CTIS411 Senior Project I

Software Requirements Specification

Web-AR Smart Indoor Gardening System

Team 5

Mert Köroğlu, 22003433

Eren Tarak, 21903229

Talha Rehman Abid, 22001505

Eray Altay, 21803051

Bilkent University

Department of Information Systems and Technologies

11.12.2023

Change History

File Name	Document Type	Deliverable	Version	Submission Date
Team5_SRS_FR1	MS Word 2013	2	1	13.11.2023
Team5_SRS_FR2	MS Word 2013	2	2	11.12.2023

Change List

- Change History's "Deliverable" format fixed.
- System Interfaces part is redone.
- Memory Constraints part is edited.
- New text added for User Characteristics.
- Added, edited, and removed Functional Requirements.
- Added General System Use Case Diagram.
- Numbering is fixed at System Model.
- EER Diagram is updated.
- Verification part is rewritten.
-

Project Team

Team 5		
Name, Surname	**Student Id Number**	**e-mail**
Eray Altay	21803051	eray.altay@ug.bilkent.edu.tr
Talha Rehman Abid	22001505	rehman.abid@ug.bilkent.edu.tr
Mert Köroğlu	22003433	mert.koroglu@ug.bilkent.edu.tr
Eren Tarak	21903229	eren.tarak@ug.bilkent.edu.tr

Project Details

Project Name	**Web-AR Smart Indoor Gardening System**
Software Name	**FloraVision**
Academic Advisor	**Dr. Duygu Albayrak**
Github URL	**https://github.com/Smart-Indoor-Gardening-System**
WEB page	

Executive Summary

In this SRS document, we provided how the FloraVision will work and expected it to perform. Our aim with this product is to create an easy interface between the users and their plants that provides overall health and growth using data that comes from a specialized hardware kit that we will produce. In the document, first we talked about the scope of our product. This consists of our hardware kit, cloud support using AWS for data analysis, the web application and AR support to easily visualize the plant's insights. Our major requirements include, reading data from our sensors using our hardware kit (FVF-3-2), sending this data to AWS to process health and growth analysis (FVF-3-3), displaying this data in our web application dashboard (FVF-7-8) and using AR charts to visualize the data on camera (FVF-9-4). In the document we also provided our database structure using an EER diagram indicating the connection and relation between the database tables in our software. In section 3, we answered who the users of the system will be, what are their attributes and what they need from the software. These users contain general plant owners and indoor farmers. In the document we also provided major libraries and APIs that will be used while creating the product. These include the MQTT library that will be used to create a subscriber and publisher relationship between the AWS and the hardware kit to send sensor data using low bandwidth and easy communication. DHT library will also be used to connect the hardware device to the Wi-Fi with the use of WifiManager library. Creating the hardware kit, we will also use Arduino Nano to carry out our IoT activities. We also provided complex use cases with more detail using Activity, Analysis Level System Sequence and Analysis Level Class Diagrams in section 6 of the document. Lastly, we provided a prototype of these use cases at the Requirements Prototype section to better exemplify and envision the most complex use cases that the product will make use of.

Table of Contents

Page Number

List of Tables

Page Number

List of Figures

Page Number

Abbreviations

AR	Augmented Reality
AWS	Amazon Web Services
EER	Enhanced Entity-Relationship
FVF	FloraVision Functional Requirement
FVN	FloraVision Non-Functional Requirement
HTTP	Hypertext Transfer Protocol
IOT	Internet of Things
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
SDK	Software Development Kit
SNS	Simple Notification Service
SRS	Software Requirements Specification
TLS	Transport Layer Security

1. Scope

For the scope we described the products of our system, what they are, how they will operate and what is their benefit.

1) The products of FloraVision includes:

- Hardware Product
- External Dynamic Wi-Fi Configuration Tool
- AWS Lambda
- Data Storage System
- Cloud Notification System
- Authentication Product
- Web Application
- AR Product

2) What each product will do:

Hardware Product:

- The software will read values from light, soil moisture, temperature, and air quality sensor data every 5 seconds.
- After reading the data, it will send this data in a JSON format to AWS IoT Core, which is a fully managed message broker service.
- The software should be able to connect to Wi-Fi using ESP8266 Wi-Fi module.

External Dynamic Wi-Fi Configuration Tool:

- The hardware device will be connected to a Wi-Fi address by entering the name and the password through the WifiManager web application, which is an external Wi-Fi configuration tool.

AWS Lambda

- AWS Lambda functions will be triggered to process incoming sensor data.

Data Storage System

- One function for real-time data visualization is to save data into DynamoDB which is a real time fully managed database.

Cloud Notification System

- AWS SNS will be used as a cloud-based notification system. Lambda functions will be configured to publish notification messages to relevant SNS topics such as anomaly alerts, task scheduling and predictions notifications.

Authentication Product

- For web authentication, a straightforward process will be employed using AWS Cognito. Users will be authenticated to the web application with Google, Facebook providers, or email/password.
- However, for enhanced security in device user identity, a more robust two-step verification authentication method will be implemented, involving a user-friendly device pairing process post-login.

Web Application:

- Will provide real-time environmental metrics that we received from the hardware sensors.
- It will show detailed health, growth analysis charts.
- It's the main product the user will be engaged with.

AR Product:

- The AR Product will scan the QR codes of hardware devices to show the metrics about a plant on the device camera.
- The AR Product will be accessible through the Web Application.

3) Benefits, goals, and objectives of FloraVision:

As indicated in the Initial Plan [1], the goal of FloraVision is to provide plant owners and indoor farmers with a tool that would allow them to gain deep insight about their plants. FloraVision will allow users to monitor the real time metric of their plants consisting of soil moisture, light intensity, air quality and temperature/humidity environmental variables. FloraVision will also provide daily, weekly, and monthly charts about the user's plant status in each duration. FloraVision will create future

health and growth predictions for the user to let them know if everything about their plant is going well in terms of condition and environment. The benefits of FloraVision include easy to use software to monitor your plants, providing valuable insight about the plants conditions and allowing plant owners to take care of their plant easier with less stress.

2. Product Perspective

a) System interfaces:

Google Gmail:

We will use Gmail for user registration.

b) User interfaces:

Web UI (React, Chakra UI):

Utilizes React.js v18.0 UI library and Chakra UI v2.8.1 component library for consistent and visually appealing user interface components through the app.

User Experience

- Implements engaging animations using Framer Motion v10.16.4 to enhance the overall user experience for gestures, transitions, and layouts.
- Progressive Disclosure: Show users only features relevant for the task at hand, one per screen to reduce cognitive load.
- Utilizes tooltips for interactions.
- Use checklists for progressive guidance.
- Easy to toggle dark and light themes.
- Important action buttons are designed in a way that takes immediate attention of the user on the page.
- Secondary action buttons which will be used for secondary actions in the page (canceling, going back, etc.)

State Management

- Utilizes Global State Management Zustand v4.4.4 library.
- Manages states for user authentication, device data, notifications and AR mode and data visualization.

User Authentication:

- **State:**
    - **isAuthenticated:** Indicates whether the user is logged in or not.
    - **userProfile:** Stores basic user information like name, email, and preferences.

- **Actions:**
    - **Login User:** Records the user's login status and details.
    - **Logout User:** Clears the user's information when they log out.

AR Visualization:

- **State:**
    - **AR Mode Enabled:** Signals whether Augmented Reality mode is turned on.
    - **AR Data:** Holds information needed for AR visualizations.
- **Actions:**
    - **Enable AR Mode:** Activates Augmented Reality for visualizing data.
    - **Disable AR Mode:** Deactivates Augmented Reality mode.

Data Visualization:

- **State:**
    - **Chart Preferences:** Allows users to customize their chart displays.
- **Action:**
    - **Customize Chart Preferences:** Enables users to set their chart preferences.

Notifications:

- **State:**
    - **Notifications:** Holds a record of recent notifications.
- **Action:**
    - **Mark Notification as Read:** Indicates that a user has seen a notification.

Device Data:

- **State:**
    - **Selected Device**: Stores details about the device currently in focus.
- **Action:**
    - **Select Device:** Select the hardware kit to display its relevant data.

Data Visualization on Web UI

utilizes apexcharts library v3.44.0 for interactive and dynamic highly customizable charts.

Chart Types:

- **Real Time Dynamic Updating Line Chart:**
- **Syncing Line Charts:**
    - Comparing Multiple Metrics Over Time
    - Multi-Sensor Data Visualization
- **Sparklines:**
    - provide a quick summary of environmental conditions or plant metrics on a dashboard in a minimalistic way based on historical data.

c) Hardware interfaces:

Hardware Kit Interface (Arduino):

- A simple button for power on/off the hardware kit and turn on/off the kit` s Wi-Fi.
- LED light indicating the hardware's on/off status.
- The Interface with the hardware will only be included in Wi-Fi connection using WifiManager Library. Using this library, software will be able to provide dynamic Wi-Fi connection. By pressing the button on the hardware kit, WifiManager will be activated and will appear on the user's Wi-Fi network list on their devices if it is in proximity.
- There, users will connect to the WifiManager by entering the SSID of the Wi-Fi. Then, this will redirect to the WifiManager web application. PCs, Phones and Tablets with Wi-Fi connection will be able use the WifiManager interface.
- There, users will enter the desired Wi-Fi's name and password to connect the hardware kit to Wi-Fi. With this, the hardware will send 24/7 sensor data to the AWS Cloud.
- Analog ports A3 and A5 will be used for soil moisture and light sensor respectively. Digital pin 9 will connect to the DHT11 Temperature/Humidity sensor. TX and RX pins will connect to the ESP8266 Wi-Fi module.

- AWS Lambda provides x86_64, arm6 hardware architectures for Node.js 18.x runtime which means Lambda functions can run on both x86_64 (64-bit Intel/AMD) and arm6 (ARM)

PC Interface:

- This web application will be able to work on Windows and MacOS devices.

Smartphone Interface:

- Web applications will be able to work on Android and IOS devices.

d) Software interfaces:

Arduino Libraries (WifiManager, DHT, ESP8266, etc.)

DHT

- version: 1.4.4
- The library will be used for operating the DHT11 temperature and humidity sensor.
- This library will make us use DHT functions that would give us Celsius, Fahrenheit and more data.

WifiManager:

- version: 2.0.16-rc.2
- will be used for connecting the hardware kit to Wi-Fi through the web.

Amazon DynamoDB

- The system interfaces with Amazon DynamoDB version 2019.11.21.
- serves as the primary data storage for the system.
- Operations include, but are not limited to, storing user data, updating device configurations, and fetching real-time sensor data.
- AWS Lambda functions will use AWS SDK v3.x in Node.js run time environment to interact and access the service.

AWS Cognito

- AWS Cognito Identity SDK for JavaScript: amazon-cognito-identity-js
- version:6.3.6

- serve as user management and authentication service.

AWS SNS SDK

- AWS SNS SDK: @aws-sdk/client-sns
- version: 3.435.0
- used for push notification services, handling real-time notifications for users.

Node.js 18.x Runtime Environment Command line tool to deploy lambda:

- aws-lambda 1.0.7

Frontend Build Tool:

Vite:

- Version: 4.5.0
- Discussion: Vite is the front-end build tool responsible for building and bundling the React application. It boosts development efficiency with its fast build times.

e) Communications interfaces.

Local Network Protocols:

- Utilizes MQTT for communication between the hardware kit and AWS IoT Core.
- Will communicate between HTTP protocol with TLS encryption.

f) Memory constraints.

- There are no memory constraints.

3. User characteristics

Our intended groups of end users contain:

1) People who want to take care of their indoor plants.

2) People who produce crops indoors.

All users of the product should know English to use the product. Because only English language will be used for user interface.

For the first type of group, the product expects at least 12+ age groups and above will use the product. Therefore, the usability of the product should be simple and easy to use to attract people from every age group. Because the first group of users may contain casual users that only want to know about their plant's real time insight, the product should clearly indicate the real time metrics to the user as easily as possible. This group of users may also contain people who want to grow their plant as healthily as possible. Therefore, plant's health and growth predictions and history should be as easily accessible and reliable as possible.

Another type of users are indoor farmers. We aim to provide comprehensive insight into farmer's crops as detailed and reliable as possible. These types of users are also expected to use multiple of our IoT hardware devices to connect several crops, therefore, the hardware product should be easy to use and easy to connect to our web application. Also, the hardware product itself should be cheap and affordable for the farmers to attract the same group of people to use our product.

System Users:

1) Root Users
2) Normal Users

Root Users are the users that are the owners of the device. They are given the status "Root User" when they are the first user to register to a device. They have specific privileges like removing other users or modifying device's name and password.

Normal users are the users that are not the first to register to a device. They only have viewing privileges for the device and do not have any privilege to remove another user or modify the device's name or password.

4. Assumptions and dependencies

This section lists the assumptions and dependencies that FloraVision depends on. Any change regarding these assumptions may cause issues regarding the development of the product.

