Northumbria University
NEWCASTLE

**A report submitted in partial fulfilment of the regulations governing the award of the Degree of BSc. (Honours) Computer Science at the University of Northumbria at Newcastle**

**KV6003: Individual Computing Project**

# Smart Plant Caring System using IoT

**General Computing Project**

**Teck Xun Tan**

**2021/2022**

# Declarations

I declare the following:

(1) that the material contained in this dissertation is the end result of my own work and that due acknowledgement has been given in the bibliography and references to **ALL** sources be they printed, electronic or personal.

(2) the Word Count of this Dissertation is .....25972.....

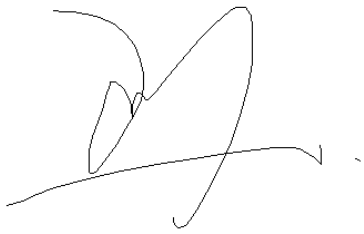
(3) that unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on the eLearning Portal (Blackboard), if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.

(4) I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other School or from other institutions using the service.

In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

(5) I have read the Northumbria University/Engineering and Environment Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been considered, evaluated and appropriately addressed in this research.

SIGNED:



Please remember to sign this declaration and include it before submitting your dissertation for binding.

# Abstract

This project will be investigate urban gardening by developing a smart plant caring system that any end-user could effortlessly utilise to participate in urban gardening in urban areas. As mentioned in the Terms of Reference (Appendix 1), the project consists of two aims, the first aim is to build a prototype Smart Plant Caring System to prove the concept of plant caring automation and the second aim is to develop a web dashboard application to visualise all the collected data to allow a user to analyse and learn the trends and patterns of the data.

To support the development and address the aims of the proposed system, a critical literature review has been conducted to obtain a better understanding of the investigating topic, such as Urban Gardening Technology and the Internet of Things (IoT), followed by challenges that should be addressed such as handling data in IoT which includes the process of visualising the data and ensure the user's data security. After that, a procedure to capture requirements was performed to obtain requirements by reviewing similar projects and conducting a survey to collect additional requirements from the user's perspective. The project has also adopted the Waterfall System Development Life Cycle to ensure that the project's implementation could be carried out smoothly. During the design phase, the project has utilised Kruchten's 4+1 Architectural View Modal to serve as a guideline to design the system using various UML diagrams such as use case diagram, class diagram, sequence diagram and database design.

Through performing project testing such as Functionality Testing and Cross Browser Testing, the project ensured that all the implemented functional requirements were working as expected without any potential errors. Moreover, user testing has also been performed during the system testing phase to get more specific user feedback and ensure that the system has reached their expectation.

The project evaluation shows that the system has met all the functional requirements and has received good feedback from the users. However, due to the project being a prototype system, there are a few limitations where the system will not be able to handle real-time data due to resource limitations, and the system will not be able to retrieve the user's current location due to unsecured webpage limitations on browser such as Google Chrome.

In conclusion, the project has successfully implemented the Smart Plant Caring System that any users could use if they wish to take part in practising urban gardening. However, an improvement could still be made in the future by applying techniques such as Artificial Intelligence to make the future of the Smart Plant Caring System (SPCS) brighter.

# Table of Contents

# Section 1: Smart Plant Caring System using IoT

# Chapter 1: Introduction

## 1.1 Project Background

The project was inspired by the recent trend of urban gardening technologies in countries such as the United Kingdom, Singapore, Hong Kong and more for the past few years (Tan and Neo, 2009, Tam and Bonebrake, 2016, Dobson et al., 2020). Through urban gardening, various unique gardening approaches. However, out of all the concepts, one that stands out the most has been discussed by multiple researchers and developers: smart gardening.

Additionally, the project was also inspired by personal experience of overwatering plants. Overwatering was one of the most common causes of plant problems. Overwatering wastes water because it exceeds the amount of water the plant initially needs. Still, it also causes harm to the plants as it will drown the plants because they cannot absorb oxygen normally from their roots which leads them to death (Garden, 2021). Previous similar works have suggested that IoT smart gardening devices can stop overwatering plants and reduce water wastage. They can decide when to start or stop dispensing water depending on the soil's moisture level through the sensors (Pravin et al., 2018).

## 1.2 Aims

Two project aims have been created based on the project background shown above. The project aims are demonstrated as listed below:

a) To build a prototype Smart Plant Caring System to prove the concept of plant caring automation

b) To develop a web dashboard application to visualise all the collected data to allow a user to analyse and learn the trends and patterns of the data.

## 1.3 Objective

Additionally, a list of objectives has also been outlined to reference the goals to achieve within this project. The list of objectives is shown in the table below.

| Objectives | Purpose |
|---|---|
| Conducting literature review | To have a better understanding of the topic and tools chosen for this project, such as urban gardening technology, Internet of Things (IoT), and Data in IoT such as Data Visualisation and Data Security. |
| Creating a requirement specification for both IoT set up and dashboard | To understand what functions are required to be implemented and achieved for both IoT set-up and dashboard |
| Develop dashboard and IoT infrastructure design documentation (e.g., class diagram, sequence diagram, database design and more) | To support and ensure the speed and efficiency of the development phase by using these diagrams constructed as a reference |
| Conducting product testing | To ensure that every feature of the product is functioning, such as requirement, functionality, and usability and produce a test plan along with test results |
| Explore and understand which sensors components are required for the project | To identify what sensors and actuators to support the use case functionality and data requirement developed |
| Develop dashboard and IoT infrastructure for data integration from sensors | To establish the connection between IoT infrastructure and dashboard to collect and send necessary data from sensors |
| Implement the design | To develop and implement the features listed using codes like C++, PHP, Python and more |
| Develop technical skills for IoT implementation | To develop technical skills like how overall IoT works, how to build an IoT infrastructure and several new coding languages like C++ |
| Develop and evaluate dashboard design with users | To develop and get users evaluation on the dashboard design to enhance the user experience |
| Summarise recommendations and future direction of the project | To summarise the recommend and analyse what future direction could be implemented for the project |

**Figure 1: Project Objective**

## 1.4 Product Overview

The proposed work is aimed to build a smart plant caring system based on IoT that will enable plant caring automation. The Smart Plant Caring System (SPCS) is proposed to detect the moisture level of the soil and water. The plants can then be automatically watered via the IoT system. Moreover, the system will also detect the level of sunlight indoor or outdoor to determine if UV lights are needed at that hour, or the users will have the option to overwrite the UV light manually using a toggle switch. Through the use of IoT, several data could also be collected. For example,

a) *Soil Moisture Level:* To detect the moisture level of the soil
b) *Temperature:* To detect the moisture level of the soil
c) *Light Intensity:* To detect the light intensity level of the environment
d) *Humidity:* To detect the humidity around the plant

These data are then visualised through a web dashboard application developed using web languages such as PHP, Javascript, CSS, and more. The dashboard has also planned to adopt charts or an image to represent and visualise all the data collected because the graph is known to present complex or numerous data that is difficult to describe in text into an easier to read and understand format.

Several features could be implemented in this dashboard application, such as customisation. The graph, as mentioned above, could be customised to show the data in different graph types such as pie charts, line charts, doughnut charts and more to match the user preference. Moreover, the user will be able to check what data needs to be shown on the dashboard, which significantly increases the possible ways to personalise their dashboard.

To support the PHP in retrieving and displaying the data, a database server will be hosted on a machine to collect and save all the real-time data sent by the Arduino through MQTT Protocol and Metrics and Event Server Agent. Additionally, several APIs will be developed to retrieve all the information for visualisation. The system architecture is shown in figure 2 below.

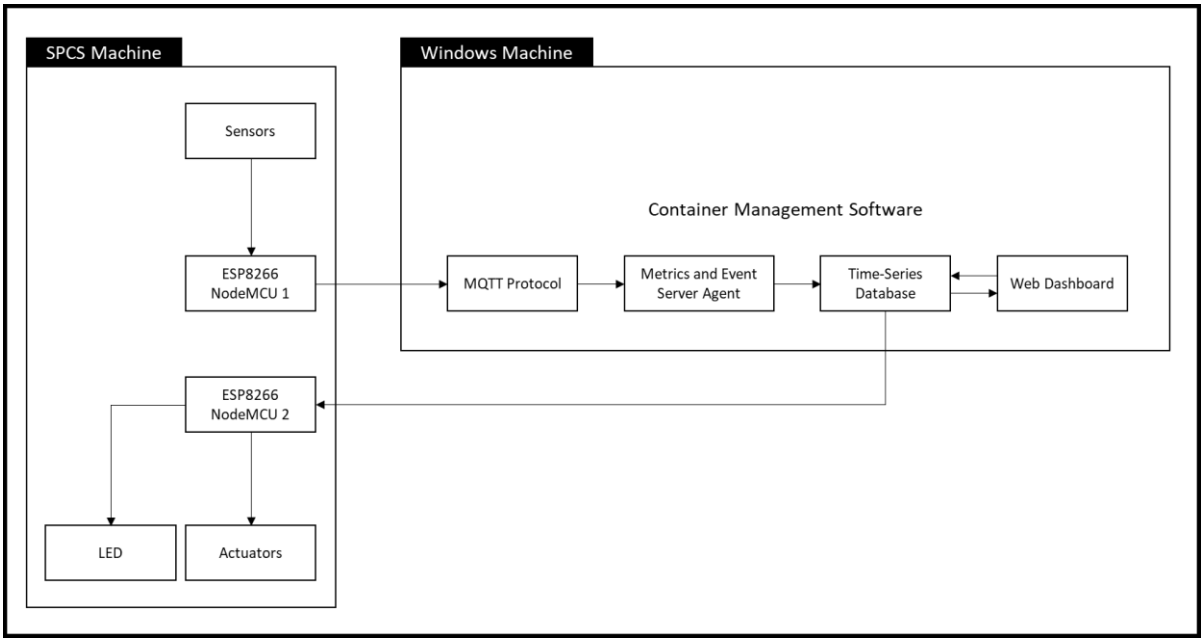



**Figure 2: System Architecture**

Finally, the project will feature a semi-sized prototype of the smart plant caring system's physical model printed utilising a 3D printer, as shown in Figure 3 below, which will hold all the sensors and motors necessary to collect the data and automate the watering system IoT technologies.

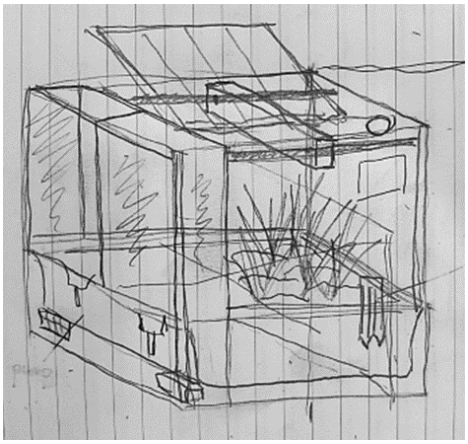


**Figure 3: Sketch of the Smart Plant Caring System**

## 1.5 Subject of the Work

The dissertation will be divided into four separate parts, where each of these sections will cover different topics. The first section of the dissertation will discuss the research and planning phase of the project, where three subchapters will be introduced in this section. The first subchapter in this section is Literature Review, which will cover different aspects of the project, including Urban Gardening Technology, Internet of Things (IOT), Data Visualisation, and Data Security. The second subchapter in the section is Requirement Specification. It will cover the requirements capturing process, the requirement prioritisation method selected, such as MoSCoW, and a brief discussion on the non-functional requirement identified using the Jakob Nielsen's 10 Usability Heuristics. And finally, the third chapter will cover the tools and hardware selected along with the testing methodology and Waterfall system development life cycle adopted in this project.

The second section consists of the synthesis part of the project, where the section will also be separated into three subchapters. The first subchapter of the synthesis section will discuss the system design phase using Kruchten's 4+1 Architectural View Modal as guidance. Several design diagrams will be introduced in this subchapter, such as the use case diagram, sequence diagram, class diagram and more. The following subchapter will introduce the product implementation phase for each identified functional requirement using the selected tools and hardware. And finally, the third subchapter of the synthesis section will introduce the product testing. It will discuss the testing phase of the project using techniques such as Functional Testing, Cross Browser Testing, and User Testing, followed by the testing results and how the identified errors are addressed.

The third section consists of the evaluation part of the project, where the section will be separated into two separated subchapters. The first subchapter of the evaluation section will evaluate the Smart Plant Caring System, discuss the system's strengths and weaknesses, and evaluate the tools used. Moreover, the second subchapter of the evaluation section will evaluate the project process, discussing several topics such as project management, project objectives, skills gained, and the lesson learned.

A conclusion will then be made in the final section of the dissertation, alongside the future recommendation for the project to explore a more comprehensive perspective than the initial proposed project.

# Section 2: Research and Planning

It is essential to study the problem and develop a proper plan before developing it. This section of the dissertation has been divided into three subchapters covering the literature review, requirement specification, and the tools and techniques used.

# Chapter 2: Literature Review

This chapter will discuss the literature review part of the project, covering different aspects of the project, including Urban Gardening Technology, Internet of Things (IoT), Data Visualisation and Data Security:

## 2.1 Urban Gardening Technology

### 2.1.1 What is Urban Gardening?

Urban gardening, also known as urban horticulture and urban agriculture, has been trending in many countries such as the United Kingdom, Singapore, Hong Kong and more for the past few years Tan and Neo (2009); (Tam and Bonebrake, 2016; Dobson et al., 2020). Urban gardening is defined as livestock goods and crop production within urban areas such as cities and towns (Zezza and Tasciotti, 2010). Various unique gardening approaches have been developed through urban gardening, which include (GroCycle, 2021; Lin et al., 2015):

***Rooftop gardening:*** Gardening is established on top of the building's roof, usually done by combining various techniques such as hydroponics, air-dynaponics systems, container gardens or green roof (Sabeh, 2020).

***Allotment or Community gardens:*** Several areas reserved for non-commercial horticulture within the cities contain sub-divided small garden plots that individuals or families will manage.

***Community orchards:*** A landscape that is usually owned by local authorities and is mainly focused on a group of people gathering to plant and cultivate local and exotic varieties of fruit trees (Communities and Local Government, 2011)

***Easement gardens:*** Unused small patches found across a range of neighbourhoods that could be improved and converted into vegetatively different gardens that support biodiversity.

***Private gardens:*** The area of land around the individual households, such as back or side gardens that have been utilised for growing plants.

### 2.1.2 Why should we practice Urban Gardening?

Urban Gardening is crucial to be practised in many countries. In today's world, several countries are heavily populated, and most lands were cleared and reserved for building structures like malls, schools, or multistorey public housing and public highway (Moss, 2021). One of the example countries is Singapore; in the latest news by Moss (2021), it is reported that only 1% of the city-state's land areas are devoted to agriculture. A significant amount of reserved land has since caused plantation problems, resulting in the country being heavily reliant on overseas suppliers. Hence, it has been proven that urban gardening should mostly be practised by people nowadays in urban areas such as houses or rooftops to start growing their food and reduce food insecurity.

An investigation carried out by Camps-Calvet et al. (2016) has shown that urban gardening is vital for a wide range of aspects, such as sustainable food and medicine (Nogeire-McRae et al., 2018); gardening pollinating (Langellotto et al., 2018), climate change reduction (Dubbeling et al., 2019) and pest control (Prezoto et al., 2019). Moreover, urban gardening is also essential in terms of health benefits. A study by Soga et al. (2017) has shown that people who practise urban gardening have a higher rate of delivering various health benefits such as increasing self-esteem, recovery from fatigue and stress, having better social networks, and significantly improving life satisfaction. Furthermore, Soga et al. (2017) have also carried out a questionnaire survey among 332 people in Tokyo to further proven that urban gardening has a severe impact on health; the result of the study has shown that people who took part and practised urban gardening was reported to have better health, compared to people who did not.

## 2.2 Internet of Things (IoT)

### 2.2.1 What is Internet of Things (IoT)?

Internet of Things (IoT) is a topic among researchers (Mahadevaswamy, 2018; Pravin et al., 2018; Santos et al., 2021) that is rapidly gaining interest. Internet of Things (IoT), introduced by Kevin Ashton in the year 1999 (Zhang et al., 2020), is envisioned as a technology that allows a wide variety of devices to communicate with each other to collect and exchange data through embedded actuators, sensors, and software across the network (Tiwary et al., 2018). Different IoT systems have been developed to integrate data. For instance, wearable fitness wristbands track and measure heart rate and the number of steps taken; autonomous cars that detect objects using complex sensors; and smart hospitals that provide full check-ups on patients (Clark, 2016).

According to Clark (2016), IoT solutions have been utilised in both individuals and organisations to collect selected essential information, enabling users to study patterns, make decisions, and avoid potential risks. Furthermore, Kodali et al. (2016) have shown that IoT could also be implemented to control or automate a daily task, such as turning home electrical appliances on and off without any human intervention, resulting in increased efficiency, productivity, and timesaving.

### 2.2.2 Features of IoT

IoT technologies consist of the following essential features:

***Connectivity:*** Connectivity is considered one of the most vital features in IoT, enabling network compatibility and accessibility (Cacovean et al., 2020). An IoT system without communication with other devices over the network will not be appropriate for executing automation tasks or collecting data. Various approaches could be applied to establish connections on IoT systems, including Wi-Fi, Bluetooth, Radio waves and more (Pedamkar, 2021).

***Sensors:*** Sensors act as a vital analytical instrument embedded in an IoT system to read analogue and digital data (Pedamkar, 2021). Various sensors have been utilised on IoT systems to gather data to solve particular problems. For instance, Electrocardiogram (ECG) sensor has been used in a hospital to record the electrical signal of a patient's heart (Neyja et al., 2017); a temperature sensor has been utilised to measure the temperature of the environment (Dagar et al., 2018); and more.

***Scalability:*** Scalability should be kept in mind when designing an IoT system. IoT systems should be able to scale up or down based on the changes in the environment on demand (Gupta et al., 2017). Pedamkar's (2021) example shows that implementing the IoT system in-home task automation could also be scaled up to automate factory tasks.

***Dynamic Changes:*** IoT components' state and quantity should be able to change dynamically based on changes in situations and conditions. For example, temperature and humidity sensors' input may vary based on locations and weather conditions (Pedamkar, 2021). The IoT components should be able to change dynamically and connect to objects within the network without any limitations (Atlam and Wills, 2020).

***Safety:*** Safety is also a vital feature to be enforced within an IoT system. Sensitive information, such as personal data, should be stored confidentially and always protected through reliable security to prevent cyberattacks and data theft from unauthorised personnel (Atlam and Wills, 2020). Security techniques such as Blockchain technology have been implemented in IoT systems to ensure that the data are protected from being altered or removed by hackers (Fakhri and Mutijarsa, 2018).

### 2.2.3 Architectures of IoT

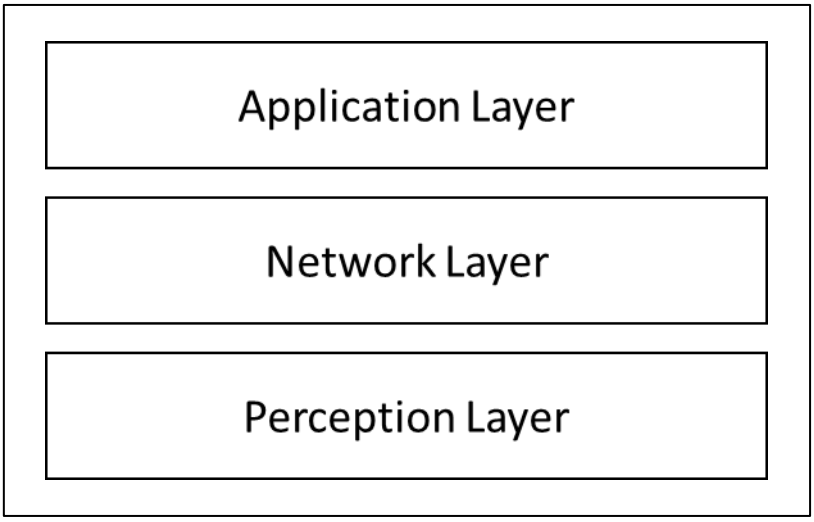



***Figure 4: IoT 3-Layer Architecture***

Lin et al. (2017) have shown that an IoT system consists of these three layers of architecture, as shown in Figure 4:

***Perception Layer:*** The perception Layer, also referred to as the sensor layer (Lin et al., 2017), is the IoT architecture's first layer. Perception Layers can communicate with the devices in the surrounding network and acquire data from the environment through sensors or actuators. The collected specified data is then transmitted to the next layer for further action. (Ghapar et al., 2018).

***Network Layer:*** Network Layer, also referred to as the transmission layer (Lin et al., 2017), serves as the middle layer in the IoT architecture. This layer is considered one of the essential layers as it is being utilised to handle tasks such as processing valuable data information delivered by the perception layer and transmitting the data to the application layer through a specified network such as Wi-Fi, Bluetooth and more (Lombardi et al., 2021; Lin et al., 2017). Furthermore, the network layer has also been utilised to establish the connection between devices and servers (Jabraeil Jamali et al., 2020).

***Application Layer:*** The application layer, also referred to as the business layer (Lin et al., 2017), served as the last layer in the IoT architecture. The application layer handles tasks such as formatting and presenting data information transferred from the network layer (Robinson, 2021) while also providing functionality such as storing data in the database, data analysation using middleware software and more. Moreover, this layer has also been utilised to define various IoT applications, such as smart cities, smart homes, and smart hospitals (Jabraeil Jamali et al., 2020).

### 2.2.4 Advantages and Challenges of IoT

IoT has brought various advantages to individual and business organisations (Tiwary et al., 2018). From the business perspective, IoT will help a company improve customer engagement by continuously capturing and analysing reliable data and trends on customer behaviour. This data information can provide the company with insight that significantly assists them in creating a business strategy to continually enhance customer experience (Borgini, 2021). Furthermore, IoT will also benefit individuals by significantly minimising human effort and saving time. Automation is one of the most outstanding features of IoT; it can be utilised to automatically complete tasks or routines required to

repeat daily, such as watering plants, turning home electrical appliances on or off, or feeding pets without any human intervention (Akram, 2021).

However, conversely, IoT has also brought several challenges to the users. Security and privacy have been two significant challenges and concerns to most users (Akram, 2021). IoT system has the nature to communicate and connect to the network consistently. Hence, IoT systems could potentially be targeted by various kinds of network attacks which increase the risk of data leakage. Moreover, complexity is also one of the challenges of IoT. An IoT system will be more complex and complicated to design or perform maintenance considering the number of modern technologies and the size of the project (Tiwary et al., 2018).

## 2.3 IoT in Urban Gardening

### 2.3.1 Existing System

As previously described in Section 1.1, various urban gardening techniques have been developed by humans to eliminate the challenges of growing food and plants for self-subsistence in urban areas. However, with the growth of the newest technology such as IoT, many IoT integrated urban gardening systems have been developed by researchers worldwide to implement and monitor irrigation systems using various sensors and actuators without human intervention while also providing a dashboard application to visualise all collected data (Sukhdev et al., 2018).

The existing system by Santos et al. (2021) has proven that IoT integrated urban gardening system has brought capabilities and potential to the agribusiness market, significantly assisting large plantations in monitoring environment status and automating daily tasks such as plant watering. The system developed by Santos et al. (2021) has given the user ability to monitor the garden environment condition and remotely activate the irrigation system via a mobile and web dashboard application. The system architecture of the IoT system is as shown in Figure 5 below:

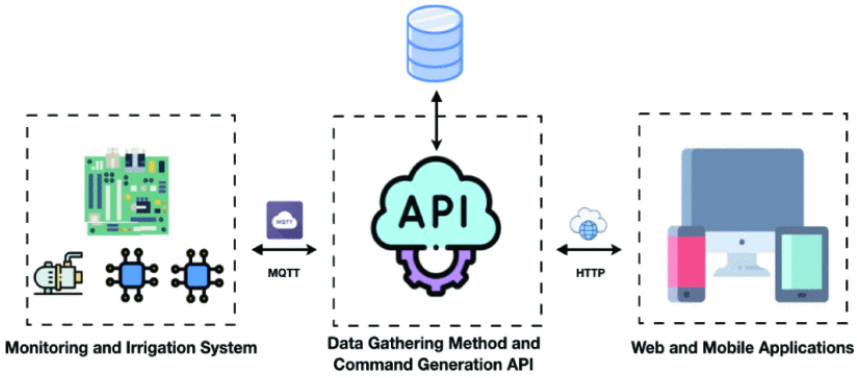



***Figure 5: Smart Gardening Monitoring and Irrigation System Architecture (Santos et al., 2021)***

The IoT system has utilised sensors and actuators, such as a soil moisture sensor (HW080) to gather moisture data of the soil and a temperature and humidity sensor (DHT11) to gather environment temperature and humidity level, connected to an Arduino Uno microcontroller supported with an ESP8266 WiFi module for internet connectivity. On the other hand, the IoT system has utilised both Representational State Transfer Application Programming Interface (API Restful) and MQTT communication protocol technologies to transmit data between the microcontroller and database.

As a result, the web dashboard application developed using a simple Hypertext Transfer Protocol (HTTP) communication interface has successfully displayed the collected sensors data measurements from the database using API. Furthurmore, Santos et al. (2021) mentioned that it is essential for users to gain access to the gathered information because users may efficiently monitor the plant environmental conditions or effortlessly solve specific problems by studying the patterns and trends collected in the data.


***Figure 6: Smart Gardening Monitoring and Irrigation Web Application Interface (Santos et al., 2021)***

## 2.4 Data Visualisation

### 2.4.1 What is Data Visualisation?

Data Visualisation is one of the essential core features of an IoT dashboard. It receives and process data transmitted by the sensors and visualises them on an interactive dashboard that shows valuable trends and patterns in real-time (Prolim, 2021). Data Visualisation is also a solution to support interpreting, visualising, and analysing big data, which is a term for complicated and large data sets collected by various sources such as IoT systems or devices across the Internet over the years (Lowe and Matthee, 2020). The primary functions of data visualisation are to enable users to see the data at a glance instead of the necessity of close reading.

Sensors on IoT systems are usually utilised to collect valuable data such as temperature, distance to another object, accelerometer, humidity, etc. These collected data are then visualised on a dashboard using different data visualisation techniques such as graphs, charts, and geographical maps (Bailey, 2019). An example provided by Bailey (2019) shows data visualisation implemented in an IoT based fish tank's dashboard where a line graph and gauge graph have been utilised to visualise data such as temperature and pH value range, respectively, as shown in Figure 7.

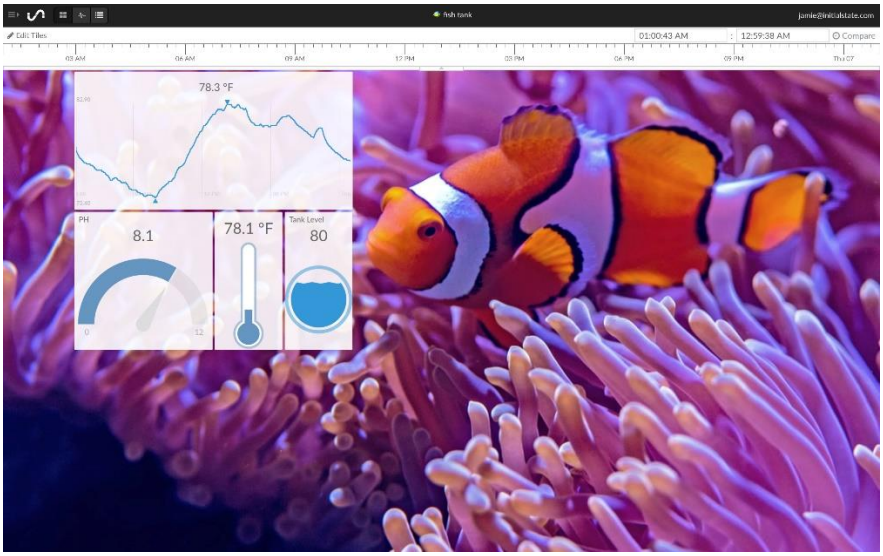



***Figure 7: Data Visualisation Example on Fish Tank Dashboard (Bailey, 2019)***

## 2.4.2 Practising a Good Dashboard Design for Data Visualisation

The dashboard is one of the essential elements of the IoT. It is a data visualisation tool that translates and displays a set of collected data transmitted and managed by the sensors and hardware connected to the network (Bailey, 2019). Dashboards usually display essential information in the form of a single-page view, providing users with a quick way to read and analyse real-time data at a glance (Laubheimer, 2017). For example, a car dashboard can provide single-page view information about their current speed or gas level for the user to rapidly identify if they are speeding or needing to stop at the gas station.

## 2.4.3 Operational Dashboard vs Analytical Dashboard

However, it is also particularly essential to understand the design practices for creating an excellent dashboard to enhance efficiency and provide accurate data information for the users. According to Nielsen Norman Group's Page, Laubheimer (2017), the most common dashboard design consists of two diverse types as listed below:

***Operational Dashboard:*** Operation Dashboard is a widely selected dashboard type that provides data information enabling users to rapidly make and execute effective decisions or predict and foresee potential risks (Yusof et al., 2018). Some dashboard allows users to carry out strictly time-sensitive tasks, for instance, monitoring patient vital status during surgery, flight traffic on an airport runway, and more. Conversely, an operational dashboard also assists users in various simple real-time analytics, such as showing current active users on a site. An example site that utilises an operational dashboard is Google Analytics Real-time Dashboard, a web service developed by Google to provide real-time web traffic analytics (O'Brien et al., 2018), as shown in Figure 8 below.

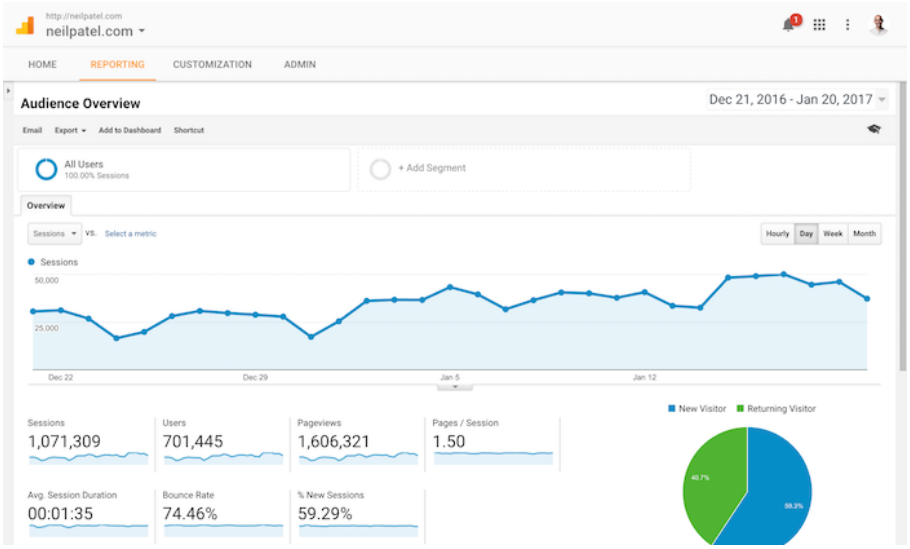



***Figure 8: Google Analytics Real-time Dashboard***

***Analytical Dashboard:*** Analytical Dashboard provides an analytical view compared with previous performance that helps the user identify the need for improvement. Unlike Operational Dashboard, the dashboard will only be required to update non-time sensitivity information once a day to keep users aware of their performance and perform analysis based on the collected data. Researchers have also been experimenting with developing the idea of an analytical dashboard in a school to enhance student's awareness of their performance compared to other classmates (Aljohani et al., 2019). Geckoboard, a company that produces dashboards to assist teams in achieving goals (Geckoboard, 2021), has shown an example of an Analytical Dashboard, as shown in Figure 9 below.

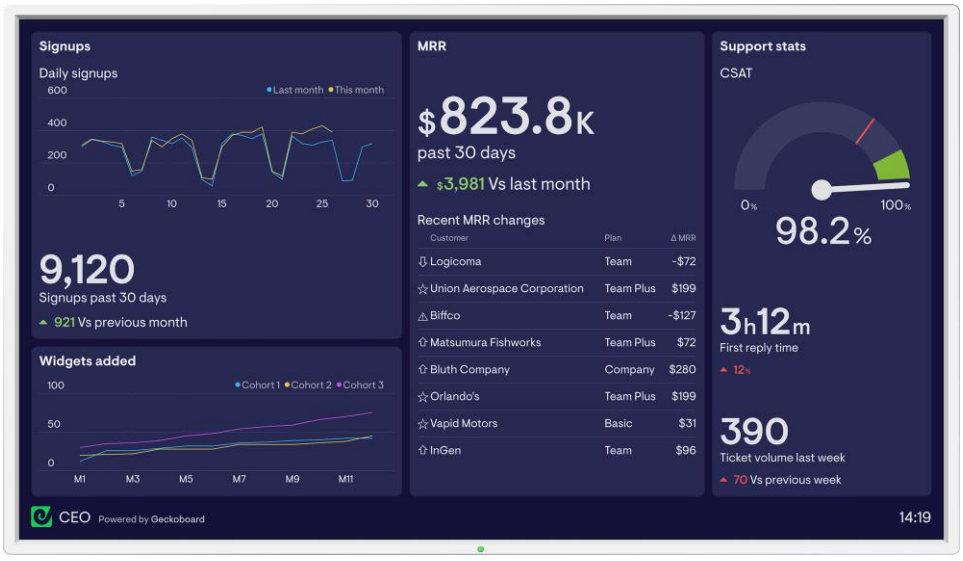



***Figure 9: CEO Dashboard Example (Geckoboard, 2021)***

Upon closely inspecting both dashboard types, the project has decided to investigate the use of an operational dashboard within the system dashboard. The operational dashboard has shown the capability to display collected data in a real-time environment that will significantly assist users in making rapid decisions by analysing and learning the data collected in real-time. Plant data such as moisture level and light intensity are considerably time-sensitive data because they are the main elements that help grow a healthy plant (Tilley, n.d.). Hence, the user could monitor these data in real-time to ensure that the plant has sufficient moisture and light all the time. Conversely, although the

analytical dashboard has served a similar functionality to the operational dashboard, it has shown to be more suitable for handling non-time sensitivity information where the data will be updated once a day to keep users aware of a particular performance.

### 2.4.4 Pre-attentive Processing

Furthermore, Laubheimer (2017) also suggested that it is essential to practice applying pre-attentive processing and quantitative representation in dashboard design. Pre-attentive processing is a human visual perception property that plays a vital role in interactive dashboard designing; it is a visual trick that enables a user to perceive an object in the shortest amount of time without requiring much attention. For instance, the longest line will gain the user's attention without a visual search on an interface that shows multiple lines with different lengths. The visualisation also has various other pre-attentive properties, such as 2D position, angle, shapes, and colours. Figure 10 represents an example of pre-attentive processing.

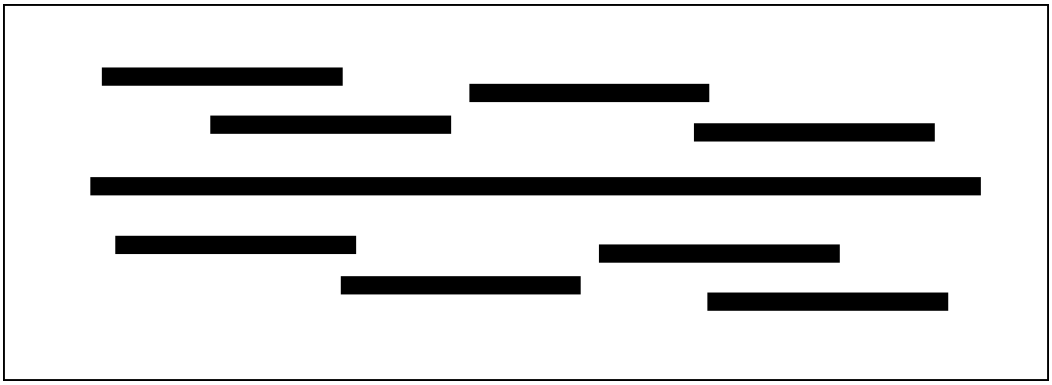


***Figure 10: Pre-attentive Processing Visual Example***

### 2.4.5 Quantitative Representation

Quantitative representation is ideal for a dashboard because it contains two pre-attentive attributes: length and 2D position estimation. Quantitative Representation consists of graphs and charts to represent complex or numerous data that is usually difficult to describe in a text (Slutsky, 2014) into an easy-to-understand format. Linear graphs suggested by Laubheimer (2017), such as a bar chart, are a unique and most reliable way to represent quantitative data on a dashboard. Implementing a bar chart in ascending or descending order allows the dashboard users to quickly identify a value and compare two different values in a single glance (Dragani, 2018), as shown in Figure 11.

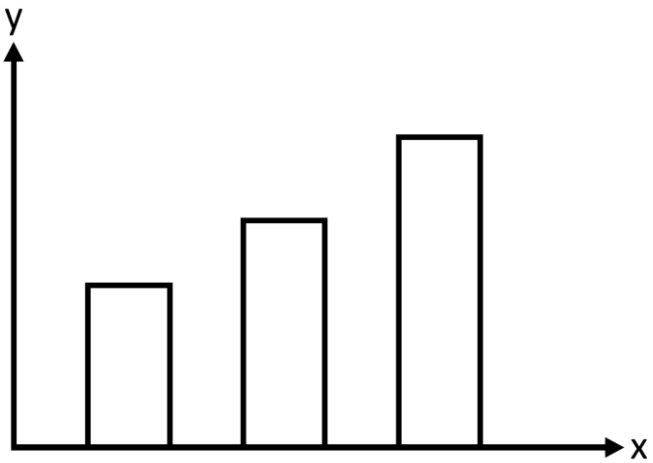



***Figure 11: Bar Chart Example***

Conversely, Laubheimer (2017) also suggests that it is not recommended to implement circular and area-based graphs such as pie charts, radar charts, or treemaps to visualise data. Although they have met the area and angle criteria within the pre-attentive attribute, it is still significantly tricky for a user to interpret values rapidly and accurately. Users cannot instantly compare quantitative data values between each item at a single glance. For instance, a treemap representing data set in a nested blocked layout and varied sizes encoding the data value (Yalçin et al., 2017), as shown in Figure 12, is significantly valuable when visualising a complex data set. However, it is not applicable for representing regular and straightforward data on a dashboard.

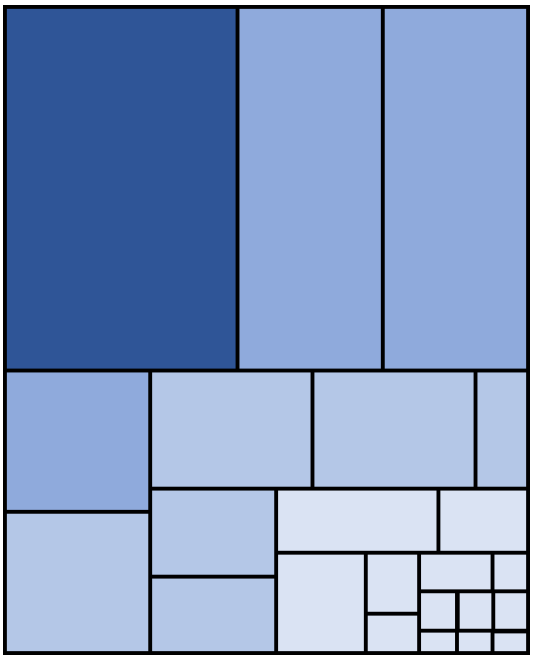


***Figure 12: TreeMap Example***

Like circular and area-based graphs discussed above, Laubheimer (2017) suggested that 3D plotting of a two-variable chart is not recommended as it will also be challenging for a user to interpret and decipher values accurately compared to regular 2D visualisation because 3D graphics will distort alignment and shapes of the visual data. As an example, it is significantly difficult for users to determine the values between the third and fourth quarter in 3D visualisation at a glance compared to 2D, as shown in Figure 13.

