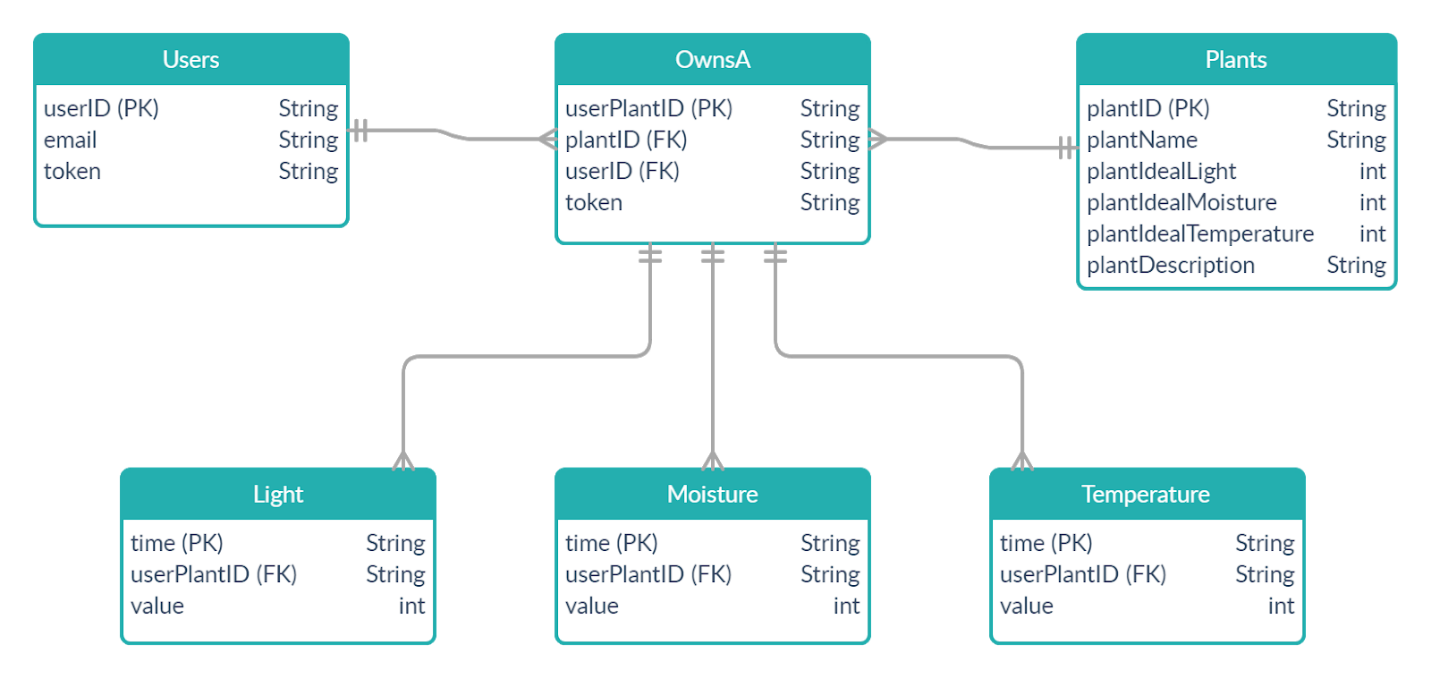
*Figure 5: Entity relationship diagram (ER-diagram) of Healthy Leaves project.*

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.

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 6: Database diagram of Healthy Leaves project.*

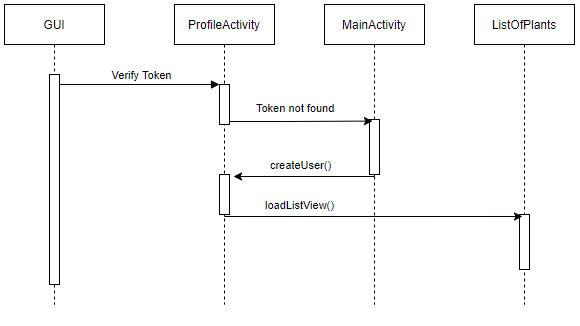
The figure above illustrates the database diagram that will be used to implement the firebase database of the Healthy Leaves project. It follows the logical relationships designed in the ER-diagram of *Figure 5*.

## 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.

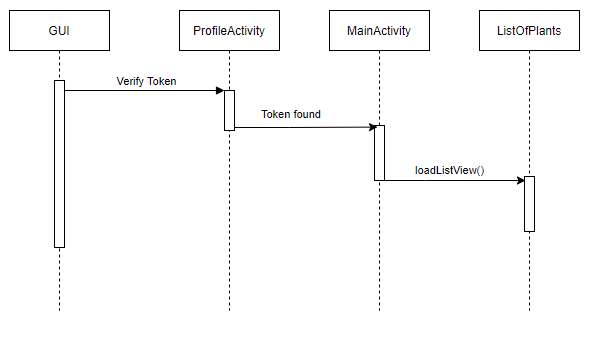
### 3.5.1 Use Case 1



*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.

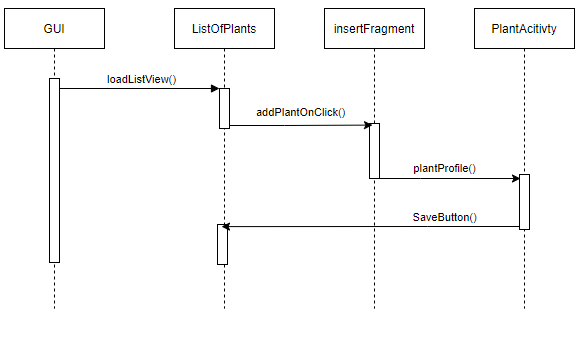
### 3.5.1 Use Case 2



*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.

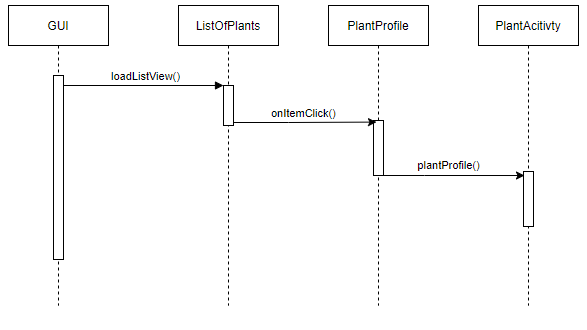
### 3.5.1 Use Case 3



*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 PlantAcitivty 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.

Table

Description automatically generated

Table 5: S-10.2 Test Case.

Table

Description automatically generated

Table 6: S-10.3 Test Case.

Table

Description automatically generated

Table 7: 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 8: S-14.1 Test Case.

Table

Description automatically generated

Table 9: S-14.2 Test Case.

A picture containing table

Description automatically generated

Table 10: S-14.3 Test Case.

Table

Description automatically generated

Table 11: S-14.4 Test Case.

A picture containing table

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 12: S-16.1 Test Case.

Table

Description automatically generated

Table 13: S-13.2 Test Case.

Table

Description automatically generated

Table 14: S-16.3 Test Case.

Table

Description automatically generated

Table 15: S-16.4 Test Case.

Table

Description automatically generated

Table 16: S-16.5 Test Case.

Table

Description automatically generated

## 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.

Table

Description automatically generated

Table 18: C-1.2 Test Case.

Table

Description automatically generated

Table 19: C-1.3 Test Case.

Graphical user interface, application, table

Description automatically generated

# 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.

Table

Description automatically generated