1. Arduino Nano will be available to create the hardware product. If it is not available, and another microcontroller is used, it will restrict the use of such libraries that we indicated that we would use for our hardware product. It also might affect the overall connection of sensors to pins if the new device provides less digital or analog pins. This new microcontroller might also feature built-in Wi-Fi support, that the Arduino Nano does not provide. This might eliminate the need for the ESP8266 Wi-Fi module that we initially agreed to use to connect Arduino to Wi-Fi.
2. The required sensors will be available for the hardware product. If some sensors are not available, this will lead to major requirement changes for our product. For example, the absence of soil moisture sensors might negatively affect the health and growth analysis that we will provide to the user. This assumption is considered for every sensor on our product.
3. The users of the product will be connected to Wi-Fi. If not, users will not be able to connect to the web application that provides insights about several factors about their plant.
4. The users of the web product will use a device with a camera. A camera is required to provide AR visualization features on the web application. If a camera device is not used, the user will not be able to use such AR features.
5. The hardware product will not malfunction. In case of the hardware product malfunction, the sensor readings may get corrupted or even connection between Arduino and AWS IoT Core can be disrupted.

5. Functional requirements

In this section, we listed the requirements that need to be satisfied in our system. These requirements include parts from our hardware, web application, AR, and data analysis sections of our product. In the first part, we displayed the requirements as test format and in the second part we displayed them using Use-Case diagrams and descriptions to exemplify in a pictural format. From *Figure 1* to Figure 6 we provided our use case diagrams. From *Table 1* to *Table 22* we provided the use case descriptions for the given diagrams.

1) Text format:

Hardware

1. (FVF-1-1) The hardware kit shall use Arduino Nano as a microcontroller.
2. (FVF-1-2) The hardware kit shall use ESP8266 Wi-fi module to connect to Wi-Fi.
3. (FVF-1-3) The hardware kit shall use MQ-2 Gas/Smoke sensor to detect CO, CO2 to detect anomalies in the air quality.
4. (FVF-1-4) The hardware kit shall be connected to power using batteries.
5. (FVF-1-5) The hardware kit shall use the DHT11 Temperature/Humidity sensor.
6. (FVF-1-6) The hardware kit shall use a light sensitive resistor to detect the presence of light and its intensity.
7. (FVF-1-7) The hardware uses a soil moisture sensor to detect the plant's soil moisture.
8. (FVF-1-8) The system shall let users power off the device.
9. (FVF-1-9) The system shall let users power on the device.

Sensor Data Reading

10. (FVF-2-1) The hardware part of the system shall read light sensor data through the A1 port on Arduino Nano.
11. (FVF-2-2) The hardware part of the system shall read humidity/temperature sensor data through the digital 9 port on Arduino Nano.

12. (FVF-2-3) The hardware part of the system shall read soil moisture sensor data through the A3 port on Arduino Nano.
13. (FVF-2-3) The hardware part of the system shall read air quality sensor data through the A2 port on Arduino Nano.

Data Transmission

14. (FVF-3-1) The hardware part of the system shall connect to Wi-Fi through digital ports on Arduino Nano. (ref. (FVF-1-6) [1])
15. (FVF-3-2) The hardware part of the system shall create a Json format sensor data every 5 seconds.
16. (FVF-3-3) The hardware part of the system shall send sensor data to AWS IOT core managed message broker service via MQTT. (ref. (FVF-1-5) [1])
17. (FVF-3-4) The hardware part of the system shall turn on/off if the button is pressed for 3 seconds.
18. (FVF-3-5) The hardware part of the system shall detect Wi-Fi if a button is pressed and released.
19. (FVF-3-6) The system shall let users connect to Wi-Fi through their phone dynamically using the WifiManager library.

Hardware-web integration

20. (FVF-4-1) The web application shall connect to hardware kits through the devices tab.
21. (FVF-4-2) The web application shall send the user to their plant's insights if the hardware kit is pressed in the devices tab.
22. (FVF-4-3) The web application shall send a GET request to AWS every 5 seconds to refresh plant data.

Hardware Kit Authentication

23. (FVF-5-1) The user shall connect to hardware using the password and id provided with the device.
24. (FVF-5-2) The system shall make the first connected user Root user.
25. (FVF-5-3) The system shall make every other user Normal user.

Web Authentication

26. (FVF-6-1) For the web application, the system shall implement user registration and login using AWS Cognito including Google, Facebook providers and email password authentication options.

Registration Page

27. (FVF-6-2) On the registration page, the system shall ask inputs regarding email, username, and password.
28. (FVF-6-3) On the registration page, the system shall give an error if the given email already exists in the AWS Cognito user pool.
29. (FVF-6-4) On the registration page if the user selected the email/password authentication, the system shall register the user if all the fields are entered regarding the rules of the registration.
30. (FVF-6-5) On the registration page if the user selected the email/password authentication, the system shall only accept passwords which contain at least one number, one symbol, one capital character and at least six characters.
31. (FVF-6-6) On the registration page if the user selected the email/password authentication, the system shall give an error if the password rules are not met after pressing the register button.

Login Page

32. (FVF-6-6) On the login page, if the user clicks on the login button, the system shall check the given email and password. (ref. (FVF-1-8) [1])
33. (FVF-6-6) On the login page, the system shall display a button called “sign up” at the bottom of the login page to navigate user to the registration page.
34. (FVF-6-7) In the login page the system shall give an option to recover existing user password if it's forgotten via a button called “forgot password”.
35. (FVF-6-8) In the login page the system shall send a confirmation code email to the user if the user clicked on the “forgot password” button. (ref. FVF-6-7).
36. (FVF-6-9) In the login page the system shall provide an input field asking for confirmation code when an email is sent to the user after they clicked on the “forgot password” button. (ref. FVF-6-7).

37. (FVF-6-10) In the login page the system shall navigate the user to a new password page if their code matches with the confirmation code in the email. (ref. FVF-6-7).
38. (FVF-6-11) In the login page the system shall navigate the user to the login page if their code does not match with the confirmation code in the email. (ref. FVF-6-7).
39. (FVF-6-12) In the login page the system shall check the new password in the new password page (ref. FVF-6-7).
40. (FVF-6-13) Web application shall keep the session of users to remember user credentials.

Failure Scenarios

41. (FVF-6-14) The system shall redirect the user to the login page if the user token expires.
42. (FVF-6-15) The system shall log out the user if the user clicks the log out button on the top navigation bar on the web application interface.
43. (FVF-6-16) If the login credentials do not match, the system will provide an error notification in the form of a text, indicating that either the password or email is incorrect.

Success Scenario

44. (FVF-6-17) Once the user is authenticated, the system shall enable the user to gain access to the system's other functionalities.

Hardware Setup on W*eb*

45. (FVF-7-1) The system shall display the connected hardware devices on the devices page in the web application.
46. (FVF-7-2) The system shall display general statistics such as temperature, soil moisture, light density and co2 density of the plant on the devices tab for each device.
47. (FVF-7-3) The system shall display a button to add new hardware devices to the application.
48. (FVF-7-4) The system shall enable the users to navigate to the selected device’s dashboard if they click on the device.

49. (FVF-7-5) The system shall enable the user to remove the selected device if they press on the "-" button.
50. (FVF-7-6) The system shall provide battery status of each connected device on the devices page on the web.

Dashboard

51. (FVF-7-8) The system shall display the selected device's real time sensor metrics including temperature, light density, soil moisture, humidity, and air quality on the "today" tab on the dashboard page. (ref. (FVF-1-7), (FVF-1-13) [1])
52. (FVF-7-9) The system shall display tabs at the top of the dashboard including daily, weekly, monthly, and yearly displaying options. (ref. (FVF-1-11) [1])
53. (FVF-7-10) The system shall display today's charts about the sensor data from 00.00 to 23.59 right below the real-time data.
54. (FVF-7-11) The system shall display the plant species name as a link at the top of the dashboard to navigate the user to relevant plant species description and characteristics.
55. (FVF-7-12) The system shall enable users to customize their dashboard via an option menu consisting of checkboxes to select the charts out of weekly, monthly, and yearly options for displaying.
56. (FVF-7-13) The system shall navigate the user to a more detailed chart about the selected metric when the user clicks the chart.

On the detailed page of the selected metric

57. (FVF-7-14) The system shall display a mixed chart combining line chart and bar plot.
58. (FVF-7-15) The system shall display the deviations from the plants corresponding optimum environmental value tagged with labels such as high, moderate, low gaps from the optimum value.
59. (FVF-7-16) The system shall present the average value of the relevant metric on a daily, monthly, and yearly basis, offering users comprehensive insights into the overall trends and patterns.

Notifications

60. (FVF-8-1) The system shall have a notification user interface modal to enable the user to track notifications.
61. (FVF-8-2) If the user clicks on the notification bell icon at the right of the dashboard navigation at the top of the page, The system shall open a notification modal including only several latest notifications with their time such as 2 minutes ago.
62. (FVF-8-3) If the user clicks on the "see more" link at the bottom of the notification modal, The system shall navigate the user notifications page where all notifications are listed.
63. (FVF-8-4) The system shall enable the user to mark all notifications as read when the user clicks on a button called "mark all as read".
64. (FVF-8-5) The system shall ensure that notifications are informative, concise, and relevant, providing users with a clear understanding of the event or update. (ref. (FVF-1-12), (FVF-1-16) [1])
65. (FVF-8-6) The system shall have real-time web push notifications enabled as the default communication method for notifying users. Additionally, users shall have the option to customize their notification preferences by adding email or SMS forms according to their choice.
66. (FVF-8-7) The system shall utilize AWS SNS (Simple Notification Service) to manage and deliver notifications efficiently. This service will be responsible for handling real-time web push notifications, emails, and SMS messages.

Augmented Reality

67. (FVF-9-1) The system shall utilize a marker based augmented reality for real time data visualization for the chrome users.
68. (FVF-9-2) The system shall display a fixed positioned button on the dashboard page to enable the user to start the AR visualization.
69. (FVF-9-3) The System shall recognize predefined markers associated with environmental data points.
70. (FVF-9-4) Upon detecting these markers through the device's camera, the system shall overlay real-time environmental metrics, including light intensity, soil moisture, and temperature, onto the markers in the AR visualization.

71. (FVF-9-5) The system shall display AR visualization content as 2d charts and metrics along with related icons.
72. (FVF-9-6) The system shall support the visualization of historical environmental data in AR, allowing users to toggle between real-time and historical view.
73. (FVF-9-7) If the user device or browser do not support AR, the system shall display an alert dialog indicating that the user device does not support AR and inform the user regarding supported browsers along with their versions and supported devices when the user clicks on the augmented reality button.
74. (FVF-9-8) If the user scans another marker, the system shall discard the previous 2d charts.

Predictions Data Science and Analysis

75. (FVF-10-1) The system shall integrate a health prediction model, leveraging scientific threshold values of relevant species and machine learning algorithms. This model should generate short-term and medium-term predictions, forecasting potential health issues based on environmental conditions before they impact the plants. (ref. (FVF-1-1), (FVF-1-3) [1])

Root User Managing Devices:

76. (FVF-11-1) The system shall enable root users to change device's name.
77. (FVF-11-2) The system shall enable root users to change device's password.

Root User Managing Users:

78. (FVF-11-3) The system shall enable root users to remove other users.
79. (FVF-11-4) The system shall enable root users to make another one root user.