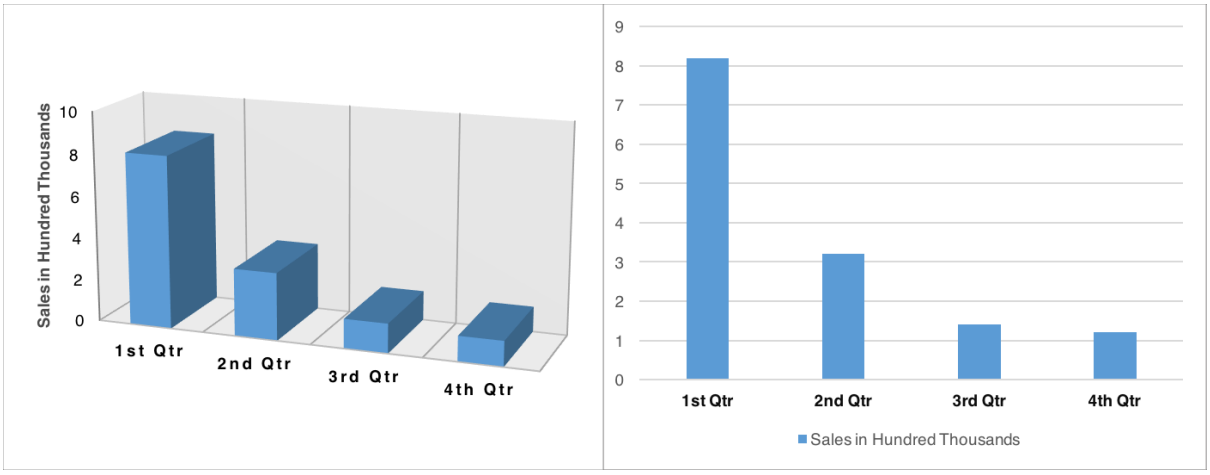



***Figure 13: 2D vs 3D Data Visualisation (Laubheimer, 2017)***

### 2.4.6 The Uses of Colour and Shapes

Finally, Laubheimer (2017) recommended applying colours and shapes to indicate categories in the dashboard. As mentioned above, colour and shape are two essential pre-attentive attributes when designing an interactive dashboard. Colours could be used to categorise status indicators where red indicates a severe problem where actions are required instantly, yellow colour indicates a moderate problem, and green colour indicates normal status where no action is necessary (Janes et al., 2013). Moreover, shapes could be used to categorise two different values and be used as an alternative grouping cue for the users with impaired colour vision (Harley, 2016). According to an experiment carried out by Harley (2016), the implementation of combining unique colours and shapes in a dashboard had significantly assisted in increasing the chance of users finding and identifying the items in the shortest amount of time by approximately 37%, proven that the combination of these two pre-attentive attributes essential to be practice designing a good dashboard.

## 2.5 Data Security

### 2.5.1 What is Data Security?

As mentioned in the previous section, with contemporary trends of adopting IoT technologies, user data security and privacy have been major preoccupation concerns by many users (Barati et al., 2019), and it is generally unavoidable. Most IoT system collects highly sensitive data and is required to transmit data periodically (Dammak et al., 2019). Hence, Data Security is a serious matter to be focused on when developing an IoT system; an unprotected IoT system is more likely to be targeted by network attacks such as Man-in-the-middle (MitM) attacks (Melnick, 2018), giving unauthorised intruders access to collected data such as personal information or sensor data and user home appliances such as door locks or CCTV (Varadharajan and Bansal, 2016). Lack of data security could potentially lead to data being abused by unauthorised intruders, endangering users' privacy and causing physical or financial harm to a user.

Fortunately, researchers have shown that IoT system today have developed several authentication techniques or security countermeasures to support and enhance data security, which includes:

### 2.5.2 Token-Based Authentication

Token-Based Authentication is a type of protocol that has been widely researched in IoT systems (Aman et al., 2018; Dammak et al., 2019) to enable users to authenticate their credentials and obtain a unique token for accessing the content. The token-based authentication is different from server-based and traditional password-based authentication techniques because the token will provide an application with an extra layer of security (Okta, 2022). Various tools could be utilised for implementing token-based authentication, such as the Open Authorisation (OAuth), Client Initiated Backchannel Authentication (CIBA) and JSON Web Token (JWT).

### 2.5.3 Blockchain

Blockchain, a distributed ledger technology, has also been widely investigated in IoT systems (Satamraju, 2020; Lin et al., 2018; Singh et al., 2019) to provide the ability to encrypt the data using SHA-256 hashing and elliptic curve cryptography (ECC) to provide robust data encryption and authentication (Khan and Salah, 2018). The cryptographically linked data blocks are stored on a distributed ledger forming a blockchain (Kshetri, 2017). As a result, data stored will remain unhackable and unalterable, significantly enhancing data security.

### 2.5.4 Device-Based Certification

A device-based certificate is one of the digital certificate types that support secure connections, information exchange and authentication between devices using a solution such as public-key infrastructure (PKI) (Crane, 2021), a set of hardware and software utilised to manage public-key encryption, distribute key and handle digital certificates (Höglund et al., 2020). For instance, an assigned key will be digitally signed to the data via a sensor device. When the device sends the data to the cloud, the digital signature is verified to check the authenticity, completing the device certificates security procedure (Varadharajan and Bansal, 2016).

Upon close review and inspection of various data security techniques, Token-Based Authentication techniques will be investigated in this project to enhance data protection. It uses an access token that acts as an authorisation key to access data and resources. It is easier to implement in various IoT applications than data security techniques like blockchain, which might be complex for smaller projects. Methods such as Blockchain and Device-Based Certification may be the better solution for enhancing security; however, the familiarity and simplicity of developing Token Based Authentication in the project is more suitable within the given time constraints.

# Chapter 3: Requirement Specification

This chapter of the report will discuss the key requirements captured through reviewing similar projects, walking through collecting additional requirements ideas from the user perspective using a survey, and using the requirement prioritisation method like MoSCoW to prioritise functional and non-functional requirements.

## 3.1 Key Requirement Capturing

Before the survey requirement capturing phase could be conducted, it is essential to begin with capturing the key requirement of the system by reviewing similar projects. Referring to the existing system identified in Chapter 2.3.1 of the literature review, the system developed by Santos et al. (2021) uses a sensory module to gather moisture data from the soil and the temperature and humidity level. The data collected by their system is considered essential in terms of plant care. According to Tilley (n.d.), elements such as water, temperature, lights, and more are all equally crucial for growing a healthy plant. As a result, the finding should be clear that such functionality to capture and visualise the data will need to be implemented as a key requirement in the SPCS system.

Additionally, a similar project by Mayuree et al. (2019) has also been reviewed. The system developed by Mayuree et al. (2019) has implemented a smart watering system by combining the data collected from the soil moisture sensor and a water sprinkler. According to Mayuree et al. (2019), the smart watering system is an exceptional solution for solving problems such as overwatering and significantly assisting users with busy schedules who will often forget to water their plants. Hence, a conclusion has been made to investigate the watering automation functionality in the SPCS system.

## 3.2 Survey Requirement Capturing

### 3.2.1 Choosing a Research Method

To further understand the requirements of the project coming from the users' perspective, a short online survey has been conducted to capture accurate requirements of what the users would like to see implemented in the project. Two different research methodologies, which are online surveys and interviews, have been compared and considered for capturing project requirements as shown below:

***Online Survey:*** According to Braun et al. (2021), the survey has been one of the widely selected research methods. It consists of a fixed amount of open-ended questions centred on a specific topic that participants may respond to using their own words rather than selecting from a set of preset options, effectively collecting rich and accurate responses. Howard (2019) has mentioned that online surveys bring a few key advantages to both researcher and participants. For instance, a survey is convenient for participants as they can answer the questions according to their schedule, resulting in a more flexible completion time. Other than that, many open-source platform, and websites, such as Google Form and Microsoft Form, have been made available to every users to create survey which makes online survey a low costing method comparing to other research methods.

***Interview:*** Interview methodology is one of the oldest research methodologies (Hamill, 2014). The interview consists of asking participants a set of prepared questions to obtain responses. An interview could be carried out in several ways, including face-to-face, internet or over the telephone, typically supported by a schedule and interview guide containing topics that will be covered during the interview. According to DeFranzo (2022), conducting face-to-face interviews will significantly benefit accurate screening and avoid answer falsification because participants who are being interviewed are unable to provide false information such as their age. Moreover, Whorton (2016) has also pointed out that researchers will have sufficient talk time with the participant during the interview, which is not achievable via survey. Through the interview session, the participant will have more time to share their thoughts compared to the survey, which considerably helps collect accurate data.

Upon closely inspecting both research methodologies, the project has chosen an online survey as the primary research method, as shown in the complete survey form in Appendix 2. Although interview methodologies such as face-to-face have proven to be more capable of getting accurate results, it is theoretically challenging to accomplish in the given time constraint due to the Covid-19 Pandemic restriction. An online survey has shown to be a better choice of research methodologies in the current situation because it's conducted over the internet. It has been proven to be more time-flexible where participants may respond to the survey when they find it suitable.

### 3.2.2 Choosing Participants using Sampling method

A sampling method has been selected to enhance and accelerate the process of choosing survey participants effectively. According to Whitehead and Whitehead (2020), sampling is a method of choosing a group of suitable participants from a large population to enable the focus of the research topic effectively. It is divided into two different categories: Probability Sampling and Non-Probability Sampling.

***Probability Sampling:*** Probability Sampling is a process typically used in quantitative research. It involves selecting random participants to participate in a survey or interview, significantly enabling researchers to make solid statistical inferences on the chosen group (McCombes, 2019).

***Non-Probability Sampling:*** Non-Probability Sampling is a process typically used in qualitative research. It does not involve participant randomisation but instead is selected based on specific criteria or convenience, significantly accelerating the process of collecting data (Whitehead and Whitehead, 2020).

Given that this project will target conducting qualitative research, two non-probability sampling methods, Convenience Sampling and Judgement (Purposive) Sampling, have been compared and considered for selecting participants.

***Convenience Sampling:*** Convenience Sampling is considered one of the most straightforward and rapid sampling methods because participants are selected depending on their willingness and availability without the requirements of setting specific criteria (Shantikumar and Barratt, 2018).

***Judgement (Purposive) Sampling:*** Judgement (Purposive) Sampling is one of the most applied sampling methods in qualitative research where the researcher gets to decide who to ask to participate in the study. Researchers may choose a set of participants with particular interests or characteristics to gain more accurate data (Shantikumar and Barratt, 2018).

Upon closely inspecting the difference between two of the selected non-probability sampling methods, the project has selected Judgement (Purposive) Sampling as the primary method for choosing participants. Convenience Sampling has been avoided in the project because although it has shown more speed and efficiency in selecting participants, it contains a higher risk of causing sampling bias where the participant may not be in the correct age group or interest, which may result in data inaccuracy. As a result, by applying Judgement Sampling, the project has decided on a set of criteria where the survey participants are required to be over 18, employed or have a busy schedule, and interested in gardening.

### 3.2.3 Ethical Consideration

Due to the survey involving human participants, several ethical consideration has been made to ensure survey participants' data are well protected. These considerations include keeping the survey results and participant consent form in a password-protected hard drive until the end of the individual computing project module. Furthermore, all written information will be provided with a unique ID number, and participants' names will not be written or appear in any reports or documents throughout the project.

Survey participants will be given a copy of the Information Participant Sheet and Consent Form upon accepting the invitation to participate in the survey. They will understand why the study is being carried out and what it will involve through the given document. A copy of the Information Participant Sheet and Consent Form could be found in Appendix 3 and 4.

### 3.2.4 Designing Survey Question

It is essential to design survey questionnaires as it may significantly affect the collected data's accuracy. A poorly designed survey will generally lead to uninterpretable results or inaccurate conclusions (Jenn, 2006). According to Amaresan (2019), a few survey design practices are recommended and should be considered to be followed to develop a quality survey. Those practices are listed below:

***Setting up a good aim for the survey:*** The first practice that needs to be fulfilled before developing the survey design is to propose an objective for the study as it could be used to ensure that the survey can remain on-topic at all times. Providing an aim could potentially assist with visualising a roadmap to designing the study, which will significantly accelerate the question design process.

***Asking closed-ended questions:*** It is recommended that most of the questions be designed towards closed-ended questions using checkbox or multiple-choice format as they have been proven to be effective in collecting quantitative results that can be both easily analysed and responded to. On the other hand, open-ended questions should not be entirely excluded when creating survey questions. Although open-ended questions will take longer to fill out, it is still a very effective method when collecting qualitative data.

***Avoiding leading and biased questions:*** Amaresan (2019) pointed out that it is easy for researchers to include leading and biased questions. For example, a leading question such as "How wonderful was your experience with the product?" has a high chance of causing the survey result to be inaccurate because the leading question will heavily influence survey participants. Alternatively, researchers should ask unbiased questions like "How would you rate your experience with the product?" which will encourage participants to answer the question as truthfully as possible.

***Using response scales:*** The response scale is considerably one of the most common methods to observe a participant's general opinion of a particular question or topic. Referring to DeCastellarnau (2018), there are four types of response scales available which are:

a) *Dichotomous Scale:* Participants are given two response options. The most common use of dichotomous scales is Yes or No and True or False.

b) *Rating Scale:* The rating scale provides survey participants with three or more categorical options. Likert Scale is an excellent example of a rating scale variant where the participants are given 5 points ordinal scale (0 to 5) to rate in response to the given topic.

c) *Branching Scale:* The scale typically used for simplifying the survey participants' task when responding to long bipolar scales.

d) *Closed Quantifiers:* A scale mainly used for objective variables such as the frequency of activities.

Upon closely studying the principles listed by Amaresan (2019), it can conclude that the survey for the project is planned to receive and understand the general feedback on the features that will be implemented into the project from the perspective of a user. Moreover, it is most likely that the question will be designed in both closed and open-ended because the survey will not only ask participants to rate the functionality based on their opinion but instead, the survey will also be asking participants about their ideas on additional features that could be implemented into the project. And finally, a set of Likert scales will be utilised throughout the survey to effectively capture users' opinions on the usefulness of the functionalities.

### 3.2.5 Survey Finding

As mentioned above, the primary aim of the survey was to capture information and understand both the functional and non-functional requirements of the SPC system coming from a user's perspective. Survey participants have successfully provided helpful information on which data should be visualised, thoughts on the product functionality and further suggestion that could be implemented in the system. Participants' survey responses can be found in Appendix 5.

## Data to be Visualised

Several data have been planned to be visualised on the system dashboard, as shown in Chapter 3.1. Participants have been given a Likert scale in the survey, as shown in Appendix 2. Participants must rate the collected sensor data on the 5 points ordinal scale depending on their usefulness (1 = Not Useful, 5 = Very Useful). The participant's responses have been collected and are as shown in the table below:

| | Soil Moisture Level | Environment Temperature | Sunlight | Humidity | Water Pump Status |
|---|---|---|---|---|---|
| Participant 1 | 5 | 4 | 3 | 5 | 4 |
| Participant 2 | 5 | 5 | 5 | 5 | 3 |
| Participant 3 | 5 | 5 | 5 | 5 | 2 |
| Participant 4 | 5 | 5 | 5 | 5 | 1 |
| Participant 5 | 5 | 4 | 5 | 5 | 2 |

***Table 1: Survey results on data to be visualised***

The collected response of the survey has shown that participants of the proposed system survey will be more interested in viewing plant-related data such as soil moisture, temperature, sunlight, and humidity rather than sensors and actuator related data like water pump status. Additionally, Participant 3 stated in one of the survey questions that all the data selected will be helpful in the system, but water pump status will most likely not because it does not directly relate to or help take care of a plant.

The result has suggested that plant caring related data is the essential data to be collected and visualised. It should be listed as the top priority in the proposed system dashboard to provide a better plant caring experience.

## How should data be visualised?

In the survey, the participants have been presented with four visualisations where they will need to select a specific visualisation method they find interested. The participant's responses have been collected and shown in the figure below.

| | Selected Visualising Method |
|---|---|
| Participant 1 | Traffic Light Face |
| Participant 2 | Traffic Light Face |
| Participant 3 | Traffic Light Face |
| Participant 4 | Traffic Light Face |
| Participant 5 | Traffic Light Face |

**Table 2: Survey results on visualisation method**

The collected response of the survey has shown that all the participants are interested in viewing the data in the Traffic Ligh Faces method. Participants 1, 2, 4 and 5 have provided a similar opinion. They

think the traffic light face visualisation method has been chosen because they felt it is significantly easier to read than the other visualising method. Furthermore, Participant 3 mentioned that any user might easily select the traffic light because rather than reading the data by looking at the graph, the user could rely on the image and colour of the displayed face to recognise the status of the plant in a faster manner compared to the other visualisation method.

Hence, the result has suggested that the dashboard should display the collected sensor data using traffic light faces, where an example could be seen in the figure below.

**Figure 14: Example of Traffic Light Faces**

## Additional Functionality

Apart from the watering system automation, which is the core feature of the proposed system. Two (2) other additional functionalities have been planned to be implemented into the system. The planned additional functions will let users enable or disable the watering process and the UV light manually through the product dashboard. However, the same method as the previous section has been applied to investigate whether the functionalities are helpful from the user perspective. Similar to the last section, a Likert scale has been used in this section, where participants must rate the usefulness of these planned functions. The participant's responses have been collected and are as shown in the table below:

| | Enable/Disable Watering Process Manually | Enable/Disable UV Light Manually |
|---|---|---|
| Participant 1 | 5 | 4 |
| Participant 2 | 4 | 2 |
| Participant 3 | 1 | 5 |
| Participant 4 | 3 | 5 |
| Participant 5 | 1 | 5 |

***Table 3: Survey Result on Additional Functionality***

The collected response of the survey has shown that participants have mixed feeling regarding enabling or disabling the water process manually function. Several participants have given their opinion based on their responses. Participants 2, 3, and 5 have a similar view where they think that it is unnecessary to implement another functionality to enable and disable the watering process if it is going to be automated. Participant 3 has also suggested that if the user can control the watering process manually, there is a possibility of causing overwatering, which could be fatal to a plant and a bad practice.

On the other hand, the additional function to enable or disable UV lights manually has received a positive response from the survey participants. Participant 3 stated that it would be an excellent addition to the proposed system because she thinks that users may not want the UV lights to be turned on all the time, even in a dark environment which will significantly help reduce power usage.

In conclusion, the result has suggested that the additional function such as manually enabling and disabling UV light should be placed as the top priority when developing the proposed system, while manually enabling and disabling the watering process function stays at a low priority.

### Participant Suggestions

Survey participants have also been asked to suggest what they want to see implemented in the proposed system to enhance the user experience. The survey has successfully captured and received several excellent suggestions from the participants, and their responses have been analysed and are described as follows.

Participant 1 suggested that users of the proposed system should be able to personalise their plant with custom names, descriptions, and photos.

Participant 2 suggested that the dashboard display the current weather forecast of their current location. He believed that this function might further enhance the caring plant experience where users may utilise that information to change the way they take care of their plants.

Participant 3 suggested that the dashboard should be easy to learn with no further comments.

Participant 4 suggested that the proposed system protect the user's data with no further comments.

Participant 5 suggested that users should be able to view a list of data in raw format upon request, where it will include data such as date and time collected and exact value.

## 3.3 Requirement Prioritisation

### 3.3.1 Choosing a Requirement Prioritisation Method

Requirement Prioritisation is a decision-making task utilised by many developers to understand a requirement's demand and define the implementation priority by considering the time and budget constraints (Bukhsh et al., 2020). It is known that requirement prioritisation should be completed in collaboration with stakeholders such as users or survey participants in this case. Generally, requirement prioritisation is completed by fulfilling two conditions: defining the scope and scheduling implementation. Developers must first define which requirements will need to be implemented into the project and which of the requirements are out of the scope. Then the developers will be required to determine which of the listed requirements are essential to the project so that they can be implemented into the project as early as possible (Schedlbauer, 2011). Additionally, developers can still reference a few criteria or objectives while prioritising requirements. The criteria are as follows (Vargas and IPMA-B, 2010):

***Urgency:*** Urgency effectively determines the urgency or priority level of a requirement. Requirements that are considered urgent will be listed as top priority where it requires to be implemented as soon as possible compared to other not urgent requirements.

***Technical Knowledge:*** It is essential to review the technical knowledge before implementing a requirement into the proposed system. It is well known that it will be easier to implement any requirements if the developers have more technical knowledge readily available, resulting in reducing and minimalising the resources required to implement a requirement.

Fortunately, researchers worldwide have developed several methodologies to accelerate the requirement prioritisation and decision-making process, as seen below. Note that the figure has been redrawn and edited from the source to show a clearer image.

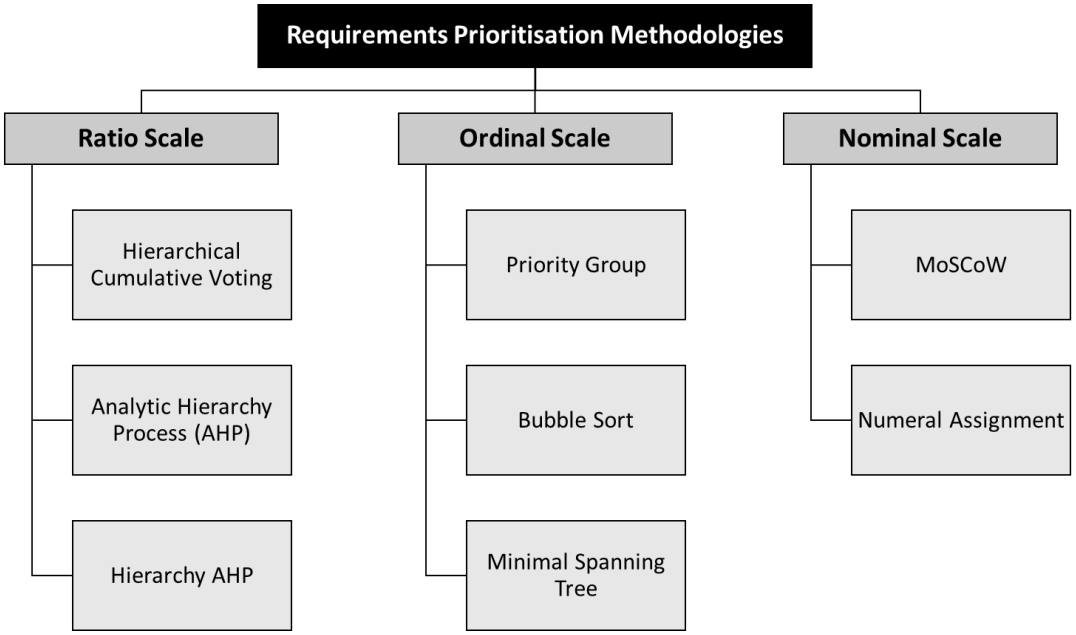



***Figure 15: Requirement Prioritisation Methodologies (Hudaib et al., 2018)***

As a result, the proposed system has selected MoSCoW as the primary methodology in requirement prioritisation due to familiarity with the method, and it is the easiest to implement within the given time constraint (Hudaib et al., 2018).

### 3.3.2 What is MoSCoW?

MoSCoW, which originated from Dynamic Software Development Methodology (DSDM) (Tufail et al., 2019), is a requirement prioritisation methodology that falls under the category of nominal scale, where it means that all the requirements will be required to be grouped into separate categories depending on their importance and priority (Hudaib et al., 2018). MoSCoW is an acronym that stands for the four different categories, “Must”, “Should”, “Could”, and “Would”, combined with two letters ‘o’, which does not stand for anything but only served as a filler to form a pronounceable word (Schillinger, 2021).


***Figure 16: MoSCoW Acronym Graphical Representation***

## Must

The requirements placed into the Must category are considered essential and must be implemented into the proposed system. Failure to implement the requirements in this category will most likely fail a system or the entire project depending on the requirement criticalness (Bukhsh et al., 2020). An example requirement in the proposed system that will be placed in this category will be the requirement to retrieve collected data from the database, which is one of the core features in the proposed system that must be implemented. However, Tufail et al. (2019) and Schillinger (2021) have suggested that developers should not assign more than 60 per cent of the requirement into the Must category because it will most likely affect a project's predictability and success.

## Should

The requirements that have been placed into the Should category are considered critical requirements that should be implemented into the system. The requirements in this category will not directly cause system failure or affect the system's acceptance but are worth being implemented as an addition. These requirements could be implemented if there is sufficient time, or they could also be delivered in a future update. An example requirement will be to visualise some of the collected data on the dashboard, which the user will mainly use for learning or analysing purposes and will not cause any significant issue to the system if it is not implemented. It is recommended that about 20 per cent of the requirements be placed into this category to ensure the project can be completed within the given time constraint (Schillinger, 2021).

## Could

The requirements placed into the Could category are considered not essential to the system, but it will be a nice-to-have feature. However, the requirements in this category are not entirely pointless. Like the Should category, it could be implemented if there are sufficient time and resources (Tufail et al., 2019). These requirements will not affect the system's operation, and they will only increase user experience and acceptance. Hence, it should not be dropped nor ignored from the requirements. An example requirement will be to let the user enable or disable UV lights manually, which will only be implemented mainly to enhance user experience and nothing else that will conflict with the overall system. It is recommended that only 10 to 20 per cent of the requirements be allocated to the Could category.

## Would

The W category can sometimes be referred to as "Would", "Want", or "Won't". The requirements that have been placed into this category should be excluded from the project scope. The requirements in this category are not unimportant, but they will not make an appearance or be implemented in the current project. Developers of the system should be reminded that requirements in this category are not a significant priority in the given time constraint. A requirements example will be the requirement to enable and disable the watering process manually. Referring to the survey result shown in Chapter 3.1.4 Additional Functionality subchapter, the requirement is unnecessary if an existing requirement will automate the process.

# 3.4 Functional Requirement

A functional Requirement is a list of features or requirements that are required to be implemented into a system to enable users to carry out specific tasks. According to Martin (2022), a functional requirement document (FRD) should consist of the following elements:

a) Details of every action or operation conducted in every screen
b) Descriptions of system outputs or reports
c) Descriptions of the system workflow
d) Descriptions of the data to be entered and handled by the system
e) A clear description of who will be able to execute a specific action

Utilising key requirements found in Chapter 3.1, a survey finding discussed in Chapter 3.1.5 and combining the MoSCoW technique discovered in Chapter 3.2.1, a set of functional requirements could be developed with a clear indication of which requirements should be prioritised. The requirements are shown below. Please note that the table below only shows a section of the critical requirements to be discussed. The full functional requirement document can be found in Appendix 6.

| Requirement | Priority |
|---|---|
| System should be able to capture soil moisture level with sensor | Must |
| System should be able to capture room brightness level with sensor | Should |
| System should be able to capture humidity with sensor | Could |
| System should be able to capture temperature with sensor | Could |
| System should be able to turn on UV Light on receiving request | Should |
| System should be able to turn on plant watering process on receiving request | Would |

***Table 4: System Hardware Functional Requirement***

| Requirement | Priority |
|---|---|
| User should be able to register for a new account | Would |
| User should be able to manage (add,edit,remove) their plant | Would |
| User should be able to view the raw data | Should |
| User should be able to turn on UV Light through dashboard | Should |
| User should be able to turn on Watering process through dashboard | Would |
| User should be able to view the sensor captured data through dashboard | Must |
| System should be able to display weather forecast | Should |
| User should be able to customise the plant information | Should |

***Table 5: System Dashboard Functional Requirement***

As seen in the table, it has shown that the system will have two separate components, which are the smart plant caring hardware and the web dashboard. Referring to the system hardware functional requirement, functions such as capturing soil moisture on the system hardware have been placed at a Must priority because the data will not only be viewed by the user but will also be used as a reference data to automate the watering process when the soil is dry. Hence, soil moisture must be captured and implemented into the project. Next, capturing humidity and temperature has been placed into

the Could category because it will not critically affect the overall system but will only be served as a reference to the user on the plant environment. As a result, it could be implemented into the system, but it will be excluded if there are insufficient time and resource. Finally, additional functionality such as turning on UV light and turning on the watering process upon receiving a request has been placed at Should and Would priority. The function of manually activating the watering process has been placed in the Would priority due to the feedback it received from the survey. The user found that the function is not a helpful addition because it could potentially cause overwatering. Hence, it could be safely placed in the Would category since it will not affect the overall system functionality.

On the system dashboard functional requirement, the functionality to view the sensor captured data through the dashboard has been placed under the Must category because it is considered one of the critical components in an IoT project and users may access it to understand the status of their plant and study the pattern to make better decisions. Next, all additional functionality recommended by the user during the survey, such as displaying weather forecasts, customising plant information and more, has been placed in the Should priority because the functions are a great addition to the system but will not affect the workflow of the system if not implemented in the given time constraint. Finally, some of the features, such as registering for a new account, have been placed into the Would priority because the system is only a proof of concept prototype project, and it will not be publicly used by any other users. This has also applied to the plant management function, which was initially planned to let users add new plant and connects to a second plant caring physical modal to visualise the sensor data. Still, since there will only be one custom-built plant caring machine available for the project within the timeframe, there is no point in implementing a function to add, edit, or remove multiple plant caring machines. However, these functions are not entirely unimportant as they will still eventually be implemented if the hardware has been mass-produced or made available to the public in the future. Hence, those functionalities can be safely placed in the Would category.

## 3.5 Non-Functional Requirement

Non-Functional Requirement is another list of features or requirements that need to be implemented. A non-functional requirement is quoted as “a set of requirements that do not concern the main functionality of a system” (Ameller et al., 2019). It only requires system to process system attributes such as security performance, usability and more. Jakob Nielsen’s ten usability heuristic practise has been used as a guide to building the non-functional requirement document.

Jakob Nielsen’s ten usability heuristic, introduced in 1994 (Kaplan, 2021), is a set of rules for enhancing design decisions. These usability heuristic rules are still widely used and applied on various websites and learning interfaces to provide a better user experience. It has also been proven that these heuristic rules are future-proof where they still can be applied in a wide variety of systems, from websites to mobile applications (Gonzalez-Holland et al., 2017). The ten usability heuristics are seen in the following table:

| | Heuristics | Description |
|---|---|---|
| UH1 | Visibility of system status | The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time. |
| UH2 | Match between system and the real world | The design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order. |
| UH3 | User Control and Freedom | Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process. |
| UH4 | Consistency and Standards | Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions. |
| UH5 | Error Prevention | Good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action. |
| UH6 | Recognition rather than recall | Minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design |
| UH7 | Flexibility and efficiency of use | Shortcuts — hidden from novice users — may speed up the interaction for the expert users such that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. |
| UH8 | Aesthetic and Minimalist Design | Interfaces should not contain information which is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. |
| UH9 | Help user recognise, diagnose, and recover from error | Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution. |
| UH10 | Help and documentation | It's best if the system doesn't need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks. |

***Table 6: Jakob Nielsen's 10 Usability Heuristics (Nielsen, 1994; Kaplan, 2021)***

By applying the 10 Usability Heuristics mentioned above and using the MoSCoW prioritisation method, a set of non-functional requirements could be produced with a clear indication of which should be prioritised and which ones could be safely excluded from the development in the given time constraint. The following table consists of the non-functional requirement to be implemented into the system, followed by the usability heuristic (represented by UH) and the prioritisation category it has been placed into. The following table will only contain the usability aspect of the non-functional requirement for evaluation purposes. The further non-functional requirements could be found in the full non-functional requirement document in Appendix 7.

| Requirement | Heuristic(s) Applied | Priority |
|---|---|---|
| User Interface should be consistent across different pages | UH4 | Must |
| Dashboard should handle error appropriately | UH5 | Must |
| Dashboard webpage should be in human readable language | UH2, UH5, UH9 | Must |
| User Interface should keep the design as simple as possible | UH8 | Should |
| Dashboard should have a responsive design | UH7 | Should |
| Dashboard should handle cancellation action appropriately | UH3 | Must |

***Table 7: Non-Functionaly Requirement***

As suggested in Table 7 above, the system usability heuristic, UH4, has been applied to the overall system to ensure that the user interface will stay consistent across each page to assist the users in expecting what the web page will look like and significantly increase the learnability (Kaplan, 2021). Hence, the functionality needs to be placed into the Must priority so that it must be priorly focused on when designing and developing the system dashboard. Next, UH5 has been applied to two requirements: the dashboard should handle the error appropriately and be in human-readable language. It is critical to capture a mistake and let a user understand what has happened to the system and what should be done by the user to solve the problem. The implementation of the functionality is believed to increase the robustness of the overall system. Hence, the functionality should also be prioritised by ensuring that each error will be well-handled throughout the development process.
Furthermore, UH7 has been applied as one of the non-functionality requirements to ensure that the system interface will be compatible across multiple devices to which the design should be responsive. There is a wide selection of devices available today, such as mobile phones and tablets, that could be chosen from to access the system dashboard, and more often, they come in various sizes. Hence, applying such usability heuristics will considerably enhance the system's compatibility and ensure that users will not have to worry about their device will not being supported.

# Chapter 4: Tools and Techniques

This chapter of the dissertation will discuss the tools and hardware selected along with the testing methodologies and development life cycle adopted throughout the project.

## 4.1 Tool Selection

### 4.1.1 Containers vs Virtual Machine

The project has planned to utilise multiple tools for development. As a result, it is ideal to find a solution that could accelerate and enhance the efficiency of the deployment process. Two different tools that could be utilised to handle application deployments, such as Docker Container and Virtual Machine, have been closely compared, as shown in the table below.

| Docker | VirtualBox |
| --- | --- |
|  |  |
| Docker is a platform specialising in assisting developers in building small-scale and lightweight software containers that significantly simplify application deployment, development, and testing. | A Virtual Machine (VM) is a virtual environment that emulates a computer system with a different operating system, such as Linux or macOS, on a physical hosting machine. |
| Docker's container could isolate applications execution under the same OS kernel, effectively eliminating the need to install any unnecessary OS that will cause the system to slow down. | Applications running on a VM typically require an operating system of its own to be operatable, resulting in slow start-up time, large file size, and difficulty to upgrade or maintain. |

**Table 8: Containers vs Virtual Machine (Clancy, 2021)**

Upon close inspection, the project has decided to investigate the use of Docker because it has brought several advantages to the project. As shown in the table above, the Docker can run all deployed applications under a single OS kernal, significantly eliminating the problems of requiring installing multiple OS presented on a VirtualBox, which effectively reduces the possibility of system slowdown. Additionally, Carey (2021) stated that Docker containers provide a solution to build an application and create images that are easy to maintain and move around to any operating system without any limitations, which significantly benefits the project in both development and deployment.

### 4.1.2 IoT Protocol

Data transmission is considered one of the essential processes in an IoT system. It consists of a process to transfer data collected from the sensor module to the server. To ensure that the data could be transmitted successfully, an appropriate reliable, and lightweight protocol should be considered. As a result, two different protocols suitable for IoT use, such as MQTT and Websocket, have been compared, as shown in the table below.

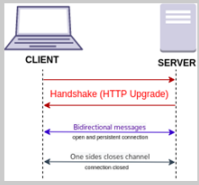
| Mosquitto MQTT | Websocket |
|---|---|
|  |  |
| The MQTT protocol is an OASIS standard messaging protocol widely used in IoT projects designed to publish or subscribe to a standard messaging transport for connecting devices with minimum network bandwidth | WebSocket is a computer communication protocol that establishes a two-way channel to achieve constant data transfer from a server to the client over a TCP connection |
| Multiple system can be able to subscribe and publish messages | Only allows point to point communication |
| Use Qualities of Services (QoS) to achieve reliable message delivery | Does not guarantee reliable message delivery |
| Issues with latency | Low latency |



**Table 9: Mosquitto MQTT vs Websocket (Gregersen, 2021; Pedamkar, n.d.)**

Upon closely inspecting the protocol available, the project has decided to select the MQTT protocol over WebSocket. As suggested in the table above, the MQTT can handle multiple devices to subscribe and publish message simultaneously, which significantly enables multiple microcontrollers to communicate with the protocol simultaneously. Moreover, it has also shown that MQTT utilises the Qualities of Services (QoS) to ensure the reliability of message delivery (Soni and Makwana, 2017). Finally, MQTT has been selected because research conducted by Gregersen (2021) has suggested that the WebSocket could not hold signals stably, causing random disconnection during data transmission, which is not ideal for a high-quality IoT system.

### 4.1.3 Time-series Database

Time series data is a list of observations collected through constant measurements, such as temperatures or moisture levels, over timely order (hourly or seconds) (Dix, 2022). According to Naqvi et al. (2017), there are several contexts in which time-series data could be applied, as shown in the table below.

| Time-Series Context | Description |
|---|---|
| Time Series Analysis | A time series analysis is used to explore how a value of a certain variable change over time. For example, patient health monitoring like electrocardiogram (ECG) is a time series analysis task where a patient's heart activity is captured over time. |
| Regression Analysis | A regression analysis can be used to investigate how a change in a variable can affect other variables simultaneously. For example, Cryptocurrency Market Analysis is a type of regression analysis task where the difference in Bitcoin (BTC) value may affect different cryptocurrency values such as Ethereum (ETH). |
| Time Series Forecasting | Time series forecasting utilises information such as historical values and patterns to predict future events. For example, Weather Forecasting is a type of time series forecasting task where the data values of weather will be used to predict the temperature in the future. |

**Table 10: Different Types of Time-Series Context**

This project requires a robust time-series database tool to handle and store time-series data collected by the sensor parts. Hence, two time-series databases, InfluxDB and OpenTSDB, have been compared, as shown in the table below.

| InfluxDB | OpenTSDB |
|---|---|
|  |  |
| InfluxDB, released by InfluxData, is an open-source database developed for handling time-series or time-stamped data | OpenTSDB is a distributed, scalable Time Series Database (TSDB) written on top of HBase. |
| Has a high write performance | Has a low write performance |
| Widely supported by large amount of programming language | Only supports a small amount of programming language |
| Support User Access Control | Does not support User Access Control |
| High popularity | Low popularity |

**Table 11: InfluxDB vs OpenTSDB (Churilo, 2018; DBEngine, 2022)**

Upon close inspection, the project has decided to choose InfluxDB as the primary database for handling time-series data because, as suggested in the table above, a benchmark carried out by Churilo (2018) has shown that InfluxDB has an excellent write performance compared to OpenTSDB when handling a large amount of data. Moreover, it has also shown that the InfluxDB is widely supported by many programming languages, such as PHP, Go, JavaScript, and more, where many external libraries have already developed to accelerate the integration process. Additionally, it has also shown that the popularity of using InfluxDB time-series databases has seamlessly increased around IoT. As proven by a database management systems popularity ranking website (DBEngine, 2022), the ranking has shown that in January 2022, InfluxDB was ranked the top 1 most used time-series database as shown in Figure 17 below, making it the top priority choice for IoT project today.

☐ include secondary database models — 39 systems in ranking, January 2022

| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| Jan 2022 | Dec 2021 | Jan 2021 | | | Jan 2022 | Dec 2021 | Jan 2021 |
| 1. | 1. | 1. | InfluxDB | Time Series, Multi-model | 30.09 | +1.70 | +3.77 |
| 2. | 2. | 2. | Kdb+ | Time Series, Multi-model | 8.77 | +0.67 | +0.81 |
| 3. | 3. | 3. | Prometheus | Time Series | 6.27 | -0.16 | +0.56 |
| 4. | 4. | 4. | Graphite | Time Series | 5.58 | -0.12 | +0.90 |
| 5. | 5. | ↑ 6. | TimescaleDB | Time Series, Multi-model | 4.22 | -0.16 | +1.29 |
| 6. | 6. | ↑ 7. | Apache Druid | Multi-model | 3.44 | -0.13 | +0.77 |
| 7. | 7. | ↓ 5. | RRDtool | Time Series | 2.08 | -0.19 | -1.11 |
| 8. | 8. | 8. | OpenTSDB | Time Series | 1.86 | -0.15 | -0.49 |
| 9. | ↑ 10. | ↑ 10. | GridDB | Time Series, Multi-model | 1.36 | +0.10 | +0.53 |
| 10. | ↓ 9. | ↓ 9. | Fauna | Multi-model | 1.36 | -0.04 | -0.55 |
| 11. | ↑ 12. | ↑ 12. | DolphinDB | Time Series | 1.24 | +0.13 | +0.50 |
| 12. | ↓ 11. | ↑ 20. | QuestDB | Time Series, Multi-model | 1.14 | +0.03 | +0.77 |
| 13. | 13. | ↑ 14. | Amazon Timestream | Time Series | 1.10 | +0.04 | +0.56 |
| 14. | ↑ 16. | | TDengine | Time Series, Multi-model | 0.69 | +0.04 | |
| 15. | ↓ 14. | ↓ 11. | KairosDB | Time Series | 0.68 | -0.02 | -0.09 |

***Figure 17: DB-Engines Ranking of Time Series DBMS (DBEngine, 2022)***

### 4.1.4 Metrics and Event Server Agent

Since it is impossible to directly pass the data collected by the MQTT protocol into the InfluxDB, a metrics and event server agent is ideal to be utilised to act as the data collecting daemon to transmit the metric data into the database. As a result, two different server agents, Telegraf and Collectd, have been compared, as seen below.

| Telegraf | Collectd |
|---|---|
|  |  |
| Telegraf is a plugin-driven server agent written in Go used to collect and send metrics data from IoT sensors or other services into databases such as MongoDB, MySQL or InfluxDB. | Collectd is a Unix daemon that collects, transfers and stores performance data of computers and network equipment. |
| Over 300 plugins available | Only 175 plugins available |
| Has large active communities | Has large communities but not very active |
| Direct support InfluxDB without plugin | Support InfluxDB through plugin |
| Wide custom metrics support | Less custom metrics support |

**Table 12: Telegraf vs Collectd (Netsil, 2017)**

Upon closely inspecting, the project has decided to choose Telegraf as the primary server agent because it has shown to be directly supported by the InfluxDB without any initial plugin installation, making the integration process effortless. Moreover, Telegraf has offered more than 300 plugins available in the market that the developer could utilise to address various issues. And finally, despite the Telegraf being relatively new compared to Collectd, it still has a large active community to provide support to any developers who require assistance.