2) Use Cases:

General System Use Case:

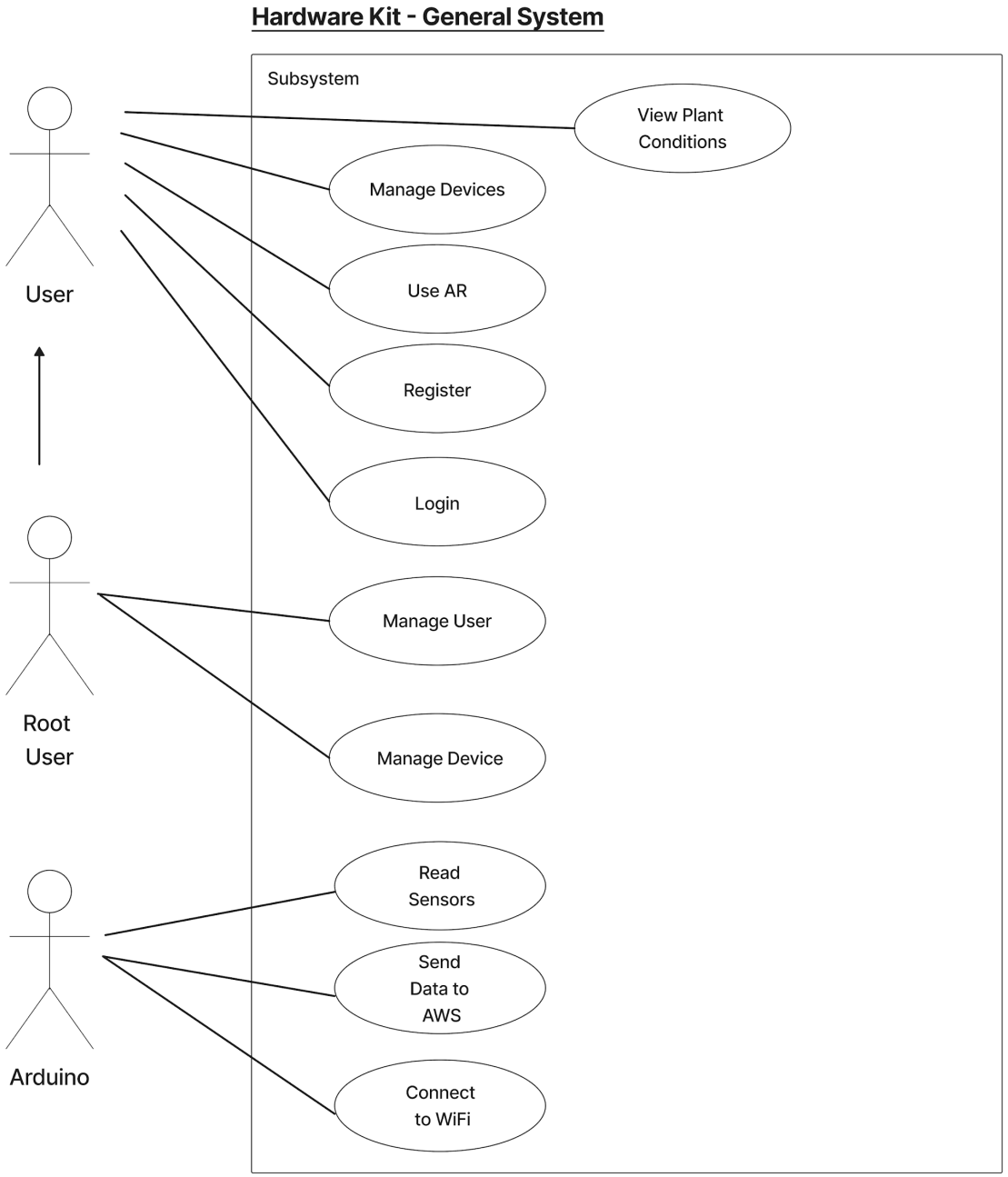


Figure 1 General System Use Case Diagram

Detailed Use Case Diagrams:

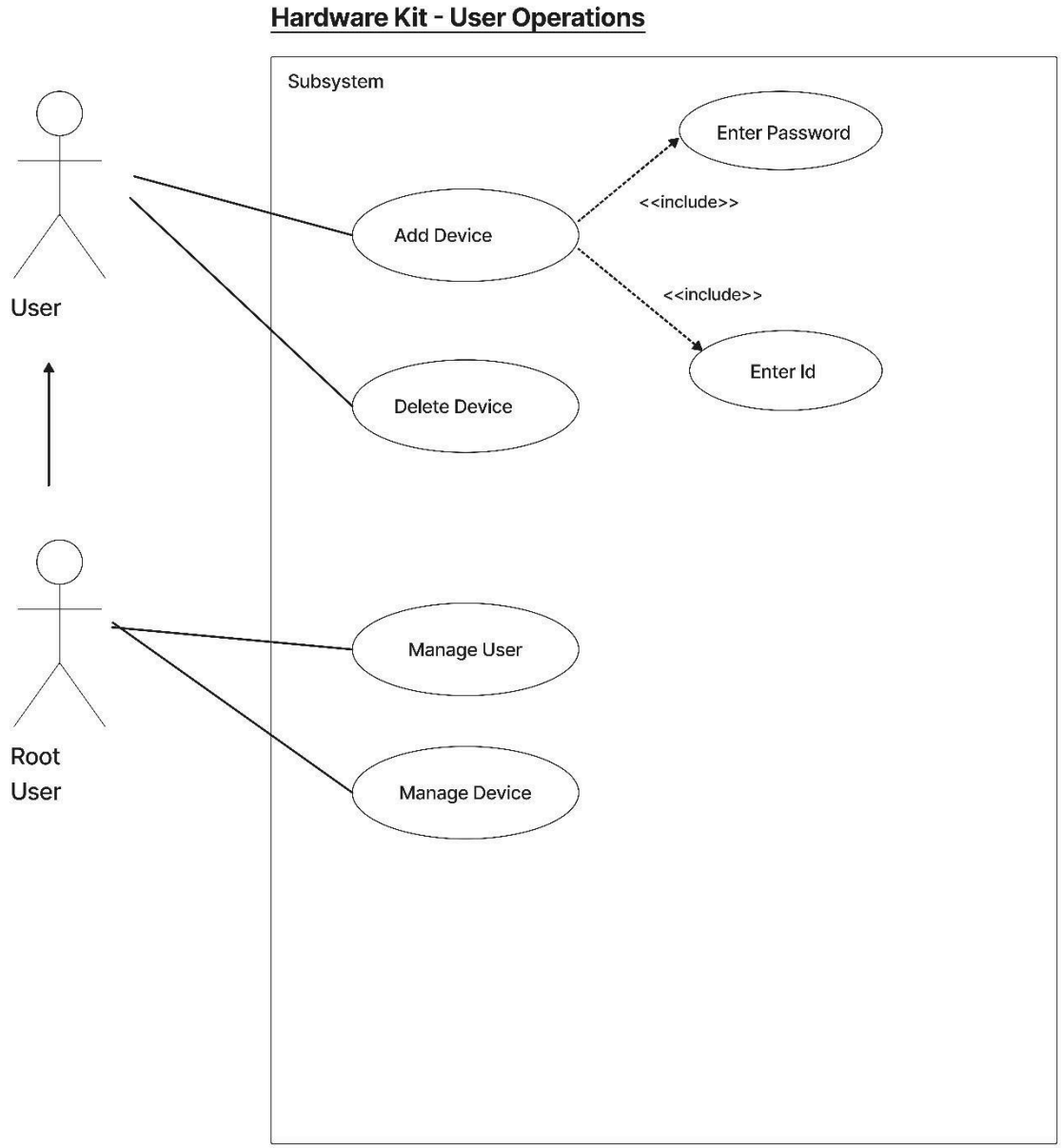


Figure 2 Use Case Diagram about Hardware-User Operations

Table 1 Use Case description for case named "Add Device"

Use Case No	1
Use Case Name	Add Device
Actors	User
Description	Defines how the user adds new device to their account
Pre-Conditions	1. Device should be turned on.

	2. The device is connected to Wi-Fi. **3.** User is logged in.
Post-Conditions	
Basic Flow	1. User goes to devices tab on the web application. 2. User enters his/her device's id and password. 3. User presses confirm button. **4.** Device is added to the system
Alternative Flow	1. User goes to devices tab on the web application. 2. User enters wrong id or password for the device. **3.** User receives an error message regarding wrong password or id
Functional Requirements	(FVF-7-3)

Table 2 Use Case description for case named "Delete Device"

Use Case No	2
Use Case Name	Delete Device
Actors	User
Description	Defines how the user deletes a device from their account
Pre-Conditions	1. Device should be an already existing device.
Post-Conditions	
Basic Flow	1. User goes to the devices tab. 2. User presses the remove (-) button to remove the device. 3. User receives a pop-up confirmation tab. 4. User presses confirm. **5.** User's device is removed from him/her account
Alternative Flow	

Functional Requirements	(FVF-7-5)

Table 3 Use Case description for case named "Manage Device"

Use Case No	3
Use Case Name	Manage Device
Actors	Root User
Description	Defines how the root user can manage the device.
Pre-Conditions	1. User has the device in his/her system. **2.** User is the root user for this device.
Post-Conditions	1. The password for the device is changed. **2.** Device name is changed
Basic Flow	1. User presses onto the device he wants to manage. 2. He/she presses “Change Password” to change the password for the device. 3. He/she enters a new password to the input field. **4.** He/she presses confirm to accept their action.
Alternative Flow	1. User presses onto the device he wants to manage. 2. He/she presses “Change Name” to change the name of the device. 3. He/she enters a new name to the input field. **4.** He/she presses confirm to accept their action
Functional Requirements	(FVF-11-1), (FVF-11-2)

Table 4 Use Case description for case named "Manage User"

Use Case No	4
Use Case Name	Manage User
Actors	Root User

Description	Defines how the root user can manage the other users.
Pre-Conditions	1. User has the device in his/her system. 2. User is the root user for this device.
Post-Conditions	
Basic Flow	1. User presses onto the device he wants to manage. 2. He/she presses “Change Root User” to change the root user of the device. 3. He/she selects the new root user. 4. He/she presses confirm to accept their action.
Alternative Flow	1. User goes to devices tab on the web application. 2. He/she presses “Remove User” to remove a user from the device. 3. He/she presses confirm to accept their action
Functional Requirements	(FVF-11-3), (FVF-11-4)

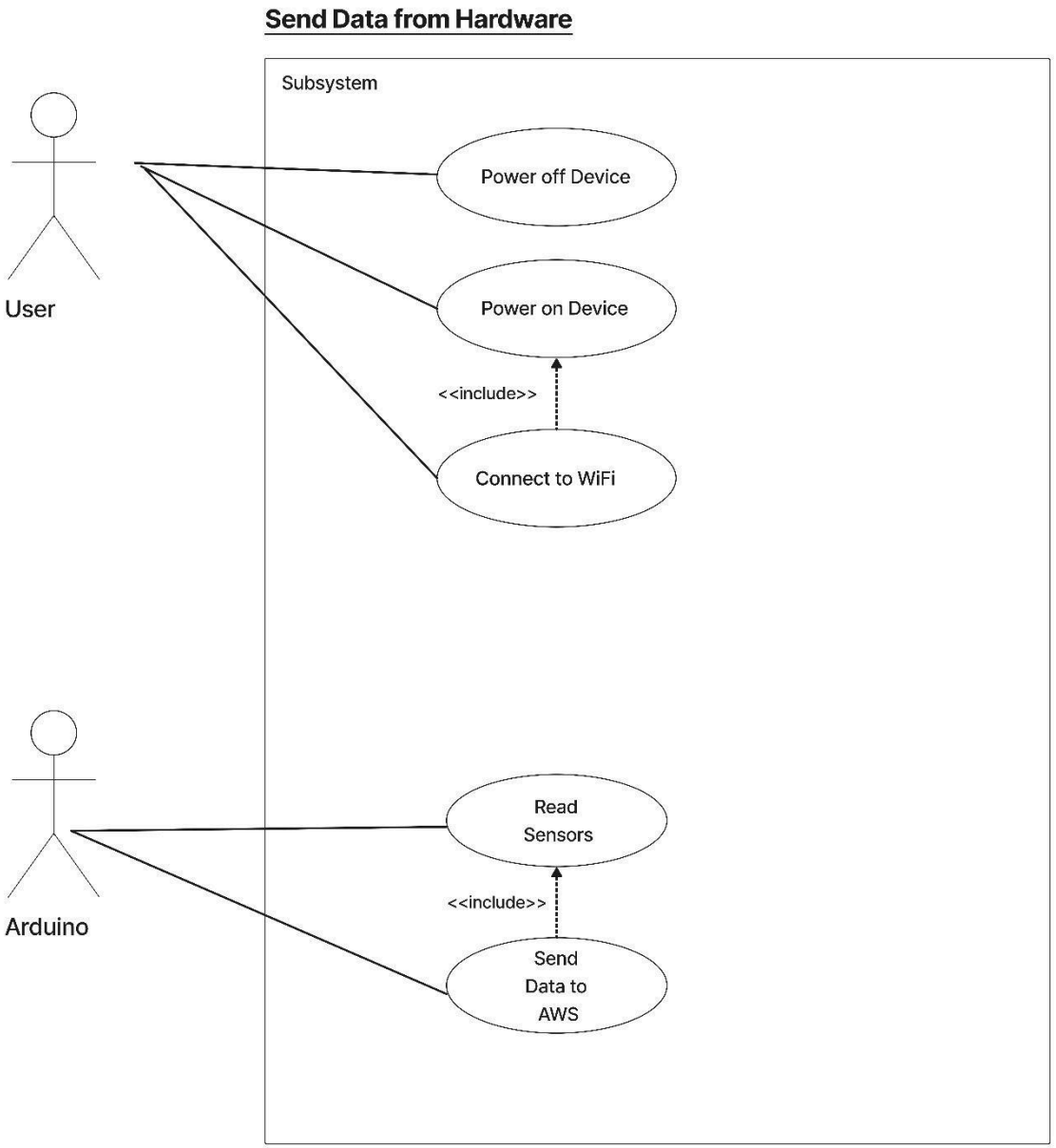


Figure 3 Use Case Diagram about Sending Data from Hardware

Table 5 Use Case description for case named "Power off Device"

Use Case No	5
Use Case Name	Power off Device
Actors	User
Description	Defines how the user power off the hardware kit.
Pre-Conditions	1. Device should be already turned on.