### 4.1.5 Backend Programming Languages

Backend programming is considered the backbone of a webpage. It is an information source mainly used to communicate with the server database to retrieve requested data information or to perform specific tasks such as handling user authentication (Stewart, 2021). This project requires a back-end programming language to implement several API endpoints. Hence, two well-known back-end programming languages such as the PHP and Node.js have been compared, as shown in the table below.

| PHP | Node.js |
|---|---|
|  |  |
| PHP is a server-side language, but it could be used as a general-purpose scripting language running JavaScript, HTML, CSS or plain text | Node.js is a server-side JavaScript run-time environment and it could only run JavaScript codes |
| Run synchronously which means the code will required to wait for their turn and will execute only after the function is executed | Run asynchronously which means it executes the entire code without waiting for a function to return |
| Integrates well with database such as SQL and expandable to support other database through external libraries | Heavily relies on JSON as the data format which significantly limited the selection of databases |
| Has larger communities | Has smaller community |

**Table 13: PHP vs NodeJS (Brewster, 2022)**

Upon close inspection, PHP has been chosen for the project because, as suggested from the table above, PHP is a general-purpose web programming language that significantly allows the backend language to create a GUI-based application. Moreover, PHP can integrate with various databases through external libraries, enabling the language to communicate with the time-series database chosen in this project. The table also suggested that PHP is one of the most popular server-sided languages compared to Node.js. In 2018, PHP was still used by over 20,000,000 websites, including Facebook, Wikipedia, Web Hosting Platforms and more (Chris, 2021), making the PHP language much more reliable than Node.js.

### 4.1.6 Frontend Programming Language

Frontend programming is referred to the client-side development of the system. It is defined as the development phase where it focuses on the website's user interface. In this project, a frontend programming language is required to handle the user interface of the web dashboard for the system. Hence, two well-known front-end programming languages such as the ReactJS and JQuery, have been compared, as shown below.

| ReactJS | jQuery |
|---|---|
|  |  |
| ReactJS developed by Facebook is one of the most trending front-end Javascript (JS) component-based libraries used for interactive user interface development. | jQuery developed by The jQuery Team is a "write less, do more" JavaScript library. It simplifies complex JavaScript tasks by wrapping them into straightforward methods. |
| React makes use of a Virtual DOM | jQuery makes use of the traditional DOM |
| Over 5 million dependant repositories on GitHub | Over 500,000 dependant repositories on GitHub |
| Suitable for larger project | Suitable for simple web project |

**Table 14: ReactJS vs jQuery**

Upon close inspection, jQuery has been chosen for the project due to the author's familiarity with the tool. ReactJS has shown more potential in terms of speed due to the nature of virtual DOM. It is more prevalent in today's web development, considering the high amount of dependent repositories available on GitHub. However, it has significantly exceeded the need for this project, given that the project will only be implementing a simple web dashboard project. Finally, jQuery has been selected due to the AJAX support integrated with the library, which significantly smoothens the handling and sending of API requests.

## 4.2 Hardware Selection

### 4.2.1 Hardware Prioritisation

Before getting into the prioritisation step, the project should first identify what hardware is needed to complete building the physical modal. Referring back to the Requirement Capturing in Chapter 3 has significantly assisted the project in accelerating the process of identifying the hardware required to be included. For example, the functional requirement has listed a few features to capture data such as soil moisture, room brightness, humidity, and temperature. For this specific reason, it is clear that the hardware will need to include four different sensors: Soil Moisture Sensor, Light Sensor, Humidity Sensor, and Temperature Sensor. Furthermore, the functional requirement has also discussed that the system should be able to water the plant automatically and turn on and off UV lights. Hence, it is clear that the project will include actuators such as Stepper Motor, Water Pump and UV Lights. Finally, the microcontroller unit is one of the essential pieces of hardware to be included. It will be mainly used as the processor to process all the information and operate the sensors and actuators components listed above.

Utilising the list of hardware identified above, combined with the use of the MoSCoW prioritisation methodology, the project could further analyse which of the following hardware should be prioritised.

In contrast, which hardware could be safely excluded within the given timeframe of this project. The following table will be divided into three different columns. Each will consist of the hardwares to be included in the project, a brief description of how they will be used, and their prioritisation category, respectively.

| Hardware | Usage | Priority |
|---|---|---|
| Soil Moisture Sensor | To capture the moisture level of the soil | Must |
| Light Sensor | To capture the brightness level of the environment | Should |
| Humidity Sensor | To capture the humidity level around the plant | Could |
| Temperature Sensor | To capture the temperature of the environment | Could |
| Stepper Motor | To move the water sprinkler back and forth when watering the plant | Could |
| Water Pump | To pump the water from a container into the sprinkler when watering the plant | Must |
| UV Lights | To provide UV exposure to the plant | Should |
| Microcontroller | To process the information and operates all the sensor and actuators | Must |

***Table 15: Hardware Usage and Prioritisation***

As seen in Table 15, the priority level for some of the hardware listed could be heavily influenced by the functional requirement document. For example, the function to capture humidity and temperature has been placed into the Could priority, which results in the priority of purchasing or implementing the humidity and temperature sensor has been put into the Could category. Additionally, the soil moisture sensor and water pump actuator have been placed into the Must priority. These sensors and actuators are the key components to capture the soil's moisture level and complete the process of automating plant watering. Finally, the stepper motor has been placed into the Could priority as it will only be used to move the water sprinkler back and forth, imitating a printer when watering the plant. However, the hardware is entirely optional, and it will not affect the overall watering process if it is not available during the implementation.

## 4.2.2 Microcontroller

A microcontroller is a type of small computer joint with a circuit chip. It could be found in various devices or systems such as home appliances, robots, vending machines, and embeded systems (Ahmed, 2021). A microcontroller is typically embedded in a device to control a specific function in a device by interpreting the data collected from the microcontroller’s input and output peripherals using its processor (Lutkevich, 2019). A microcontroller plays an essential role in this project. The microcontroller will be the central point to interact with various sensors and upload the data to a database through WiFi communication. There are multiple types of microcontrollers available nowadays that consumers can purchase and use instantly without any hesitation. However, each microcontroller released may come with a different firmware version, and the number of available General Purpose Input/Output (GPIO) slots might vary depending on the microcontroller manufacturer. As a result, it is essential to study and examine multiple microcontrollers to find the exact microcontroller that suits the project's needs. In this project, three different types of microcontrollers have been inspected and compared, as shown in the table below:

| Microcontroller | Arduino UNO WiFi Rev2 | ESP1 WiFi Module | NodeMCU ESP8266 Module |
|---|---|---|---|
| Image |  |  |  |
| Price Range | £22 - £30 (High-cost) | £7 - £8 (Low-cost) | £6 - £7 (Low-cost) |
| Available GPIO(s) Slot | 14 Digital 6 Analog | 3 Digital 0 Analog | 13 Digital 1 Analog |
| Wi-Fi Connectivity | Yes | Yes | Yes |
| Other Description | No extra wiring needed to connect the module. | Extra wiring required to connect the module and firmware are outdated for today uses | No extra wiring needed to connect the module. |

***Table 16: Microcontroller Comparison Table***

Upon closely inspecting and experimenting with these three different types of microcontrollers, the NodeMCU ESP8266 Module has been selected as the main microcontroller for the IoT project. This specific microcontroller has been chosen because of the gap in the price range. The NodeMCU ESP8266 Module is relatively lower cost than the other Wi-Fi enabled modules, such as the Arduino Uno Wi-Fi Rev2, with about £16 difference in price. Additionally, the microcontroller consists of built-in Wi-Fi support. It is already integrated onto the chipboard, making configuring Wi-Fi connectivity straightforward with no extra wiring required, unlike the ESP1 module. And finally, the NodeMCU also consists of 13 digital and one analog programmable GPIO slots, significantly supporting connectivity of one to many sensors at once, which is sufficient for this project.

### 4.2.3 Sensors

A sensor is a device that detects and collects input data from the environment. Each type of sensor can capture different types of data, such as light, motion, and more (Wigmore, 2022). A sensor could typically be classified into two variants of sensors, Analog and Digital sensors. An analog sensor will produce analog output by constantly producing an output voltage or signal corresponding to the measuring quantities, such as temperature and speed (Teja, 2021). On the other hand, digital sensors are a type of electrochemical sensor that will produce data in binary form (0 and 1). However, unlike the analog sensor, the conversion and transmission of the data such as distance and humidity are processed digitally, effectively removing and overcoming the limitation found in the analog sensors (Chakraborty, 2020). Like the microcontroller, every sensor must be examined and compared thoroughly because each sensor has unique specifications. For example, the temperature measurable by a temperature sensor might varies depending on the manufacturer, brand, or type of the sensor. Hence, in this project, all the sensors listed in the hardware prioritisation section in Chapter 4.2.1 have been closely researched and compared between their respective brands by referencing their

datasheet and prices on websites such as Amazon and RS Components, as shown in the following sections.

## Soil Moisture Sensor

There are two different soil moisture sensors, which are the capacitive and resistive soil moisture sensors, as shown in the following table:

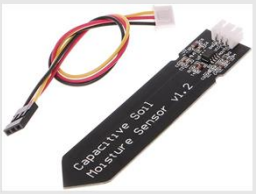
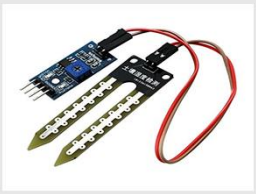
| Sensor | Capacitive Soil Moisture Sensor | Resistive Soil Moisture |
|---|---|---|
| Image |   |  |
| Brand | AZDelivery | AZDelivery |
| Price Range | £2.99 (Low-cost) | £4.49 (High-cost) |
| Voltage | 3.3V – 5V | 3.3V – 5V |
| Durability | High | Low |

***Table 17: Soil Moisture Sensor Comparison Table***

Note that the comparison table did not include a specification row as there is no noticeable difference in the measurable range of these two types of soil moisture sensors. However, the only significant difference between these sensors is the price range and durability. As the table suggested, there is a £1.50 difference in price range, making capacitive soil moisture a more accessible choice to consumers and developers. Additionally, personal experience with the resistive soil moisture has also shown that the sensor has significantly lower durability than the capacitive sensor because the sensor probes on the resistive soil moisture sensor will almost certainly get corrosive over time when interacting with moisture over a certain period. As a result, the capacitive soil moisture sensor will be utilised as the primary sensor to capture soil moisture data due to its higher durability and considerably lower cost.

## Light Sensor

Two different light sensor module, which is the TSL2561 and BH1750, has been compared thoroughly with each other, as shown in the table below:

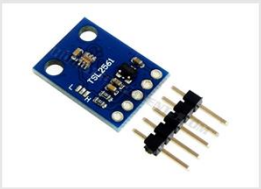
| Sensor | TSL2561 | BH1750/GY-30 |
|---|---|---|
| Image |  |  |
| Brand | Youmile | HALJIA |
| Price Range | £2 (Low-cost) | £3.99 (High-cost) |
| Supported Measurement Type | Lux, Infrared | Lux |
| Analog-to-Digital Conversion | Yes | Yes |

***Table 18: Light Sensor Comparison Table***

The table above suggests that the price range of TSL2561 is slightly lower cost comparing to the BH1750, with about a £1.99 difference in cost. Both light sensor modules support analogue-to-digital conversion, making the sensor convenient to consumers and developers. It will be compatible with outputting data on digital and analogue pins on any microcontroller. TSL2561 supports both Lux and Infrared for measurement, whereas BH1750 only supports Lux measurement. However, although it seems that TSL2561 is a better choice than the BH1750 at first glance, the number of stock available for TSL2561 during the project's timeframe is exceptionally limited due to the popularity of the sensor, making it difficult to purchase for development. As a result, the HALJIA BH1750 has been chosen as the primary sensor to capture light intensity data because it was the best choice that fits most of the use cases on this project.

## Temperature and Humidity Sensor

Three different types of sensors, which are the DHT11, HIH-4030, and LM335Z, have been compared, as shown in the following table:

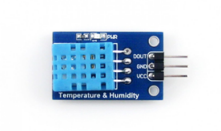
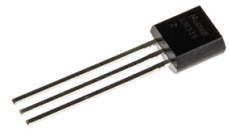
| Sensor | DHT11 | HIH-4030 | LM335Z |
|---|---|---|---|
| Image |  |  |  |
| Brand | AZDelivery | SparkFun | Texas Instrument |
| Price Range | £4.99 | £15.72 | £0.70 |
| Supported Measurement Type | Humidity, Temperature | Humidity | Temperature |
| Measurable Range | Humidity: 20% to 90%<br>Temperature: 0 to 50°C | Humidity: 0% to 100%<br>Temperature: N/A | Humidity: N/A<br>Temperature: -40 to 100°C |

***Table 19: Humidity and Temperature Sensor Comparison Table (Honeywell International Inc., 2008; D-Robotics, 2010; Texas Instrument, 2015)***

Table 19 suggested that DHT11 could be easily chosen as the primary sensor to capture data because it supports the humidity and temperature measurement type compared to the HIH-4030 and LM335Z sensors. Although both the HIH-4030 and LM335Z sensor has the capability of supporting a more comprehensive measurable range with about 10% to 50% difference in terms of temperature and humidity, the DHT11 has still proven to be the best selection as it fits in most of the use case for a reasonable price. It is convenient for most users because it can collect two different types of data in one module. As a result, the DHT11 has been selected in this project to capture both the humidity and temperature data.

### 4.2.4 Actuators

The actuator is a type of hardware that covers incoming signals and energy provided by the system and produces particular action or motion (Roorkee, 2017). An actuator could be in many forms. For example, a stepper motor is considered an electrical actuator because it converts the energy provided by an electric source into motion (Dickson, 2021). Similar to a sensor, it is crucial to compare and review the two actuators listed on the hardware prioritisation as every actuators released by different manufacturers might vary in terms of performance. As a result, in this project, both the actuators listed in the hardware prioritisation section in Chapter 4.2.1 have been closely researched and compared between their respective brands through personal findings and experience, as shown in the following sections.

#### Stepper Motor

Two different stepper motor, which is the 28BYJ-48 5V Motor and the 17HS4023 Stepper Motor, has been closely tested and compared in this project, as seen in the table below:

| Stepper Motor | 28BYJ-48 | 17HS4023 |
|---|---|---|
| Image |  |  |
| Brand | Adafruit | Usongshire |
| Price Range | £5 - £8 (Low-cost) | £11.99 (High-cost) |
| Voltage | 5V | 12V |
| Speed | One speed | Adjustable by limiting the current on the driver |

***Table 20: Stepper Motor Comparison Table***

A similar experiment method has been applied while testing the capability of these two different stepper motors where both rotary parts have been pushed down with a solid object to test how much pressure each stepper motor can withstand. The experiment found that 28BYJ-48 significantly slowed down when pressure was applied to the rotary parts. At the same time, the 17HS4023 continues to rotate without any noticeable slowdown, probably due to the amount of current or voltage it is supplying to power the rotary parts. Moreover, the table has also suggested that although the 28BYJ-48 is lower cost, it is limited to one speed, whereas the 17HS4023 speed could be easily adjustable directly from the motor driver. Hence, upon closely inspecting and experimenting with both stepper motors, this project has concluded using the 17HS4023 as the main stepper motor to move around the water sprinkler.

## Water Pump

Two different water pump, which is the Gikfun Pump and Pssopp Pump, has been closely tested and compared in this project, as seen in the table below:

| Water Pump Brand | Gikfun | Pssopp |
|---|---|---|
| Image |  |  |
| Price Range | £10 (High-cost) | £8.50 (Low-cost) |
| Voltage | 12V | 12V |
| Time taken to complete draining 500ml of water | 9 minutes | 5 minutes |

***Table 21: Stepper Motor Comparison Table***

The experiment has been carried out on both water pumps to assess their speed and force when pumping the water. As suggested in the table, the investigation has found that despite both water pumps taking the same 12V voltage, Gikfun still performed relatively slowly when attempting to pump 500ml of water compared to the Pssopp brand water pump. This experiment has shown that the Gikfun brand water pump will most likely take longer to complete the plant watering process on the system. Hence the Pssopp brand water pump has been chosen for this project.

## 4.3 Testing Methodology

### 4.3.1 Functionality Testing

Testing takes an essential role in most of the development cycle. It consists of several steps to verify and validate if the software has been implemented appropriately by ensuring that all the implemented has met the user requirements and could be executed without errors (Anand and Uddin, 2019). The testing will be divided into three phases. The first testing phase will consist of Functionality Testing carried out at the end of the developing stage by the developer, where the test is carried out to validate the system against the functional requirements (Hamilton, 2022). When conducting a functionality test, a test case will typically be constructed with pre-selected test inputs to be tested along with a description of expected outcomes. As a result, the use case testing technique has been applied to the functionality testing phase to accelerate the process of identifying test cases through the use cases documentation that will be created later in Chapter 5.2.

### 4.3.2 Cross Browser Testing

The next phase will consist of Cross Browser Testing, which will also be carried out at the end of the development stage in targeting to determine if the system will work as intended across different browsers such as Safari, Google Chrome, and Firefox (SuccessiveTech, 2021). Cross-browser testing must be conducted, primarily when the system will most likely be used by various devices running on different browsers. According to SuccessiveTech (2021), each browser will handle information differently, resulting in some of the browsers not displaying some of the features as expected. The same set of test cases used for the functionality test will be reused to ensure that every functionality requirement works across different browsers when conducting cross-browser testing.

### 4.3.3 User Testing

The last phase will consist of User Testing carried out at the end of the development stage by a group of users. User Testing is carried out to test the system functionalities and user interface to ensure the system has reached their expectation and usability (Hotjar, 2022). When conducting user testing in this project, the user will be given the same test cases used in the Functionality Testing to serve as a guide to test through the system. At the end of the testing, users will be given a survey to fill out their opinion on the system as seen in Appendix 8.

## 4.4 Development Life Cycle

A software development life cycle (SDLC) is a process usually adopted by software developers to develop high-quality software. The SDLC serves as a guideline for planning each software development phase (Mbaabu, 2021). A basic SDLC module usually consists of five stages: Analysis and Requirement, Software Designing, Implementation, System Testing and Deployment and Maintenance (Ihsan and Kadir, 2018). In this project, the two most well-known development life cycle methodologies which are the Agile Model and Waterfall Model, have been compared and considered by reference to the comparison table by Ankers (n.d.), as shown in the table below:

| AGILE | WATERFALL |
|---|---|
| Continuous cycle, which enables either incremental or iterative delivery | Delivered in a sequential or linear path with development and minimal testing implemented |
| Flexible | Structured and rigid process |
| Incorporates regular inspection and adaption to produce the desired end-product | Testing is conducted once the product has been built |
| Accommodates project requirement changes | No changes to the scope or the project requirements can be made after the project has started |
| Stakeholder involvement and communication is increased throughout the development | Stakeholders are engaged during the initial stages of the project requirement phase |
| Minimal project documentation to be completed | Requires comprehensive project documentation to be completed to understand the requirements |
| Cost-effective to address change as development of the product is being built | Costly to make changes to the final product, if it does not meet expectations |
| Slower delivery due to the number of changes implemented | Faster delivery |
| Suitable for projects on a much larger scale | Suitable for small-scale project delivery |

***Table 22: Agile vs Waterfall Comparison (Ankers, n.d.)***

According to Hoory and Bottorff (2022), the main difference between these two methodologies is that Agile assists a project in working on tasks simultaneously in each phase. The waterfall is a linear system where every task and documentation must be completed before moving into the next step. As suggested in Table 11, agile is significantly suitable in projects that are more extensive scale, and project requirements could be subject to changes throughout the development cycle, making the agile method more flexible, allowing the project to experiment with a different path on development. In contrast, the waterfall is more appropriate for a small-scale project, and no changes should be made after the development has started.

Upon closely inspecting both the Agile and Waterfall methodologies, a decision has been made to adopt the waterfall method as the primary development life cycle for this project. Although agile has shown more potential in terms of flexibility throughout the whole life cycle, it has significantly exceeded the need for this project, given that the project has aimed to collect all the requirements at the beginning of the cycle and will only engage with their stakeholders during the requirement capturing phase and after the development phase. Hence, this has shown that the project is more suitable for using a development life cycle such as Waterfall.

# Section 3: Synthesis

Using the knowledge gained from literature review and the requirement captured, a robust prototype system was developed using the tools and techniques selected to resolve the identified problem. This section of the dissertation has been divided into a few subchapters, covering the system design, product implementation and product testing.

# Chapter 5: System Design

This chapter of the dissertation will look at how Kruchten's 4+1 architectural view model has been applied to the system design phase using several UML diagrams and database design, followed by the design of the user interface using wireframes.

## 5.1 4+1 Architectural View Model

The project has utilised the 4+1 architectural view model (Kruchten, 1995) as a reference to guide through the system design phase. According to Jayawardene (2021), the 4+1 view model is usually applied to design to describe software architecture from four different views: Logical, Process, Development and Physical. Additionally, diagrams such as the use cases or scenarios are generally used as the "plus one" view to showcase the design from an end-user point of view. The descriptions for each view are as shown in the following table:

| Views | Description |
|---|---|
| Logical | The logical view essentially supports the functional requirements from the end-users' perspective. UML diagram such as the **class diagram** could be applied into the logical view to display a set of classes and their relationship with each other. |
| Process | The process view focus on the system behaviours and non-functional aspects of the system such as the performance and usability. UML diagram such as the **sequence diagram** is applied into the process view to describe the behaviour of a system. |
| Development | The development view focus on the organisation of the software module in the software development environment. Element such as the **wireframe** could be used in this process to define the user interface design of the system. |
| Physical | The physical view primarily focus on the software components also visualise the physical connections between each components. UML diagram such as the **Entity Relationship Diagram** could be applied in the physical form to describe each components relationship and how each components will be communicating with each other. |
| Scenario | The scenario "plus one" view focus on describing the relationship of all four views under each scenarios. Element such as the **Use Case Diagram** could be applied in this view to visualise how each view interact with each other. |

***Table 23: 4+1 View Model Description***

The 4+1 architectural view model has shown significant potential in the Waterfall development life cycle. Due to the nature of each waterfall phase that has to be completed before moving on to the next phase, the 4+1 model could be referred to as an ideal guideline to give the developers an idea of what should be prepared and designed during the system designing phase, accelerating the designing process. As a result, this project has developed several diagrams, as shown in Appendix 9 to 15.

## 5.2 Use Case Diagram and Descriptions

A use case diagram is a Unified Modelling Language (UML) diagram demonstrating how users could interact with the system (Gaskin, 2021). A use case diagram will consist of components such as the Actors, Functionalities in the form of use cases, and the actors' relationship with the use cases. In this project, two separate sets of use cases diagram have been developed using the requirements captured in Chapter 3 to illustrate a user's interaction with the SPCS dashboard and a microcontroller with the webserver. The example below shows the use case diagram of the user. Both use case diagrams can be found in Appendix 9.

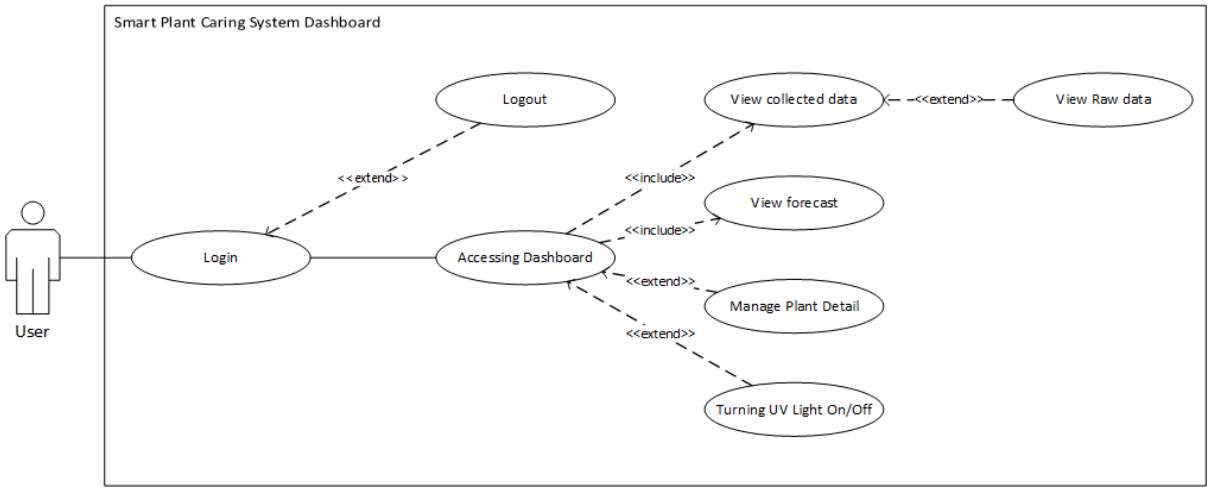



***Figure 18: Use Case Diagram between User and SPCS Dashboard***

As suggested in the use case above, the diagram describes how the User will interact with the SPCS Dashboard, including all the possible interactions of the system. In the figure above, a user has been set as the primary actor who can access the plant dashboard by first logging into the system. Within the plant dashboard, the User will be presented with every data that has been collected by the sensors and view the forecast retrieved from a weather API as requested by participant 2. And finally, there will be three optional use cases where the user will be given a choice to view the raw data as requested by participant 5, toggle UV lights or manage the plant detail as requested by participant 1.

After that, several use case descriptions have also been written to briefly explain the basic flow on how the users will trigger each use case and their respective alternative paths and preconditions, as shown in one of the example use cases below. The entire use case description document can be found in Appendix 10.

| | |
|---|---|
| **Use Case** | Accessing the dashboard |
| **Use Case Description** | User may access the dashboard to view all the collected sensor data uploaded by the microcontroller including soil moisture, light intensity, water sprinkler status and UV light status. |
| **Actor** | User |
| **Precondition** | The user must be logged in |
| **Basic Flows** | • The user decided to look at the status of the plant.<br>• The user clicked on the Plant Dashboard button shown on the panel.<br>• The system verifies if an authenticated token is presented.<br>• The user is directed into the dashboard page<br>• The user views the collected data displayed on the dashboard. |
| **Alternative Path** | **1. The system fails to find an authenticated token**<br>The user is redirected into an error page with error 401.<br><br>**2. The system detects that the token has expired**<br>The user is redirected into an error page with error 401.<br><br>**3. The system fails to retrieve the dashboard page**<br>The user is redirected into an error page with error 500. |

***Figure 19: Example Use Case Description***

## 5.3 Sequence Diagram

Following the use case diagram, a series of sequence diagrams, as seen in Appendix 11, has been constructed to illustrate the interaction between each system object in sequential order. A sequence diagram consists of a combination of components such as the lifeline represented by a vertical box and horizontal arrows to show the when messages will be sent between objects (Panigrahi et al., 2018). It is relatively challenging to create a sequence diagram as developers must consider how each component, such as the webpage, database, and hardware, will have to work together to achieve specific functionality.

The figure below shows one of the example sequence diagrams on how the microcontroller will interact with other components in aiming to automate the plant watering process:

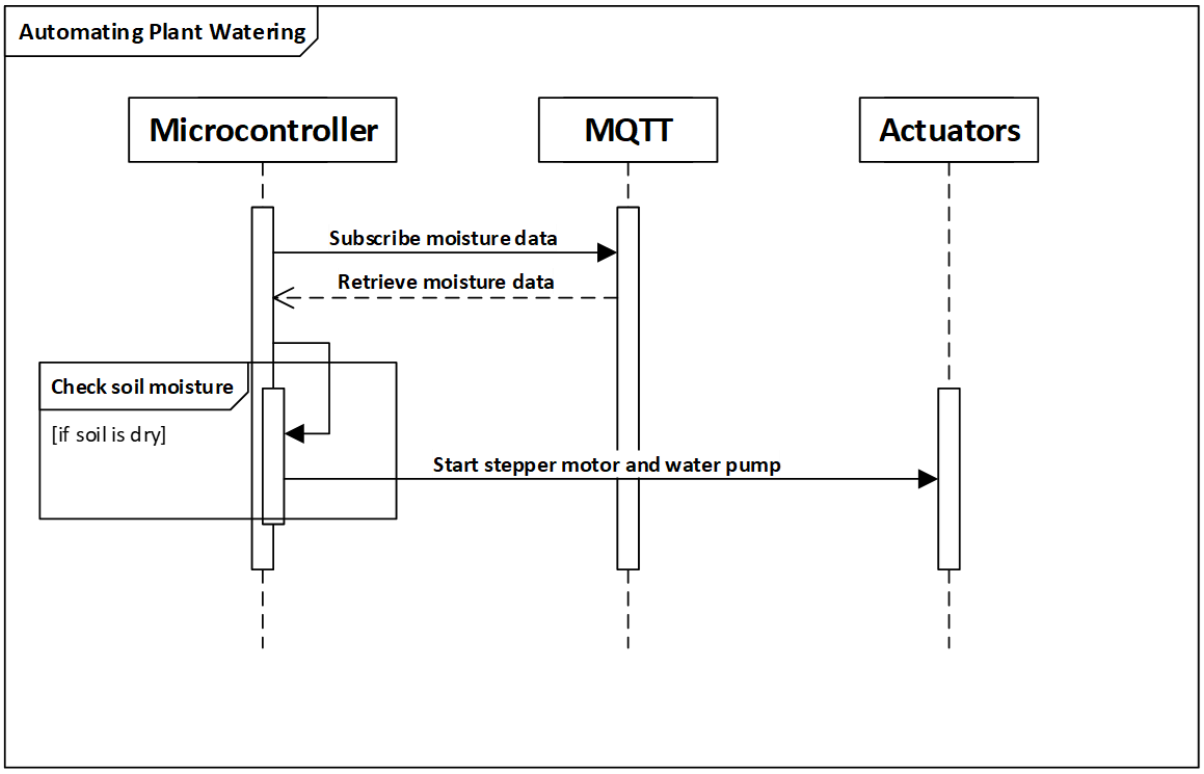



***Figure 20: Sequence Diagram – Automating Plant Watering***

The figure above shows a brief sequence of how the plant watering will be automated. To begin automating the process, it is essential to keep track of the soil's moisture level. In the beginning, the microcontroller will be subscribing to the moisture data on the MQTT broker to receive the newest value collected by the moisture sensor. The microcontroller will then check if the retrieved value has exceeded a specific number, indicating that the soil is now dried. If the soil moisture returns dry, it will send a message to the actuators connected to start the stepper motors and water pump. Conversely, if the soil moisture returns wet, it will repeat the whole sequence until it has been detected that the soil is dried again.

## 5.4 Class Diagram

The project has also created a UML class diagram as seen in Appendix 12 to illustrate the relationship between each entity, showing a clear image of how the system is structured. A rectangle block represents typically a class in the diagram with three separate sections where the first section describes the name of the class, the attributes of the class in the second section and the methods of the class in the final section (Bell, 2003), as shown in the figure below:

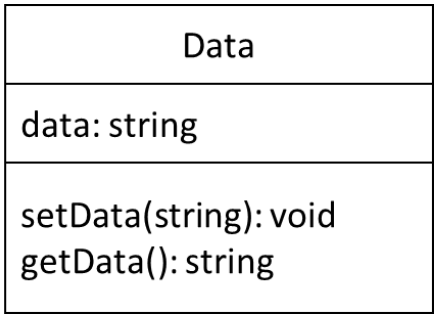



***Figure 21: Example Class***

The SOLID principle (Martin, 2000) consists of five different principles that serve as a guide when designing a class diagram which are the Single Responsibility Principles (SRP), Open-Closed Principle (OCP), Liskov Substitution Principle (LSP), Interface Segregation Principle (ISP) and the Dependency Inversion Principle (DIP) to ensure that the project can be easily maintainable and extendable in the future. The description for each principle is as shown below:

| Principles | Description |
|---|---|
| Single Responsibility Principle (SRP) | A class should only have one responsibility and only one reason to be changed |
| Open-Closed Principle (OCP) | A class should be open for extension but closed for modifications |
| Liskov Substitution Principle (LSP) | A class should be able to use any derived class instead of a base class without any modifications |
| Interface Segregation Principle (ISP) | A class should not be forced to depend upon the interfaces that they do not use |
| Dependency Inversion Principle (DIP) | A class must depend on abstractions, not on concretions |

***Table 20: SOLID Principle Description***

The first three of the SOLID principle have been applied in this project when developing the class diagram. The first object-oriented design principle used in the project class diagram is the Single Responsibility Principle (SRP). The SRP ensures that all the classes should have one and only one responsibility, as shown in the figure below:

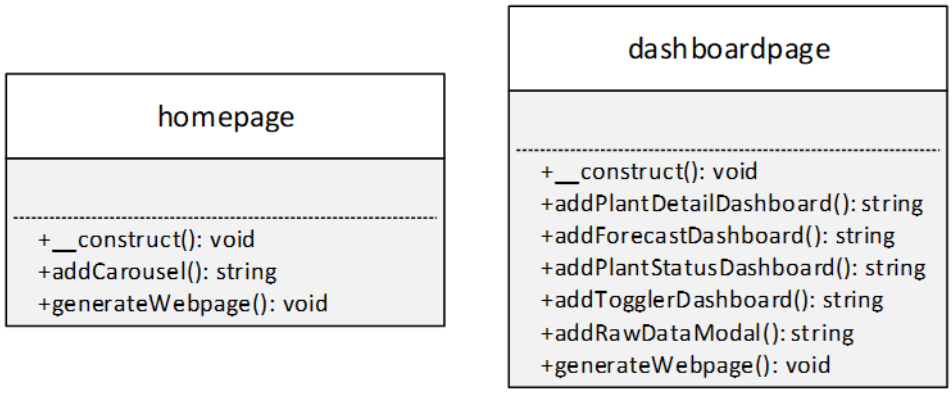



***Figure 22: Example of SRP in SPCS Class Diagram***

As the figure above suggests, the Homepage class and DashboardPage class are specifically designed to only handle generating and adding web page elements to their respective page without any other methods that are out of their responsibility. The SRP is considerably a good practice because it makes the implementation of the system more straightforward while preventing unexpected side effects in future changes. Additionally, the design practice will make the code easier to understand, which will significantly reduce potential bugs and development speed (Janssen, 2020).

The following principle practised when designing the class diagram for this project is the Open-Closed Principle (OCP). As the description suggested, the OCP ensures that the class entities should be designed in a way to open for an extension but closed for modification, as shown in the example below:

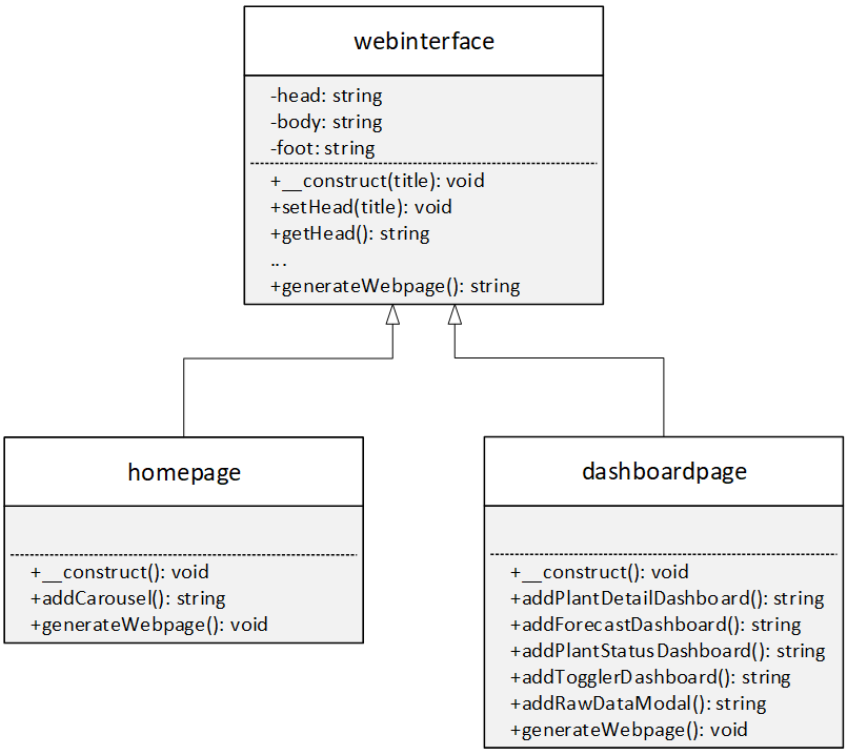



***Figure 23: Example of OCP in SPCS Class Diagram***

The figure above shows that the class diagram is designed so that both the Homepage and DashboardPage class can override the generate webpage method in the WebInterface class without modifying the source. The OCP is a good practice to be kept in mind when designing a class diagram as it will benefit a class to be maintainable without having the potential risk of breaking the existing codes (Vathanakamsang, 2017).

And finally, the Liskov Substitution Principle (LSP) is the third principle on the list where it ensures that the subclasses should be substitutable for their base classes. Although the LSP has not been directly applied to the class diagram for this project yet, it still has shown that the WebInterface class has been designed to make sure that the original code will not be affected if any class is added to the WebInterface class as shown in the example below:

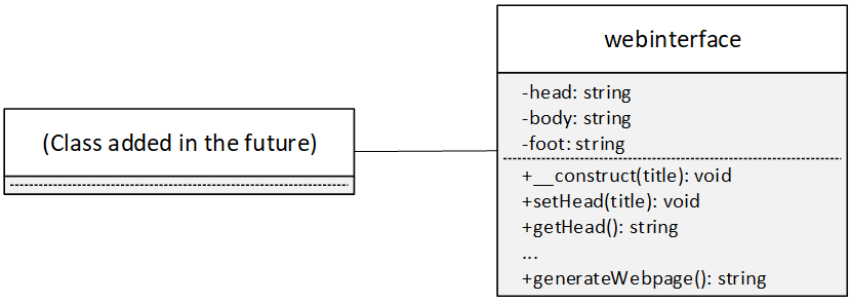



***Figure 24: Example of LSP in SPCS Class Diagram***

For clarification, the last two practices, such as the Interface Segregation Principle (ISP) and Dependency Inversion Principle (DIP), will not be achievable in the class diagram for this project due to the lack of utilising abstract and interface class. However, both of these practices are not irrelevant, as they will need to be revisited if a new abstract or interface class were to be added to the system in the future.

## 5.5 Database Design

Referring to tools selected in Chapter 4.1.4, the project has chosen the InfluxDB as the primary database to store data collected by the sensors. Database design such as the Entity Relationship Diagram is not applicable for InfluxDB given that the selected database is considered as a Time-Series Database, unlike the MySQL, which is a type of Relational Database. A relational database will use a primary key as an index to query and sort data. In contrast, the time-series database uses timestamps (David, 2020) which indicates that there is no possible way to draw a relation between each data in the time-series database, given that each index on the time series data will be different when a data is added.

However, the project will still use MySQL as an alternative database to store data such as user credentials and plant information. Those data are not time series related and should not be stored within a database such as the InfluxDB. Hence, the project has developed an Entity Relationship Diagram (ERD) as seen in Appendix 13 to illustrate the relationship between two tables which are the User Account table and the Plant Information table, as shown in the figure below:

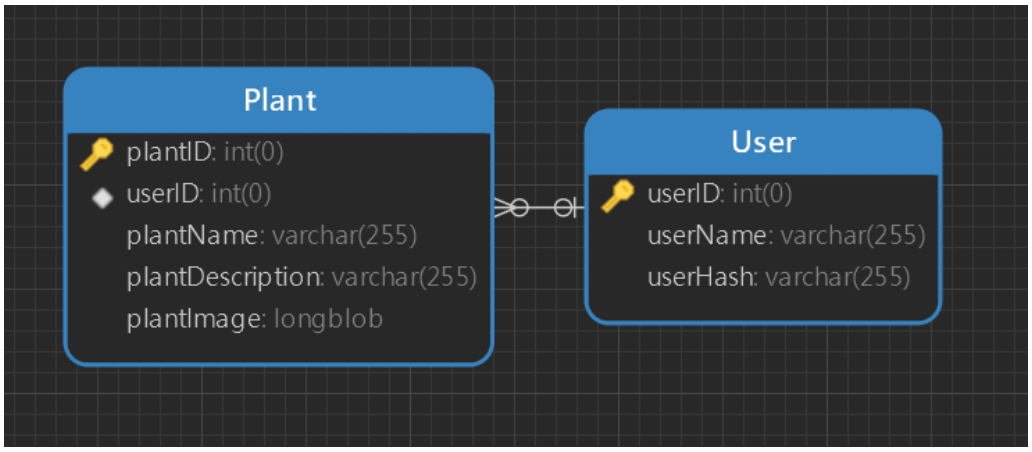



***Figure 25: Entity Relationship Diagram***

As suggested by the figure above, the User table will handle user information and credentials using data columns such as user ID, name, and hash. The UserID in the User table has been specified as the table's primary key to classify each row of data uniquely. Additionally, the Plant table has been designed to handle plant information using data columns such as plantID, plantName, plantDescription, and plantImage. However, different from the User table, the Plant table has been specified with a user ID to represent the plant owner. It will be using user ID from the User table as a foreign key to create a relation between these two tables.

## 5.6 User Interface Design

Finally, the system has developed a series of user interface designs to illustrate how will a system be designed. According to Ramón et al. (2013), wireframing is an excellent method for creating an

application's user interface. A wireframe could be divided into two parts, Low and High Fidelity, which both the documents can be found in Appendix 14 and 15.

### 5.6.1 Low Fidelity

The user interface design phase begins by creating the low fidelity interface for the system. An example of the dashboard low fidelity design has been shown in the figure below. The complete low fidelity design can be found in Appendix 14.

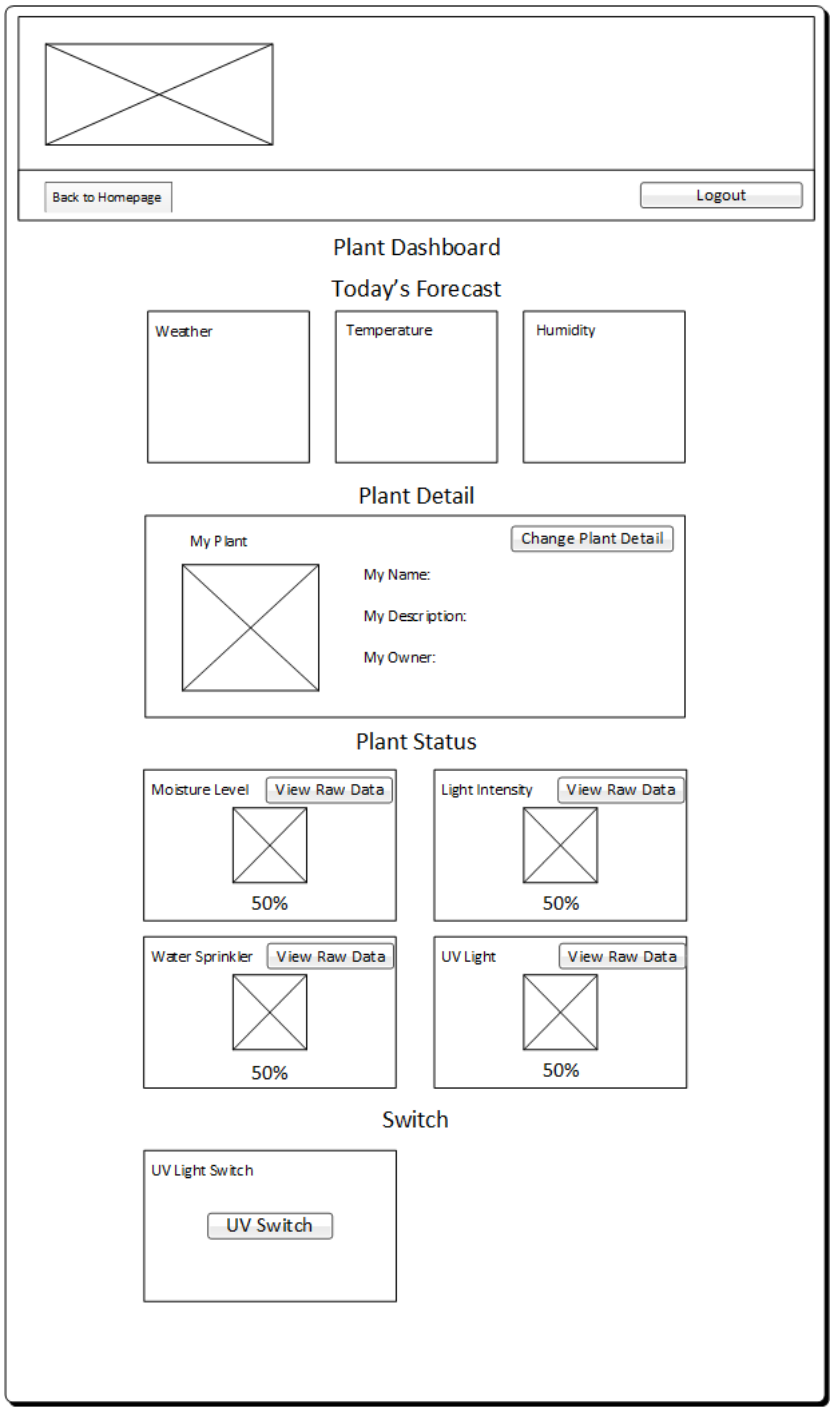



***Figure 26: Low Fidelity Dashboard Interface Design***

Several elements have been applied when creating the dashboard interface design. The first element that has been applied is the use of Pre-attentive Processing suggested by Laubheimer (2017), covered in Chapter 2.4.4. The pre-attentive processing consists of a visual trick which enables a user to perceive an object in the shortest amount of time without requiring much attention. As seen in the figure above, the dashboard interface has applied a similar pre-attentive processing technique as the example shown in Chapter 2.4.4. The dashboard has been divided into three separate categories, with each component designed to display in different lengths. The use of pre-attentive processing is believed to significantly enhance user experience because it has shown exceptional potential in assisting users to find what they are looking for in the shortest amount of time without conducting a long visual search.

Additionally, several principles from Jakob Nielson's 10 Usability Heuristics have also been kept in mind when designing the low fidelity design for the system. According to the fourth and sixth principles in usability heuristics, it has stated that each interface of a system should remain consistent and minimise the user's memory load (Nielsen, 1994). Hence, the user interface of the system has been designed to ensure that the majority of the elements will remain the same across each web page to accommodate these two usability heuristics. An example can be seen in the figure below.

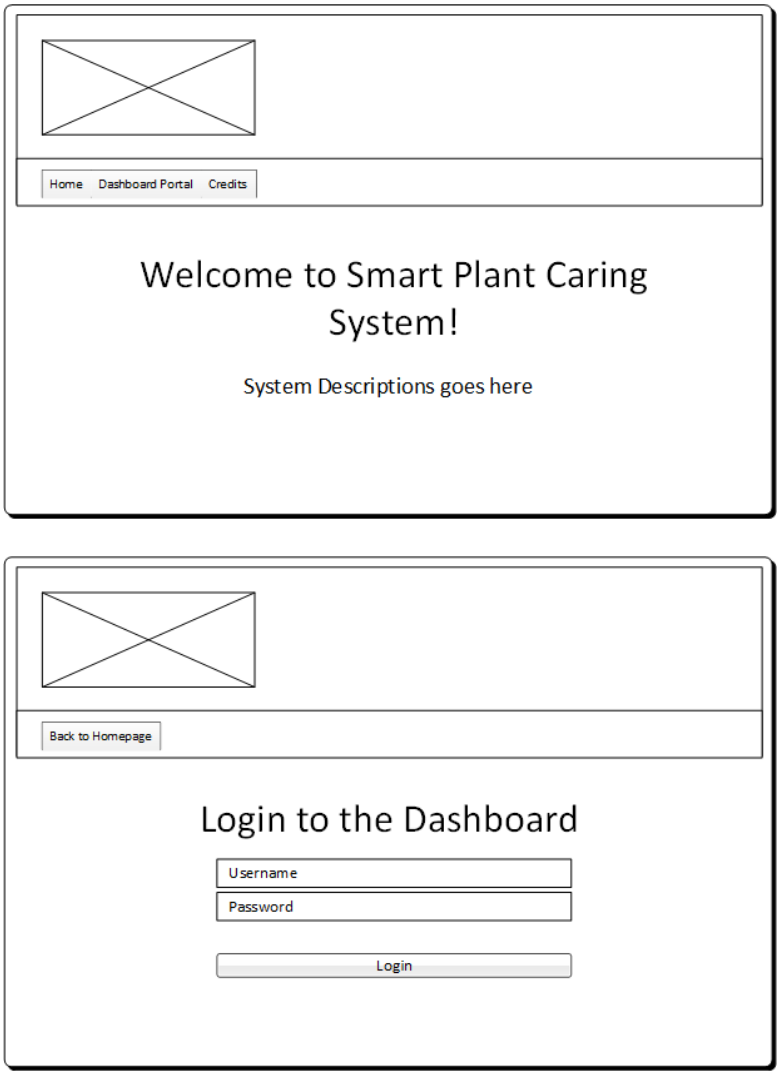



***Figure 27: Low Fidelity Home and Login Interface Design***

As suggested in the figure above, the system webpages such as the home and login page shown in the example have been designed similarly to ensure that each interface will remain the same consistency while significantly enhancing user experience by reducing user's memory load where users will not require to remember information between pages.

### 5.6.2 High Fidelity

The user interface design continues by creating a high fidelity interface for the system. An example of the dashboard high fidelity design has been shown in the figure below. The complete high fidelity design can be found in Appendix 15.

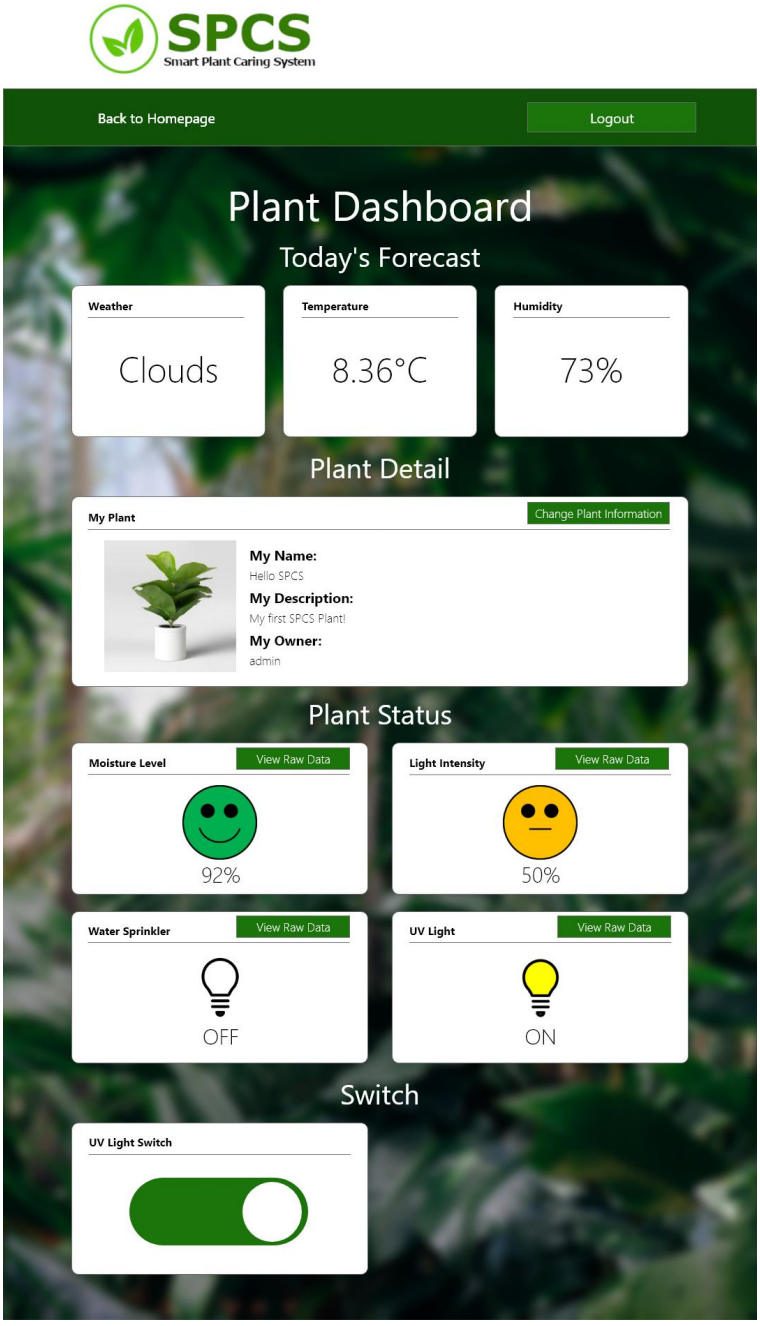



***Figure 28: High Fidelity Dashboard Interface Design***

The high fidelity design of the interface seen in the figure above has shown that the interface has applied the uses of colour and shapes suggested by Laubheimer (2017) discussed in Chapter 2.4.6 of the literature review to practise good dashboard design. Referring back to the visualisation method chosen in Chapter 3.1.5, the survey participant has selected the Traffic Light Faces as the primary method for visualising the collected sensor data. Hence, a set of "faces" has been designed by applying the use of colour and shapes, as seen in the table below.

|  |  |  |
|---|---|---|
| Soil is moist | Soil is about to dry | Soil is dry |

***Table 21: Emoji Traffic Light designed using colour and shapes***

The colours applied on the "face" shown in the table above are utilised to categorise certain elements such as the soil moisture level where red indicates the soil is fully dried, yellow indicates the soil is partially dried, and green indicates the soil is moist. Additionally, the uses of different face shapes have also been implemented into the design of the visualising "faces" to serve as an alternative grouping cue for users with impaired colour vision, which significantly enhances the accessibility of the SPCS dashboard.

Finally, the eighth principle of Jakob Nielson's Usability Heuristic has also been applied to the high fidelity design. The principle ensures that the system is aesthetic and contains a minimalist design. The high fidelity design of the system has been implemented while ensuring that the user interface will keep a design where it will only contain information or functionalities that are relevant and essential to the users.

# Chapter 6: Product Implementation

This dissertation chapter will look into how the product was implemented, including Docker Configuration, investigating Token-Based Authentication, and the functionality implementation for both hardware and web dashboard.

## 6.1 Building the physical set-up

A physical set-up has been built for this project to serve as the plant caring machine. At the beginning of the building step, a design model was built using Autodesk which has significantly enhanced the design process. After the design phase has completed, the model was then printed using a 3D printer. Although the printing process has took almost 5 full days to complete, the result still remains satisfying. The final printing physical product could be seen in the figure below:

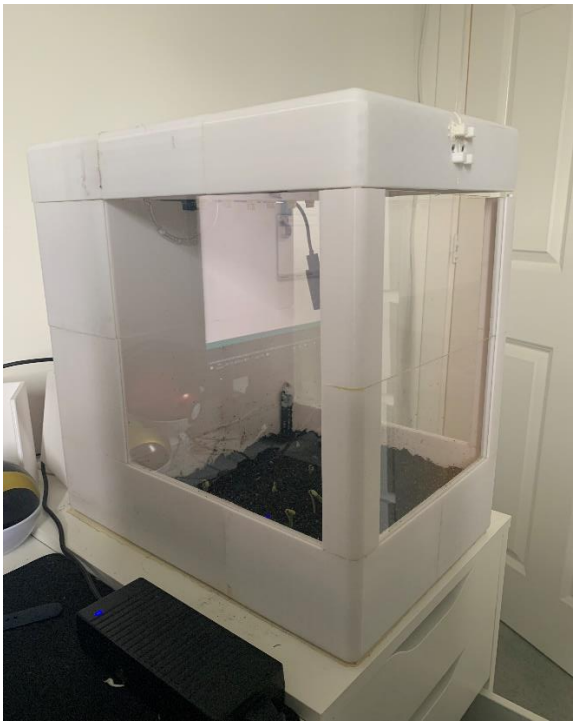


***Figure 29: SPCS System Physical Set up***

The design of the physical set-up has taken its inspired from a computer case. Howeve, to ensure that the sensor and actuator could be fit into the physical set up, an extra compartment slot has been created at the back of the set-up to allow more space for the sensor modules, as seen in the figure below. Hence, completing the implementation of the physical set-up.

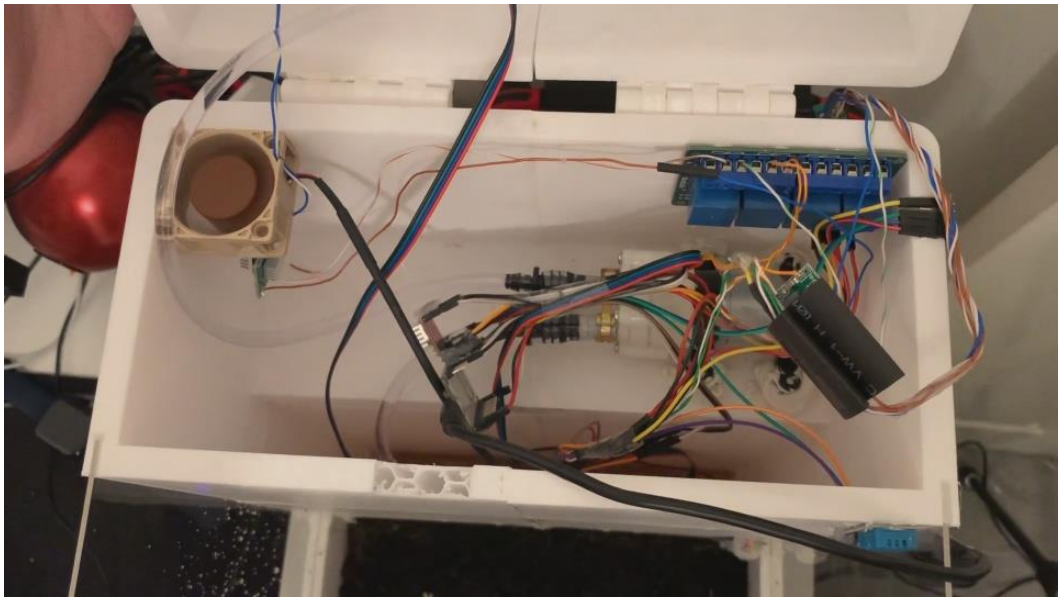


***Figure 30: SPCS Back Compartment***

## 6.2 Configuring Docker

Docker is considered the essential part of the system because it will be used to run all the tools selected, including MQTT, InfluxDB, PHP, and more, as discussed in Chapter 4.1.1. In this project, the Docker tool, Compose, has been utilised for accelerating the defining and running multi-container

applications process. In the beginning, a YAML file has been prepared to define each application's service and assign their respective ports, as shown in the example code snippet below:

```yaml
services:
  php:
    image: php:8.1.3-apache
    container_name: php
    restart: 'unless-stopped'
    ports:
      - 80:80
    volumes:
      - ./dashboard:/var/www/html/
      - ./php.ini:/usr/local/etc/php/php.ini

  influxdb:
    image: influxdb:latest
    container_name: influxdb
    restart: 'unless-stopped'
    ports:
      - "8086:8086"
    volumes:
      - ./influxdb:/var/lib/influxdb
    environment:
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=admin
      - DOCKER_INFLUXDB_INIT_PASSWORD=admin123
      - DOCKER_INFLUXDB_INIT_ORG=spcs-org
      - DOCKER_INFLUXDB_INIT_BUCKET=spcs

  telegraf:
    image: telegraf
    container_name: telegraf
    restart: 'unless-stopped'
    volumes:
      - ./telegraf:/etc/telegraf
      - ./telegraf/telegraf.conf:/etc/telegraf/telegraf.conf:ro
    depends_on:
      - influxdb
    links:
      - influxdb
    ports:
      - '8125:8125'
```

***Code 1: docker-compose.yml application service configuration file***

After all, the application service has been appropriately configurated in the docker-compose.yml files. The command "docker-compose up -d" is executed on the Linux system to start and install the application services specified in the configuration file. Upon completion, the "docker ps" command is performed to check if each container has been successfully created, as shown in the figure below. Hence, concluding the Docker configuration process.

```
ubuntu@ubuntu:~$ sudo docker ps
sudo: unable to resolve host ubuntu: Name or service not known
CONTAINER ID   IMAGE              COMMAND                  CREATED      STATUS       PORTS                                                                                  NAMES
35dd576c2c73   php:8.1.3-apache   "docker-php-entrypoi…"   8 days ago   Up 19 hours  0.0.0.0:80->80/tcp, :::80->80/tcp                                                      php
88b5b2ef4d59   eclipse-mosquitto  "/docker-entrypoint.…"   9 days ago   Up 19 hours  0.0.0.0:1883->1883/tcp, :::1883->1883/tcp, 0.0.0.0:9001->9001/tcp, :::9001->9001/tcp   MQTT
e3c761e7eee3   telegraf           "/entrypoint.sh tele…"   9 days ago   Up 19 hours  8092/udp, 8125/udp, 8094/tcp, 0.0.0.0:8125->8125/tcp, :::8125->8125/tcp                telegraf
613704d47a4b   mysql:8.0-oracle   "docker-entrypoint.s…"   9 days ago   Up 19 hours  0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp                                   mysql
dd530dc4bf1e   influxdb:latest    "/entrypoint.sh infl…"   9 days ago   Up 19 hours  0.0.0.0:8086->8086/tcp, :::8086->8086/tcp                                              influxdb
```

***Figure 31: Docker is running every container***

## 6.3 Token-Based Authentication

As discussed in Chapter 2.5, the project will be investigating the use of token-based authentication in the system. The development of token-based authentication started by creating an API called authentication to act as the gateway for handling user authentication. The API endpoint discussed in this section is the endpoint for verifying and authenticating users. The endpoint uses the Firebase

JSON Web Token library for PHP to handle user tokens. It has been proven to be a high-performance and scalable solution for user authentication and access control (Jánoky et al., 2018).

```php
//Handling authenticate request
if (isset($_REQUEST['authenticate'])) {

    if(isset($_POST['username']) && isset($_POST['password'])){
        //If username and password parameter is providedAuthenticate the user
        $this->setResponse($this->authenticateUser($_POST['username'],$_POST['password']));

    }else{
        //Return 400 - Incorrect Parameter
        $this->setResponse($this->showError(400));

    }
```

***Example API URL:***
**192.168.0.216/index.php/api/authentication?authenticate**

***Code 2: API Endpoint for authenticating user***

When the authentication API has been requested with the username and password variable provided, the function “authenticateUser” will be called to authenticate the user and generate the necessary access token. The “authenticateUser” function are seen in the code snippet below:

```
     * @param string $username The user's username
     * @param string $password The user's password
     *
     * @return array Returns the status of the API
     */
    private function authenticateUser($username,$password){
        try{
            //Retrieve the user information (ID and Hash)
            $retrievedResult = $this->retrieveUserHash($username);

            if(!$retrievedResult){
                //If retrievedResult returned false, return 204 - No Content
                return $this->showError(204);

            }

            //Verify the user password with hash
            if(password_verify($password, $retrievedResult['userHash'])) {

                //If verification returns true, generate the token
                $key = SECRET_KEY;
                $payload = array(
                    "user_id" => $retrievedResult['userID'],
                    "user_name" => $username,
                    "exp" => time() + 86400
                );

                //Start generating JSON Web Token
                $jwt = JWT::encode($payload, $key, 'HS256');

                //Insert generated token into the 'data' array
                http_response_code(200);

                //Setting up the session
                $session = new Session();
                $_SESSION['token'] = $jwt;

                //Return success message
                return array("authentication" => true);

            }else{
                //Return 401 - Not authorised message
                return $this->showError(401);

            }
        }catch (Exception $e){
            //Return 500 - Internal Server Error
            return $this->showError(500);

        }
    }
```

***Code 3: authenticateUser function***

The function will first retrieve the user hash stored in the MySQL database by calling the "retrieveUserHash" function, where the code can be found in the code snippet below:

```php
/**
 * retrieveUserHash
 *
 * To retrieve the user ID and hashed password.
 *
 * This function will be used to retrieve the user's ID and hashed password
 * from the SQL database.
 *
 * @param string $username The user's username
 * @param string $password The user's password
 *
 * @return array Returns the status of the API
 */
private function retrieveUserHash($username){
    try{
        //Retrieve the user's ID and hashed password
        $database = new Database();

        //Preparing the SQL query
        $query = "SELECT userID, userHash FROM User WHERE userName = :name";
        $parameter = ["name" => $username];

        //Execute the SQL
        $result = $database->executeSQL($query, $parameter)->fetch();

        if(!empty($result)){
            //Return the first result
            return $result;

        }else{
            //If result is empty, return false
            return false;

        }
    }catch (Exception $e){
        //Return false
        return false;

    }
}
```

***Code 4: retrieveUserHash function***

Upon successfully retrieving the user hash, the password_verify() function verifies the given password with the retrieved hash. If the verification process returns true, the token generating and encoding process will start using the preset secret key and payload. The attribute 'exp' has been supplied into the payload to ensure that the token will only have a lifetime of 24 hours, equivalent to 86400 seconds before expiring. The generated token is stored using the session variable instead of keeping it on the user's browser to enhance the user's security. Hence, completing the authentication process.

## 6.4 Arduino IoT

### 6.4.1 Uploading sensor data

One of the core functions of the SPCS system is to capture sensor data such as soil moisture, light intensity, temperature, and humidity, as discussed in Chapter 3.1.5. The implementation of this functionality begins by connecting the necessary sensor module to the microcontroller, as seen in the figure below.

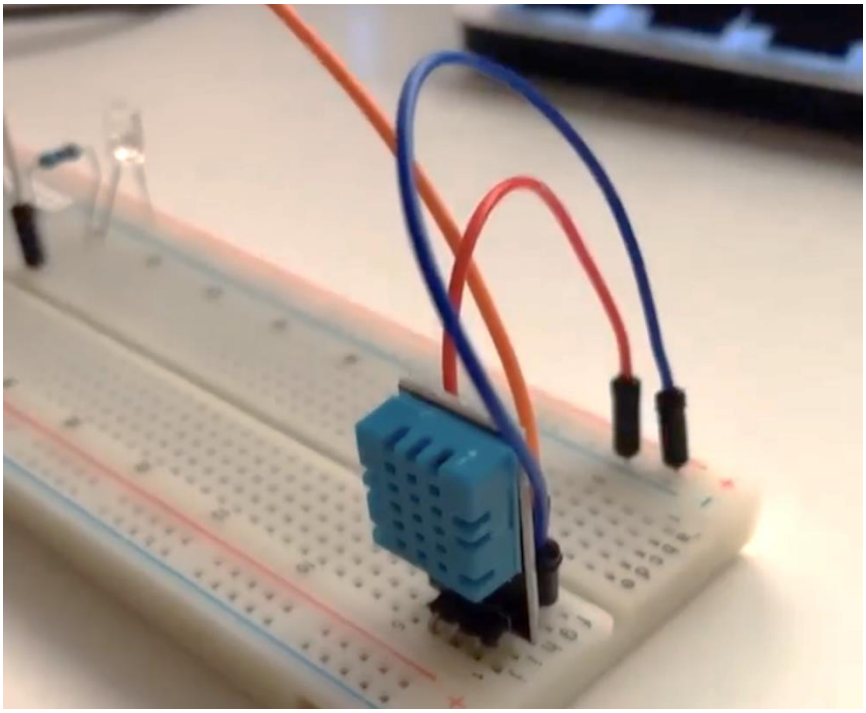


***Figure 32: Connecting Sensor Module***

Upon connecting the sensory module, it is essential to keep in mind that an internet connection will be required to upload data to the database. Hence, the implementation ensures that the microcontroller is first connected to the internet. Using the Arduino IDE, a few lines of code have been compiled into the microcontroller, as seen in the code snippet below:

```
//WiFi Configuration
char wifi_ssid[] = "testESP";
char wifi_pass[] = "abc12345";

void setup() {
    // Debug console
    Serial.begin(9600);

    //Start internet connection
    WiFi.begin(wifi_ssid, wifi_pass);
    Serial.println("Connecting");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.print("Connected to WiFi network with IP Address: ");
    Serial.println(WiFi.localIP());
}
```

***Code 5: Connecting Microcontroller to the Internet***

After connecting the microcontroller to the internet, the microcontroller is now capable of beginning the process of uploading data to the database. The figure below illustrates how the collected data will be stored in the time-series database, InfluxDB, through publishing data to MQTT and Telegraf to send data into the database.


***Figure 33: Process of uploading data to InfluxDB***

However, since Telegraf has already been appropriately configurated during the Docker Application Service configuration process, the microcontroller will now only be required to focus on publishing the collected sensor data to the MQTT service. As a result, an MQTT library for Arduino has been downloaded into the Arduino IDE. Several lines of code have been compiled and uploaded into Arduino, as seen in the code snippet below.

```
void publishData(const char publishTopic[], int publishValue){
    mqttClient.beginMessage(publishTopic);
    mqttClient.print(publishValue);
    mqttClient.endMessage();
}
```

***Code 6: Publishing sensor data into MQTT***

Finally, a small test has been conducted to confirm that the data has been successfully uploaded into the database by inspecting the data flow through the InfluxDB interface. The figure below shows that the data has been successfully stored in the database. Hence the implementation of the upload sensor data functionality is completed.

| table | _measurement | _field | _value | _start | _stop | _time | host | topic |
|---|---|---|---|---|---|---|---|---|
| 0 | mqtt_consumer | value | 315 | 2022-03-27T02:12:23.887Z | 2022-04-26T02:12:23.887Z | 2022-04-17T06:00:00.000Z | e3c761e7eee3 | soil-moisture |
| 0 | mqtt_consumer | value | 600 | 2022-03-27T02:12:23.887Z | 2022-04-26T02:12:23.887Z | 2022-04-17T07:00:00.000Z | e3c761e7eee3 | soil-moisture |
| 0 | mqtt_consumer | value | 200 | 2022-03-27T02:12:23.887Z | 2022-04-26T02:12:23.887Z | 2022-04-17T17:00:00.000Z | e3c761e7eee3 | soil-moisture |
| 0 | mqtt_consumer | value | 500 | 2022-03-27T02:12:23.887Z | 2022-04-26T02:12:23.887Z | 2022-04-17T18:00:00.000Z | e3c761e7eee3 | soil-moisture |
| 0 | mqtt_consumer | value | 348 | 2022-03-27T02:12:23.887Z | 2022-04-26T02:12:23.887Z | 2022-04-17T19:00:00.000Z | e3c761e7eee3 | soil-moisture |
| 0 | mqtt_consumer | value | 215 | 2022-03-27T02:12:23.887Z | 2022-04-26T02:12:23.887Z | 2022-04-17T20:00:00.000Z | e3c761e7eee3 | soil-moisture |
| 0 | mqtt_consumer | value | 200 | 2022-03-27T02:12:23.887Z | 2022-04-26T02:12:23.887Z | 2022-04-17T21:00:00.000Z | e3c761e7eee3 | soil-moisture |

***Figure 34: Validating Data Flow through InfluxDB interface***

## 6.4.2 Automating Watering process

Referring to the additional functionality identified in Chapter 3.1.5, the following functionality consists of automating the watering process. The implementation of this functionality begins by referring to the designed sequence diagram of the automation watering process, as shown in figure 20 in Chapter 5.3, to have a general idea of how the process should be implemented.

As seen in the sequence diagram, the implementation of this functionality should be broken down into two separate steps. The first step of the implementation should ensure that the microcontroller is subscribed to the soil moisture topic to retrieve a copy of the latest data. The same MQTT library installed in the previous section has been used to subscribe to the moisture data, as seen in the code snippet below.

```
Serial.print("Subscribing to topic:");
Serial.println(topic);
Serial.println();

mqttClient.subscribe(topic);
```

***Code 7: Subscribing to topic using MQTT***

The next step of the implementation includes activating the actuators and water pumps to start the watering automation process. To achieve this, the retrieved subscribe data is compared against a set of values to verify if the soil has dried up. The comparison table is as seen in the table below.

| Moisture Value | Lower than 300 | Between 300 and 600 | Higher than 600 |
|---|---|---|---|
| Soil Moisture Level | Moist | Partially Dried | Dried |

***Table 22: Soil Moisture Level Table***

As the table suggested, the watering automation process should not begin until the soil has completely dried up. Hence, a few lines of code have been compiled and uploaded into the microcontroller to compare the retrieved value before activating the watering process, as seen in the code snippet below. Hence, completing the implementation process of this functionality.

```
void wateringAutomationProcess(retrievedMoistureValue)
{
  //If the soil moisture has exceed an amount of 600, starts the watering process
  if(retrievedMoistureValue >= 600)
  {
      //Set the relay to LOW (The relay is active low)
      digitalWrite(stepperRelay, LOW);
      digitalWrite(waterPumpRelay, LOW);
  }
}
```

***Code 8: Watering Automation Process function***

## 6.5 SPCS IoT Dashboard

### 6.5.1 Visualising the collected data stored in InfluxDB

To retrieve and visualising the sensor data stored on the webserver, the development started by creating another API called plantdata to act as the gateway to retrieve data stored in InfluxDB. Several endpoints have been developed within the API to ensure that the data is accessible by JQuery using the AJAX method. The API has utilised the InfluxDB library to add InfluxDB support to the PHP environment.

Before the system could retrieve the stored data on the database, “InfluxDB2\Client” was used to initialise an InfluxDB client object to connect to a running InfluxDB instance using parameters such as URL, and user token, destination bucket, and organisation name as seen in the code snippet below.

```php
$this->client = new Client([
    "url" => "http://192.168.0.216:8086",
    "token" => [redacted],
    "bucket" => "spcs",
    "precision" => WritePrecision::NS,
    "org" => "spcs-org",
    "debug" => false
]);
```

***Code 9: Connecting to InfluxDB Instance***

After the connection has been successfully initialised, the API can now proceed to retrieve data from the database. The endpoint introduced in this section is the endpoint for retrieving collected sensor data stored in InfluxDB.

```php
//If the URL requested for "latest"
if(isset($_REQUEST['latest'])){

    //Check if 'topic' parameter is provided
    if(isset($_GET['topic'])){

        //Call the getLatestData function with the specified 'topic' parameter and set into result variable
        $result = $this->getLatestData($_GET['topic']);

        if($result == null){
            //If result returned null, set response to display error 204 - No content
            $this->setResponse($this->showError(204));

        }else{
            //If result is not null, set response to the result
            $this->setResponse($result);

        }

    }else{
        //Set response to display error 400 - Incorrect Parameter
        $this->setResponse($this->showError(400));
    }
```

***Example API URL:***
**192.168.0.216/index.php/api/plant?latest&topic=soil-moisture**

***Code 10: API Endpoint for retrieving the latest data from InfluxDB***

When the 'latest' API has been requested with a topic variable provided, the function "getLatestData" will be called to retrieve the data from InfluxDB. The "getLatestData" function is as seen in the code snippet below:

```php
private function getLatestData($topic){

    //Preparing InfluxDB Query API
    $queryApi = $this->client->createQueryApi();

    //Querying to retrieve the last row of the data for a specific topic
    $result = $queryApi->queryStream(
        'from(bucket:"spcs")
            |> range(start: 1970-01-01T00:00:00.000000001Z)
            |> filter(fn: (r) => r["topic"] == "'.$topic.'")
            |> last()'
    );

    //Return the value of the retrieved data
    foreach($result->each() as $record){
        return ["value" => $record->getValue()];
    }

}
```

***Code 11: getLatestData function***

According to InfluxDB documentation, the InfluxDB uses InfluxData's functional data scripting language, Flux, to query and analyse data. To retrieve data, a Flux query should consist of these three lines, which are the data source, time range, and data filters. The code snippet below shows the query for retrieving the latest data from a particular topic.

```
$result = $queryApi->queryStream(
    'from(bucket:"spcs")          <-- Data Source
        |> range(start: 1970-01-01T00:00:00.000000001Z)   <-- Time range
        |> filter(fn: (r) => r["topic"] == "'.$topic.'")   <-- Data filter
        |> last()'                <-- Retrieve the last row
);
```

***Code 12: Function to retrieve the latest data with a specific topic***

As seen in the code snippet, the first line of the query represents the InfluxDB data source. The data source line will require a "bucket" parameter, equivalent to a "table" in MySQL. Since InfluxDB is a time-series data, it is necessary to specify a time range when querying for data. The documentation has mentioned that the Flux query will not be executable without a specified range. Hence, the start time of the range has been set to '1970-01-01T00:00:00.000000001Z' to ensure that all the data will be retrieved without missing out on any essential details. The query will need to include the 'filter()' function to narrow down results based on the selected data column or attributes on the third line. For this case, the function has been developed to accept a "$topic" parameter which will then be parsed into the Flux 'filter()' function to specify which topic should be retrieved by the query. The 'last()' function has been included on the last line to ensure that the query will only retrieve the last row of data.