Post-Conditions	1. Device is turned off.
Basic Flow	1. User presses the power off button for 3 seconds to turn off the device.
Alternative Flow	
Functional Requirements	(FVF-1-9)

Table 6 Use Case description for case named "Power on Device"

Use Case No	6
Use Case Name	Power on Device
Actors	User
Description	Defines how the user power on the hardware kit.
Pre-Conditions	1. Device should be already turned off.
Post-Conditions	1. Device is turned on.
Basic Flow	1. User presses the power on button for 3 seconds to turn on the device.
Alternative Flow	
Functional Requirements	(FVF-1-10)

Table 7 Use Case description for case named "Connect to Wi-Fi"

Use Case No	7
Use Case Name	Connect to Wi-Fi
Actors	User
Description	Defines how the user will connect to Wi-Fi
Pre-Conditions	1. Device should be already turned on.
Post-Conditions	1. Device is connected to Wi-Fi
Basic Flow	1. User presses the power off button for less than 3 seconds to search Wi-Fi. 2. User selects the device name that popped up in their Wi-Fi on their phone/PC. 3. User enters password for the device. 4. User is redirected to WifiManager to add a Wi-Fi for the hardware. 5. User selects the hardware he/she wants to connect. 6. User enters the SSID and password for this Wi-Fi. 7. User presses confirm button. 8. If SSID and password is correct the device is connected to the Wi-Fi
Alternative Flow	1. User presses the power off button for less than 3 seconds to search Wi-Fi. 2. User selects the device name that popped up in their Wi-Fi on their phone/PC. 3. User enters password for the device. 4. Unable to redirected to WifiManager interface because of wrong password.

Functional Requirements	(FVF-3-1)

Table 8 Use Case description for case named "Read Sensors"

Use Case No	8
Use Case Name	Read Sensors
Actors	Arduino
Description	Defines how the Arduino will read the sensor data.
Pre-Conditions	1. Device should be already turned on. 2. Device is connected to a Wi-Fi
Post-Conditions	1. Device is reading sensor data.
Basic Flow	1. Arduino starts reading sensor data from the sensors.
Alternative Flow	
Functional Requirements	(FVF-2-1), (FVF-2-2), (FVF-2-3), (FVF-2-4)

Table 9 Use Case description for case named "Send Data to AWS"

Use Case No	9
Use Case Name	Send Data to AWS
Actors	Arduino
Description	Defines how the Arduino will send the sensor data to AWS.
Pre-Conditions	1. Device should be already turned on. 2. Device is connected to Wi-Fi. 3. Device is reading sensor data.

Post-Conditions	1. Device is sending sensor data to AWS.
Basic Flow	1. Arduino subscribes to AWS's MQTT address. 2. If connection is successful, it combines the data read from its sensor to a JSON format. 3. It creates and sends a new JSON format data to AWS through MQTT.
Alternative Flow	
Functional Requirements	(FVF-3-3)

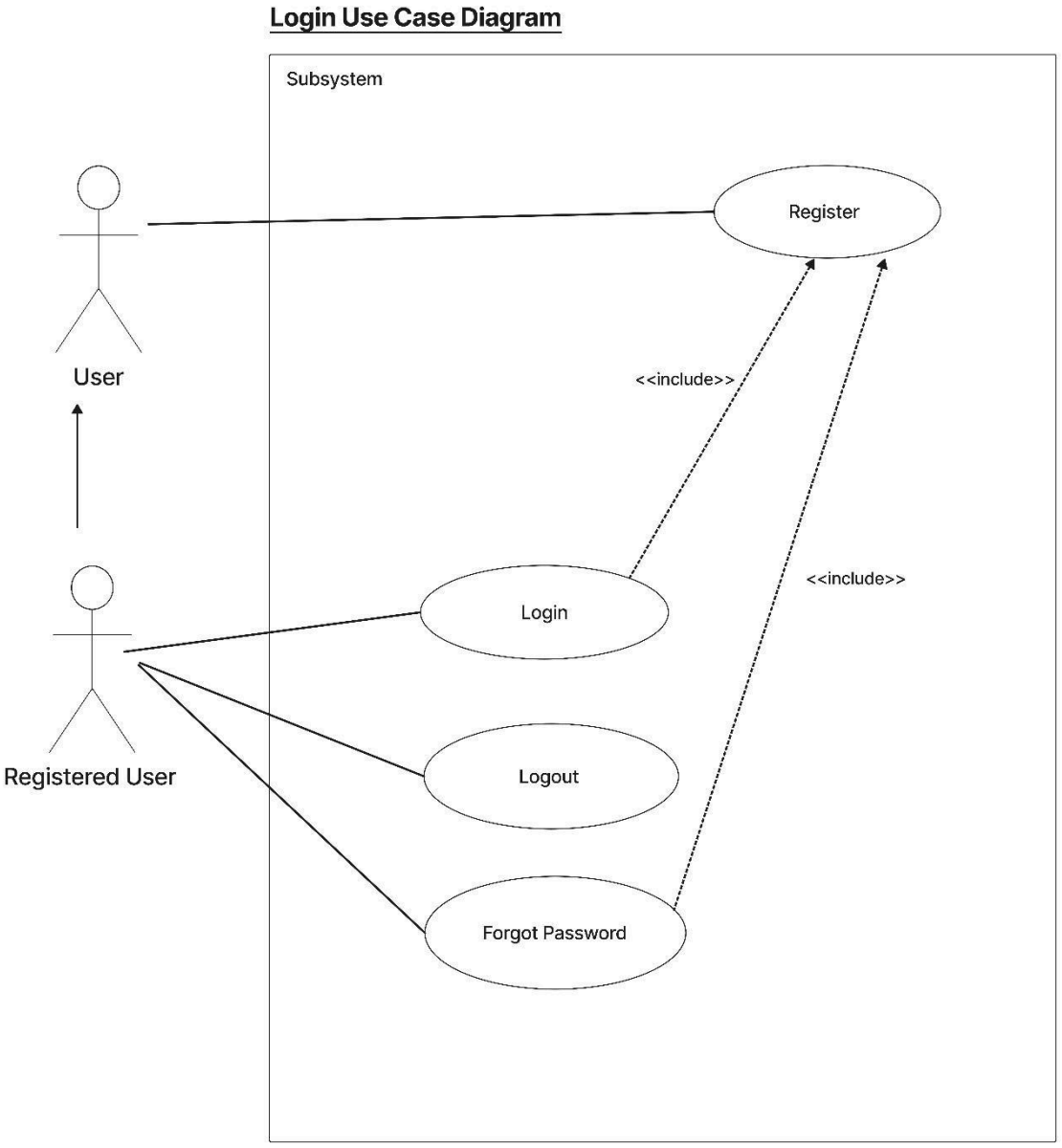


Figure 4 Use Case Diagram about User Login

Table 10 Use Case description for case named "Register"

Use Case No	10
Use Case Name	Register
Actors	User
Description	Defines how the user will register to the system.

Pre-Conditions	1. User should not have an account.
Post-Conditions	1. User is now registered to use FloraVision.
Basic Flow	1. User goes to Registration page of the FloraVision web application. 2. User enters his/her email, password, username. 3. If everything is correct user will register to FloraVision.
Alternative Flow	1. User goes to Registration page of the FloraVision web application. 2. User enters an email that is already in use by someone else. 3. The system prompts with an error message.
Functional Requirements	(FVF-6-2), (FVF-6-3), (FVF-6-4), (FVF-6-5), (FVF-6-6)

Table 11 Use Case description for case named "Login"

Use Case No	11
Use Case Name	Login
Actors	Registered User
Description	Defines how the user will login to the system.
Pre-Conditions	1. User is already a registered user.
Post-Conditions	1. User will login to the system.
Basic Flow	1. User goes to Login page of FloraVision web application.

	2. He/she enters her email and password. 3. Clicks confirm button. 4. He/she login to the system.
Alternative Flow	1. User goes to Login page of FloraVision web application. 2. He/she enters her email and password. 3. Clicks confirm button. 4. System prompts as wrong email or password.
Functional Requirements	(FVF-6-6)

Table 12 Use Case description for case named "Logout"

Use Case No	12
Use Case Name	Logout
Actors	Registered User
Description	Defines how the user will logout of the system.
Pre-Conditions	1. User is already logged in. 2. User is already registered.
Post-Conditions	1. User will logout from the system.
Basic Flow	1. User presses the logout button from the navigation bar. 2. User is logged out of the system.
Alternative Flow	
Functional Requirements	(FVF-6-15)

Table 13 Use Case description for case named "Forgot Password"

Use Case No	13
Use Case Name	Forgot Password
Actors	Registered User
Description	Defines how the user will recover their password.
Pre-Conditions	1. User is already registered.
Post-Conditions	2. User will have a new password.
Basic Flow	1. User will go into the login page. 2. He/she will press the “Forgot password” button. 3. The system will send an email with a confirmation code. 4. If the confirmation code in inputted correctly, he/she will be redirected to new password page. 5. He/she will input her new password.
Alternative Flow	1. User will go into the login page. 2. He/she will press the “Forgot password” button. 3. The system will send an email with a confirmation code. 4. If the confirmation code in inputted wrongly, he/she will not be directed to a new password page.
Functional Requirements	(FVF-6-7), (FVF-6-8), (FVF-6-9), (FVF-6-10), (FVF-6-11), (FVF-6-12)

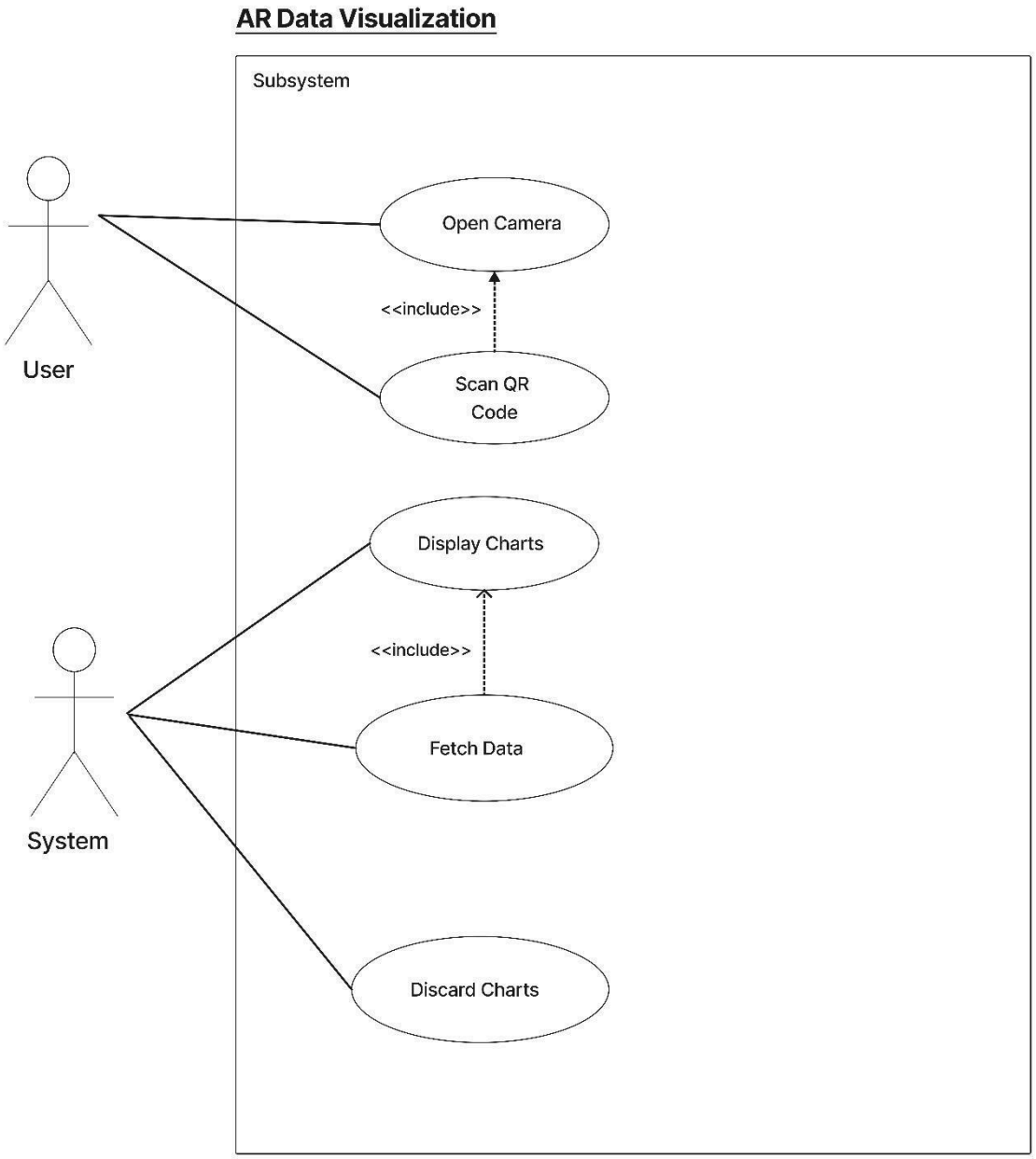


Figure 5 Use Case diagram for AR Data Visualization

Table 14 Use Case description for case named "Open Camera"

Use Case No	14
Use Case Name	Open Camera
Actors	User
Description	Defines how the user will access to AR.
Pre-Conditions	1. User is already logged in. 2. User already have devices in the system.
Post-Conditions	
Basic Flow	1. User presses the camera icon on the dashboard page. 2. Camera opens.
Alternative Flow	
Functional Requirements	(FVF-9-2)

Table 15 Use Case description for case named "Scan QR Code"

Use Case No	15
Use Case Name	Scan QR Code
Actors	User
Description	Defines how the user will see the plant data.
Pre-Conditions	1. User is logged in. 2. User opened the camera from dashboard page.
Post-Conditions	1. User sees their plant‘s data.
Basic Flow	1. User get closer to the QR code of their device.

	2. QR code is scanned automatically.
Alternative Flow	
Functional Requirements	(FVF-9-3), (FVF-9-4)

Table 16 Use Case description for case named "Fetch Data"

Use Case No	16
Use Case Name	Fetch Data
Actors	System
Description	Defines how the system get the devices data.
Pre-Conditions	1. User has scanned the QR code of their device.
Post-Conditions	1. System fetches the device data.
Basic Flow	1. After the QR code is scanned by the user, the system gets the id of the device. 2. Id of the device has been queried from AWS. 3. System received the data.
Alternative Flow	1. Scanned QR code is not related with the product, so system ignored.
Functional Requirements	(FVF-9-4)

Table 17 Use Case description for case named "Display Charts"

Use Case No	17
Use Case Name	Display Charts
Actors	System
Description	Defines how the system will show the plant's charts.
Pre-Conditions	1. User has scanned the QR code of their device. 2. System has fetched the data of the device.
Post-Conditions	1. System displays the charts.
Basic Flow	1. After fetching the data, system displays the chart on a static position of the QR code.
Alternative Flow	
Functional Requirements	(FVF-9-5)

Table 18 Use Case description for case named "Discard Charts"

Use Case No	18
Use Case Name	Discard Charts
Actors	System
Description	Defines how the system will discard the plant's charts.
Pre-Conditions	1. System has already displayed a chart.
Post-Conditions	1. System discards the chart.
Basic Flow	1. After user scanned one QR code of their device and received charts.

	2. If the user scans another QR code of another device, system will discard the other charts of the previous device.
Alternative Flow	
Functional Requirements	(FVF-9-8)

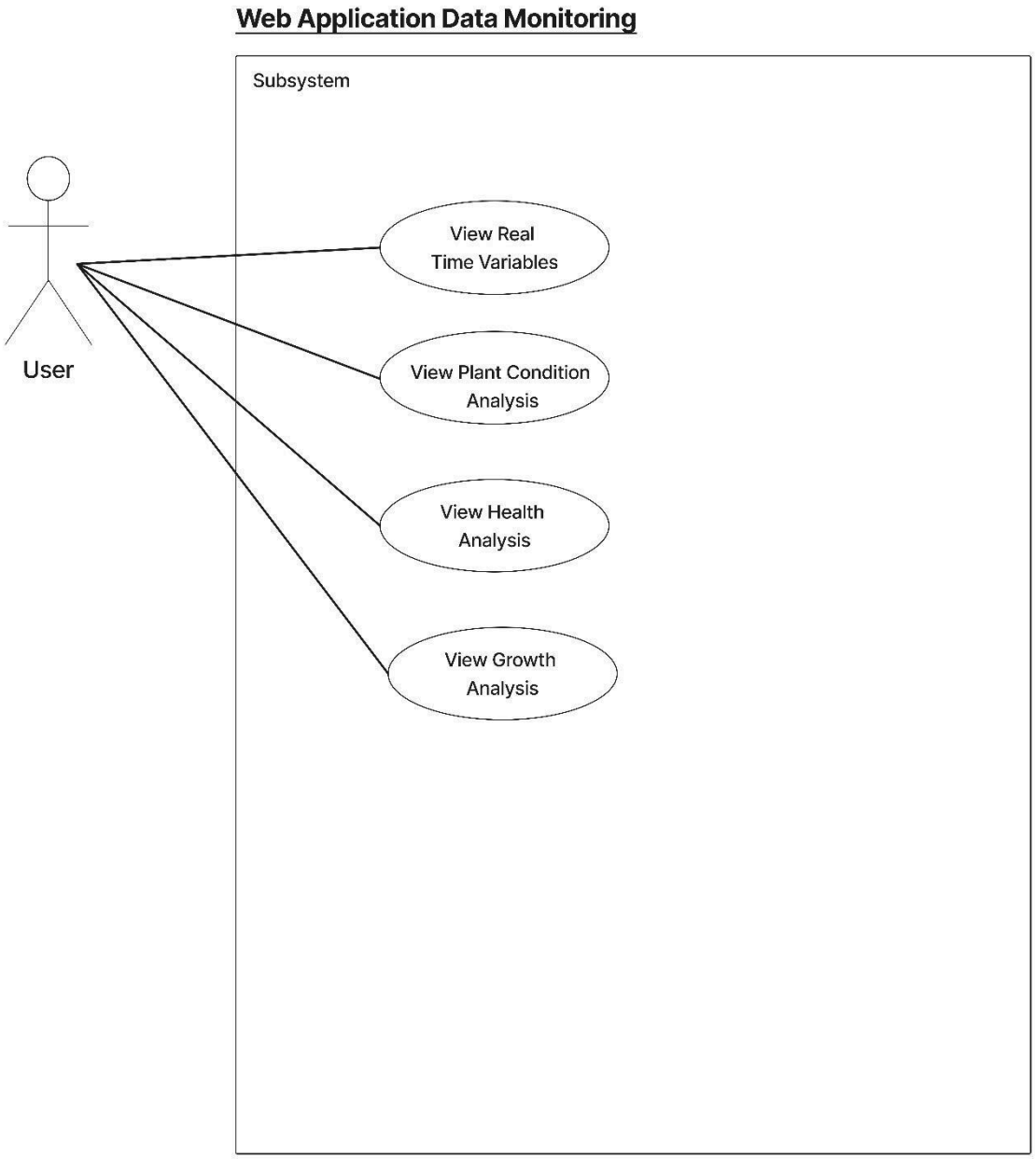


Figure 6 Use Case diagram of Web Application Data Monitoring

Table 19 Use Case description for case named "View Real Time Variables"

Use Case No	19
Use Case Name	View Real Time Variables
Actors	User
Description	Defines how the user will see their plants real time data.

Pre-Conditions	1. User is logged in. 2. User at least have one device in their system.
Post-Conditions	1. System shows the real time data of the plant.
Basic Flow	1. Users will click on the device they want to look at from their devices tab. 2. On the dashboard of the application, they will see real time variables about their plant.
Alternative Flow	
Functional Requirements	(FVF-7-8)

Table 20 Use Case description for case named "View Plant Condition Analysis"

Use Case No	20
Use Case Name	View Plant Condition Analysis
Actors	User
Description	Defines how the user will see their plants conditional analysis.
Pre-Conditions	1. User is logged in. 2. User at least have one device in their system.
Post-Conditions	1. System shows the plant's condition analysis.
Basic Flow	1. Users will click on the device they want to look at from their devices tab. 2. On the dashboard of the application, they will see real time variables about their plant.

	3. On the top of the dashboard there will be tabs saying daily, weekly, and monthly. 4. User will press on of them.
Alternative Flow	
Functional Requirements	(FVF-7-9)

Table 21 Use Case description for case named "View Health Analysis"

Use Case No	21
Use Case Name	View Health Analysis
Actors	User
Description	Defines how the user will see their plants health analysis.
Pre-Conditions	1. User is logged in. 2. User at least have one device in their system.
Post-Conditions	1. System shows the plant's health analysis.
Basic Flow	1. Users will click on the device they want to look at from their devices tab. 2. On the dashboard of the application, they will see real time variables about their plant. 3. On the top of the dashboard there will be tabs saying daily, weekly, and monthly. 4. User will press one of them. 5. The system will provide the user with a chart containing their plant's health scores.
Alternative Flow	

Functional Requirements	(FVF-10-1)

Table 22 Use Case description for case named "View Growth Analysis"

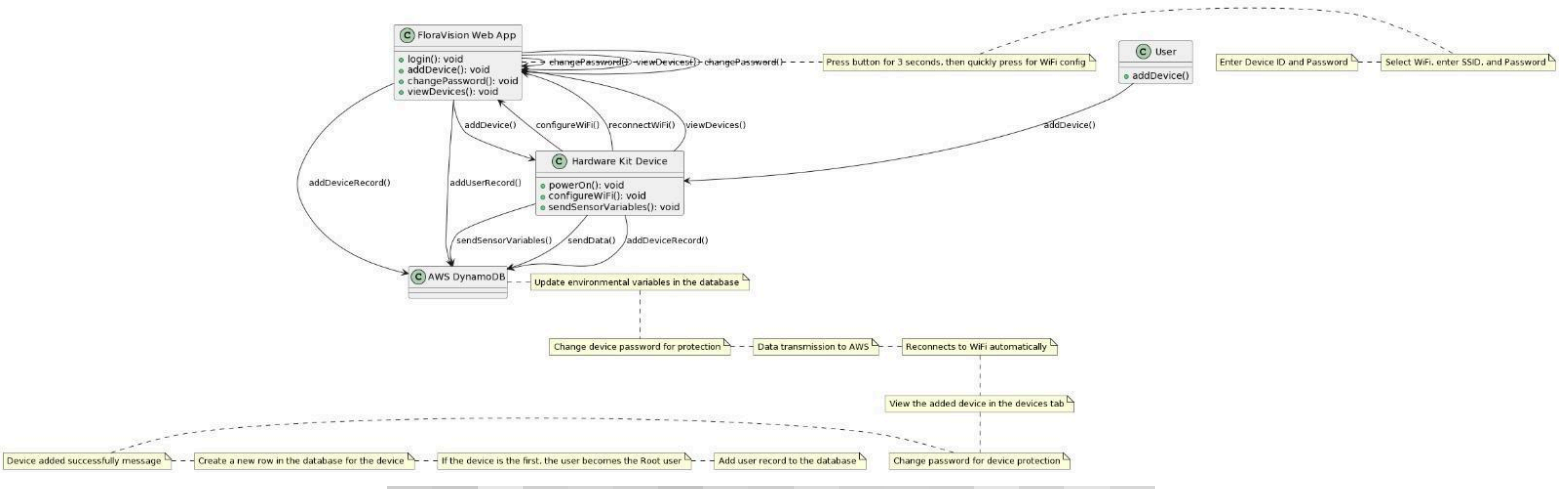
Use Case No	22
Use Case Name	View Growth Analysis
Actors	User
Description	Defines how the user will see their plants growth analysis.
Pre-Conditions	3. User is logged in. 4. User at least have one device in their system.
Post-Conditions	2. System shows the plant's growth analysis.
Basic Flow	6. Users will click on the device they want to look at from their devices tab. 7. On the dashboard of the application, they will see real time variables about their plant. 8. On the top of the dashboard there will be tabs saying daily, weekly, and monthly. 9. User will press one of them. 10. The system will provide the user with a chart containing their plant's growth scores.
Alternative Flow	
Functional Requirements	(FVF-10-1)

6. System Model

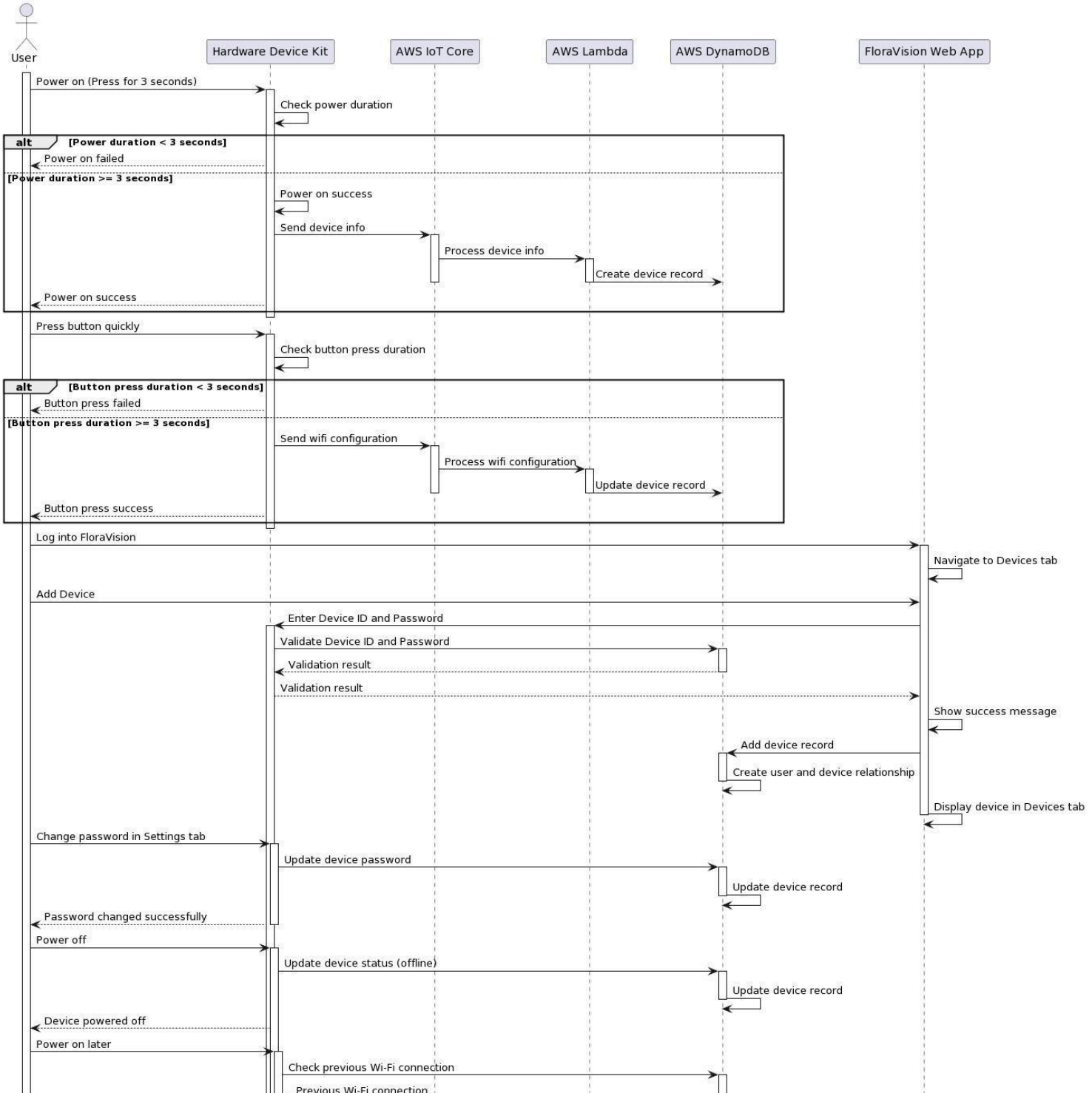
In this part we provided Activity Diagrams, Analysis Level System Sequence Diagrams, and Analysis Level Class Diagrams for two most complex use cases of the FloraVision. We considered that most complex use cases in our system are “Add Device” use case defined at Table 1 which is part of the Use Case Diagram provided at Figure 1. And “View Health Analysis” use case defined at Table 21 which is a part of the Use Case Diagram provided at Figure 5.