Figure 35 below shows an example result retrieved by the endpoint's function requested through Postman API Tools:


***Figure 35: The result of the endpoint running through the Postman API Tool***

To visualise the retrieved data, a Javascript script has been implemented where it will call the "updateVisualiseData" function on load to send four separate GET requests to the 'latest' API endpoint using the AJAX function to retrieve different data as seen in the code snippet below:

```
$("document").ready(function(){
    updateVisualiseData();
    setInterval(updateVisualiseData,3000);
})

function updateVisualiseData(){
    updateMoistureData();
    updateLightData();
    updateSprinklerStatus();
    updateUVStatus();
}

function updateMoistureData(){
    //Get moisture data using AJAX
    $.ajax({
        //URL to the API Endpoint
        url: 'api/plant?latest&topic=soil-moisture',
        async:false,

        //If AJAX successfully retrieved data
        success: function (result) {

            if(result['value'] < 300){
                //If value is less than 300, show Happy face
                changeFaceVisual(".moisture-visual-face",0);

            }else if(result['value'] >= 300 && result['value'] < 600){
                //If value is between 300 to 600, show Neutral Face
                changeFaceVisual(".moisture-visual-face",1);

            }else if(result['value'] >= 600){
                //If value is more than 600, show Sad Face
                changeFaceVisual(".moisture-visual-face",2);
            }

            //Display the value in percentage
            $('p.moisture-percent').text(Math.abs((result['value']/10)-100)+"%");
        }
    })
}
```

***Code 13: updateMoistureData function to retrieve and visualising the data***

After that, to meet the requirement of visualising data using Traffic Light Faces, as discussed in the survey results for visualising method in Chapter 3.2.5, the retrieved value such as the Moisture Level and Brightness Level has been compared against a set of preset numbers to decide which "face" image should be displayed. After comparison, the 'changeFaceVisual' function, as shown in the code snippet

below, has been called to change the displaying “faces” by modifying the CSS display property using the JQuery CSS() method and hence getting the result shown in figure 36.

```
/**
 * changeFaceVisual
 *
 * The function is used to toggle between the Traffic Light Face
 *
 * @param {string} visualClass The class of the image
 * @param {int} face The type of face to be displayed (0 - Happy, 1 - Neutral, 2 - Sad)
 */
function changeFaceVisual(visualClass,face){
    //Reset every image display properties back to none to hide every face image
    $(visualClass).eq(0).css("display","none");
    $(visualClass).eq(1).css("display","none");
    $(visualClass).eq(2).css("display","none");

    switch(face){
        case 0:
            //Change the Happy image display property to Block to only display Happy image
            $(visualClass).eq(0).css("display","block");
            break;
        case 1:
            //Change the Neutral image display property to Block to only display Neutral image
            $(visualClass).eq(1).css("display","block");
            break;
        case 2:
            //Change the Sad image display property to Block to only display Sad image
            $(visualClass).eq(2).css("display","block");
            break;
    }
}
```

***Code 14: changeFaceVisual function to change the displaying “Faces” using CSS Display***

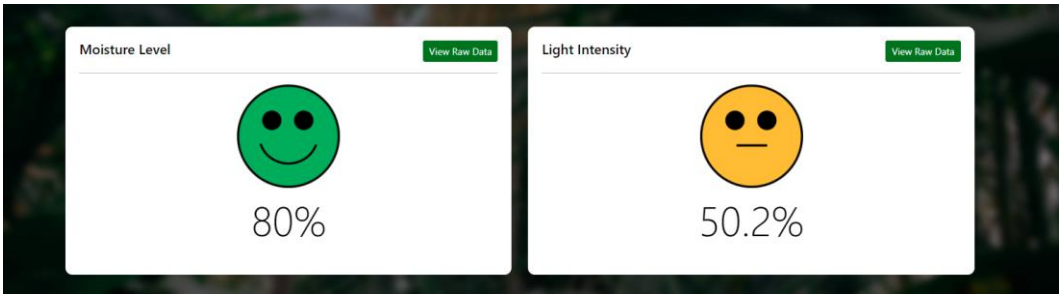



***Figure 36: Result of Visualising Data on Dashboard***

And finally, to update the data without constantly refreshing the website, the setInterval() function has been included in the JavaScript script to continually recall the “updateVisualData” function every three seconds, as shown in the code below:

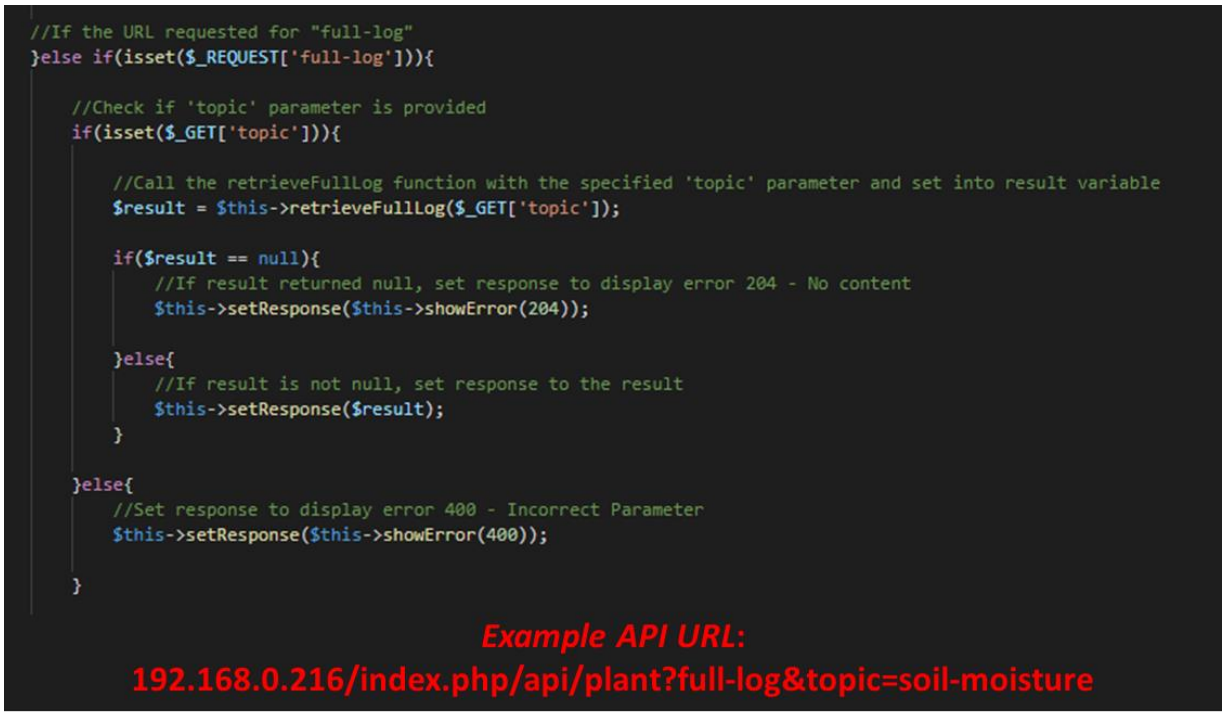
```
$("document").ready(function(){
    updateVisualiseData();
    setInterval(updateVisualiseData,3000);
})
```

***Code 15: setInterval to update the data***

## 6.5.2 View Raw Data

To achieve the view raw data requirements requested by Participant 5 (Chapter 3.2.5). A similar approach as the previous section has been taken where another endpoint called ‘full-log’ has been

added to the plantData API to retrieve all the data points of the topic stored in InfluxDB, as seen in the code snippet below:



```php
//If the URL requested for "full-log"
}else if(isset($_REQUEST['full-log'])){

    //Check if 'topic' parameter is provided
    if(isset($_GET['topic'])){

        //Call the retrieveFullLog function with the specified 'topic' parameter and set into result variable
        $result = $this->retrieveFullLog($_GET['topic']);

        if($result == null){
            //If result returned null, set response to display error 204 - No content
            $this->setResponse($this->showError(204));

        }else{
            //If result is not null, set response to the result
            $this->setResponse($result);
        }

    }else{
        //Set response to display error 400 - Incorrect Parameter
        $this->setResponse($this->showError(400));

    }
```



***Code 16: API Endpoint for retrieving complete data log from InfluxDB***

When the 'full-log' API has been requested with a topic variable provided, the function "retrieveFullLog" will be called to retrieve all the data points of the specified topic from InfluxDB. The "retrieveFullLog" function is as seen in the code snippet below:

```php
private function retrieveFullLog($topic){

    //Initialising an empty array
    $resultArray = [];

    //Preparing InfluxDB Query API
    $queryApi = $this->client->createQueryApi();

    //Querying to retrieve the every data point for a specific topic
    $result = $queryApi->queryStream(
        'from(bucket:"spcs")
            |> range(start: 1970-01-01T00:00:00.000000001Z)
            |> filter(fn: (r) => r["topic"] == "'.$topic.'")'
    );

    foreach($result->each() as $record){
        //Pushing results into the empty array
        array_push($resultArray,
            [
                "timestamp" => $record->getTime(),
                "value" => $record->getValue()
            ]
        );
    }

    //Returning the
    return $resultArray;
}
```

***Code 17: retrieveFullLog function***

However, different from the 'latest' endpoint, the query implemented in the retrieveFullLog function has been slightly modified. The "last()" tag has been removed to retrieve every data point stored in that topic seen in the code snippet below.

```php
$queryApi = $this->client->createQueryApi();

//Querying to retrieve the every data point for a specific topic
$result = $queryApi->queryStream(
    'from(bucket:"spcs")
        |> range(start: 1970-01-01T00:00:00.000000001Z)
        |> filter(fn: (r) => r["topic"] == "'.$topic.'")'
);

foreach($result->each() as $record){
```

"Last()" tag has been removed

***Code 18: Function to retrieve the complete data log with a specific topic***

The result of the query is then being formatted and rebuilt into the empty array so that the returned development will only consist of data that are needed to be displayed. Figure 37 shows an example of a soil moisture data log retrieved by the endpoint's function requested through the Postman API Tool.

```
{
    "timestamp": "2022-04-17T05:32:14.556690272Z",
    "value": 615
},
{
    "timestamp": "2022-04-17T05:50:23.783064168Z",
    "value": 315
},
{
    "timestamp": "2022-04-17T06:09:25.211324401Z",
    "value": 215
},
{
    "timestamp": "2022-04-17T06:40:41.372914041Z",
    "value": 600
},
{
    "timestamp": "2022-04-17T16:38:56.098299278Z",
    "value": 200
},
```


***Figure 37: The data log retrieved by the endpoint running through Postman API Tool***

To display the data on the dashboard, a separate button has been created on the dashboard where users will be able to decide if they would like to view the raw data, as shown in the figure below:

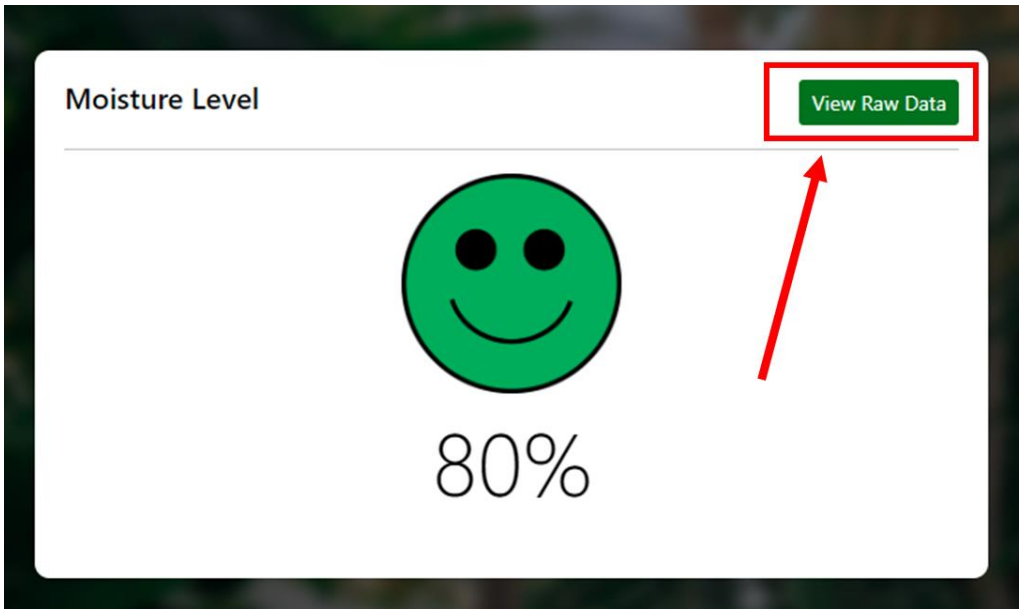



***Figure 38: View Raw Data button***

When the button is clicked, the JavaScript function 'retrieveRawData' will be called to send a GET request to the "full-log" API endpoint using AJAX and generate a table using the retrieved result. The retrieveRawData function could be seen in the code snippet below:

```
/**
 * retrieveRawData
 *
 * A function used to retrieve a specific topic's full data point
 *
 * @param {string} topic The topic to retrieve data log
 */
function retrieveRawData(topic){
    //Retrieve all data point
    $.ajax({
        //URL to the API Endpoint with specified topic
        url: 'api/plant?full-log&topic='+topic,
        async:false,

        //If AJAX successfully retrieved data
        success: function (result) {

            //Generate the table
            let fullLogTable =
                "<div class='table-responsive'>\n"+
                    "<table class='table table-striped'>\n"+
                        "<thead>\n"+
                            "<th>Timestamp</th>\n"+
                            "<th>Value</th>\n"+
                        "</thead>\n"+
                        "<tbody>\n";

            //Generate table row for each data in the result
            $.each(result, function(index){
                fullLogTable +=
                    "<tr>\n"+
                        "<td>"+result[index].timestamp+"</td>\n"+
                        "<td>"+result[index].value+"</td>\n"+
                    "</tr>\n";
            })

            //Generate end tags
            fullLogTable += "</tbody>\n"+
                "</table>\n"+
                "</div>"

            //Push table into a container named 'raw-data-table' within the generated modal
            $('.raw-data-table').html(fullLogTable);
        }
    })
}
```

***Code19: retrieveRawData Javascript function***

The table generating process is divided into three steps, as shown in the code snippet below, where it will first create the opening HTML tags and appropriate table headers. Then, it will generate a new table row for each data in the retrieved result. And finally, the end tags for the table body and container. The generated table is then pushed into a container named ‘raw-data-table’ in a pre-generated modal, hence getting the result shown in figure 39

```
//If AJAX successfully retrieved data
success: function (result) {

    //Generate the table
    let fullLogTable =
        "<div class='table-responsive'>\n"+
            "<table class='table table-striped'>\n"+
                "<thead>\n"+
                    "<th>Timestamp</th>\n"+
                    "<th>Value</th>\n"+
                "</thead>\n"+
                "<tbody>\n";

    //Generate table row for each data in the result
    $.each(result, function(index){
        fullLogTable +=
            "<tr>\n"+
                "<td>"+result[index].timestamp+"</td>\n"+
                "<td>"+result[index].value+"</td>\n"+
            "</tr>\n";
    })

    //Generate end tags
    fullLogTable += "</tbody>\n"+
        "</table>\n"+
        "</div>"

    //Push table into a container named 'raw-data-table' within the generated modal
    $('.raw-data-table').html(fullLogTable);
}
```

Step 1

Step 2

Step 3

*Code 20: Process to generate a new data log table*

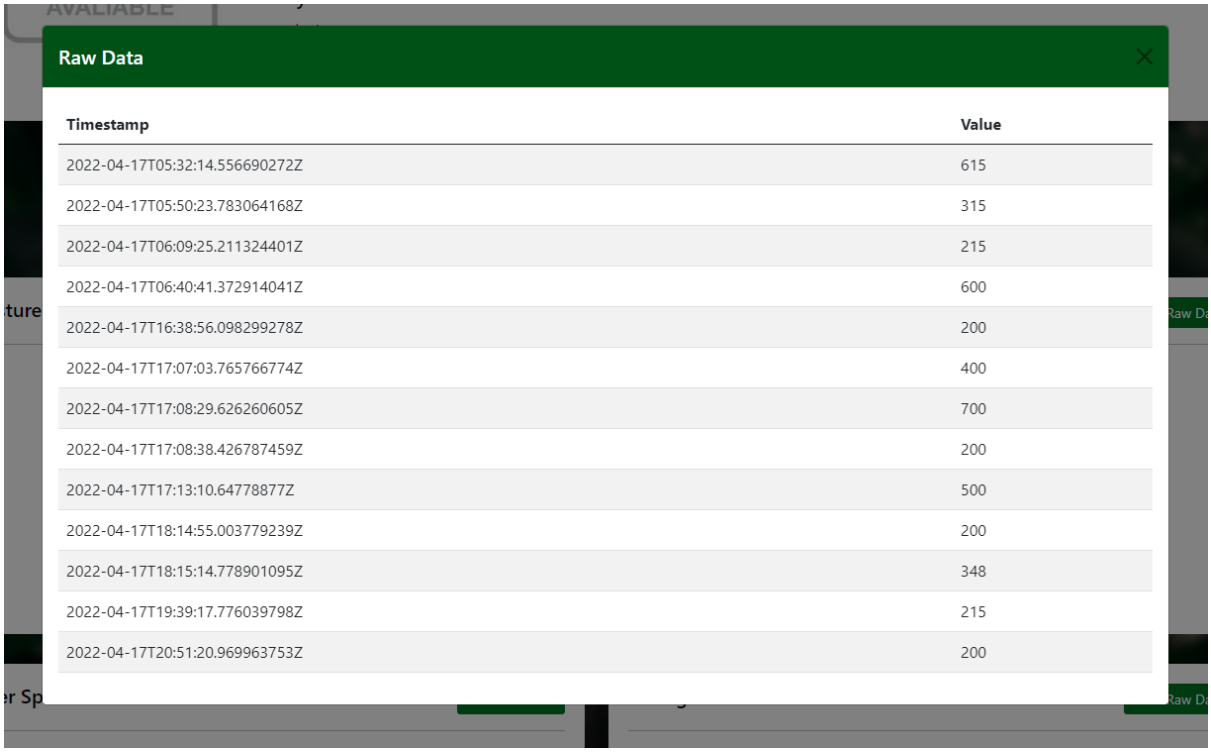



*Figure 39: Result of Displaying Raw Data in a Modal View*

### 6.5.3 Display Weather Forecast

To display the weather forecast as requested by Participant 2 (Chapter 3.2.5), the OpenWeather API has been selected to get the current weather forecast of a specific location. The OpenWeather API has been chosen for this project because it provides Free tier users 1,000,000 lightning-fast API calls per month, whereas the other API will have lesser calls available on the Free tier account. Before the development of this functionality can be started, it is essential first to review the documentation created by the API provider to have a general idea of how to access the API and understand how the data are structured, as shown in figure 40 and 41 below:


***Figure 40: Current Weather Data API Documentation (OpenWeather, 2022)***

```
{
    "coord": {
        "lon": -1.6077,
        "lat": 54.9769
    },
    "weather": [
        {
            "id": 800,
            "main": "Clear",
            "description": "clear sky",
            "icon": "01d"
        }
    ],
    "base": "stations",
    "main": {
        "temp": 8.65,
        "feels_like": 5.84,
        "temp_min": 7.82,
        "temp_max": 9.21,
        "pressure": 1015,
        "humidity": 87,
        "sea_level": 1015,
        "grnd_level": 1009
    },
    "visibility": 10000,
    "wind": {
        "speed": 5.15,
        "deg": 52,
        "gust": 10.52
    },
```


***Figure 41: Example OpenWeather API Response running on Postman API Tool***

A JavaScript script has been implemented where it will call the function retrieveWeatherData() on load to send a GET request to the OpenWeather API endpoint using the AJAX function to retrieve the current weather forecast.

```
function retrieveWeatherData(){
    $.ajax({
        url: 'https://api.openweathermap.org/data/2.5/weather?lat=54.9768729&lon=-1.6077272&appid=[redacted]&units=metric',
        async:false,
        success: function(result){
            data = result;
        }
    })
}
```

***Code 21: Sending request to OpenWeather API using AJAX***

After that, three separate JavaScript function, such as “updateWeather”, “updateTemperature”, and “updateHumidity”, has been called to replace the default text, as shown in Figure 41 into the respective retrieved data. The code snippet below shows the codes for the three functions.

```
function updateWeather(){
    let weather = data['weather'][0]['main'];
    $('p.today-weather').text(weather);
}

function updateTemperature(){
    let temperature = data['main']['temp'];
    $('p.today-temperature').text(temperature+"°C");
}

function updateHumidity(){
    let humidity = data['main']['humidity'];
    $('p.today-humidity').text(humidity+"%");
}
```

***Code 22: Replace the default value with the data retrieved***

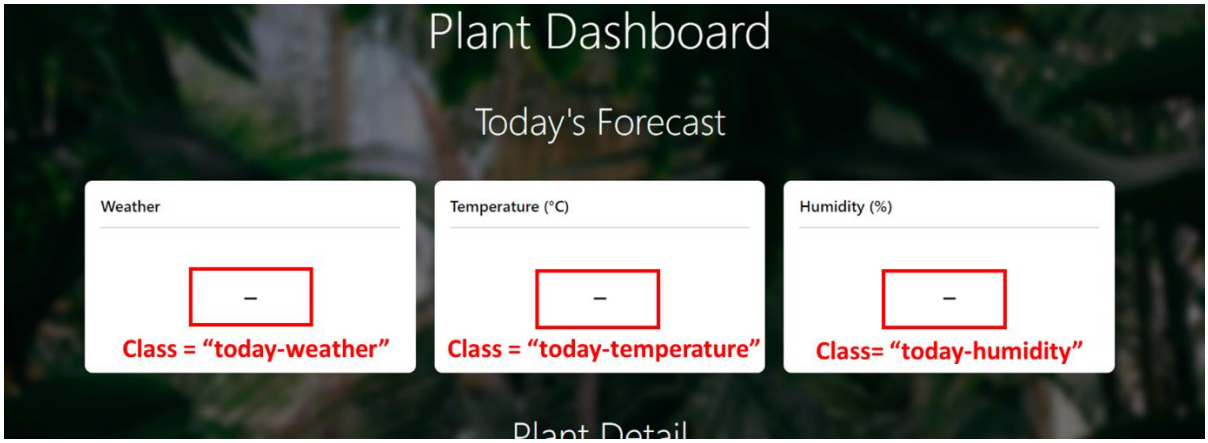



***Figure 41: Forecast Default Text and their respective classes***

And finally, a similar approach used in the visualising collected data section has also been applied in this section where the JavaScript script will refresh the data every 10 seconds using "setInterval()" as shown in the code snippet below and hence getting the result as shown in figure 42.

```
//Execute when the webpage is loaded
$('document').ready(function(){
    updateWeatherData();
    setInterval(updateWeatherData,10000);
})

function updateWeatherData(){
    retrieveWeatherData();
    updateWeather();
    updateTemperature();
    updateHumidity();
}
```

***Code 23: Run setInterval every 10 seconds to update weather data***

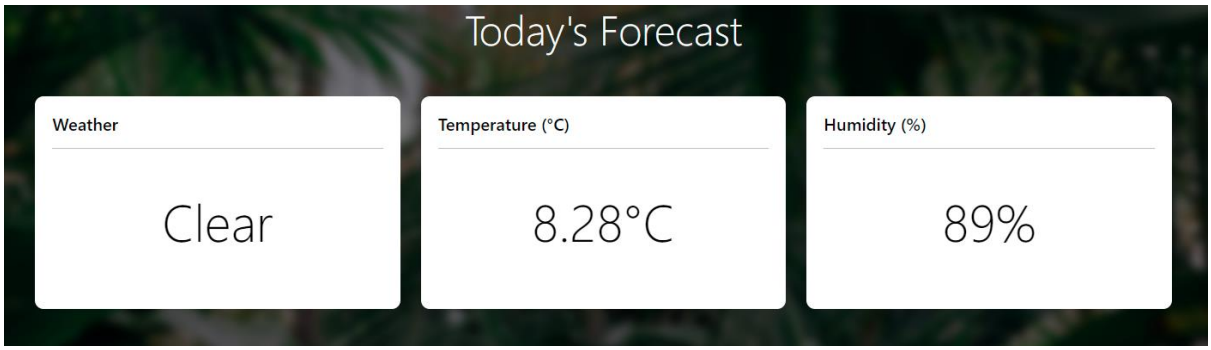



***Figure 42: Result of Display Weather Forecast feature***

### 6.5.4 Customisable Plant Information

To implement the customisable plant information functionality suggested by Participant 1 (Chapter 3.2.5). The implementation started by creating a new API endpoint to the existing "plantdata" class to retrieve the plant information stored in MySQL, as seen in the code snippet below.

```php
//If the URL requested for "plant-info"
}else if(isset($_REQUEST['plant-info'])){
        $this->setResponse($this->retrievePlantInformation($session->getUserID(),$session->getUserName()));
```

**Example API URL:**
**192.168.0.216/index.php/api/plant?plant-info**

***Code 24: API Endpoint to retrieve plant information***

When the "plant-info" endpoint has been requested, it will retrieve the user ID and username using getUserID() and getUserName() from the Session class and then use them as the parameter. After that, the function "retrievePlantInformation" will be called to retrieve plant information using SQL query. As seen in the code snippet below, the SQL query is prepared using PDO to sanitise the parameter to prevent potential SQL injection attacks.

```php
private function retrievePlantInformation($userID,$userName){
    $database = new Database();
    try{
        //Preparing the SQL query
        $query = "SELECT * FROM Plant WHERE userID = :userID";
        $parameter = ["userID" => $userID];

        //Execute the SQL
        $result = $database->executeSQL($query, $parameter)->fetch();

        //Check if result returns empty
        if(!empty($result)){

            //Set the plant ID into a session variable
            $_SESSION['plantID'] = $result['plantID'];

            //Return a formatted array
            return array(
                "plantName" => $result['plantName'],
                "plantDescription" => $result['plantDescription'],
                "plantOwnerName" => $userName,
                "plantImage" => $result['plantImage']
            );

        }else{
            //If result is empty, return false
            return $this->showError(204);

        }
    }catch (Exception $e){
        //Return false
        return $this->showError(500);
    }
}
```

***Code 25: retrievePlantInformation function***

After the API endpoint implementation has been completed, the endpoint is then requested by a JavaScript script called 'plantInformation' to retrieve the plant information and pre-render the information on the dashboard, as seen in the figure below:

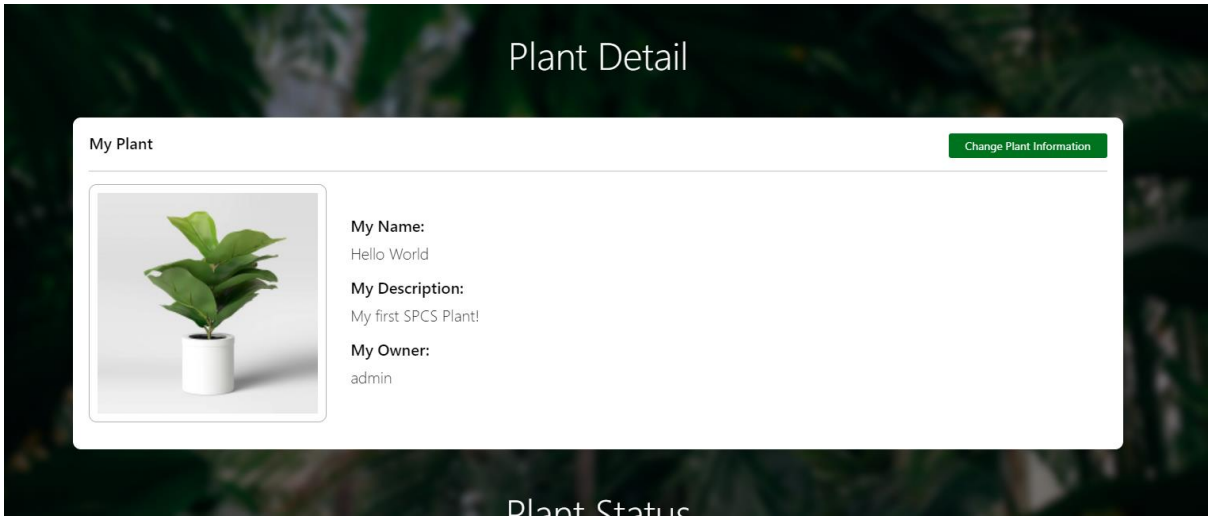



***Figure 43: Plant Information on SPCS Dashboard***

Additionally, a functionality to edit the plant information has also been implemented to provide users with full customisation to the plant information. The implementation phase for this functionality begins by creating a new database class called “plantdb” to handle the update process of the plant information, as seen in the code snippet below:

```php
<?php
class PlantDB extends Database{

    public function updatePlantInfo($id,$name,$description,$image){
        try{
            //Preparing the SQL query and parameter
            $query = "UPDATE Plant SET plantName = :name, plantDescription = :description";

            //Check if the image is null
            if($image != null){
                $query .= ", plantImage = :image WHERE plantID = :id";
                $parameter = ["id" => $id, "name" => $name, "description" => $description, "image"=>$image];
            }else{
                $query .= " WHERE plantID = :id";
                $parameter = ["id" => $id, "name" => $name, "description" => $description];
            }

            //Execute the query
            if($this->executeSQL($query, $parameter)){
                return true;
            }else{
                return false;
            }

        }catch(Exception $e){
            //Return false if SQL execution caught error
            return false;
        }
    }
```

***Code 26: plantdb class to handle SQL execution***

Similar to the approach taken in the View Raw Data functionality in Chapter 6.4.2, a modal has been created to handle the plant information edit form, as shown in the figure.


***Figure 44: Plant Information Editing Modal***

When the "Save Information" submit button is clicked, the updatePlantInfo function, as shown in code 27, will be called using the plant ID set in the session variable and the input value as a parameter. However, before the image file could be successfully stored in the database, the image was first being converted into a long blob format using the base64_encode() function, as seen in the imageToBlob function below:

```
private function imageToBlob(){
    if(is_uploaded_file($_FILES['plant-image']['tmp_name'])){
        return base64_encode(file_get_contents(addslashes($_FILES['plant-image']['tmp_name'])));
    }else{
        return null;
    }
}
```

***Code 27: imageToBlob function***

And finally, to prevent users from uploading files that are not an image, such as files with executable extension (.exe), an onChange function has been added to a JavaScript script to check the extension of the uploaded file and conduct error handling. If the function has identified a non-image extension file, an appropriate error will be displayed, signifying that an invalid file has been selected, as shown in the figure below. As a result, the implementation stage for the customisable plant information functionality has been completed.

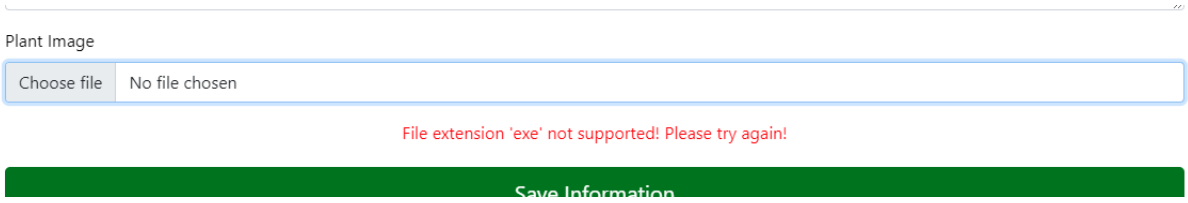



***Figure 45: Invalid File Extension Error Handling***

## 6.5.5 Manual Toggling UV Light

As seen in the additional functionality identified in Chapter 3.1.5, the dashboard will include a feature where users can toggle the UV light manually through the dashboard. Another API endpoint called

"toggle-light" has been implemented into the system to achieve that functionality, as seen in the code snippet below.

```
//If the URL requested for "toggle-light"
}else if(isset($_REQUEST['toggle-light'])){

        //Check if 'switch' parameter is provided
        if(isset($_GET['switch'])){

            //Call the toggleLight function with the specified 'switch' parameter and set into result variable
            $this->setResponse($this->toggleLight($_GET['switch']));

        }else{

            //Set response to display error 400 - Incorrect Parameter
            $this->setResponse($this->showError(400));

        }

                    Example API URL:
        192.168.0.216/index.php/api/plant?toggle-light&switch=1
```

***Code 28: API Endpoint for handling UV Light toggling***

When the 'toggle-light' API endpoint has been requested with a switch variable provided, the function "toggleLight" will be called to toggle the UV light by publishing an MQTT message to the "uv-status" topic using an external MQTT PHP library as seen in the code snippet below:

```
private function toggleLight($data){

    //Preset variables
    $host     = "192.168.0.216";
    $port     =  1883;
    $username = "admin";
    $password = "admin123";

    //Initialising connection to MQTT
    $mqtt = new phpMQTT($host, $port, "spcs");

    //Start connecting
    if ($mqtt->connect(true,NULL,$username,$password)) {

        //Publish the UV light data
        $mqtt->publish("uv-status",$data, 0);

        //Close the connection
        $mqtt->close();

    }else{
        //Return if mqtt connection run into error
        return $this->showError(500);

    }

}
```

***Code 29: toggleLight function***

To allow users to interact with the UV light toggling functionality, a switch has been added to the dashboard along with a JavaScript script called “toggler” to pre-toggle the light switch and handle change events as shown in the code snippet below:

```
$('document').ready(function(){

    //Pre-toggle the UV light switch by retrieving the latest "uv-status" data
    $.ajax({
        url: 'api/plant?latest&topic=uv-status',
        async:false,
        success: function (result) {
            if(result['value'] == 1){
                //Toggle the light switch to ON if the value returns 1
                $('#switch').attr('checked',true);
            }
        }
    })

})

$('#switch').change(function(){
    $.ajax({
        //Send "toggle-light" request
        url: 'api/plant?toggle-light&switch='+(Number(this.checked)),
        async:false
    })
})
```

***Code 30: toggler JavaScript function***

As the code suggested, the JavaScript script will begin by pre-rendering the UV light switch by retrieving the UV status through the “latest” API endpoint using the AJAX method on load. If the retrieved value is 1, the checked property of the UV light switch will be toggled to true using the attr() JQuery function. Additionally, the script has also been designed to handle on change event where it will send an API request to the “toggle-light” endpoint to alter the current status of the UV light. As a result, the functionality has been successfully implemented, as seen in the figure below.

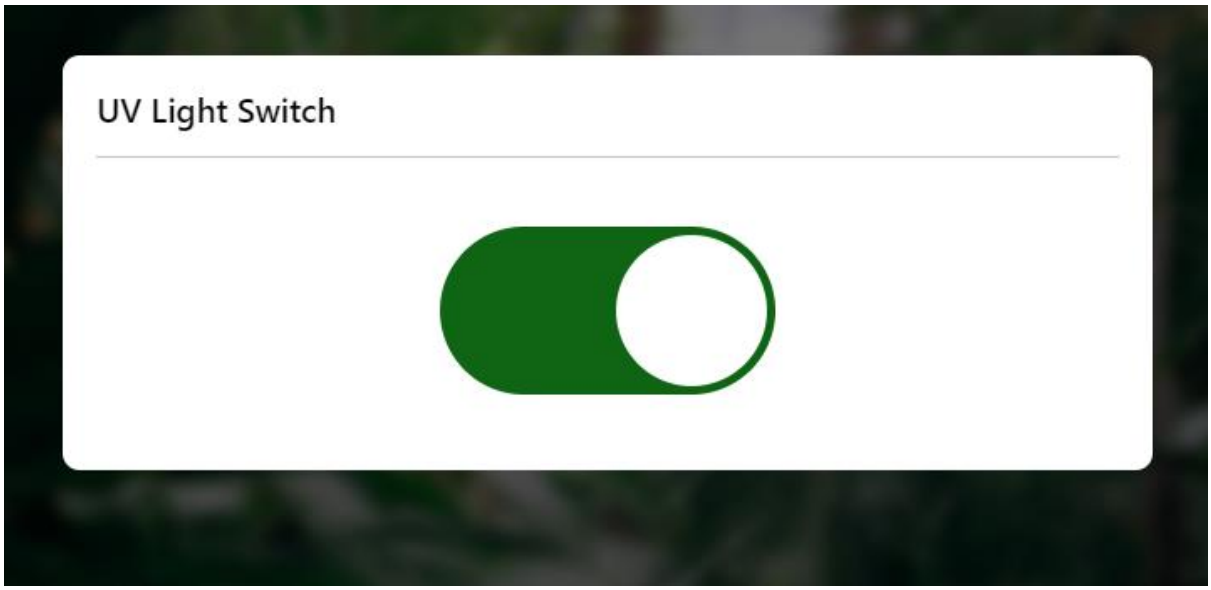



***Figure 46: UV Light Switch on Dashboard***

### 6.5.6 Error Handling

Referring to the non-functional requirement identified in Chapter 3.5, it is clear that error should be appropriately handled to cooperate with the fifth element in the Jakob Nielsen's usability heuristic, Error Prevention (Nielsen, 1994). For that reason, a showError function has been created in the apiresponse class to handle API errors, as seen in the code snippet below, ensuring that an appropriate response code and messages will be returned according to the error codes. For example, if an invalid request has been made to the API, the API will respond with status code 501, as shown in figure 47.

```
    }

    protected function showError($errorCode){
        switch($errorCode){
            case 204:
                http_response_code($errorCode);
                return null;
            case 405:
                http_response_code($errorCode);
                return array("Message" => "Sorry! Method not allowed!");
            case 501:
                http_response_code($errorCode);
                return array("Message" => "Sorry! Request method not found");
            case 400:
                http_response_code($errorCode);
                return array("Message" => "Incorrect parameters");
            case 401:
                http_response_code($errorCode);
                return array("Message" => "Not authorised!");
            case 500:
                http_response_code($errorCode);
                return array("Message" => "Internal Server Error!");
        }
    }
}
```

***Code 31: showError function to display error message***

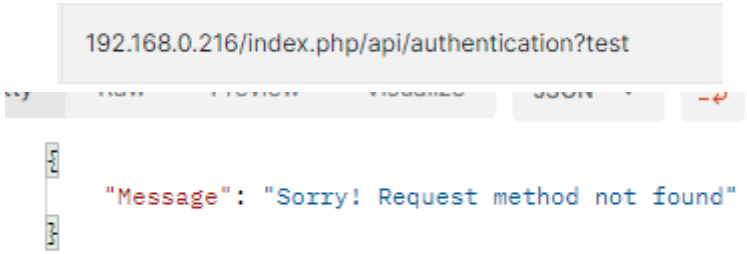



***Figure 47: Invalid Request made to API***

Utilising the status code return from the API, the dashboard will now also be able to display the error accordingly by utilising the statusCode function in AJAX, as shown in an example JavaScript code snippet for login handling where the AJAX statusCode function has been used to retrieve the status code returned by the API and make specific actions depending on the type of status code.

```
e.preventDefault();
$.ajax({
    type: "POST",
    url: 'api/authentication?authenticate',
    data: formData,
    dataType: 'json',
    encode: true,
    async: false,
    statusCode:{
        200: function(){
            window.location.href = "panel";
        },
        204: function(){
            $(errorMessage).html("<p>Incorrect Username/Password. Please try again!</p>");
        },
        401: function(){
            $(errorMessage).html("<p>Incorrect Username/Password. Please try again!</p>");
        },
        500: function(){
            window.location.href = "error";
        }
    }
})
```

***Code 32: JavaScript for Logging In***

However, when dealing with error handling, it is also essential to keep in mind that the non-functional requirement has mentioned that the dashboard should be in a human-readable format to cooperate with the ninth principle in the Jakob Nielsen's usability heuristic, which is to help the user recognise, diagnose, and recover from an error. Hence, the code has ensured that when the API has returned with status code 204 (No Content) or 401 (Unauthorised), an error message "Incorrect Username or Password" will be displayed, allowing the user to understand that they have entered the wrong username or password as shown in the figure below.

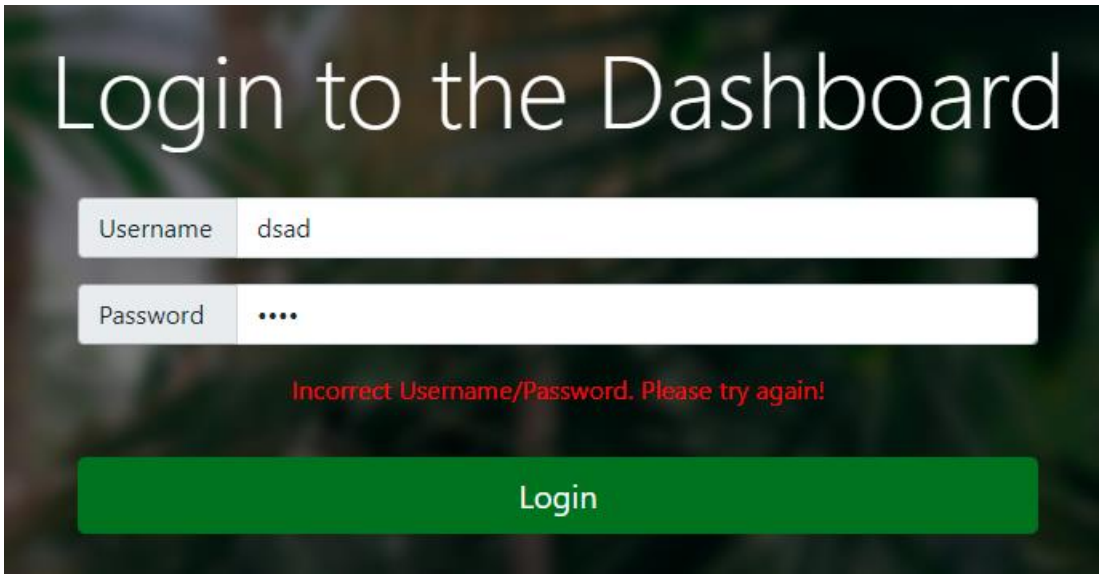



***Figure 48: Incorrect Username/Password. Please try again***

And alternatively, if the API has returned a critical error such as code 500 (Internal Server Error), the user will be redirected to an error page where an appropriate message will be displayed, signifying that an error has occurred on the server-side as shown in the figure below.

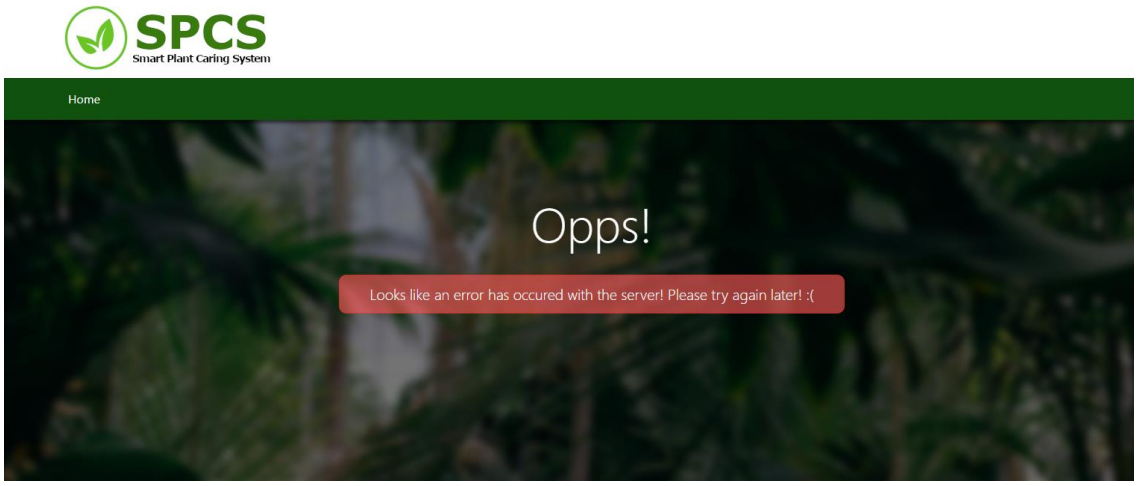



***Figure 49: Error Page with an appropriate error message***

# Chapter 7: Product Testing

This dissertation chapter will cover the testing results using the techniques discussed in Chapter 4.3 and how the error was resolved.

## 7.1 Functional Testing

As previously mentioned, the first phase of the testing will be including a Functional Testing carried out by the developer. Functional testing is a type of testing that ensures implemented functionalities can be executed without any errors. A set of test components are evaluated against an expected outcome during the functional testing. The result of the test case could be categorised as Pass or Fail depending on the outcome of the test case.

By applying the use case testing technique discussed in Chapter 4.3.1, a series of test cases containing number and name of the corresponding use case, a test description, test input, expected outcome, actual outcome and the pass or fail section, was created by referencing to the use case documentation seen in Appendix 10. Figure 50 below shows an example list of the test case. The whole test cases document along with the results can be found in Appendix 16.

| Test Case ID | Use Case Description | Test Description | Test Input | Expected Outcome | Actual Outcome | Pass/Fail |
|---|---|---|---|---|---|---|
| 1 | 1 | Accessing the login screen | Click on 'Dashboard Portal' button | Redirect to login page | Page redirected successfully | Pass |
| 2 | 1 | Username is black | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass |
| 3 | 1 | Password is blank | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass |
| 4 | 1 | Username is incorrect | Username: **test** | Error message "Incorrect Username/Password. Please try again" is displayed | Error message displayed | Pass |
| 5 | 1 | Password is incorrect | Password: **test** | Error message "Incorrect Username/Password. Please try again" is displayed | Error message displayed | Pass |
| 6 | 1 | Correct credentials | Username: **admin** Password: **admin123** | Redirect to dashboard page | Page redirected successfully | Pass |

***Figure 50: Example Test Case***

As discussed in the Functional Requirement in Chapter 3.4, the system has been separated into two sections, the hardware and the dashboard. As a result, two tests will be conducted separately to ensure that all requirements can be executed as expected.