1. Use Case “Add Device”:

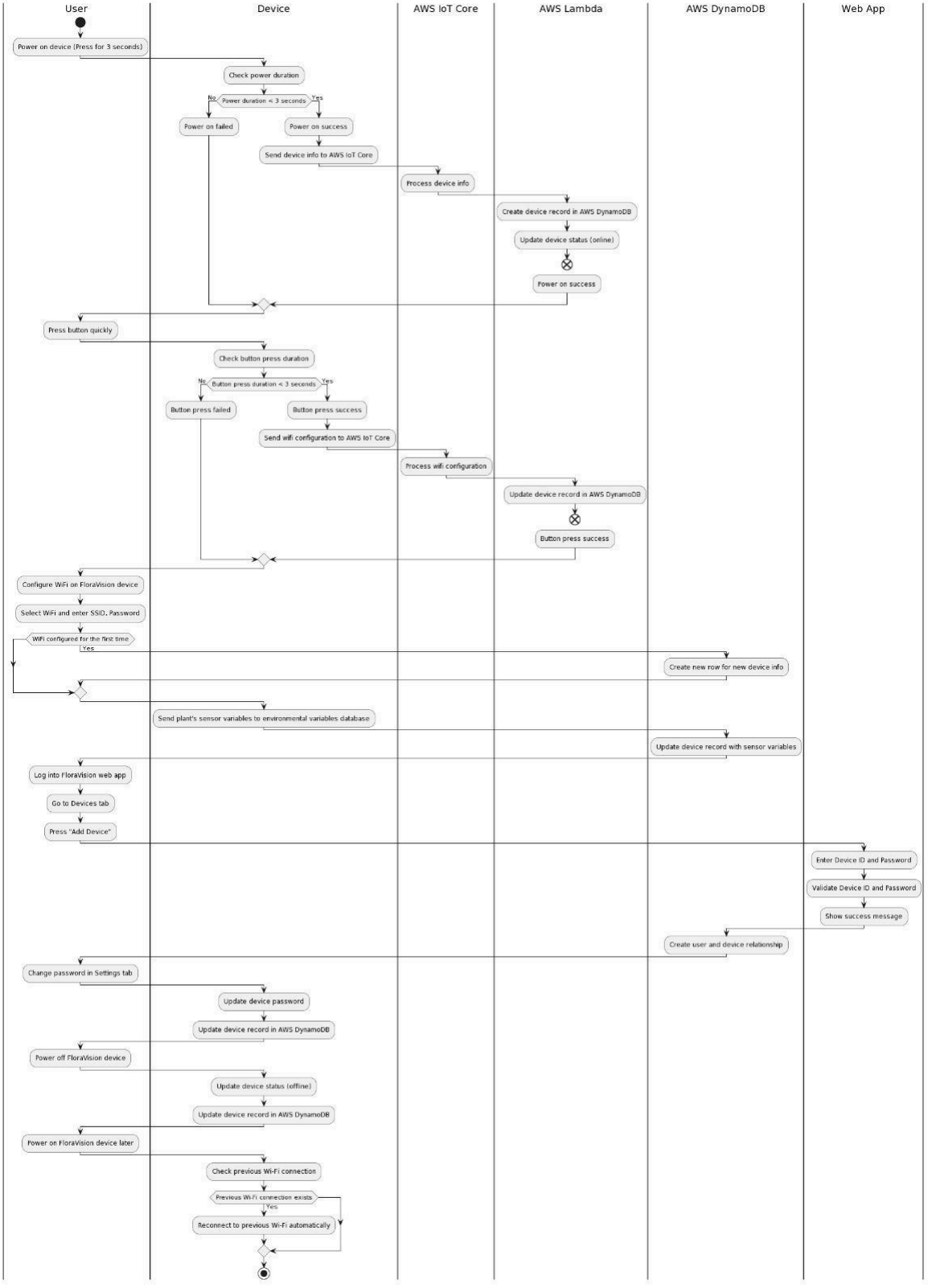
1.1 Class Diagram:



1.2 Sequence Diagram:

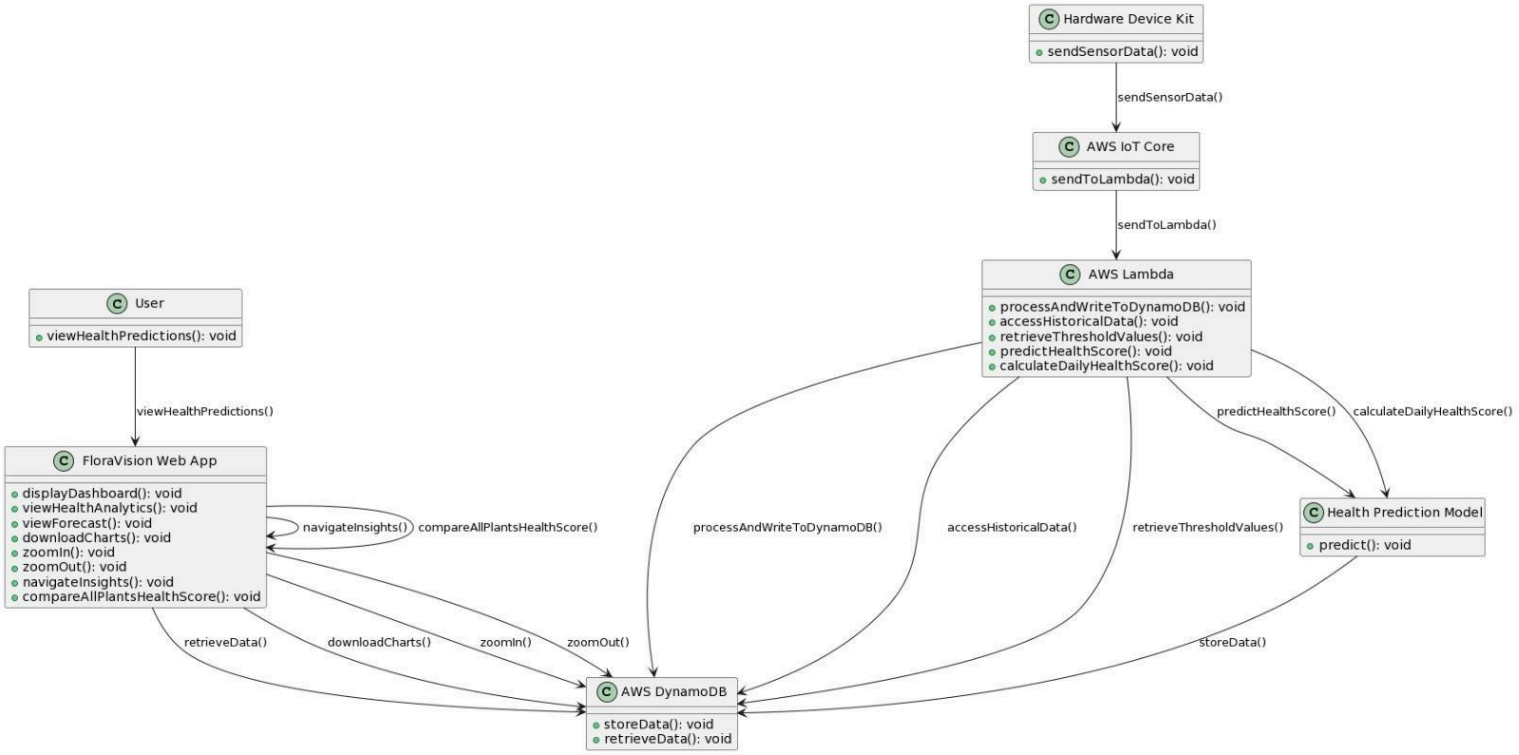


1.3 Activity Diagram:

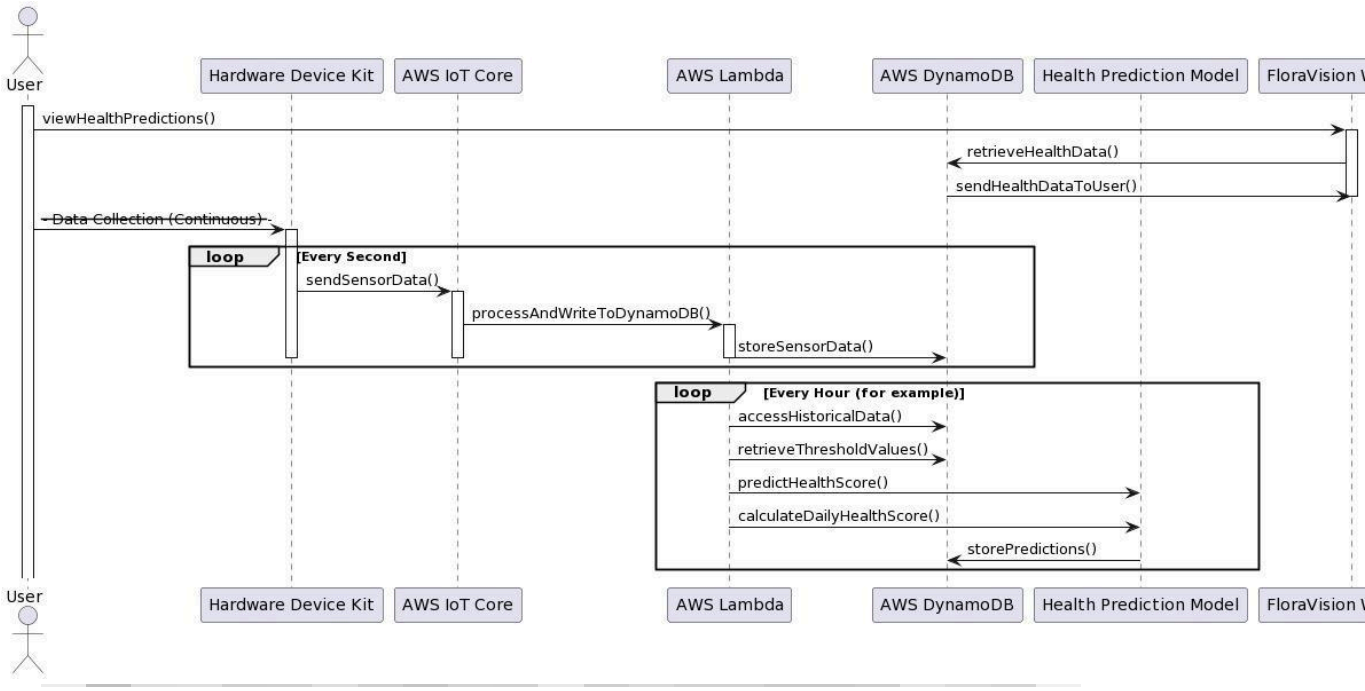


2. Use Case "View Health Analysis":

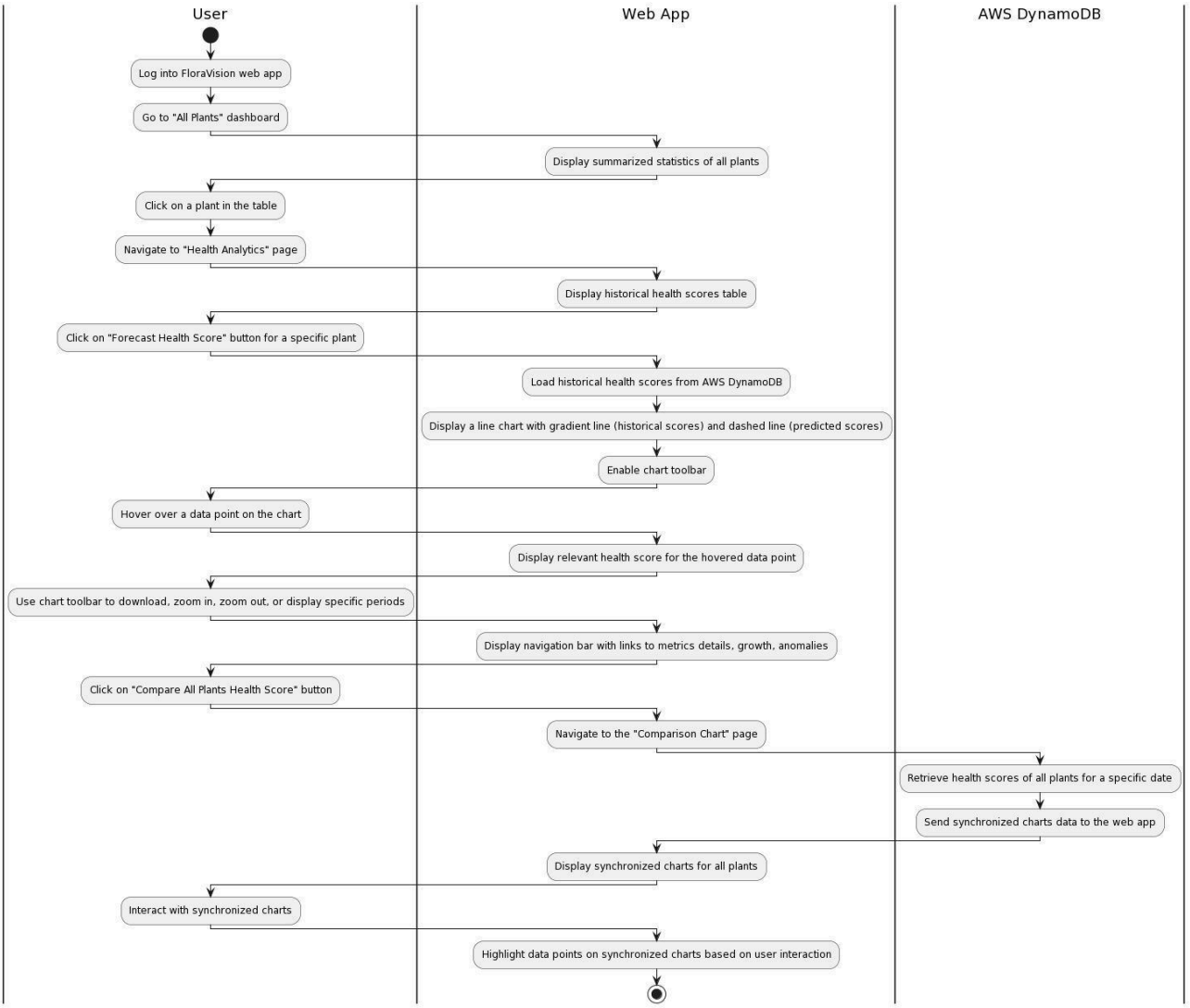
2.1 Class Diagram:



2.2 Sequence Diagram:



2.3 Activity Diagram:



7. Requirements Prototypes

In this part we exemplify the 2 most complex use cases, these include “Add Device” and “View Health Analysis” use cases.

1st Prototype – Add Device:

In the Figure 12 we provided how the device adding system will persist. This includes powering on the hardware kit to connect it to Wi-Fi to adding the device to our web application.

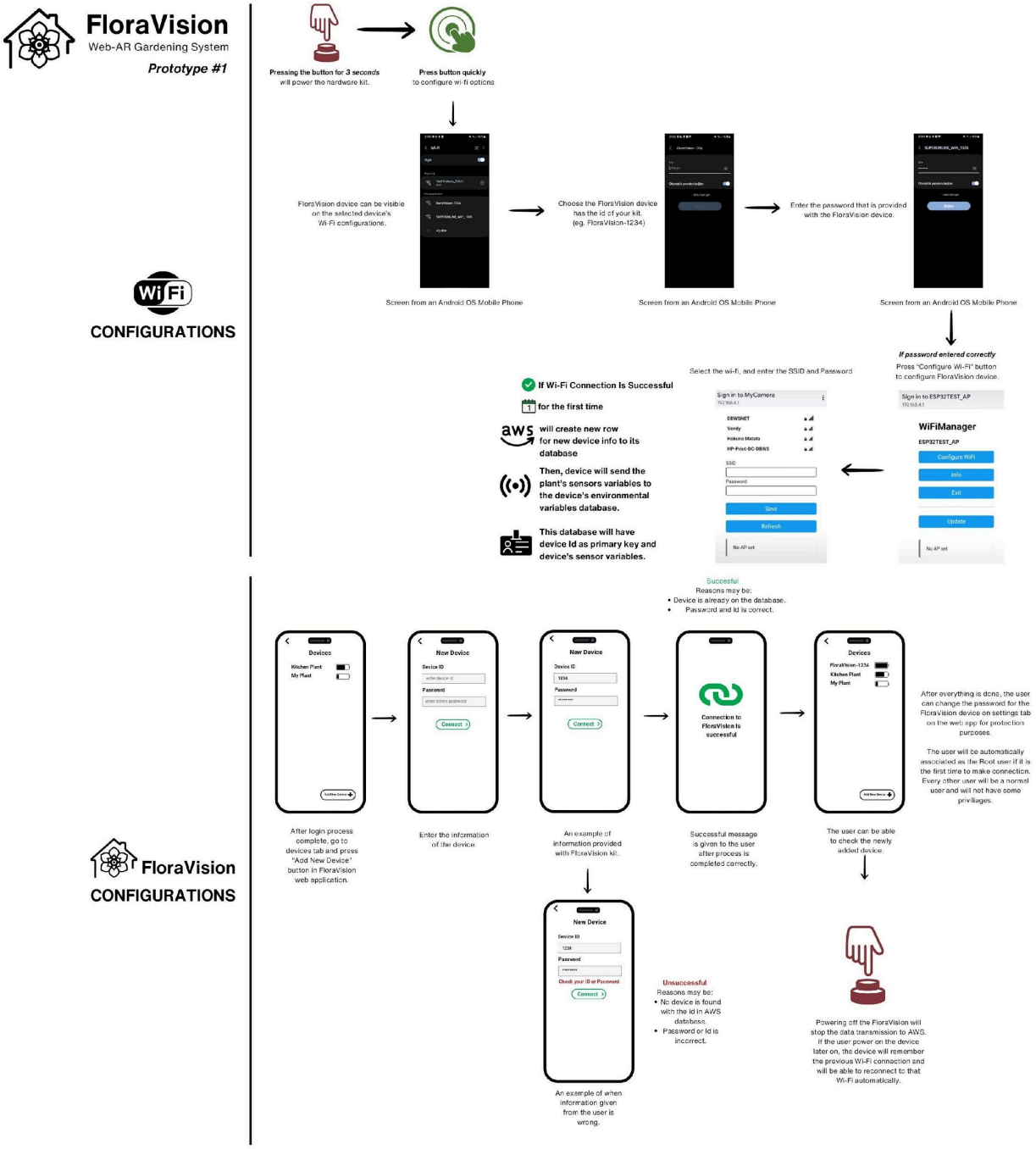


Figure 13 Prototype 1: Add Device

2nd Prototype – View Health Analysis:

In the Figure 13 we provided how the health analysis calculations will be handled and how this analysis will be provided to the user.

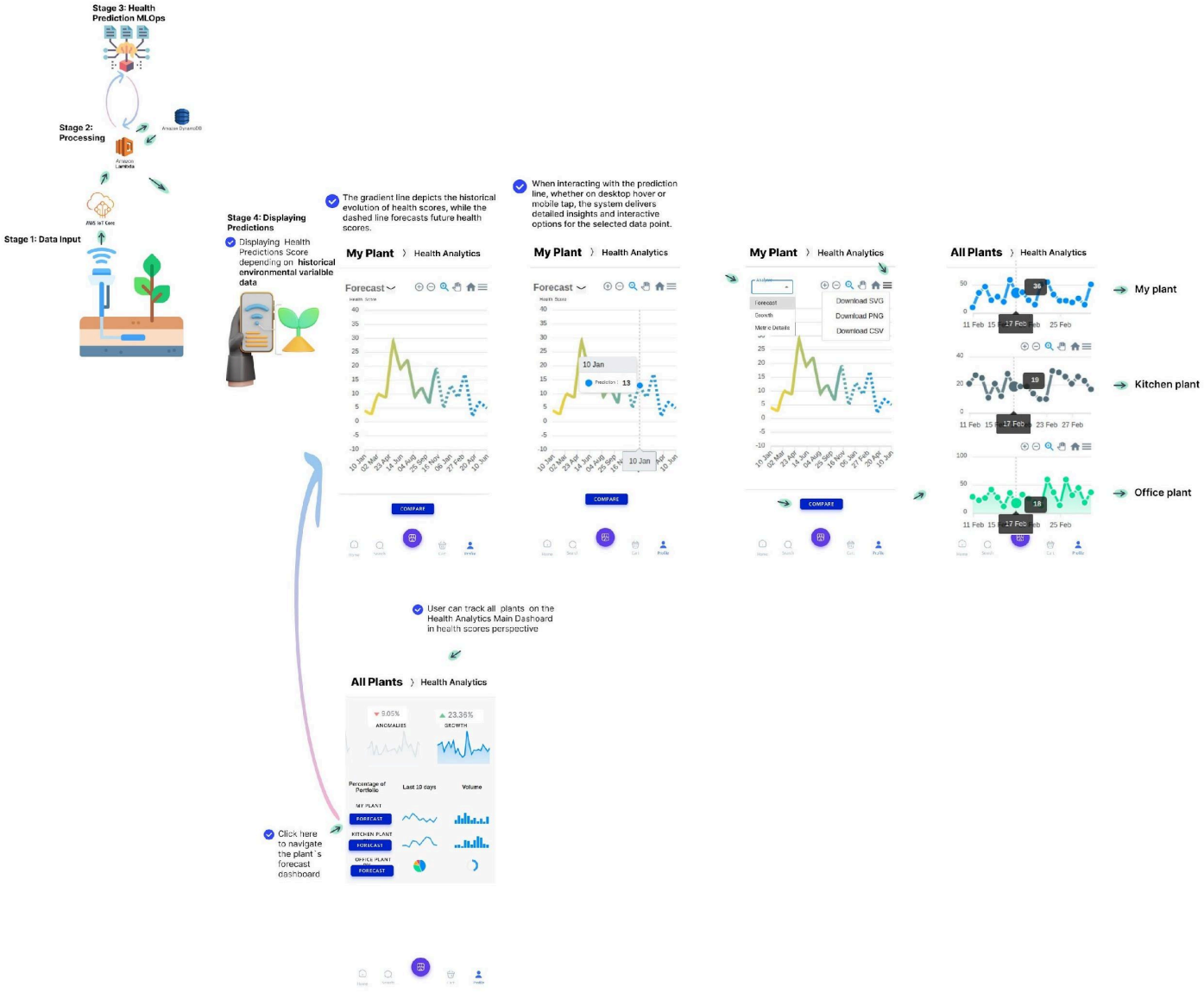


Figure 14 Prototype 2: View Health Analysis

8. Non-functional Requirements

In this section, we provided non-functional requirements that the system will rely on. These non-functional requirements are divided into 7 groups. Usability, performance, software system attributes, constraints, error handling and other requirements.

1. Usability requirements

- (FVN-1-1) The web application shall be compatible with major browsers (Chrome, Firefox, Safari) to provide a consistent experience across different platforms except the AR features.

2. Performance requirements

- (FVN-2-1) The system should handle a minimum of 10 simultaneous users. AWS Lambda, DynamoDB, and SNS are designed to scale horizontally to accommodate varying workloads.
- (FVN-2-2) The system shall hold the device's data history for at least 1 month.

3. Software system attributes

a) Reliability

- (FVN-3-1) The system shall be reliable by providing health and growth insights about the plant using reliable academic sources for creating reliable algorithms.
- (FVN-3-2) The system shall not include mid and high priority level failures/defects at launch.

b) Availability

- (FVN-4-1) The system shall have real time administration by sending notifications to system administrators when in the event of a system failure.

c) Security

- (FVN-5-1) The system shall log every system failure.

d) Maintainability

We did not identify any maintainability related non-functional requirements.

e) Portability

- (FVN-6-1) The web application shall enable users to log in from other devices that are compatible to use the web product.
- (FVN-6-2) The code for each hardware device shall be only for that device only. This is because every hardware device will have its own id for data.