### 7.1.1 Dashboard

Other than the Use Case Testing applied, the functional testing carried out the dashboard will also be including API Testing where the testing will be used to test the functionality of the API implemented on a system using a tool such as Postman. Hence, by combining the two testing techniques, a total of 41 tests has been carried out, and the result has been summarised as seen in the figure below.

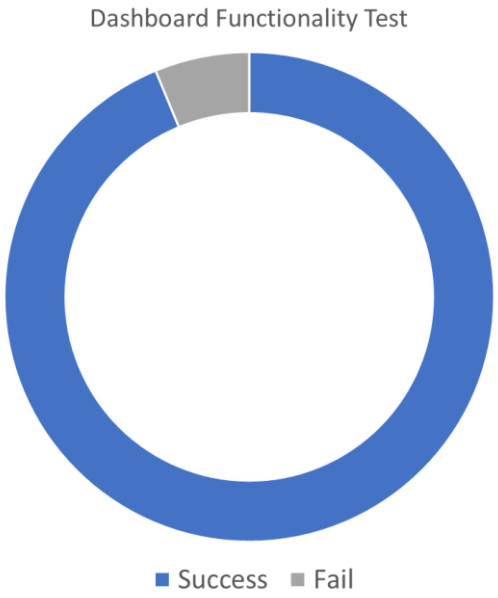



***Figure 51: Dashboard Functionality Test Result***

As a result, out of 41 test case, 39 has passed the test case, whereas two failures have been successfully identified. All failures identified in the dashboard functionality testing phase were corrected before moving on to the hardware functionality testing. The failure correcting process is as shown below.

#### User Accessing the Dashboard without logging in

A failure has been identified when accessing the dashboard using an URL without logging in. The dashboard has failed to validate the user token before displaying the dashboard web page's contents. The expected outcome for this use case assumes that the dashboard should redirect users to an error page if the token is either not presented or expired. As a result, an attempt to resolve this failure has begun by first inspecting the codes for the dashboard page, as seen in the code snippet below:

```
public function __construct($title){
    parent::__construct($title);
    $this->generatePlantPage();
}
```

***Code 33: Constructor function for the Dashboard page***

As the code suggested, the authentication function is missing from the constructor function for the dashboard page, which results in the dashboard class will only generate the page without first validating if the user has logged in. This was corrected by adding the isAuthenticated function implemented in the Session class to check if a user is authenticated before generating the webpage, as seen in the modified code snippet below:

```php
public function __construct($title){
    $session = new Session();
    if($session->isAuthenticated()){
        parent::__construct($title);
        $this->generatePlantPage();
    }else{
        header("Location: error");
    }
}
```

***Code 34: Modified Constructor function for the Dashboard page***

## Plant Information Setting Modal Not Displaying

Another failure has been identified where the plant information setting modal will not display when the user clicks on the "Change Plant Information" button located on the Plant Information panel, as shown in the figure below:

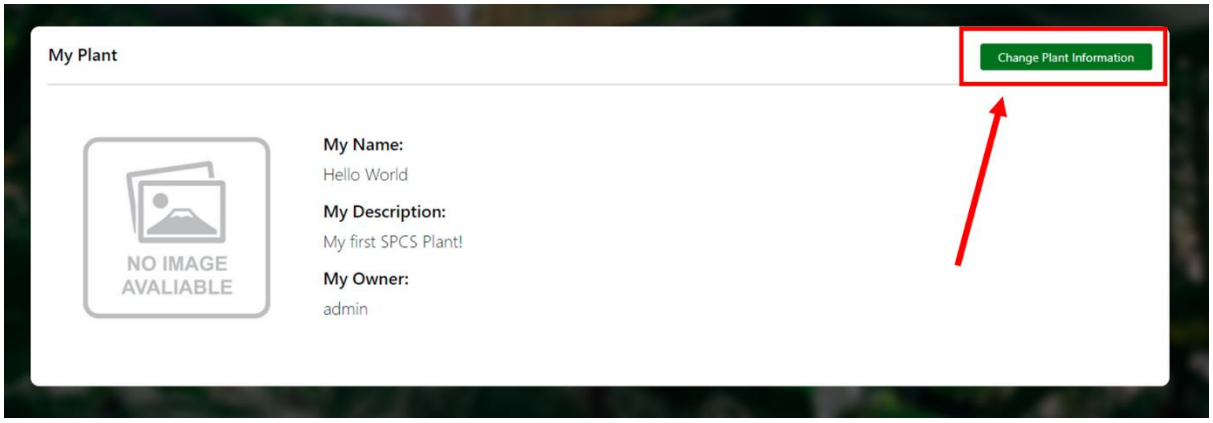



***Figure 52: Plant Information Panel "Change Plant Information" button***

The expected outcome for this use case has assumed that the "Change Plant Information" modal should display upon user click. As a result, an attempt to resolve this failure has begun by inspecting the HTML codes using the Inspect Element feature, as seen in the figure below:

```
<div class="header-button-container">
  <button type="button" class="btn btn-sm setting" data-bs-toggle="modal" data-bs-target="#datailModal">Change Plant Information</button>
  == $0
</div>
```

***Figure 53: "Change Plant Information" HTML codes on Inspect Element***

As the code suggested in the figure, a typo has been spotted for the data-bs-target property on the "Change Plant Information" button HTML tag, failing to display the modal due to the incorrect modal ID provided. This was corrected by simply modifying the ID specified in the data-bs-target property, as seen in the code snippet below:


***Code 35: Modified data-bs-target property***

### 7.1.2 Hardware

After all the failure identified has been successfully resolved on the Dashboard functionality testing. The functionality testing is once again carried out on the system's hardware to verify if the requirement of the hardware implemented is working as expected. Such requirement includes if the sensors are capturing and uploading data to the server properly. However, since it is not possible to make any custom inputs to the hardware system, a slightly different testing approach has been utilised where the data flow between the microcontroller and InfluxDB database is monitored through the InfluxDB interface to ensure that all the data has been successfully uploaded as seen in the figure below.

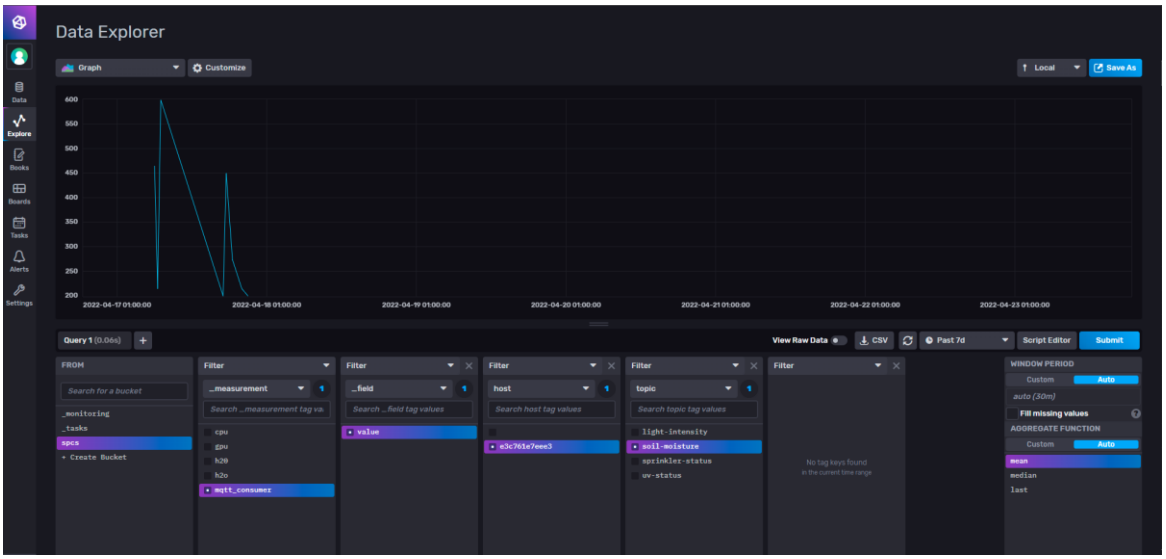



***Figure 54: InfluxDB Interface***

Hence, a total of seven tests has been carried out, and the result has been summarised as seen in the figure below.

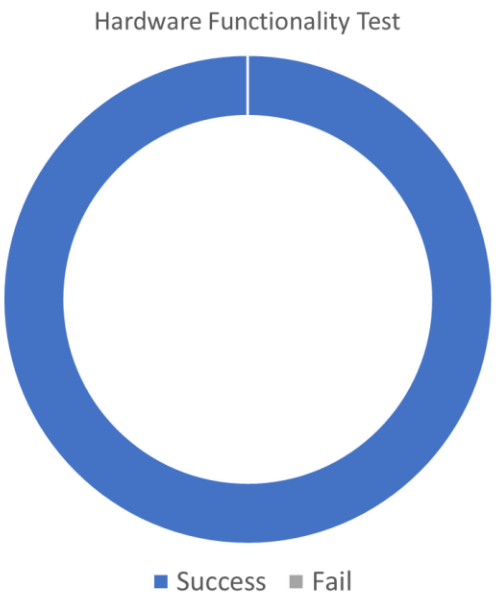



***Figure 55: Hardware Functionality Test Result***

As a result suggested, out of seven test cases, the hardware has passed the test case, whereas no failure has been successfully identified. It has shown that all the collected sensor data have been successfully stored in the database. The actuators installed on the hardware can be toggled depending on the value received.

## 7.2 Cross Browser Testing

As discussed in the testing methodology in Chapter 4.3, the second phase of the testing will be including cross-browser testing on the system dashboard, where the same set of test cases used in the Functionality Testing phase is applied in various browsers such as Safari and Firefox to determine if the system will work as intended across different browser. The form of testing is considered essential in this project as it has shown an exceptional potential in accommodating the seventh principle of the Jakob Nielsen's Usability Heuristics, Flexibility and Efficiency of use, where a system should be responsive, effectively allowing a user to run the system on devices or environment of their choice. However, the test case was slightly modified before the testing could be conducted. Two new rows are added to the Pass and Fail section to fully record if the test cases have been executed accordingly on two different browsers, as shown below.

| Test Case ID | Use Case Description | Test Description | Test Input | Expected Outcome | Actual Outcome | Pass/Fail (Safari) | Pass/Fail (Firefox) |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Accessing the login screen | Click on 'Dashboard Portal' button | Redirect to login page | Page redirected successfully | Pass | Pass |
| 2 | 1 | Username is black | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass | Pass |
| 3 | 1 | Password is blank | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass | Pass |
| 4 | 1 | Username is incorrect | Username: **test** | Error message "Incorrect Username/Password. Please try again" is displayed | Error message displayed | Pass | Pass |
| 5 | 1 | Password is incorrect | Password: **test** | Error message "Incorrect Username/Password. Please try again" is displayed | Error message displayed | Pass | Pass |
| 6 | 1 | Correct credentials | Username: **admin** Password: **admin123** | Redirect to Dashboard page | Page redirected successfully | Pass | Pass |
| 7 | 1 | User is already logged in | Click on 'Dashboard Portal' button | Redirect to Dashboard page | Page redirected successfully | Pass | Pass |

***Figure 56: Example Modified Test Case***

Additionally, test cases made for API testing in the previous testing have also been removed since the cross browser will be focusing more on the system's user interface aspect and less on the backend. The whole test cases document for cross-browser testing along with the results can be found in Appendix 17. As a result, a total of 23 tests have been carried out, and the result has been summarised as seen in the figure below. Please note that Google Chrome is not tested in the Cross Browser Testing phase because it is the primary browser used when implementing the system dashboard.

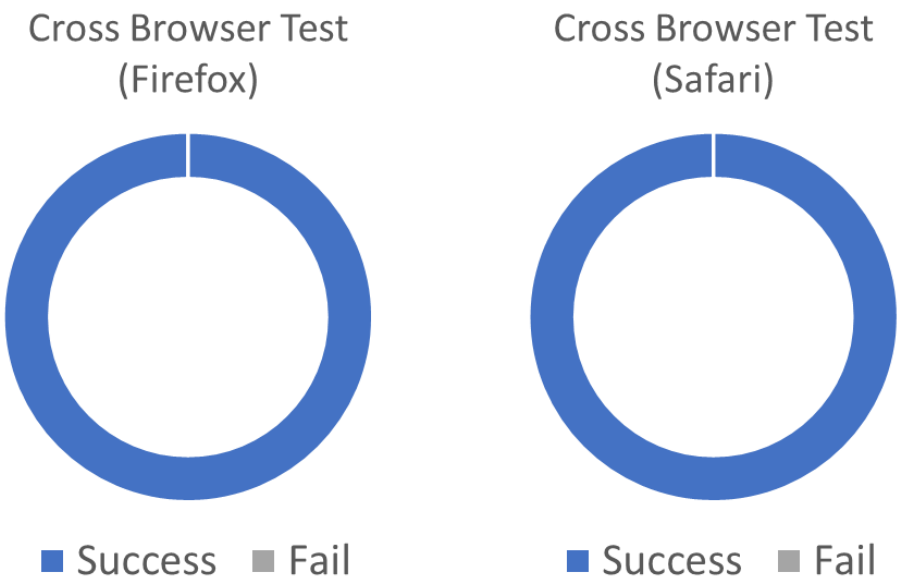



***Figure 57: Cross Browser Testing Result on Firefox and Safari***

As suggested by the cross-browser testing result shown in the figure above, both browsers have successfully performed every test case without running into an error. However, two problems outside of the test cases have been identified when running the dashboard through Safari on iOS devices

where some elements are not displaying. The correcting process of those identified problems is shown below.

## Error Message Visual Bug

The first problem outside of the test case was identified when the system attempted to access the error page through Safari using an iPhone XS on vertical mode. The error message was not displaying correctly where the message was cut off at the edge of the screen, as shown in the figure below:

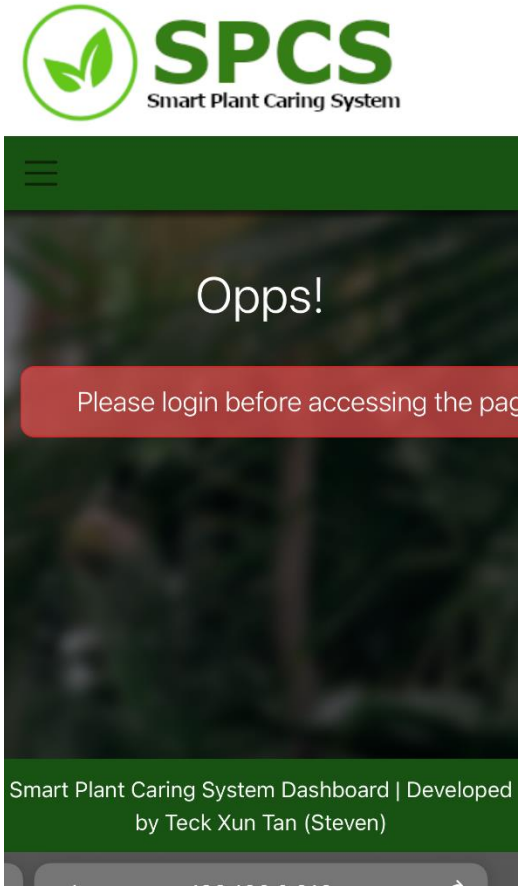



***Figure 58: Error Message Visual Bug on a mobile device***

Although the problem may seem “harmless” to the overall functionality, it may still cause potential issues to the users in the future because they will not be able to read the entire message and understand what should be done to resolve the error, which as a result, violating the ninth principle of the Jakob Nielsen’s usability heuristic as discussed in Chapter 3.5. The correcting process begins by inspecting the CSS code for the error container handler, as seen in the code snippet below.

```
/**Error**/
.error-container{
    border: 1px solid rgba(255, 0, 0, 0.37);
    border-radius: 10px;
    background-color: rgba(248, 83, 83, 0.651);
    margin: 30px auto;
    width:max-content;
}
```

***Code 36: Error Container CSS Code***

As suggested by the code above, the width property has been specified in the error container by accident, causing the container's width to always depend on the child's content, significantly limiting the container's responsiveness. This was corrected by removing the width property from the container, resolving the visual bug as seen in the result below.

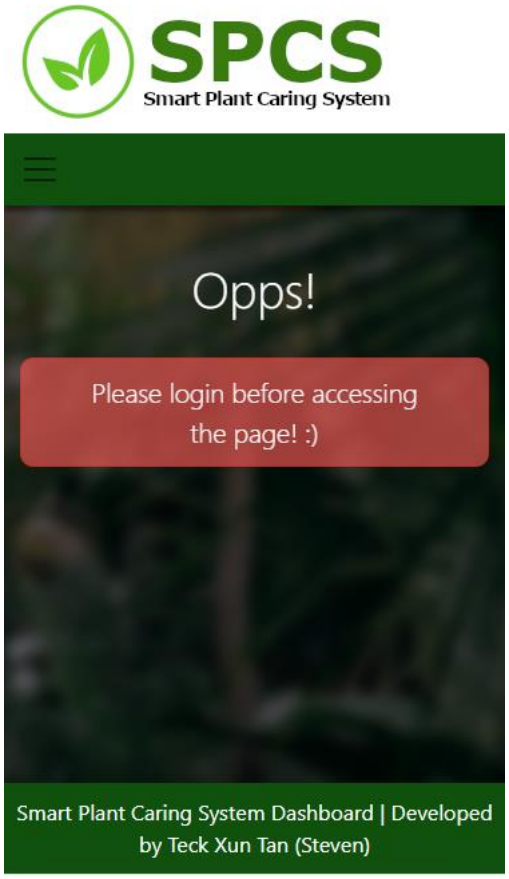



***Figure 59: Visual Bug fixed on a mobile device***

## Background Image Bug on iOS Safari

The second problem outside of the test case was identified when accessing the dashboard page through any iOS device. The background was not working as intended, where the background image would not move according to the scroll input, leaving a massive grey space, as seen in the figure below.


***Figure 60: Background Image Bug on iOS Safari***

Several devices were tested against this problem; however, the problem will only seem to be triggered by iOS devices running on the Safari browser. According to research, this is a common problem on iOS devices because Safari has disabled background-attachment fix property by default to save resources. Hence, the CSS for the background image has been modified to resolve this problem, as seen in the code snippet below:

```
.img-fluid{
    background-image: linear-gradient(rgba(0,0,0,0.5), rgba(0,0,0,0.5)), url("../assets/background.jpg");
    background-position: center;
    background-repeat: no-repeat;
    background-attachment: fixed;
}

.img-fluid{
    background-image: linear-gradient(rgba(0,0,0,0.5), rgba(0,0,0,0.5)), url("../assets/background.jpg");
    background-repeat: no-repeat;
    background-size: cover;
    z-index: -1;
    position: fixed;
    top: 0;
    left: 0;
    height: 100%;
    width: 100%;
}
```

***Code 37: Modified CSS to resolve Background Image Bug***

## 7.3 User Testing

The last phase of the testing will be including User Testing. As seen in the response shown in Appendix 18, the participant mentions in the survey that the system dashboard has been developed excellently given that no visible system flaws have been found throughout the testing process. The test participant continues by giving his feedback on the system's user interface. The participant mentioned that the web dashboard was well-designed. The participant could navigate through the web page and complete most tasks without feeling overwhelmed by the system interface.

The participant also pointed out that he could find every functionality by recognising the length of the container at a glance without requiring much attention. The participant mentions that this is an excellent decision for the user interface design. Users will not need to spend extra time reading every title to locate the function they are finding.

Finally, the participant also mentioned that he had a typo in the username text box during the testing of the login page, causing the system not to recognise the username. However, the participant noticed that an error message has successfully displayed with the appropriate message. The participant commented that the error displayed is useful because it has significantly and effectively helped the participant identify and resolve the error that occurred without any issues.

# Section 4: Evaluation

# Chapter 8: Smart Plant Caring System Evaluation

This chapter of the dissertation will be focused on the evaluation of the product's strengths and weaknesses by utilising findings from product implementation found in Chapter 6 and product testing results found in Chapter 7 as supporting evidence, followed by potential solutions that could have been taken to overcome the identified problems and finally a brief evaluation of the choice of tools selected in the project.

## 8.1 Product Strengths

### 8.1.1 Fulfilling the Functional Requirements

As discussed in Chapter 3.4, a set of functional requirements required to be implemented into a system to enable users to carry out specific tasks have been developed, as seen in Appendix 6. A robust requirement capturing method in literature reviewing and conducting an online survey has been applied to identify the system's functional requirements. According to the researcher Liao and Hsieh (2017), online-based surveys have shown more potential in collecting reliable, quality, and valid data compared to paper-based surveys. Hence, adopting an online survey in the project has significantly ensured that the requirements gathered from the user perspective are high quality and should be investigated in the project is targeting to address the project's aims as the top priority.

| Requirement | Priority | Implemented? |
|---|---|---|
| User should be able to register for a new account | Would | No |
| User should be able to manage (add,edit,remove) their plant | Would | No |
| User should be able to view the raw data | Should | Yes |
| User should be able to turn on UV light through dashboard | Should | Yes |
| User should be able to turn on watering process through dashboard | Would | No |
| User should be able to view the sensor captured data through dashboard | Must | Yes |
| System should be able to display weather forecast | Should | Yes |
| User should be able to customise the plant information | Should | Yes |

| Requirement | Priority | Implemented? |
|---|---|---|
| System should be able to capture soil moisture level with sensor | Must | Yes |
| System should be able to capture room brightness level with sensor | Should | Yes |
| System should be able to capture humidity with sensor | Could | Yes |
| System should be able to capture temperature with sensor | Could | Yes |
| System should be able to turn on UV light on manually | Should | Yes |
| System should be able to turn on plant watering process manually | Would | No |

***Table 22: Functional Requirements Implemented in the System***

As seen in the table shown above, the prioritisation method, MoSCoW, adopted in this project has also demonstrated that it is capable of significantly accelerating the process of fulfilling the functional requirements by planning and prioritising each of the requirements in the most efficient sequence, hence providing a clear image on which requirements should be implemented as the top priority without being distracted by requirements at the lower priority. For example, from the table above, the requirements that have been categorised under the Must and Should category have been implemented into the system because they are considered part of the core requirements. In contrast, the requirements listed in the Would category have been ignored and not implemented within the given time constraints because these requirements are primarily additional functionality that does not directly affect the overall system.

Additionally, the product testing carried out in Chapters 7.1 and 7.2 has further proven that the requirements have been implemented appropriately where most of the requirements have been met with minimum to no errors, making the system more stable and enhancing the overall usability of the system. Moreover, the testing has also ensured that users may interact with the functionalities implemented in the system will have a fantastic user experience without worrying about any potential errors with the system.

### 8.1.2 Positive User Feedback on User Interface

Based on the findings seen in Chapter 7.3, one of the system testing participants has provided excellent feedback on the implemented system user interface. The participant has mentioned that he can navigate through the webpage without any problems and he could also complete most of the tasks without feeling overwhelmed by the system interface. The user feedback collected has proven that the system has successfully utilised Jakob Nielsen's 10 User Heuristic, as discussed in Chapter 3.5, to enhance the user experience. For example, the system has successfully fulfilled the eighth principle in the user heuristics mentioned in Chapter 5.6.2. The interfaces should be keeping a minimalist design to ensure that users will not be confused or distracted by elements that are not essential to the system.

Another example is that the participant mentioned that the error shown on the login screen was helpful. He was able to identify the source of the error and resolve it within a short time. This has been demonstrated that the system has successfully integrated and fulfilled the fifth and ninth principles of the user heuristics mentioned in Chapter 6.4.6 Error Handling, where an error has been successfully caught and prevented. An appropriate error message has also been displayed, effectively assisting the users in identifying and resolving the error.

Additionally, participant feedback has proven that the pre-attentive processing visual trick, as discussed in Chapter 2.4.4 and applied in Chapter 5.6, is considerably effective. The participant pointed out that he could locate an object in the shortest amount of time without requiring much attention.

As a result, the feedback from the participant has proven that the system has been designed and implemented appropriately from the user perspective. It has also shown that the techniques and principles applied, such as the pre-attentive processing technique and Jakob Nielsen's User Heuristic, have been appropriate and significantly enhanced the user experience.

## 8.2 Product Weakness

### 8.2.1 Real-Time Data Handling

In Chapter 2.4.3, the project has planned to investigate the use of an operational dashboard. As discussed in the literature review, an operation dashboard should show be capable of displaying collected data in a real-time environment that will significantly assist users in making a rapid decision by analysing and learning the data collected in real-time. However, the system implemented in this project has shown to be only capable of collecting and displaying data in near real-time instead of in real-time. There is a noticeable delay between the data collected by the sensors with the data visualised on the dashboard.

Real-time data refers to data captured and presented when it has been acquired, which means that the data captured are required to be exchanged in a timely manner between systems (Shin and Ramanathan, 1994). To handle data in real-time, high-performance hardware and high-speed internet connection is typically required to ensure that data can be transferred between the system with minimum to no latency (SentinelOne, 2021). However, real-time data is almost certainly not realisable or challenging to be implemented in this project due to the hardware, such as the ESP8266 microcontroller selected in Chapter 4.2.2, does not hold the sufficient processing power in handling and uploading real-time data generally due to the low specification single-core processor integrated on the selected microcontroller.

The problem identified above has shown that there is also a potential weakness in the microcontroller review chapter (Chapter 4.2.2), where the specification of the microcontroller has not been sufficiently reviewed and compared, consequently causing the hardware selected to have lower specifications than initially expected.

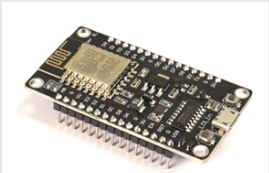
| Microcontroller | NodeMCU ESP8266 Module | NodeMCU ESP32 Module |
|---|---|---|
| Image |  |  |
| Performance | Single Core Processor (80MHz) | Dual-Core Processor (160MHz – 240MHz) |
| Available GPIO(s) Slot | 13 Digital 1 Analog | Over 24 pins with 6 Analog to Digital Converter (ADC) pins |
| Wi-Fi Connectivity | Yes | Yes |

***Table 23: ESP8266 vs ESP32 Microcontroller***

In conclusion, the potential solution to resolving the identified problem is to integrate the system hardware with a higher processing power microcontroller, such as the ESP32 module integrated with a higher specification processor compared to the ESP8266 module, as shown in the figure above. The integration of the ESP32 module will significantly ensure that the system will have more excellent

capability to handle real-time data, further proving the benefits and utilising the full potential of an operational dashboard.

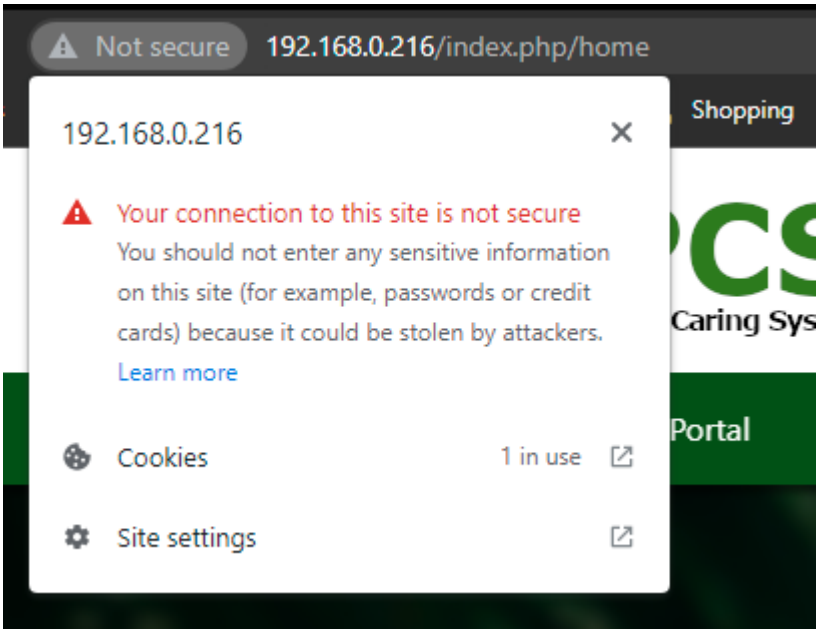
### 8.2.2 Unsecured Webpage Limitation

An unsecured webpage has been identified as one of the weaknesses of the project because it has brought several limitations to the implementation of a functional requirement requested by Participant 2 in Chapter 3.2.5. Participant 2 has suggested that the system should have a functionality where the dashboard will display the information of the current weather forecast of their current location. Despite that the proposed functionality has been successfully implemented into the system, as seen in Chapter 6.4.3, the functionality is not using the user location; instead, it uses a fixed location, as shown in the code below:

```
function retrieveWeatherData(){
    $.ajax({
        url: 'https://api.openweathermap.org/data/2.5/weather?lat=54.9768729&lon=-1.6077272&appid=eaf7333227d25dee9d935ceb06b9bdfa
        async:false,
```

***Code 38: Weather Forecast Function Using Static Location***

The decision was made because retrieving user location is nearly unrealisable due to the limitations of an unsecured web page on browsers such as Google Chrome. Research conducted based on this problem has found that the user location is not retrievable due to Google Chrome has a few security policies implemented into the browser in targeting to protect user's privacies by automatically blocking an unsecured webpage from accessing sensitive data such as the user's location as seen in the figure below.



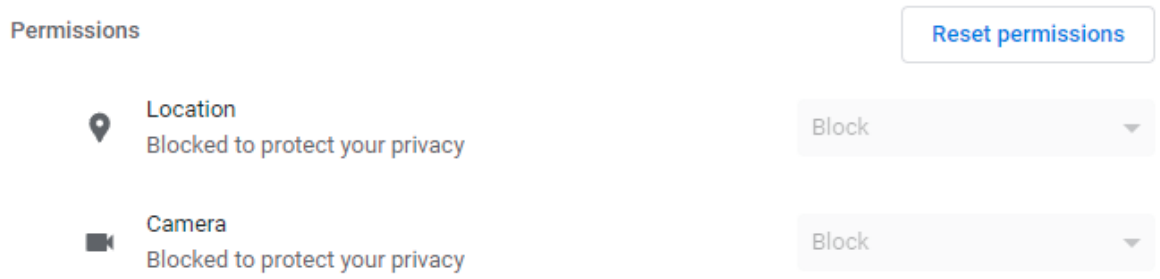



***Figure 61: Unsecured Webpage Limitation***

A potential solution for the problem identified is to enable an encrypted connection using an SSL digital certificate. An SSL, a Secured Socket Layer, is a digital certificate utilised to encrypt the web browser and server communication. The SSL certificate can ensure that the connection is protected from various cyberattacks such as the Man-in-the-Middle (Gupta, 2020). In conclusion, by establishing an encrypted connection using an SSL certificate on the system dashboard webpage, the limitation identified on the unsecured webpage will be lifted, and it will also significantly enhance a users' security, ensuring that their sensitive data can be further protected.

### 8.2.3 Local Connection Limitation

The subsequent weakness identified in the system is the local connection limitation. Due to the system being a proof of concept prototype, the system web dashboard was implemented and hosted on a local machine. It has caused specific limitations where the system is only accessible by connecting to the local network.

The identified weakness is not considered critical because it does not directly affect the system's overall functionality. However, the weakness has caused a significant impact on the user testing phase of the project. As seen in the test result shown in Chapter 7.3, the project could only retrieve a single response from one participant living in the same household. The weakness has been demonstrated that it is capable of causing the project to retrieve insufficient user feedback because the system is technically impossible to be accessed or adequately tested by an external users if they are not connected to the local network.

In conclusion, a potential solution for the problem identified is to host the dashboard through web hosting services such as Hostinger, GoDaddy and more. Domantas (2022) defines web hosting as an online service that enables developers to upload their web applications onto the internet and publicly make the website available. Hence, if the project is revisited, the author should host the dashboard on a web hosting service to ensure that robust feedback could be collected from more test participants.

## 8.3 Evaluating the Tools Used

As shown in Chapter 4.1, several tools have been selected to create a robust Smart Plant Caring System. The chosen tools have been proven appropriate and valuable given that the system has fulfilled most of the functional requirements as mentioned in Chapter 8.1.1.

A tool such as the Docker has proven to be appropriate in the project because instead of installing tools one at a time using a virtual machine as discussed in Chapter 4.1.1, the Docker Compose tool can

deploy multiple tools simultaneously using only a single YAML configuration file that has significantly accelerated the deployment process as seen in Chapter 6.1.

Next, tools selected, such as the MQTT, Telegraf and InfluxDB, have also been appropriate in the project to upload and store data captured by the sensor part. For example, as seen in Chapter 6.3.1, MQTT has been utilised for subscribing to the data published by the microcontroller. The collected data is then uploaded to InfluxDB using a server agent, Telegraf, as the central data collection daemon service. The example has significantly shown that the selected tools can work with each other appropriately to handle the data storage functionality in the most efficient method.

And finally, web-based programming languages such as JQuery and PHP have been appropriately used in developing both the frontend and backend of the dashboard, respectively. For example, as seen in Chapter 6.4, PHP has been utilised to establish connections to the database, and several API endpoints have been implemented to handle authentication and data retrieving. On the other hand, JQuery has also been utilised in the project to handle HTML manipulation and event handling. Additionally, the JQuery function, such as the AJAX, has also been used to send a request to the implemented API endpoint.

# Chapter 9: Project Process Evaluation

## 9.1 Project Management

### 9.1.1 The Adoption of Waterfall Development Life Cycle

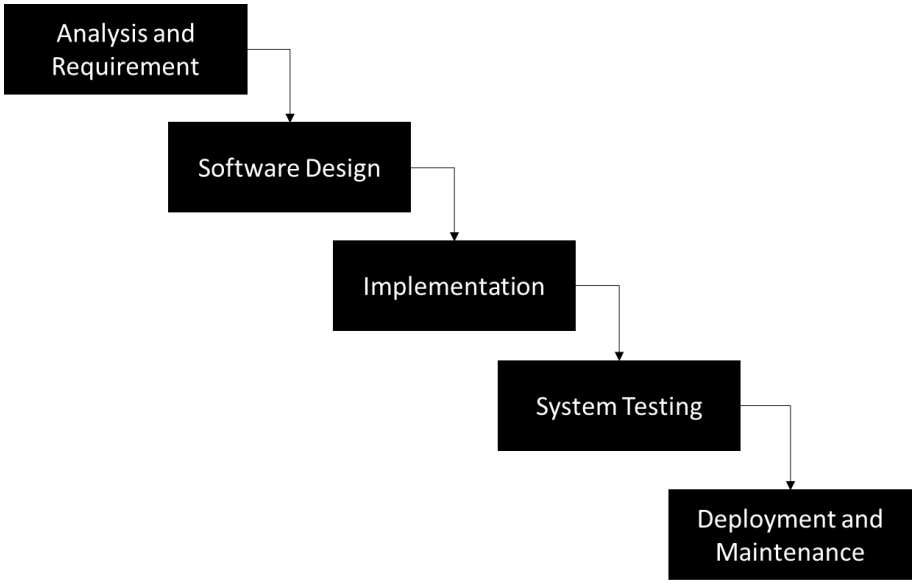



***Figure 62: The Waterfall Development Life Cycle***

As discussed in Chapter 4.4, the project has decided to adopt the waterfall method as the primary development life cycle for this project because it is considered a linear project management process, as seen in the figure above, that is appropriate to be applied in this project.

During the implementation phase of the system carried out in Chapter 6, the author has found a potential weakness to the selected waterfall methodology where it could hold a relatively high risk

and uncertainty if the requirements identified were subject to change or it was found not well-documented during the Implementation or Testing phase. This shows that the methodology will perform poorly when handling changes in requirements and design primarily due to the linear sequential flow nature of the development life cycle, where each phase must be completed before moving on to the next one in a strict order.

However, despite the weakness found in the waterfall development life cycle. The Waterfall model has brought several benefits to the project, including supporting the author in creating a robust development plan. Given that the waterfall model is relatively new to the author during the project timeframe, the author has found the waterfall development cycle to be the most beginner-friendly methodology that individual developers could most certainly adopt. Moreover, the author has also found that the waterfall methodology could be managed effortlessly due to the model's rigidity, where each phase contains a fixed amount of deliverables and recommended review process that could efficiently serve as a guideline to developers on their projects to accelerate the development process of a project. Additionally, the waterfall is a type of development life cycle model that heavily encourages developers to focus on each phase one at a time without being overwhelmed by the process that comes after the current steps, which significantly assists the author in improving productivity and enhancing the quality of the development process.

In conclusion, although the waterfall model has shown potential weakness in handling unforeseen changes, the author still firmly believes that the adoption of the Waterfall development life cycle has been appropriate, given the project has aimed to collect all the requirements at the beginning of the cycle and will only engage with the survey participants during the requirement capturing phase and after the development phase as discussed in Chapter 4.4 which fits the nature of the selected methodology. Moreover, the methods chosen have effectively supported the author in accelerating the development speed, improving productivity, and enhancing the overall development quality as evaluated above.

## 9.1.2 Time Management

The project has consistently cross-referenced with the Gantt Chart created in Section 11 of the Terms of Reference (Appendix 1). Gantt Chart has been seen as the most helpful tool for scheduling and planning the project process to ensure that the project could be carried out according to plan, as seen in the figure below.

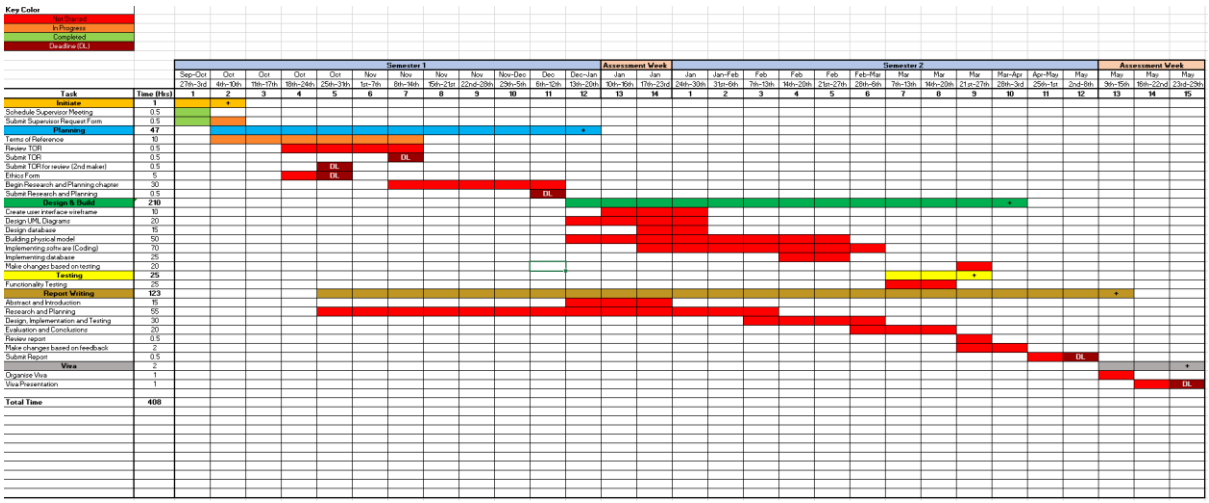



***Figure 63: Project Gantt Chart***

However, the author has encountered issues with time management starting from Semester 2, Week 2. The discrepancies between the author's task and the Gantt chart have become more noticeable during that timeframe because the author has spent a majority amount of time working on the code implementation instead of working on the Research and Planning part of the dissertation, which is initially planned to due for completion on Semester 2 Week 3. The author's supervisor has taken note of the problem and suggested that the author make specific changes in terms of realigning each of the project tasks to fit the current situation. As a result, changes have been made to the original Gantt Chart, as seen in the updated figure below.

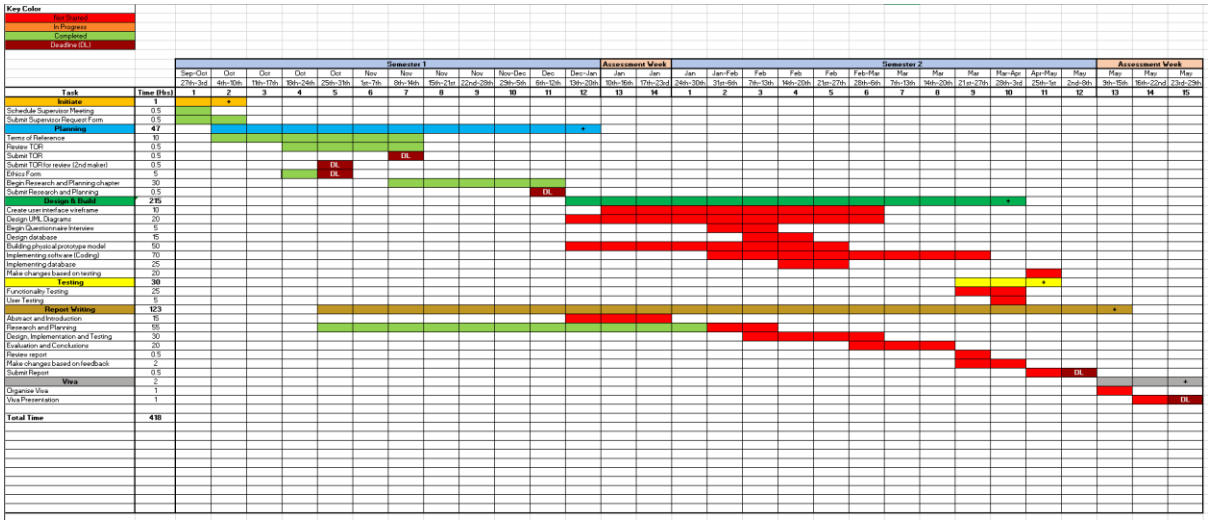



***Figure 64: Updated Project Gantt Chart***

As seen from the changes above, several tasks such as the "Create user interface wireframe", "Design UML Diagrams", and "Implementing software (Coding)" have been given a slightly longer completion time to allow more time to be used to make progression on Report Writing section of the project.

The evaluation above shows that the author lacks skill in time management, which has caused the undertaking project task to have significant discrepancies compared to the Gantt Chart. Suppose the project were to be revisted again, the author will allocate more time towards the report writing section and reduce the time taken for system implementation to ensure that the dissertation could be completed on time.

## 9.2 Project Objectives

A total of 10 objectives have been presented in Section 5 of the Terms of Reference (Appendix 1). 9 out of 10 objectives have been addressed at this point of the project except for a summarisation of the recommendations and future direction of the project, which will be later discussed on Chapter 10. The following table consists of the objectives and how they have been addressed using every aspect of the dissertation as evidence.

| Objectives | How it has been addressed? |
|---|---|
| Conducting literature review | **Chapter 2: Literature Review** of the dissertation has discussed the literature review covering through different aspect of the project including Urban Gardening Technology, Internet of Things (IOT), Data Visualisation and Data Security. The literature review has succeeded in providing the author with a better understanding on the investigating topic and challenges that should be addressed. |
| Creating a requirement specification for both IoT set up and dashboard | **Chapter 3: Requirement Specification** of the dissertation has shown that the requirements have been successfully captured by reviewing to similar project and conducting a survey to collect requirements opinion from the user perspective. The section of the dissertation has significantly assisted the author in understanding what functions are required to be implemented and achieved for both IoT set-up and web dashboard. |
| Develop dashboard and IoT infrastructure design documentation (e.g., class diagram, sequence diagram, database design and more) | **Chapter 5: System Design** of the dissertation has shown that the project has successfully utilised the 4+1 Architectural View Modal as a guideline to design several UML diagrams and database design, followed by the design of the user interface using wireframes. The use of design documentation is considered as essential in this project because it has successfully supported and enhanced the speed and efficiency of the development phase by using these diagrams constructed as a reference. The designed documents could be found in Appendix (number) to (number). |
| Conducting product testing | **Chapter 7: Product Testing** of the dissertation has show that project has successfully conducted product testing using techniques such as Functional Testing, Cross-Browser Testing and User Acceptance Testing along with how error was resolve. Additionally, the chapter has also discussed the used of test plan and the test result produced to ensures that every functionalities implemented in the system are working as intended . |
| Explore and understand which sensors components are required for the project | **Chapter 4.2: Hardware Selection** of the dissertation has successfully addressed this objective by demonstrating the process of researching and identifying the sensors and actuators that should be utilised to fulfil the requirement functionalities. |
| Develop dashboard and IoT infrastructure for data integration from sensors | **Chapter 6.3 Arduino IoT** and **Chapter 6.4 SPCS IoT Dashboard** of the dissertation has successfully addressed the objective by explaining the process of establishing the connection between IoT infrastructure and dashboard to collect and send necessary data from sensors installed on the microcontrollers. |
| Implement the design | **Chapter 6: Product Implementation** of the dissertation has addressed the objective by walking through the steps taken to implement the requirement functionalities identified in **Chapter 3: Requirement Specification** using tools selected in **Chapter 4.1: Tool Selection**. |
| Develop technical skills for IoT implementation | The dissertation has shown that the author has successfully developed technical skills for IoT related system through the literature review found in **Chapter 2.2 Internet of Things** and the development process found in **Chapter 6.3 Arduino IoT**. |
| Develop and evaluate dashboard design with users | **Chapter 7.3: User Acceptance Testing** and **Chapter 8.1.2: Positive User Feedback on User Interface** of the dissertation has successfully addressed the objective by conducting a testing and retrieve user evaluation on the design of the web dashboard to ensure that the non-functionality identified in **Chapter 3.5: Non-Functional Requirements** and the techniques identified in **Chapter 2.4 Data Visualisation** has enhanced the user experience. |

***Table 24: Project Objectives Addressed***

## 9.3 Skill Gained

Through the process of implementing the project. Several skills have been gained from many aspects, including the tools and techniques selected to assist the development of the system. For example, the author has successfully developed the skill to use a tool such as Docker when developing the system. Before this project, the author has never had experience with the Docker tool. However, due to the continuously rising popularity of the tool, the author has decided to investigate the use of Docker. Hence, the author has spent some time researching the tool by viewing a tutorial on web pages such as StackOverflow and YouTube to understand how to implement Docker into the project. At the end of the development, the author has significantly mastered Docker, which has been used in this project to efficiently deploy multiple tools simultaneously, as discussed in Chapter 6.1 and 8.3.

The second skill gained by the author is the skill to utilise the knowledge gained from previous studies or personal experience in a large project. The author thinks this skill is considered essential because the author has been constantly reminded to fully utilise the skills acquired, as identified in Section 6 of the Terms of Reference (Appendix 1), to create a robust IoT system and address potential problems throughout the project. For example, the author developed a fully functional web dashboard using PHP language, as shown in Chapter 6.4, because the author has fully applied the skills and knowledge that he has acquired from the KF6012 Web Application Integration class.

## 9.4 Lessons Learned

The first lesson learned by the author is the importance of proper time management. Time management is considered the essential part of any project because it is the main method utilised to organise and plan specific tasks. As shown in Chapter 9.1.2, the author has lacked time management skills which have caused discrepancies with the initial Gantt Chart created. This has brought significant risk to the project because a project lacking time management will hold a higher chance of causing potential project failure. As a result, the importance of proper time management should always be kept in the author's mind to ensure that time management should be considered a top priority in future projects to ensure that the project could be carried out smoothly.

The second lesson learned by the author is the importance of getting user feedback. As shown in survey requirements captured in Chapter 3.2 and user testing results found in Chapter 7.3, the project has utilised the information and feedback gathered through the process and made changes to the system to ensure that the system will be built according to the user instead of developer perspective. As a result, the author has learned that it is essential to gather opinions and feedback from the end-user, which will significantly enhance the system to be more user-friendly and easier to use by end-users.

# Section 5: Conclusion

# Chapter 10: Conclusion and Recommendation

## 10.1 Conclusion

In conclusion of the literature review shown in Chapter 2 of the dissertation, the literature review has provided a critical review of the problem area to be investigated. A topic such as Urban Gardening Technology has been reviewed appropriately by the author to understand the technologies and techniques developed by researchers worldwide for eliminating the challenges of growing food and plants for self-subsistence in urban areas. Furthermore, topics such as Internet of Things (IoT), Data Visualisation, and Data Security have also been closely reviewed to understand the IoT technology as a whole and hightlight the potential challenges and solutions when attempting to visualise the collected data or ensuring the user's data security. The knowledge gained from conducting the literature review has sufficiently and successfully assisted and provided the author with a better understanding of the research topic and potential challenges that will need to be addressed during the project development.

Followed by the requirement specification chapter shown in Chapter 3 of the dissertation. Several requirements have been successfully captured and concluded during this chapter by reviewing academic papers on existing projects and conducting an online survey with a group of participants who has a specific interest in gardening. The requirement specification phase has also been concluded to be one of the essential steps to the overall project because it can assist the author in further understanding functionalities that are required to be implemented and achieved for both IoT physical prototype set-up and web dashboard.

Furthermore, the dissertation has conducted an in-depth review of the tools and techniques available to the project. The chapter has brought conclusions to the tools and hardware selection section. The appropriate tools and hardwares such as Docker, Mosquitto MQTT Protocol, Telegraf and more have been chosen to assist and accelerate the development of the system. The evaluation of tools found in Chapter 8.3 has also concluded that the tools selected have been proven appropriate and valuable, given that the system has fulfilled most of the functional requirements.

In conclusion to the Synthesis section of the dissertation, the project has completed each development phase by using the Waterfall Development Life Cycle. The system design chapter shown in Chapter 5 has shown the system design process using various UML diagrams such as the class diagram, sequence diagram, database design and more by following the Kruchten's 4+1 Architecture View Model. Followed by the product implementation, as shown in Chapter 6 of the dissertation, it has discussed the process of implementing the product using the tools and hardware selected. All functional requirements have been fulfilled during the process, successfully producing the physical prototype and web dashboard application for the Smart Plant Caring System. As a result, the process has successfully addressed the two aims outlined in Section 5 of the Terms of Reference (Appendix 1) and Chapter 1.2, which the two aims are as shown below:

a) To build a prototype Smart Plant Caring System to prove the concept of plant caring automation

b) To develop a web dashboard application to visualise all the collected data to allow a user to analyse and learn the trends and patterns of the data.

In conclusion to the testing phase, Functional Testing and Cross Browser Testing have been utilised to allow for early detection of potential system errors. During the Functional Testing process, the use case testing technique has been successfully applied to accelerate the process of identifying test cases. The result shown from the testing has concluded that the system has been developed appropriately due to the low error rate shown in the testing result, as seen in Chapter 7.1 and 7.2. Additionally, user testing has also been adopted to get more specific feedback on the system directly from the user's perspective and ensure that the system has reached their expectation.

Evaluation of the system significantly allows the strengths and weaknesses to be identified. The evaluation in this chapter has concluded that the product system has successfully fulfilled all the functional requirements and has received positive user feedback on the user interface of the system. However, the evaluation has also concluded that the system has a few potential weaknesses. The system lacks real-time data handling in the dashboard and the limitations raised by unsecured web pages and local network hosting.

And finally, the evaluation of the project process has concluded that the decision to adopt waterfall as the development life cycle has been appropriate, judging by the model has significantly assisted the author in effectively accelerating the development speed, improving productivity, and enhancing the overall quality of the development. On the other hand, the evaluation has also concluded that the author lacks time management skills during the project. The evaluation is valuable because it allows the author to review the choice made throughout the project's development and significantly assists in critical self-reflection.

## 10.2 Recommendation

There is still a wide range of directions that could be investigated in the system from this point onwards. For future recommendations, the IoT system presented in this project could investigate the use of Artificial Intelligence (AI) to bring more advanced functionalities using machine learning algorithms. Similar to IoT, Artificial Intelligence of Things (AIoT) has recently become a point of discussion among many researchers. An AIoT is a combination of Artificial Intelligence (AI) with the Internet of Things (IoT) which enables the AIoT device to collect data, analyse data and execute the decision without any human intervention (Marr, 2019).

Researchers have already begun to experiment with the concept of AIoT in various agriculture-related systems. For instance, an example research paper written by Debauche et al. (2020) has shown an AIoT agriculture system capable of performing advanced functionalities such as smart irrigation, plant disease, and pests identification using images and data captured by the sensors.

As a result, the finding above has provided strong evidence to make the author believe that the use of AI should be investigated as one of the future recommendations in this project to enable an opportunity to make the future of the Smart Plant Caring System (SPCS) brighter with the support of Artificial Intelligence (AI) algorithms.

# Reference


AHMED, M. 2021. *What is a Microcontroller?* [Online]. Medium. Available at: https://medium.com/nerd-for-tech/what-is-a-microcontroller-547ae100b8e9 (Accessed 7 April 2022).

AKRAM, M. W. 2021. *Advantages and Disadvantages (IOT)* [Online]. Medium. Available at: https://medium.com/@MohamedWasim001/advantages-and-disadvantages-iot-c84b2d4f588a (Accessed 3 December 2021).

ALJOHANI, N. R., DAUD, A., ABBASI, R. A., ALOWIBDI, J. S., BASHERI, M. & ASLAM, M. A. 2019. An integrated framework for course adapted student learning analytics dashboard. *Computers in Human Behavior,* 92(pp. 679-690.

AMAN, M. N., TANEJA, S., SIKDAR, B., CHUA, K. C. & ALIOTO, M. 2018. Token-based security for the internet of things with dynamic energy-quality tradeoff. *IEEE Internet of Things Journal,* 6(2)**,** pp. 2843-2859.

AMARESAN, S. 2019. *The 9 Most Important Survey Design Tips & Best Practices* [Online]. HubSpot. Available at: https://blog.hubspot.com/service/survey-design (Accessed 2 April 2022).

ANAND, A. & UDDIN, A. 2019. Importance of Software Testing in the Process of Software Development. *International Journal for Scientfic Research and Development,* 12(6).

ANKERS, N. n.d. *Agile vs Waterfall: Choosing the Right Methodology for Project Success* [Online]. Overwhelmed PM. Available at: https://www.overwhelmedpm.com/blog/agilevswaterfall (Accessed 11 April 2022).

ATLAM, H. F. & WILLS, G. B. 2020. IoT security, privacy, safety and ethics. *In:* FARSI, M., DANESHKHAH, A., HOSSEINIAN-FAR, A. & JAHANKHANI, H. (eds.) *Digital twin technologies and smart cities.* Cham: Springer International Publishing**,** pp. 123-149.

BAILEY, J. 2019. *What is an IoT Dashboard?* [Online]. Initial State. Available at: https://www.initialstate.com/blog/what-is-an-iot-dashboard/ (Accessed 2 December 2021).

BARATI, M., PETRI, I. & RANA, O. F. Developing GDPR compliant user data policies for Internet of things. Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing, 2019. 133-141.

BELL, D. 2003. UML basics: An introduction to the Unified Modeling Language. *The Rational Edge*.

BRAUN, V., CLARKE, V., BOULTON, E., DAVEY, L. & MCEVOY, C. 2021. The online survey as a qualitative research tool. *International Journal of Social Research Methodology,* 24(6)**,** pp. 641-654.

BREWSTER, C. 2022. *Node.js vs. PHP: Which Is Better For Your Business?* [Online]. Trio. Available at: https://trio.dev/blog/nodejs-vs-php (Accessed 2 February 2022).

BUKHSH, F. A., BUKHSH, Z. A. & DANEVA, M. 2020. A systematic literature review on requirement prioritization techniques and their empirical evaluation. *Computer Standards & Interfaces,* 69(pp. 103389.

CACOVEAN, D., IOANA, I. & NITULESCU, G. 2020. IoT system in diagnosis of Covid-19 patients. *Informatica Economica,* 24(2)**,** pp. 75-89.

CAMPS-CALVET, M., LANGEMEYER, J., CALVET-MIR, L. & GÓMEZ-BAGGETHUN, E. 2016. Ecosystem services provided by urban gardens in Barcelona, Spain: Insights for policy and planning. *Environmental Science & Policy,* 62(pp. 14-23.

CAREY, S. 2021. *What is Docker? The spark for the container revolution* [Online]. Info World. Available at: https://www.infoworld.com/article/3204171/what-is-docker-the-spark-for-the-container-revolution.html (Accessed 29 December 2021).

CHAKRABORTY, J. 2020. *Analog Sensors Vs Digital Sensors* [Online]. Iot Edu. Available at: https://iot4beginners.com/analog-sensors-vs-digital-sensors/ (Accessed 9 April 2022).

CHURILO, C. 2018. *InfluxDB Markedly Outperforms OpenTSDB in Time Series Data & Metrics Benchmark* [Online]. InfluxData. Available at: https://www.influxdata.com/blog/influxdb-markedly-outperforms-opentsdb-in-time-series-data-metrics-

benchmark/#:~:text=InfluxDB%20outperformed%20OpenTSDB%20in%20all,response%20times%20for%20tested%20queries. (Accessed 4 April 2022).
CLANCY, M. 2021. *What's the Diff: VMs vs. Containers* [Online]. Backblaze. Available at: https://www.backblaze.com/blog/vm-vs-containers/ (Accessed 29 December 2021).
CLARK, J. 2016. *What is the Internet of Things (IoT)?* [Online]. IBM. Available at: https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/ (Accessed 20 November 2021).
COMMUNITIES AND LOCAL GOVERNMENT. 2011. *Community orchards How to guide: setting up your own community orchard*, DEPARTMENT FOR COMMUNITIES AND LOCAL GOVERNMENT, United Kingdom. Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/11466/1973262.pdf (Accessed 20 November 2021).
CRANE, C. 2021. *What Is a Device Certificate? Device Certificates Explained* [Online]. hashedout. Available at: https://www.thesslstore.com/blog/what-is-a-device-certificate-device-certificates-explained/ (Accessed 7 December 2021).
D-ROBOTICS. 2010. *DHT11 Humidity & Temperature Sensor* [Online]. D-Robotics UK. Available at: https://datasheetspdf.com/pdf-file/785590/D-Robotics/DHT11/1 (Accessed 10 April 2022).
DAGAR, R., SOM, S. & KHATRI, S. K. Smart farming–IoT in agriculture. 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), 2018. IEEE, 1052-1056.
DAMMAK, M., BOUDIA, O. R. M., MESSOUS, M. A., SENOUCI, S. M. & GRANSART, C. Token-based lightweight authentication to secure IoT networks. 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), 2019. IEEE, 1-4.
DAVID, S. B. 2020. *What is Time Series Database (TSDB)* [Online]. Weka. Available at: https://www.weka.io/learn/what-is-time-series-database-tsdb/ (Accessed 13 April 2022).
DBENGINE. 2022. *DB-Engines Ranking of Time Series DBMS* [Online]. DB-Engines. Available at: https://db-engines.com/en/ranking/time+series+dbms (Accessed 25 January 2022).
DECASTELLARNAU, A. 2018. A classification of response scale characteristics that affect data quality: a literature review. *Quality & quantity,* 52(4)**,** pp. 1523-1559.
DEFRANZO, S. 2022. *Advantages and Disadvantages of Face-to-Face Data Collection* [Online]. Snap Surveys. Available at: https://www.snapsurveys.com/blog/advantages-disadvantages-facetoface-data-collection/ (Accessed 18 March 2022).
DICKSON, R. 2021. *What is an Actuator and How They Work* [Online]. Firgelli Automations. Available at: https://www.firgelliauto.com/en-gb/blogs/actuators/examples-of-actuators-and-how-they-work (Accessed 10 April 2022).
DIX, P. 2022. *What is time series data?* [Online]. influxdata. Available at: https://www.influxdata.com/what-is-time-series-data/ (Accessed 25 January 2022).
DOBSON, M. C., EDMONDSON, J. L. & WARREN, P. H. 2020. Urban food cultivation in the United Kingdom: Quantifying loss of allotment land and identifying potential for restoration. *Landscape and Urban Planning,* 199(pp. 103803.
DOMANTAS. 2022. *What Is Web Hosting – Web Hosting Explained for Beginners* [Online]. Hostinger Tutorials. Available at: https://www.hostinger.co.uk/tutorials/what-is-web-hosting/ (Accessed 2 May 2022).
DUBBELING, M., VAN VEENHUIZEN, R. & HALLIDAY, J. 2019. Urban agriculture as a climate change and disaster risk reduction strategy. *Field Actions Science Reports. The journal of field actions,* Special Issue 20)**,** pp. 32-39.
FAKHRI, D. & MUTIJARSA, K. Secure IoT communication using blockchain technology. 2018 International Symposium on Electronics and Smart Devices (ISESD), 2018. IEEE, 1-6.
GASKIN, J. 2021. *Everything You Need to Know About Use Case Diagrams* [Online]. Venngage. Available at: https://venngage.com/blog/use-case-diagram/ (Accessed 12 April 2022).
GECKOBOARD. 2021. *About Geckoboard* [Online]. Geckoboard. Available at: https://www.geckoboard.com/about/ (Accessed 2 December 2021).

GHAPAR, A. A., YUSSOF, S. & BAKAR, A. A. 2018. Internet of Things (IoT) architecture for flood data management. *Int. J. Future Gener. Commun. Netw,* 11(1)**,** pp. 55-62.

GONZALEZ-HOLLAND, E., WHITMER, D., MORALEZ, L. & MOULOUA, M. Examination of the use of Nielsen's 10 usability heuristics & outlooks for the future. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 2017. SAGE Publications Sage CA: Los Angeles, CA, 1472-1475.

GREGERSEN, C. 2021. *Websocket vs. MQTT vs. CoAP: Which is the Best Protocol?* [Online]. nabto. Available at: https://www.nabto.com/websocket-vs-mqtt-vs-coap/ (Accessed 1 December 2021).

GROCYCLE. 2021. *Urban Farming Ultimate Guide and Examples* [Online]. GroCycle. Available at: https://grocycle.com/urban-farming/ (Accessed 15 October 2021).

GUPTA, A., CHRISTIE, R. & MANJULA, P. 2017. Scalability in internet of things: features, techniques and research challenges. *Int. J. Comput. Intell. Res,* 13(7)**,** pp. 1617-1627.

GUPTA, N. 2020. *How SSL works?* [Online]. Medium. Available at: https://namangupta01.medium.com/how-ssl-works-23d8e5ed0cfa (Accessed 29 December 2022).

HAMILL, H. 2014. *Interview methodology*, Oxford University Press.

HAMILTON, T. 2022. *What is Functional Testing? Types & Examples (Complete Tutorial)* [Online]. Guru99. Available at: https://www.guru99.com/functional-testing.html (Accessed 10 April 2022).

HARLEY, A. 2016. *Visual Indicators to Differentiate Items in a List* [Online]. Nielsen Norman Group. Available at: https://www.nngroup.com/articles/visual-indicators-differentiators/ (Accessed 2 December 2021).

HÖGLUND, J., LINDEMER, S., FURUHED, M. & RAZA, S. 2020. PKI4IoT: Towards public key infrastructure for the Internet of Things. *Computers & Security,* 89(pp. 101658.

HONEYWELL INTERNATIONAL INC. 2008. *HIH-4030/31 Series* [Online]. Honeywell International Inc. Available at: https://www.sparkfun.com/datasheets/Sensors/Weather/SEN-09569-HIH-4030-datasheet.pdf (Accessed 10 April 2022).

HOORY, L. & BOTTORFF, C. 2022. *Agile vs. Waterfall: Which Project Management Methodology Is Best For You?* [Online]. Forbes Advisor. Available at: https://www.forbes.com/advisor/business/agile-vs-waterfall-methodology/ (Accessed 11 April 2022).

HOTJAR. 2022. *Usability testing: your 101 introduction* [Online]. hotjar. Available at: https://www.hotjar.com/usability-testing/ (Accessed 12 April 2022).

HOWARD, C. 2019. *Advantages and Disadvantages of Online Surveys* [Online]. cvent. Available at: https://www.cvent.com/en/blog/events/advantages-disadvantages-online-surveys (Accessed 18 March 2022).

HUDAIB, A., MASADEH, R., QASEM, M. H. & ALZAQEBAH, A. 2018. Requirements prioritization techniques comparison. *Modern Applied Science,* 12(2)**,** pp. 62.

IHSAN, S. N. & KADIR, T. A. A. 2018. Adoption of Software Development Life Cycle (SDLC) Model in Games Development Framework for Serious Games Applications. *Advanced Science Letters,* 24(10)**,** pp. 7300-7304.

JABRAEIL JAMALI, M. A., BAHRAMI, B., HEIDARI, A., ALLAHVERDIZADEH, P. & NOROUZI, F. 2020. IoT Architecture. *Towards the Internet of Things: Architectures, Security, and Applications.* Cham: Springer International Publishing**,** pp. 9-31.

JANES, A., SILLITTI, A. & SUCCI, G. 2013. Effective dashboard design. *Cutter IT Journal,* 26(1)**,** pp. 17-24.

JÁNOKY, L. V., LEVENDOVSZKY, J. & EKLER, P. 2018. An analysis on the revoking mechanisms for JSON Web Tokens. *International Journal of Distributed Sensor Networks,* 14(9)**,** pp. 1550147718801535.

JANSSEN, T. 2020. *SOLID Design Principles Explained: The Single Responsibility Principle* [Online]. Stackify. Available at: https://stackify.com/solid-design-principles/ (Accessed 12 April 2022).
JAYAWARDENE, P. D. 2021. *4+1 Architectural view model in Software* [Online]. Medium. Available at: https://medium.com/javarevisited/4-1-architectural-view-model-in-software-ec407bf27258 (Accessed 12 April 2022).
JENN, N. C. 2006. Designing a questionnaire. *Malaysian family physician: the official journal of the Academy of Family Physicians of Malaysia,* 1(1)**,** pp. 32.
KAPLAN, K. 2021. *10 Usability Heuristics Applied to Complex Applications* [Online]. Nielson Norman Group. Available at: https://www.nngroup.com/articles/usability-heuristics-complex-applications/ (Accessed 3 April 2022).
KHAN, M. A. & SALAH, K. 2018. IoT security: Review, blockchain solutions, and open challenges. *Future generation computer systems,* 82(pp. 395-411.
KODALI, R. K., JAIN, V., BOSE, S. & BOPPANA, L. IoT based smart security and home automation system. 2016 international conference on computing, communication and automation (ICCCA), 2016. IEEE, 1286-1289.
KRUCHTEN, P. B. 1995. The 4+ 1 view model of architecture. *IEEE software,* 12(6)**,** pp. 42-50.
KSHETRI, N. 2017. Can blockchain strengthen the internet of things? *IT professional,* 19(4)**,** pp. 68-72.
LANGELLOTTO, G. A., MELATHOPOULOS, A., MESSER, I., ANDERSON, A., MCCLINTOCK, N. & COSTNER, L. 2018. Garden pollinators and the potential for ecosystem service flow to urban and peri-urban agriculture. *Sustainability,* 10(6)**,** pp. 2047.
LAUBHEIMER, P. 2017. *Dashboards: Making Charts and Graphs Easier to Understand* [Online]. Nielsen Norman Group. Available at: https://www.nngroup.com/articles/dashboards-preattentive/ (Accessed 2 December 2021).
LIAO, P.-W. & HSIEH, J.-Y. 2017. Does Internet-Based Survey Have More Stable and Unbiased Results than Paper-and-Pencil Survey? *Open Journal of Social Sciences,* 5(01)**,** pp. 69.
LIN, B. B., PHILPOTT, S. M. & JHA, S. 2015. The future of urban agriculture and biodiversity-ecosystem services: Challenges and next steps. *Basic and applied ecology,* 16(3)**,** pp. 189-201.
LIN, J., SHEN, Z., ZHANG, A. & CHAI, Y. Blockchain and IoT based food traceability for smart agriculture. Proceedings of the 3rd International Conference on Crowd Science and Engineering, 2018. 1-6.
LIN, J., YU, W., ZHANG, N., YANG, X., ZHANG, H. & ZHAO, W. 2017. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE internet of things journal,* 4(5)**,** pp. 1125-1142.
LOMBARDI, M., PASCALE, F. & SANTANIELLO, D. 2021. Internet of Things: A General Overview between Architectures, Protocols and Applications. *Information,* 12(2)**,** pp. 87.
LUTKEVICH, B. 2019. *microcontroller (MCU)* [Online]. TechTarget. Available at: https://www.techtarget.com/iotagenda/definition/microcontroller (Accessed 7 April 2022).
MAHADEVASWAMY, U. 2018. Automatic IoT based plant monitoring and watering system using Raspberry Pi. *International Journal of Engineering and Manufacturing,* 8(6)**,** pp. 55.
MARR, B. 2019. *What Is The Artificial Intelligence Of Things? When AI Meets IoT* [Online]. Forbes. Available at: https://www.forbes.com/sites/bernardmarr/2019/12/20/what-is-the-artificial-intelligence-of-things-when-ai-meets-iot/?sh=a81b9ddb1fd0 (Accessed 29 April 2022).
MARTIN, M. 2022. *What is a Functional Requirement in Software Engineering? Specification, Types, Examples* [Online]. Guru99. Available at: https://www.guru99.com/functional-requirement-specification-example.html (Accessed 4 April 2022).
MARTIN, R. C. 2000. Design principles and design patterns. *Object Mentor,* 1(34)**,** pp. 597.
MAYUREE, M., AISHWARYA, P. & BAGUBALI, A. Automatic plant watering system. 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), 2019. IEEE, 1-3.

MBAABU, O. 2021. *Understanding the Software Development Life Cycle* [Online]. Section. Available at: https://www.section.io/engineering-education/understanding-software-development-life-cycle/ (Accessed 11 April 2022).
MCCOMBES, S. 2019. *Sampling Methods | Types and Techniques Explained* [Online]. Scribbr. Available at: https://www.scribbr.com/methodology/sampling-methods/ (Accessed 18 March 2022).
MELNICK, J. 2018. *Top 10 Most Common Types of Cyber Attacks* [Online]. netwrix. Available at: https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/ (Accessed 3 December 2021).
MOSS, D. 2021. *Can 1% of Singapore's Land Feed Its Population?* [Online]. Bloomberg Opinion. Available at: https://www.bloomberg.com/opinion/articles/2021-04-29/singapore-s-investment-in-urban-farming-isn-t-just-trendy (Accessed 12 October 2021).
NAQVI, S. N. Z., YFANTIDOU, S. & ZIMÁNYI, E. 2017. Time series databases and influxdb. *Studienarbeit, Université Libre de Bruxelles,* 12(
NETSIL. 2017. *Collector Comparison: Telegraf vs Collectd vs DD-agent* [Online]. Medium. Available at: https://blog.netsil.com/collector-comparison-telegraf-vs-collectd-vs-dd-agent-f49c866657b0 (Accessed 10 April 2022).
NEYJA, M., MUMTAZ, S., HUQ, K. M. S., BUSARI, S. A., RODRIGUEZ, J. & ZHOU, Z. An IoT-based e-health monitoring system using ECG signal. GLOBECOM 2017-2017 IEEE Global Communications Conference, 2017. IEEE, 1-6.
NIELSEN, J. Enhancing the explanatory power of usability heuristics. Proceedings of the SIGCHI conference on Human Factors in Computing Systems, 1994. 152-158.
NOGEIRE-MCRAE, T., RYAN, E. P., JABLONSKI, B. B. R., CAROLAN, M., ARATHI, H. S., BROWN, C. S., SAKI, H. H., MCKEEN, S., LAPANSKY, E. & SCHIPANSKI, M. E. 2018. The Role of Urban Agriculture in a Secure, Healthy, and Sustainable Food System. *BioScience,* 68(10)**,** pp. 748-759.
O'BRIEN, P., YOUNG, S. W., ARLITSCH, K. & BENEDICT, K. 2018. Protecting privacy on the web: a study of HTTPS and Google Analytics implementation in academic library websites. *Online Information Review*.
OKTA. 2022. *What is Token-Based Authentication?* [Online]. okta. Available at: https://www.okta.com/identity-101/what-is-token-based-authentication/ (Accessed 2 April 2022).
OPENWEATHER. 2022. *OpenWeather* [Online]. OpenWeather. Available at: https://openweathermap.org/ (Accessed 13 April 2022).
PANIGRAHI, S. S., SHAURYA, S., DAS, P., SWAIN, A. K. & JENA, A. K. Test scenarios generation using UML sequence diagram. 2018 International Conference on Information Technology (ICIT), 2018. IEEE, 50-56.
PEDAMKAR, P. 2021. *IoT Features* [Online]. Educba. Available at: https://www.educba.com/iot-features/ (Accessed 3 December 2021).
PEDAMKAR, P. n.d. *MQTT vs Websocket* [Online]. Educba. Available at: https://www.educba.com/mqtt-vs-websocket/ (Accessed 5 April 2022).
PRAVIN, A., JACOB, T. P. & ASHA, P. 2018. Enhancement of plant monitoring using IoT. *International Journal of Engineering and Technology (UAE),* 7(3)**,** pp. 53-55.
PREZOTO, F., MACIEL, T. T., DETONI, M., MAYORQUIN, A. Z. & BARBOSA, B. C. 2019. Pest control potential of social wasps in small farms and urban gardens. *Insects,* 10(7)**,** pp. 192.
RAMÓN, O. S., MOLINA, J. G., CUADRADO, J. S. & VANDERDONCKT, J. GUI Generation from Wireframes. 14th Int. Conference on Human-Computer Interaction Interaccion'2013, 2013.
ROBINSON, S. 2021. *Common application layer protocols in IoT explained* [Online]. IoT Agenda. Available at: https://internetofthingsagenda.techtarget.com/feature/Common-application-layer-protocols-in-IoT-explained (Accessed 3 December 2021).
ROORKEE, I. 2017. *Actuators and Sensors* [Online]. Medium. Available at: https://medium.com/@ariesiitr/actuators-and-sensors-246eef0badd9#:~:text=Actuators%20are%20something%20that%20control%20or%20move

%20things%20around%20in,electric%20or%20even%20mechanical%20signal. (Accessed 10 April 2022).
SABEH, N. 2020. Rooftop plant production systems in urban areas. *Plant factory.* Elsevier**,** pp. 129-135.
SANTOS, A. M., PAIVA, L. O., REIS, J. R., SILVA, N. E., SANTOS, A. E., LUIZ, D. F., CARVALHO, C. B. & JÚNIOR, W. S. Smart Garden Monitoring and Irrigation System in Multiplatform Application Using IoT. 2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), 2021. IEEE, 1-2.
SATAMRAJU, K. P. 2020. Proof of concept of scalable integration of internet of things and blockchain in healthcare. *Sensors,* 20(5)**,** pp. 1389.
SCHEDLBAUER, M. 2011. *Requirements Prioritization Strategies* [Online]. Available at: https://www.projecttimes.com/articles/requirements-prioritization-strategies/ (Accessed 21 March 2022).
SCHILLINGER, F. 2021. *Methodical prioritization with the MoSCoW method* [Online]. Medium. Available at: medium.com/devlix-blog/methodical-prioritization-with-the-moscow-method-6d390f0d740f (Accessed 21 March 2022).
SENTINELONE. 2021. *Real-Time Processing: Difference & (Dis)Advantage Over Batches* [Online]. SentinelOne. Available at: https://www.sentinelone.com/blog/real-time-processing/ (Accessed 1 May 2022).
SHANTIKUMAR, S. & BARRATT, H. 2018. *Methods of sampling from a population* [Online]. Health Knowledge. Available at: https://www.healthknowledge.org.uk/public-health-textbook/research-methods/1a-epidemiology/methods-of-sampling-population (Accessed 18 March 2022).
SHIN, K. G. & RAMANATHAN, P. 1994. Real-time computing: A new discipline of computer science and engineering. *Proceedings of the IEEE,* 82(1)**,** pp. 6-24.
SINGH, S., RA, I.-H., MENG, W., KAUR, M. & CHO, G. H. 2019. SH-BlockCC: A secure and efficient Internet of things smart home architecture based on cloud computing and blockchain technology. *International Journal of Distributed Sensor Networks,* 15(4)**,** pp. 1550147719844159.
SLUTSKY, D. J. 2014. The effective use of graphs. *Journal of wrist surgery,* 3(02)**,** pp. 067-068.
SOGA, M., COX, D. T., YAMAURA, Y., GASTON, K. J., KURISU, K. & HANAKI, K. 2017. Health benefits of urban allotment gardening: Improved physical and psychological well-being and social integration. *International journal of environmental research and public health,* 14(1)**,** pp. 71.
SONI, D. & MAKWANA, A. A survey on mqtt: a protocol of internet of things (iot). International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017), 2017.
STEWART, L. 2021. *Front End Development vs Back End Development: Where to Start?* [Online]. Course Report. Available at: https://www.coursereport.com/blog/front-end-development-vs-back-end-development-where-to-start (Accessed 10 April 2022).
SUCCESSIVETECH. 2021. *Cross Browser Testing* [Online]. Medium. Available at: https://medium.com/@successivetech/cross-browser-testing-edb06801061c#:~:text=Cross%20Browser%20Testing-,What%20Is%20Cross%20Browser%20Testing%3F,application's%20compatibility%20with%20different%20browsers. (Accessed 10 April 2022).
SUKHDEV, N., NAHATA, N., SRIDHARA, S. & SWAMY, G. IoT enabled smart gardening. 2018 Fourteenth International Conference on Information Processing (ICINPRO), 2018. IEEE, 1-2.
TAM, K. C. & BONEBRAKE, T. C. 2016. Butterfly diversity, habitat and vegetation usage in Hong Kong urban parks. *Urban ecosystems,* 19(2)**,** pp. 721-733.
TAN, L. H. & NEO, H. 2009. "Community in Bloom": local participation of community gardens in urban Singapore. *Local Environment,* 14(6)**,** pp. 529-539.

TEJA, R. 2021. *What is a Sensor? Different Types of Sensors and their Applications* [Online]. Electronics Hub. Available at: https://www.electronicshub.org/different-types-sensors/ (Accessed 8 April 2022).

TEXAS INSTRUMENT. 2015. *LMx35, LMx35A Precision Temperature Sensors* [Online]. Texas Instrument. Available at: https://www.ti.com/lit/ds/symlink/lm335.pdf?HQS=dis-mous-null-mousermode-dsf-pf-null-wwe&ts=1650396072934&ref_url=https%253A%252F%252Fwww.mouser.com%252F (Accessed 10 April 2022).

TILLEY, N. n.d. *What Makes Plants Grow: Plant Growing Needs* [Online]. Gardening Know How. Available at: https://www.gardeningknowhow.com/special/children/how-plants-grow.htm (Accessed 12 April 2022).

TIWARY, A., MAHATO, M., CHIDAR, A., CHANDROL, M. K., SHRIVASTAVA, M. & TRIPATHI, M. 2018. Internet of Things (IoT): Research, architectures and applications. *International Journal on Future Revolution in Computer Science & Communication Engineering,* 4(3)**,** pp. 23-27.

TUFAIL, H., QASIM, I., MASOOD, M. F., TANVIR, S. & BUTT, W. H. Towards the selection of optimum requirements prioritization technique: a comparative analysis. 2019 5th International Conference on Information Management (ICIM), 2019. IEEE, 227-231.

VARADHARAJAN, V. & BANSAL, S. 2016. Data Security and Privacy in the Internet of Things (IoT) Environment. *In:* MAHMOOD, Z. (ed.) *Connectivity Frameworks for Smart Devices: The Internet of Things from a Distributed Computing Perspective.* Cham: Springer International Publishing**,** pp. 261-281.

VARGAS, R. V. & IPMA-B, P. Using the analytic hierarchy process (AHP) to select and prioritize projects in a portfolio. PMI global congress, 2010. 1-22.

VATHANAKAMSANG, A. 2017. *Benefits of the Open/Closed Principle* [Online]. Medium. Available at: https://medium.com/@a.vathanaka/benefits-of-the-open-closed-principle-dc9284d47598 (Accessed 12 April 2022).

WHITEHEAD, D. & WHITEHEAD, L. 2020. Data collection and sampling in qualitative research. *Nursing and Midwifery Research Methods and Appraisal for Evidence-Based Practice. 6th Edition. Sydney: Elsevier***,** pp. 118-135.

WHORTON, K. 2016. *Qualitative Interview Pros and Cons* [Online]. Asae Center. Available at: https://www.asaecenter.org/resources/articles/an_plus/2016/january/qualitative-interview-pros-and-cons. (Accessed 18 March 2022).

WIGMORE, I. 2022. *sensor* [Online]. TechTarget. Available at: https://www.techtarget.com/whatis/definition/sensor (Accessed 8 April 2022).

YALÇIN, M. A., ELMQVIST, N. & BEDERSON, B. B. Raising the Bars: Evaluating Treemaps vs. Wrapped Bars for Dense Visualization of Sorted Numeric Data. Graphics Interface, 2017. 41-49.

YUSOF, E., OTHMAN, M. & YUSOF, A. 2018. Operational dashboard: Accelerator for shop floor workers. *International Journal of Engineering & Technology,* 7(2.29)**,** pp. 4-6.

ZEZZA, A. & TASCIOTTI, L. 2010. Urban agriculture, poverty, and food security: Empirical evidence from a sample of developing countries. *Food policy,* 35(4)**,** pp. 265-273.

ZHANG, W. E., SHENG, Q. Z., MAHMOOD, A., ZAIB, M., HAMAD, S. A., ALJUBAIRY, A., ALHAZMI, A. A. F., SAGAR, S. & MA, C. The 10 research topics in the Internet of Things. 2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC), 2020. IEEE, 34-43.