4. Constraints

- (FVN-7-1) The product shall adhere to privacy laws and rules about data sharing, keeping, and handling.
- (FVN-7-2) The system shall only provide English as a language for all products.

5. Error Handling Requirements.

- (FVN-8-1) If a user provides the wrong email or password while logging in, the system shall want the user.
- (FVN-8-2) If a user provides a non-existing or already in use email while registering, the system warns the user.
- (FVN-8-3) If a user provides a password with non-required type, the system shall warn the user.
- (FVN-8-4) If the AWS Core receives an incorrectly formatted message from the hardware product, the AWS Core shall ignore the message and provide a log about the ignored message.
- (FVN-8-5) If a user tries to access unauthorized locations in the web application, the system shall warn and redirect the user to the login page.
- (FVN-8-6) If a user enters and provides two different passwords while registering, the system shall warn the user.

6. Other Non-Functional Requirements

There are no other non-functional requirements.

9. Logical Database Requirements

In this part, we indicated the database structure of FloraVision. In the first part we provided our tables and their details. In the Figure 14 we provided our EER diagram for FloraVision

1. Users Table (DynamoDB):

- Partition Key: UserID (from Cognito)

- Attributes:

- **Username:** Username created by the user.
- **Email:** Email of the user.
- **Password:** Password of the user.
- **TimezonePreference:** Captures the user's time zone preference for displaying time-sensitive information.
- **MetricPreference:** Allows users to choose between different units for metrics (e.g., Celsius vs. Fahrenheit for temperature).
- **DashboardCustomization:** Captures any specific customization preferences the user has for the dashboard layout or features.
- **NotificationPreference:** Stores the preferred notification channel(s) for the user (e.g., email, SMS, web push). Default is web push notifications.

2. Devices Table (DynamoDB):

- Partition Key: UserID

- Sort Key: DeviceID

- Attributes:

- **DeviceType:** Type of the device
- **DeviceName:** Name of the device
- **Status:** Is it on or off

- **Credentials:** Who is the root user.

3. SensorData Table (DynamoDB):

- Partition Key: UserID

- Sort Key: DataID

- Attributes:

- **Timestamp:** The time that the data is created.
- **Temperature:** Temperature detected by the device.
- **Humidity:** Humidity detected by the device.
- **LightIntensity:** Light Intensity detected by the device.
- **SoilMoisture:** Soil Moisture detected by the device.
- **AirQuality:** Air Quality detected by the device.

4. Notifications Table (DynamoDB):

- Partition Key: UserID

- Sort Key: NotificationID

- Attributes:

- **Timestamp:** The time that the notification is created.
- **Content:** Content of the issued notification.
- **Status:** Status of the notification (is it read by the user)

5. Predictions Table (DynamoDB):

- Partition Key: UserID

- Sort Key: PredictionID (String)

- Attributes:

- **Timestamp:** The time that the prediction is created
- **PredictedValue:** Predicted value for the given prediction.

6. ScientificThresholds Table (DynamoDB):

- **Description:** Keeps the optimum values of the specific plant.
- **Partition Key:** PlantID (String): A unique identifier for each type of indoor plant. This could be a common name, scientific name, or any unique identifier you choose.
- **Attributes:**
  - **TemperatureMin** (Number): The minimum temperature threshold for the plant.
  - **TemperatureMax** (Number): The maximum temperature threshold for the plant.
  - **HumidityMin** (Number): The minimum humidity threshold for the plant.
  - **HumidityMax** (Number): The maximum humidity threshold for the plant.
  - **LightIntensityMin** (Number): The minimum light intensity threshold for the plant.
  - **LightIntensityMax** (Number): The maximum light intensity threshold for the plant.
  - **SoilMoistureMin** (Number): The minimum soil moisture threshold for the plant.
  - **SoilMoistureMax** (Number): The maximum soil moisture threshold for the plant.
  - **CO2DensityMin** (Number): The minimum CO2 density threshold for the plant.
  - **CO2DensityMax** (Number): The maximum CO2 density threshold for the plant.

7. Anomalies Table (DynamoDB):

- **Description:** Keeps every anomaly detected by the device.
- **Partition Key:** UserID
- **Sort Key:** AnomalyID
- **Attributes:**
    - **Timestamp:** The time the anomaly is detected.
    - **AnomalyType:** Which type of anomaly this is.
    - **DetectionConfidence:** Percentage of confidence for this anomaly.
    - **NotificationStatus:** Is notification read by the user.
    - **AnomalyDetail:** Detail about the anomaly.

7. Historical Health Scores Table (DynamoDB):

- **Description:** Keeps the history of the health scores provided for the plant.
- **Partition Key:** UserID
- **Sort Key:** ScoreID
- **Attributes:**
    - **Timestamp:** The time the anomaly is detected.
    - **HealthScore:** Health score created for the plant.
    - **Temperature:** Temperature of the plant's environment.
    - **LightIntensity:** Light Intensity of the detected.
    - **SoilMoisture:** Soil moisture of the plant.
    - **AirQuality**: Air Quality of the plant's environment.
    - **Humidity**: Humidity of the plant's environment.
    - **PredictionConfidence**: Confidence for this health score in percentage
    - **NotificationStatus**: Is the notification is read by the user.

10. Verification

To ensure that the application is working as intended, we specified our verification process that would help us ensure that the product will be clear of defects/failures. We divided the verification process into 3 sections, Hardware, Data Analysis and Web Application Verifications.

1. **Hardware Testing:**
    - Integration Test: Ensure all hardware components (Arduino Nano, sensors, Wi-Fi module) work seamlessly together.
    - Unit Tests: Test each sensor individually to verify accurate readings.
    - Power Testing: Validate power-on and power-off functionalities.
    - Connectivity Testing: Test Wi-Fi connection and data transmission.
2. **Sensor Data Reading:**
    - Unit Tests: Test each sensor's data reading independently for accuracy.
    - Integration Test: Verify that all sensors can be read simultaneously without interference.
3. **Data Transmission:**
    - Unit Tests: Test the creation of JSON format data and MQTT message sending individually.
    - Integration Test: Verify the entire data transmission process from sensor reading to AWS.
4. **Hardware-Web Integration:**
    - Integration Test: Ensure the web application can successfully connect to the hardware kits.
    - User Interface Testing: Validate that user interactions with the devices tab work as intended.
5. **Hardware Kit Authentication:**
    - Unit Tests: Test password and ID authentication individually.

- Integration Test: Verify the overall authentication process, including assigning root and normal user roles.

6. Web Authentication:

- Unit Tests: Test registration and login functionalities individually.
- Integration Test: Verify the end-to-end user authentication process.
- Session Management Testing: Ensure user sessions are maintained correctly.

7. Failure and Success Scenarios:

- Edge Case Testing: Test scenarios such as token expiration, incorrect login credentials, and successful authentication.
- Logout Testing: Verify the log-out functionality works as intended.

8. Hardware Setup on Web:

- Integration Test: Ensure the web application displays connected devices and their statistics correctly.
- User Interface Testing: Validate user interactions with adding, removing, and navigating devices.

9. Dashboard and Charts:

- Integration Test: Confirm that real-time and historical sensor data is displayed accurately.
- User Interface Testing: Validate user interactions with customizing the dashboard and navigating through charts.

10. Notifications:

- Unit Tests: Test notification generation and display individually.
- Integration Test: Verify the end-to-end notification system, including marking notifications as read.

Anomaly Notification Testing with AWS SNS and Lambda:

1. **Objective:** Validate the generation and delivery of anomaly notifications when the system detects abnormal plant conditions using AWS SNS and associated Lambda functions.
2. **Test Cases:**
    - **Generate Anomaly Scenario:**
        - Simulate abnormal sensor readings that trigger an anomaly.
        - Ensure the system correctly identifies the anomaly.
        - Confirm the initiation of the anomaly notification process.
    - **AWS SNS Topic Integration:**
        - Verify that the anomaly detection system triggers the AWS SNS topic for anomaly notifications.
        - Ensure that the SNS topic is configured correctly.
    - **Lambda Subscription:**
        - Confirm that the Lambda function is correctly subscribed to the SNS topic.
        - Validate that the Lambda function is triggered upon the anomaly event.
    - **Lambda Function Execution:**
        - Test the Lambda function's execution to process the anomaly event.
        - Verify that the Lambda function formats the notification content correctly.
    - **AWS SNS Notification:**
        - Check if the Lambda function successfully publishes the anomaly notification to the SNS topic.
        - Ensure that the SNS service delivers the notification to subscribed endpoints (e.g., users).
    - **User Interaction:**

        - Validate that the user receives and can view the anomaly notification through the chosen communication channel.
        - Confirm that the notification is appropriately marked as read after the user views it.
    - **Notification Content:**
        - Verify the content of the notification for accuracy and clarity.
        - Test different formats or styles of notifications supported by SNS.
    - **Edge Cases:**
        - Simulate various anomaly scenarios, including different types and levels of anomalies.[1]

11. Augmented Reality:

- Compatibility Testing: Ensure AR features work on supported browsers and devices.
- Marker Recognition Testing: Validate marker-based AR visualization.

12. Predictions Data Science and Analysis:

- Integration Test: Verify the integration of the health prediction model into the system.

13. Root User Management:

- Unit Tests: Test changing device name, password, and user roles individually.
- Integration Test: Verify overall root user management functionalities.[2]

---

[1] GenAI tool: ChatGPT 3.5
Prompt: First inserted every functional requirement we had and prompted "Based on these requirements write integration and unit cases considering AWS services."
Rationale: To find an answer.

[2] GenAI tool: ChatGPT 3.5
Prompt: First inserted every functional requirement we had and prompted "According to these functional requirements we will create Verification Methods. Can you create them for us. which kind of tests should be used: Integration test etc."
Rationale: To find an answer.

AWS Lambda Functions Unit Testing:

Lambda function execution

For testing the lambda function, we need a way to execute/call a lambda function from our local. For that, we can use a third-party package called lambda-tester.

Lambda-Tester 4.0.1: Aiding in running lambda functions locally.

Overriding and mocking the dependencies

Dependencies can be any third-party libraries or DB calls or even an API call. To override and mock these dependencies we will use proxyquire 2.1.3 package.

Proxyquire will help us import the lambda function without calling (invoking) it and also help us mock the dependencies used inside the lambda function.

- Chai 4.3.10: Assertion library to verify code correctness.
- Mocha 10.2.0: For creating a test suite and running test cases.

Additionally, Elastic APM (Application Performance Monitoring) will be employed on lambda functions to monitor and track the performance of AWS Lambda functions, ensuring optimal functionality and identifying areas for improvement.

AWS LAMBDA FUNCTIONS UNIT TEST CASES

- Sensor Data Processing Unit Test:
  - Objective: Ensure the correct processing of sensor data.
  - Test Cases:
    - Verify the correctness of data processing algorithms.
    - Check the handling of different sensor inputs.
- Data Storage Unit Test:
  - Objective: Confirm accurate data storage in AWS DynamoDB.
  - Test Cases:
    - Validate the functionality of the code responsible for storing data in DynamoDB.
    - Verify the correctness of data format and attributes.

- Health Prediction Model Execution Unit Test:
  - Objective: Validate the execution of the health prediction model through AWS Lambda.
  - Test Cases:
    - Test the individual components of the health prediction model.
    - Ensure the model produces accurate predictions in isolation.
- Scientific Threshold Values Retrieval Unit Test:
  - Objective: Test the retrieval of scientific threshold values from DynamoDB.
  - Test Cases:
    - Validate the functionality of the code responsible for fetching threshold values.
    - Verify the correctness of retrieved values.[3]

[3] GenAI tool: ChatGPT 3.5
Prompt: First inserted every functional requirement we had and prompted "Write a more detailed guideline for unit testing implementation on AWS Lambda functions."
Rationale: To find an answer.

11. Discussions

1. Limitations and Constraints

- While collecting the requirements for the SRS we were in midterm week and our schedules were busy. Therefore, we were not able to work on this document as we intended.
- While collecting the requirements, some requirements for the product needed background information about the subject. For example, for AR and Data analysis we couldn't provide the requirements as detailed because the subjects were not familiar to us and our requirements about them were vague.

2. Health and Safety Issues

- There were no health and safety issues while collecting the requirements for this document.

3. Legal Issues

- While developing the requirements, we did not face any legal issues.

4. Economic Issues and Constraints

- While developing the document we did not face any economic issues or constraints.

5. Sustainability

- While developing the requirements, team meetings were held online. Therefore, we did not use any automobiles. This cut costs and helped environmental sustainability.
- We did not use any paper or printed material while developing this document, and only copies of the documents were a Google Doc and a Word document.

6. Ethical Issues

- While developing this document, we did not face any ethical issues.

7. Multidisciplinary Collaboration

- We did not use any multidisciplinary collaboration while developing our requirements.

12. References

[1] M. Köroğlu, E. Tarak, T. R. Abid, E. Altay, Initial Plan of Team 5, 2023-2024 CTIS Senior Projects, 2023.