# Appendicies

## Appendix 1: Terms of Rerence

Northumbria University
NEWCASTLE

# Project Terms of Reference

| | |
|---|---|
| **Module Name** | Individual Computing Project |
| **Module Number** | KV6003 |
| **Name** | Teck Xun Tan |
| **Student ID** | W20003691 |
| **Course** | BSc Hons Computer Science |
| **Project Title** | Smart Plant Caring System using IoT |
| **Supervisor Name** | Kay Rogage |
| **Second Marker Name** | Matthew Pointon |
| **Project Type** | General Computing Project |

**1. Project Title**

Smart Plant Caring System using IoT

**2. Project Background**

Urban gardening, also known as urban horticulture and urban agriculture, has been trending in many countries such as the United Kingdom, Singapore, Hong Kong and more for the past few years (Tan and Neo, 2009, Tam and Bonebrake, 2016, Dobson et al., 2020). The process consists of growing various types of plants in an urban environment like a rooftop and balcony. Through urban gardening, various unique gardening approaches have been developed, including vertical farming, community gardening, hydroponics, rooftop farming, and more (GroCycle). However, out of all the concepts, there is one that stands out the most and has been discussed by various researchers and developers: smart gardening.

A smart garden is a technological gardening based device usually controlled by an application on either a computer or mobile phone across the internet (Grant, 2020). Many smart gardening technologies devices and applications, such as smart plant monitors, gardening apps, robotic weeders, and more (L.Grant, 2021), are being researched day by day to enhance the concept of urban gardening worldwide. Moreover, the Internet of Things (IoT), a type of modern wireless communication, has also been widely applied to smart gardening for data collecting, analysing, and decision making (Patel and Patel, 2016). IoT adoption is increasingly on the rise, with various devices interacting and communicating to share data without human intervention(Gubbi et al., 2013, Abd Rahim et al., 2020). The purpose of IoT in smart gardening is to collect data such as humidity level, soil moisture level, light, temperature and more through different sensors (Pravin et al., 2018).

In today's world, several countries are heavily populated, and the majority of lands were cleared and reserved for building structures like malls, schools, or multistorey public housing and public highway. One of the example countries is Singapore; on recent news, it is reported that only 1% of city-state's land areas are devoted to agriculture. A significant amount of reserved land has since caused plantation problems, resulting in the country being heavily reliant on overseas suppliers (Moss, 2021). For that reason, it is ideal to propose an idea to build a product that could solve this problem by enabling users to participate in urban gardening in their own houses to save spaces and cost.

Another problem that had inspired this project idea is that according to an online newspaper, The Independent, a survey has shown that average British employees have spent 34 hours and 26 minutes working a week. They will also work overtime for an additional nine hours a month, totalling 4,890 hours in their whole career (Bailey, 2018). This has indicated that nowadays, people are occupied with work and have less spare time considering gardening as their hobby. Alongside the busy schedule, Stickel and Ludwig [13] found that people who practise urban gardening will not certainly hold knowledge of agriculture since they have different backgrounds. Hence, the idea of automating plant watering is much necessary to eliminate the need for manual watering. Furthermore, developing a web dashboard application to visualise the data collected by the sensors is ideal for enabling users to easily monitor and analyse the plant's data through the dashboard.

Overwatering was known to be one of the most common causes of plant problems. Overwatering not only wastes water because it exceeds the amount of water that the plant initially needs, but it also causes harm to the plants as it will drown the plants because they are unable to absorb oxygen normally from their roots which leads them to death (Garden, 2021). Previous similar works have suggested that IoT smart gardening devices can stop overwatering plants and reduce water wastage as they can decide when to start or stop dispensing water depending on the soil's moisture level through the sensors (Pravin et al., 2018).

One of the challenges of developing a smart IoT-based project is collecting meaningful and real-time data. It is proven that meaningful data is exceptionally vital. The data is primarily used in data analytics in IoT, allowing users to identify patterns or trends from the data collected by the device sensors, ensuring users will be well equipped with the knowledge required to make effective decisions. Moreover, meaningful data is also crucial in making a smart device as it plays a considerable role in machine learning, enabling Artificial Intelligence (AI) to make better and accurate predictions.

I am confident that this product is beneficial for everyone who practices urban gardening because the product will significantly increase the number of locations where people can perform urban gardening, enabling indoor to be one of the urban environments. Moreover, this product is also beneficial for users who wish to learn more about gardening or taking care of a plant but does not have enough spare time because the product automates plant caring like watering. Users may learn and analyses the trends and patterns of the data from the dashboard to understand the requirements to take care of a plant successfully.

**3. Proposed Work**

The proposed work is aimed to build a smart plant caring system based on IoT that will enable plant caring automation. The smart plant caring system is proposed to detect the moisture level of the soil and water. The plants can then be automatically watered via the IoT system, or the user can override this functionality and choose to water the plant by activating a 'water now' feature. Moreover, the system will also detect the level of sunlight indoor or outdoor to determine if UV lights are needed at that hour.

According to Kavita (2019), water and sunlight are two essential elements for plants to grow healthy. Water acts as a plant's transportation medium to carry water to the different parts of plants while also helping maintain its temperature. Meanwhile, sunlight enables plants to achieve photosynthesis to produce glucose required to generate energy and cellular structures. As mentioned in the problem above, people are occupied with work or may have a tight schedule which may occasionally be causing them to forget to water the plant or move the plant near sunlight. For this reason, it is ideal for the product to automate these tasks as one of the requirements, which will significantly increase the efficiency of taking care of a plant for busy people.

Through IoT, several data could also be collected, for example,

- Soil moisture level – To detect the moisture level of the soil

- Soil temperature – To detect the temperature of the soil
- Environment temperature – To detect the temperate of the environment around the device
- Water level – To detect the water level
- Sunlight – To detect the environment sunlight level
- Humidity – To detect the humidity around the plant
- Water pump status – To monitor the status of the water pump
- Battery level – To monitor the battery level

A similar device created by Anusha and Dr Mahadevaswamy (2018) has suggested similar data parameters required to be captured. Their device is built to capture and monitor temperature, humidity, soil moisture and intensity of light data collected by the sensor, which are vital for growing healthy plants.

These data are then visualised through a web dashboard application developed using web languages such as PHP, Javascript, CSS, and more. The dashboard has also planned to adopt charts to represent and visualise all the data collected because the graph is known to present complex or numerous data that is difficult to describe in text into an easier to read and understand format.

Several features could be implemented in this dashboard application, such as customisation. The graph, as mentioned above, could be customised to show the data in different graph types such as pie charts, line charts, doughnut charts and more to match the user preference. Moreover, the user will be able to check what data needs to be shown on the dashboard, which significantly increases the possible ways to personalise their dashboard.

The project will also include a survey which will be needed to find out and understand the users' views and opinions on what attractive or convenient features could be implemented into the dashboard of the smart plant caring system to enhance the overall user experience. The survey will be conducted online via Google Form featuring several questions for the participants to answer. This process is estimated to have 5 to 10 survey participants who are required to be over 18, employed or have a busy schedule, and interested in gardening. At the end of the development phase, participants will be asked to review and test the functionality of the implemented feature. This step ensures that the features suggested by the participators have reached their expectations and functioning correctly.

The user interface of the web dashboard application is planned to be designed based on the example shown in Figure 1 below.


**Figure 1: Example of the web dashboard application**

Other than that, to support the PHP on retrieving and displaying the data, a server hosting on a Raspberry Pi will be utilised to collect and save all the real-time data sent by the Arduino, and an API will be developed to retrieve all the information for visualisation. The example shown in figure 2:

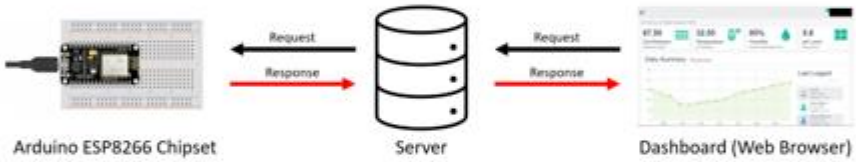



**Figure 2: How is data proposed to be retrieved and visualised**

The project will feature a semi-sized prototype of the smart plant caring system's physical model printed utilizing a 3D printer, as shown in Figure 3 below, which will hold all the sensors and motors necessary to collect the data and automate the watering system IoT technologies.

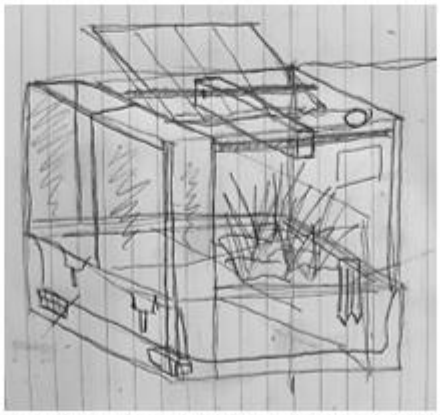


**Figure 3: Sketch of the Smart Plant Caring System Model**

The necessary parts that are required for this project will be categorised using the MoSCoW method. This method is essential as it creates a better understanding of the product's requirements and identifies the project's priorities. The MoSCoW method is divided into four (4) sections which are,

- **M** – Must have the requirement in the project
- **S** – Should have the requirement, but project success will not rely on it
- **C** – Could have the requirement, if it does not affect other functionality on the project
- **W** – Would like to have the requirement, which would not be developed this time

It is crucial to have a clear set of agreed and prioritised requirements in the project, combined with the overall objective, quality criteria, and timescale to create a successful project which makes the reason for selecting the MoSCoW method for this project.

In this project, computing technologies like the Internet of Things (IoT) will be used. It is the most critical technology used in this project as it represents the central core of the project handling works like plant caring automation and data collection and visualisation. Another exciting technology that will be explored in this project will be Artificial Intelligence (AI). The project is also planned to train an AI to predict and recommend the best plant planted at that temperature or during that season.

There are a few areas that will be required to be investigated to carry out the project. The literature review will cover the importance of urban gardening, how it affects the future and how it works. This literature review topic helps the project identify how the overall product should work, for example, what sensor part is vital for a smart urban gardening system and how similar devices solve their problem. Moreover, the literature review will also cover IoT. Reviewing this part of literature is believed to have a deep understanding of how overall IoT works, which is essential when building and developing the Smart Plant Caring System. Furthermore, the importance of data collecting, analysis and visualising will also be part of the literature review to have a deep understanding of what data is worth collecting and visualising on the dashboard. In addition, the literature review will also feature a brief description of the framework selected for this project and the reason for selecting them. Reviewing this part of literature is believed to have a better understanding of the frameworks chosen and also the benefits. Lastly, the literature review will discuss what constitutes a good dashboard design which will be needed to review to justify the design to the dashboard of this project.

Testing is considered one of the most vital aspects when developing a project. It is used to perform functional tests and identify errors in the features that have been implemented into the project. This project will feature a test plan documentation used to identify what should be tested and how to execute the test. Changes will then be made to the dashboard or IoT set up based on the identified error from the test results to ensure that the products are functioning correctly.

Overall, this project is considered moderate size since there are several challenges such as building the physical model of the Smart Plant Caring System, gathering necessary sensor parts, and building the web dashboard application alongside establishing the connection

between a server with an Arduino device and web application. However, I am confident that the project is realistically achievable, just as planned within the 400 hours.

**4. Project Aims**

- To build a prototype Smart Plant Caring System to prove the concept of plant caring automation
- To develop a web dashboard application to visualise all the collected data to allow a user to analyse and learn the trends and patterns of the plant data

**5. Objectives**

| Objectives | Purpose |
|---|---|
| Conducting literature review | To have a better understanding of the topic and tools chosen for this project, such as urban gardening technology, Internet of Things (IoT), Data in IoT and tools used like programming languages |
| Creating a requirement specification for both IoT set up and dashboard | To understand what functions are required to be implemented and achieved for both IoT set-up and dashboard |
| Develop dashboard and IoT infrastructure design documentation (e.g., class diagram, sequence diagram, database design and more) | To support and ensure the speed and efficiency of the development phase by using these diagrams constructed as a reference |
| Conducting product testing | To ensure that every feature of the product is functioning, such as requirement, functionality, and usability and produce a test plan along with test results |
| Explore and understand which sensors components are required for the project | To identify what sensors and actuators to support the use case functionality and data requirement developed |
| Develop dashboard and IoT infrastructure for data integration from sensors | To establish the connection between IoT infrastructure and dashboard to collect and send necessary data from sensors |
| Implement the design | To develop and implement the features listed using codes like C++, PHP, Python and more |
| Develop technical skills for IoT implementation | To develop technical skills like how overall IoT works, how to build an IoT infrastructure and several new coding languages like C++ |

| | |
|---|---|
| Develop and evaluate dashboard design with users | To develop and get users evaluation on the dashboard design to enhance the user experience |
| Summarise recommendations and future direction of the project | To summarise the recommend and analyse what future direction could be implemented for the project |

**6. Skills**

| **Skill Required** | **Has the skill already been acquired?** | **The scale of fully competent**<br>**(1 Not competent – 5 Fully competent)** |
|---|---|---|
| Understanding of IoT | Skills have been acquired from *KV6006 – Internet of Things* | 4 |
| Knowledge to program a webpage (e.g., HTML, CSS, PHP) | Skills have been acquired from *KF5002 – Web Programming* and *KF6012 – Web Application Integration* | 5 |
| Knowledge of Arduino programming language, C++ | Skills will be acquired during the project through online courses or tutorials such as https://www.learncpp.com/ | 3 |
| Knowledge to manage a database using queries (e.g., SQLite) | Skills have been acquired from *KF5002 – Web Programming* and *KF6012 – Web Application Integration* | 5 |
| Knowledge to construct UML diagrams (e.g., Class Diagram, Sequence Diagram, and more) | Skills have been acquired from *KF5008 – Program Design and Development* | 5 |
| Knowledge to establishing a connection between server and devices | Skills will be acquired during the project through online courses or tutorials and also researching through books | 2 |

| | | |
|---|---|---|
| Understanding of Human Computing Interaction (HCI) | Skills have been acquired from *KV5003 – Human-Computer Interaction* | 4 |
| Understanding of system development life cycle (e.g., Agile) | Skills will be acquired during the project through online courses or tutorials | 4 |
| Knowledge to build a circuit connecting Arduino with sensors selected | Skills have been acquired from *KV6006 – Internet of Things*, additional skills could be acquired through online courses or tutorials and also researching through books | 3 |
| Knowledge to write and construct a report or documentation | Skills have been acquired from *KV5001 – Academic Language Skills for Computer and Information Science* | 5 |

**7. Source of Information/Bibliography**


ABD RAHIM, N., ZAKI, F. A. & NOOR, A. 2020. Smart App for Gardening Monitoring System using IoT Technology. *system,* 29**,** 7375-7384.

BAILEY, G. 2018. *British People will work for average of 3,507 days over a lifetime, survey says* [Online]. The Independent. Available: https://www.independent.co.uk/life-style/british-people-work-days-lifetime-overtime-quit-job-survey-study-a8556146.html [Accessed 15 October 2021].

BAZZAZ, M. 2019. *The Role of Data Analytics in IoT* [Online]. IoT for all. Available: https://www.iotforall.com/role-data-analytics-iot [Accessed 16 October 2021].

DOBSON, M. C., EDMONDSON, J. L. & WARREN, P. H. 2020. Urban food cultivation in the United Kingdom: Quantifying loss of allotment land and identifying potential for restoration. *Landscape and Urban Planning,* 199**,** 103803.

GARDEN, M. B. 2021. *Overwatering* [Online]. Missouri Botanical Garden. Available: https://bit.ly/3n0Qkps [Accessed 16 October 2021].

GRANT, A. 2020. *Self Watering Indoor Garden: How Do You Use A Smart Garden* [Online]. Gardening Know How. Available: https://www.gardeningknowhow.com/special/containers/what-is-a-smart-garden.htm [Accessed 12 October 2021].

GROCYCLE. *Urban Farming Ultimate Guide and Examples* [Online]. GroCycle. Available: https://grocycle.com/urban-farming/ [Accessed 15 October 2021].

GUBBI, J., BUYYA, R., MARUSIC, S. & PALANISWAMI, M. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems,* 29, 1645-1660.

L.GRANT, B. 2021. *Smart Gardening Guide – Learn About Gardening With Technology* [Online]. Gardening Know How. Available: https://www.gardeningknowhow.com/garden-how-to/info/gardening-with-technology.htm [Accessed 11 October 2021].

MOSS, D. 2021. *Can 1% of Singapore's Land Feed Its Population?* [Online]. Bloomberg Opinion. Available: https://www.bloomberg.com/opinion/articles/2021-04-29/singapore-s-investment-in-urban-farming-isn-t-just-trendy [Accessed 12 October 2021].

PATEL, K. K. & PATEL, S. M. 2016. Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges. *International journal of engineering science and computing,* 6.

PRAVIN, A., JACOB, T. P. & ASHA, P. 2018. Enhancement of plant monitoring using IoT. *International Journal of Engineering and Technology (UAE),* 7, 53-55.

SLUTSKY, D. J. 2014. The effective use of graphs. *Journal of wrist surgery,* 3, 067-068.

TAM, K. C. & BONEBRAKE, T. C. 2016. Butterfly diversity, habitat and vegetation usage in Hong Kong urban parks. *Urban ecosystems,* 19, 721-733.

TAN, L. H. & NEO, H. 2009. "Community in Bloom": local participation of community gardens in urban Singapore. *Local Environment,* 14, 529-539.

WEBB, R. 2020. *12 Challenges of Data Analytics and How to Fix Them* [Online]. ClearRisk. Available: https://bit.ly/3BMUAzh [Accessed 16 October 2021].

**8. Resources**

| Software | How will it be provided? | Purpose | If the item is not available |
|---|---|---|---|
| Arduino IDE | The software is free to download through the official website or Microsoft Store. | Default software uses to create functions for Arduino using C++ language. It is used in this project to create functions to automate the plant caring components like watering and UV light switching and upload data collected by the sensors to the server. | The software is the core IDE for Arduino, and it will be available permanently. This item cannot be not available. |
| JetBrains PhpStorm | A student license is obtainable for free using student | I have been using software to create websites or web applications; it handles | A license will be available for students |

| | | | |
|---|---|---|---|
| | email at the official website. The software mentioned will also be free with the student license activated. | web languages like HTML, CSS, and PHP. It is used to develop a web dashboard application for this project. | permanently. This item cannot be not available. |

| **Hardware** | **How will it be provided?** | **Purpose** | **If the item is not available** |
|---|---|---|---|
| Arduino Uno | Available to be purchased at Amazon or tech stores | Hardware acts as the central core of the IoT system, controlling the behaviour of sensors and other parts and sending data collected to the server. For example, Arduino Uno will need to activate a water sprinkler and stepper motor when the moisture sensor detects dry soil. | The student already owns the hardware. This item cannot be available. |
| Humidity Sensor | Available to be purchased at Amazon or tech stores | Hardware is used to detect and collect data on the humidity level inside the planting compartment of the Smart Plant Caring System device. | The hardware will be requested from the university equipment loans centre or loan from the supervisor. |
| Moisture Sensor | Available to be purchased at Amazon or tech stores | Hardware is used to detect and collect data on the moisture level of the soil. | The hardware will be requested from the university equipment loans centre or loan from the supervisor. |
| Temperature Sensor | Available to be purchased at Amazon or tech stores | Hardware is used to detect and collect the environment's temperature around the Smart Plant Caring System device. | The hardware will be requested from the university equipment loans centre or loan from the supervisor. If |

| | | | |
|---|---|---|---|
| | | | hardware is still not available, it could be removed from the project, and it would not affect the core feature of the IoT system. |
| Sunlight Sensor | Available to be purchased at Amazon or tech stores | Hardware is used to detect and collect data on the sunlight level of the environment around the Smart Plant Caring System device. | The hardware will be requested from the university equipment loans centre or loan from the supervisor. |
| Solar Panel | Available to be purchased at Amazon or tech stores | Hardware used to practice reusable energy turning solar power into energy charging the battery of the Smart Plant Caring System device. | The hardware will be requested from the university equipment loans centre or loan from the supervisor. If hardware is still not available, it could be removed from the project, and it would not affect the core feature of the IoT system. |
| Water Pump | Available to be purchased at Amazon or tech stores | Hardware is used to pump water into the water sprinkler to water the plants. | The hardware will be requested from the university equipment loans centre or loan from the supervisor. |
| Stepper Motor | Available to be purchased at Amazon or tech stores | Hardware is used to move the water dispenser compartment to ensure that every plant part is watered. | The hardware will be requested from the university equipment loans centre or loan from the supervisor. If |

| | | | |
|---|---|---|---|
| | | | hardware is still not available, it could be removed from the project, and it would not affect the core feature of the IoT system. |
| Water Sprinkler | Available to be purchased at Amazon or tech stores | Hardware is used to dispense water to the plant. | The hardware will be requested from the university equipment loans centre or loan from the supervisor. |
| UV Light LED | Available to be purchased at Amazon or tech stores | Hardware is used to provide UV lights to the plant. | The hardware will be requested from the university equipment loans centre or loan from the supervisor. |

**9. Project Report Structure and Contents**

| Report Structure | Description |
|---|---|
| • Abstract | This section of the project will feature a precise and concise summary of the project. |
| • Introduction<br>o Project Background<br>o Aims & Objectives<br>o Product Overview<br>o The approach taken and tools used | This section of the project will feature the **Introduction** of the project.<br>This section will also introduce a few subsections to examine the project's background, aims, and objectives, discuss the problem context, and reasons for undertaking the project. Finally, also a summary of the approach taken and tools used. |
| • **Research and Planning** | This section will briefly explain what will be included in the **research and planning** section of the report. |

| | |
|---|---|
| • Literature Review<br>  ○ Urban Gardening Technology<br>  ○ Internet of Things (IoT)<br>  ○ Data in IoT<br>  ○ Framework<br>  ○ Good Dashboard Design | This section of the project will introduce a **literature review** of the project.<br>A few subsections will review different topics, including urban gardening technology, evaluating the definition and importance of urban gardening, how it affects the future, and how it works.<br>The next topic will be reviewing the topic of **Internet of Things (IoT)**. Several topics like what is IoT and what system has utilised IoT technology could be expected from this subsection.<br>The third subsection will be reviewing the uses of **Data in IoT**. This subsection of the literature review will investigate the importance of data collecting and analytics in IoT.<br>In the fourth subsection, different **frameworks** used to develop the dashboard, such as ReactJS and Bootstrap, will be reviewed to understand the selected framework better.<br>Lastly, characteristics for **good dashboard design** will be reviewed to justify the design for the dashboard. |
| • Existing Systems | This section of the project will be discussing **existing systems** in the smart plant system sector. |
| • Requirement Specification<br>  ○ Application Structure<br>  ○ Functional Requirement<br>  ○ Non-Functional Requirement<br>  ○ Tools and Techniques<br>  ○ Development Life Cycle | This section of the project will be discussing the **requirement specification** of the project.<br>The subsection **application structure** will explain how the application structure is planned along with a use case diagram.<br>The following subsection consists of both **functional and non-functional requirements** listing the software's functions and defining systems attributes like security, reliability, performance, and more.<br>The third subsection will be introducing the **tools and techniques** used to develop the project.<br>Finally, the last subsection will introduce the **development life cycle** (e.g., Agile, Waterfall) |

| | |
|---|---|
| | that will be adopted through the development phase of the project |
| • **Design, Implementation and Testing** | This section will briefly explain what will be included in the **report's design, implementation, and testing section**. |
| • Analysis Models | This section will introduce the **analysis models** (e.g., Sequence Diagram) that have been created to support the development of the system. |
| • Design Specification | This section will introduce the **design specification** (e.g., Wireframe) part of the project to provide an overview of the design decisions to meet the project requirement specification. |
| • Product Hardware<br>  ○ Sensors<br>  ○ Microcontroller | This section will introduce the **project hardware** used to achieve IoT functionality.<br>The first subsection will introduce the **sensors** that have been used in this project. The sensors are expected to have a brief description of the function to clarify the purpose of using them in this project.<br>The following subsection will introduce the core hardware, **Microcontroller**. This subsection will provide a brief description of the hardware, explain how this hardware is used in this project. |
| • Product Code<br>  ○ Hardware<br>  ○ Software | This section will briefly introduce the **product code** of the project.<br>The subsection, **Hardware**, will provide codes used to program the Arduino (C++) alongside the code functionality and explanation.<br>Moreover, the subsection, **Software**, will provide codes used to program the web dashboard application (which is PHP). |
| • Testing<br>  ○ Functionality Testing<br>  ○ Testing Result | This section will introduce the **testing** process of the project.<br>The first subsection will introduce the **functionality testing** phase of the project, like testing strategy, a technique used. This subsection is expected to provide a list of the |

| | |
|---|---|
| | test case in the appendix to show the steps taken for testing.<br><br>The following subsection will discuss the **testing results**. This subsection is expected to discuss the result obtained from the testing and conclude the testing. |
| • **Evaluation and Conclusions** | This section will briefly explain what will be included in the **evaluation and conclusions** section of the report. |
| • Evaluation of Smart Plant Caring System<br>  ○ Product Functionality<br>  ○ Development Challenges | This section of the report will discuss an **evaluation of the Smart Plant Caring System,** like the build quality and fitness for the product.<br><br>**The product functionality** subsection will discuss the product's overall functionality and how the product has met the requirements set out in this document.<br><br>**The development challenges** subsection will discuss what challenges have been met during the development of this project. |
| • Evaluation of Project Process<br>  ○ Project Management<br>  ○ Learning Challenges | This section of the report will discuss the **evaluation of the project process,** such as project management aspects, use and suitability of methods and tools and more.<br><br>The first subsection, **Project Management**, will discuss the effectiveness of adopting the selected project development lifecycle in this project.<br><br>The second subsection, **Learning Challenges**, will discuss the difficulties and challenges throughout the learning phase. A table of what has been learned is expected to be included in this section. |
| • Conclusion and Recommendations<br>  ○ Conclusion<br>  ○ Recommendations | The final section of the report will provide a **conclusion and recommendations** for this project.<br><br>The **conclusion** subsection will conclude this project alongside discussing what has been achieved and not achieved in the project. |

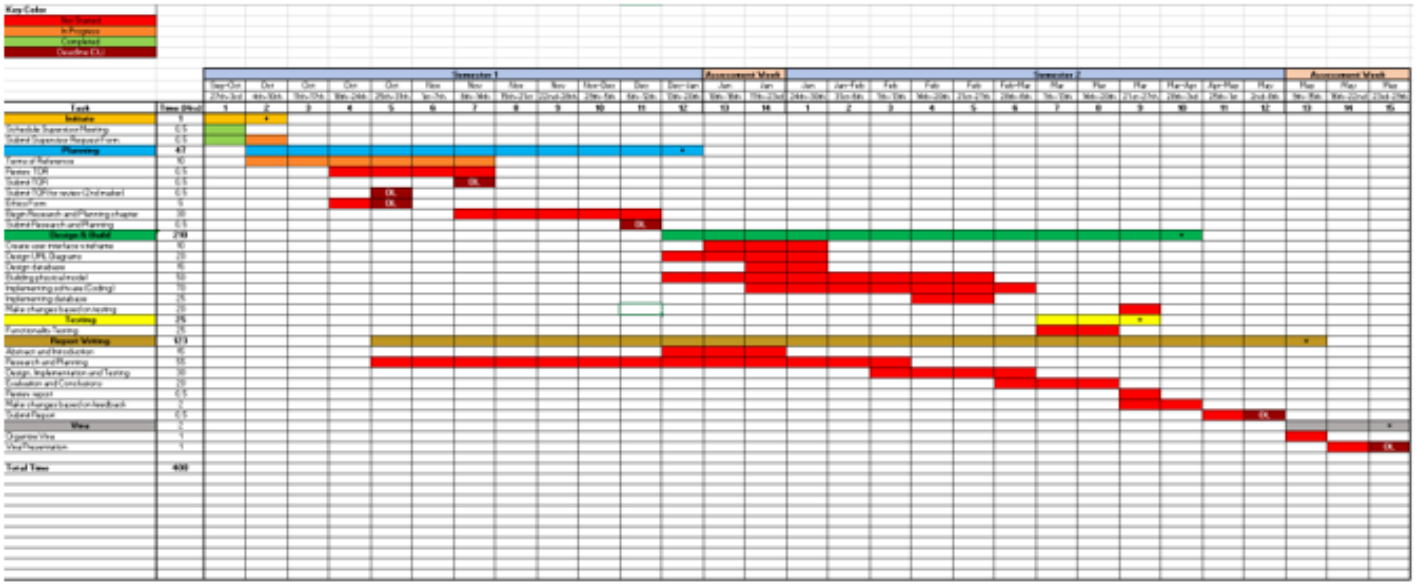
| | |
|---|---|
| | The final subsection **Recommendations** will discuss further research relating to both product and the broader field. |

**10. Assessment Criteria for Product**

- Fitness for Purpose
  - Meeting of Requirements as identified during the project (25m)
    - A dashboard will be provided with a physical prototype which will be demonstrated at the viva on request. A video of the physical set-up will also be submitted with the report.
  - Quality of Functionality (15m)
    - Product Functionality – All the functions are working correctly
    - Usability – The dashboard is easy to use
    - Security – The product will secure user's information
    - Safety – The product is safe for users
  - Human-Computer Interaction (HCI) (10m)
    - Learnability – The dashboard function and interface can be quickly learned by the users
    - Accessibility – The dashboard should be user friendly towards people with disabilities
    - Interface Design – The dashboard should have a good design practice
    - Error Handling – The dashboard should handle errors well and provide the user with instructions on what should be done if an error has occurred

- Build Quality
  - Quality of requirements specification & analysis (10m)
    - Functional and Non-Functional Requirement documentation
    - Use Case documentation
  - Quality of design (15m)
    - UML Diagrams (e.g., Sequence Diagram and more)
    - Database design (e.g., Database Model)
    - Low and High Fidelity design
  - Quality of implementation (15m)
    - Correspondence of implementation to the design
  - Quality of testing (10m)
    - Test Plan Documentation
    - Test Results Report

## 11. Project Plan – Activity Schedule



## Terms of Reference Review Form

This form is to be completed **by 2nd marker** after the TOR group review meeting and then sent to the student and the supervisor for signature. After the TOR group review, students should make any necessary changes to the TOR, online ethics and risk assessment forms (as required by your Supervisor and Second Marker), then submit the final approved TOR to Blackboard by the dates given in the module schedule whilst also completing the online ethics and risk assessment approval process.

**Date of TOR Review:**

**Review Outcomes:**

| | |
|---|---|
| The topic is appropriate to the student's programme. | Yes / |
| The project contains sufficient practical work using computing skills relevant to the programme. | Yes / |
| An appropriate topic for the literature review has been identified. | Yes / |
| An explanation of the contribution of the analysis/literature review to the project work has been given. | Yes / |

**The TOR (select one):**

| | |
|---|---|
| • Accepted without changes. | |
| • Needs the changes listed below. | Y |
| • Cannot be made satisfactory and a new topic is required. | |

**Ethics Draft PDF (select one):**

| | |
|---|---|
| • Has been reviewed and can be approved via the Ethics Online System. | |
| • Requires revision before approval can be granted via the Ethics Online System. | Y |
| • Has been reviewed; the project should be referred to the Faculty Research Ethics Committee (FREC) via the Ethics Online System. | |
| • The project has already been referred to FREC via the Ethics Online System. | |
| • Has not yet been provided. | |
| • Other (please explain). | |

**Risk Assessment Draft PDF (select one):**

| | |
|---|---|
| • Has been reviewed and can be approved via the Ethics Online System. | |
| • Requires revision. | |
| • Is required but has not yet been provided. | Y |
| • Not required. | |

**Changes required/identified issues:**

| |
|---|
| **Changes to proposed aim(s):**<br>Develop a prototype IoT smart watering plant system – make sure you are clear that you state this is a proof-of-concept prototype so you are not committing to a fully functioning system. |
| **Changes to proposed objectives:**<br>Don't need develop project plan as an objective.<br>Include an objective to develop and evaluate dashboard design with users.<br>Include an objective to say summarise recommendations and future direction of the project. |
| **Changes to deliverables:** |
| **Changes to structure and contents of project report:** |
| **Changes to project plan:** |
| **Resource issues:** |
| **Other comments:**<br>Align the marking scheme percentages against the deliverables.<br><br>Is the testing and evaluation part of the design specification – need to clarify this in TOR. |

The ethics form looks OK except I need to see the information sheet and consent form you are providing to your users before the ethics form can be submitted.
Suggest that the hardware sensor components are suggested components but these will be revised and prioritised using a MoScoW approach during the requirements analysis.

In the literature review discuss what constitutes a good dashboard design to justify your dashboard design. So look at sources that cover interaction design foundation and Neilson/Norman group dashboard design/data visualisation.

You have already made decisions about what the system does – how were these requirements arrived at? It needs to be clear what the process was and where the requirements came from.

You can clearly collect a lot of different types of data – but what use cases do these support? If you did a full analysis of the literature and existing systems you could develop a set of use case requirements that help to identify what data is relevant for each use case.

Survey questions come after the building of the system – make sure that the requirements capture process comes before the IoT set-up.

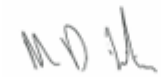
Risk form needs completing.

It's a general computing project – the deliverables align to this type of project better.

**Signatures:**

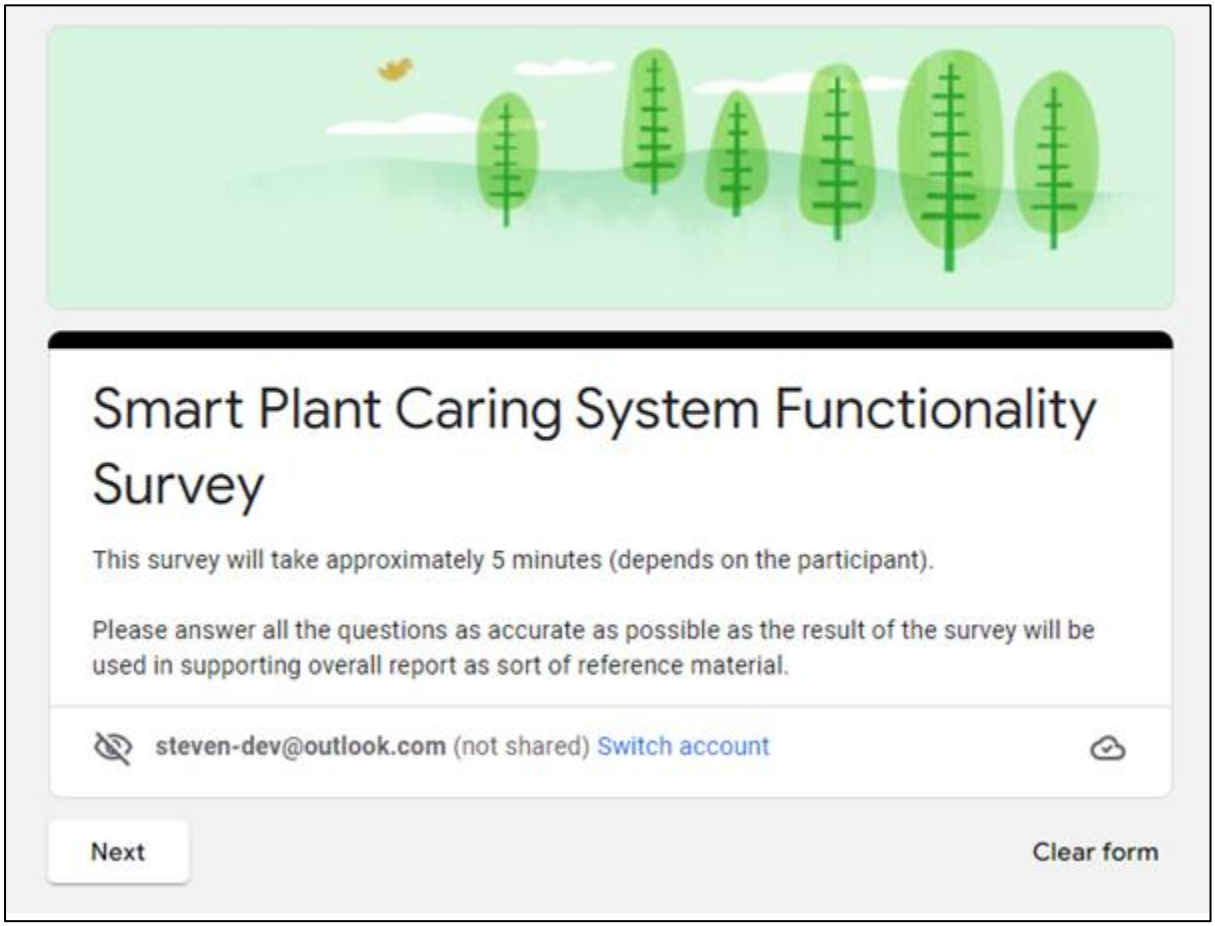
**Student** **Teck Xun Tan**

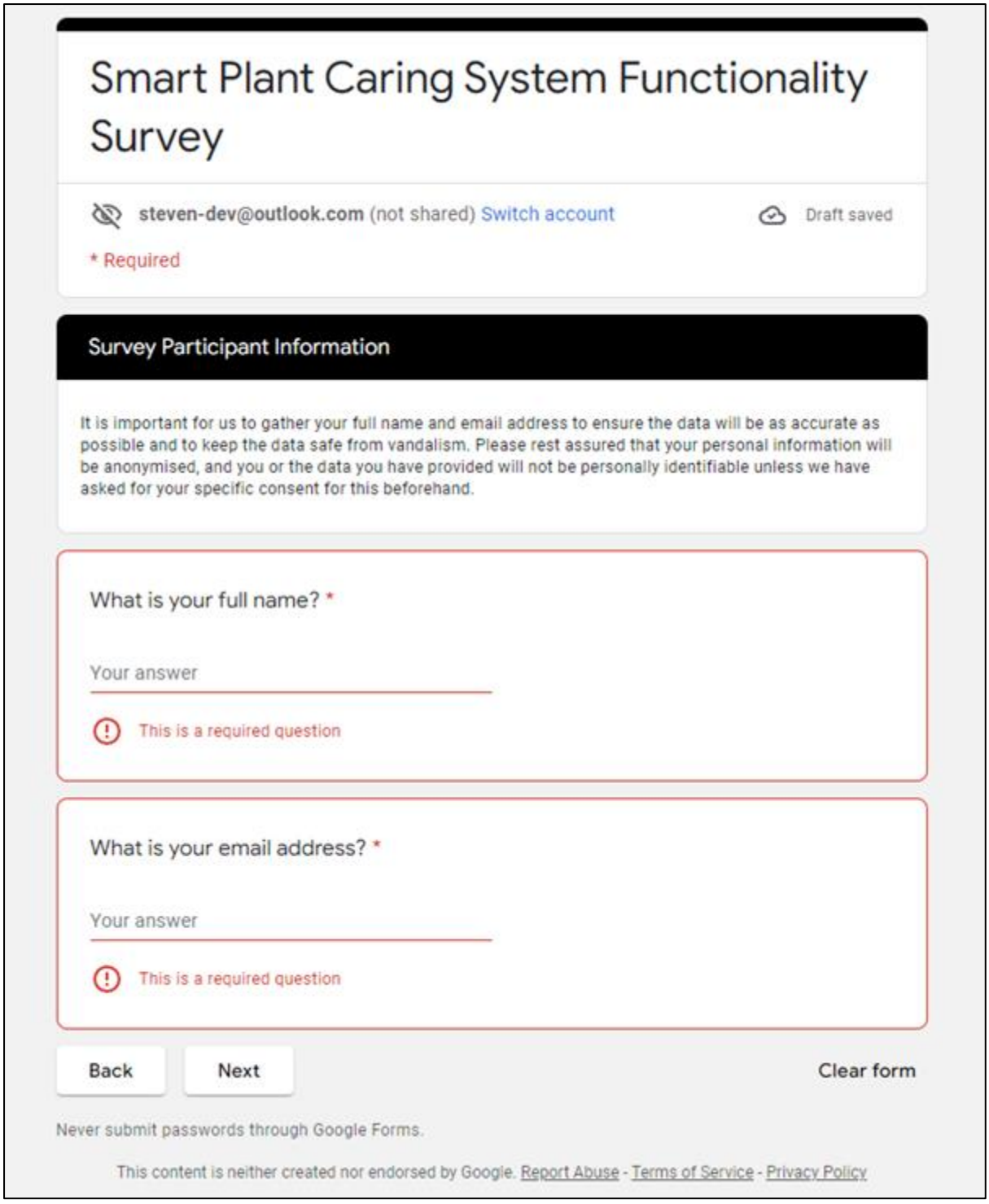
**Supervisor Kay Rogage**



**Second Marker** **Matt Pointon**

## Appendix 2: Survey Questionnaires




Smart Plant Caring System Functionality Survey
steven-dev@outlook.com (not shared) Switch account
Draft saved
* Required
Survey Participant Information
It is important for us to gather your full name and email address to ensure the data will be as accurate as possible and to keep the data safe from vandalism. Please rest assured that your personal information will be anonymised, and you or the data you have provided will not be personally identifiable unless we have asked for your specific consent for this beforehand.
What is your full name? *
Your answer
This is a required question
What is your email address? *
Your answer
This is a required question
Back
Next
Clear form
Never submit passwords through Google Forms.
This content is neither created nor endorsed by Google. Report Abuse - Terms of Service - Privacy Policy

# Smart Plant Caring System Functionality Survey

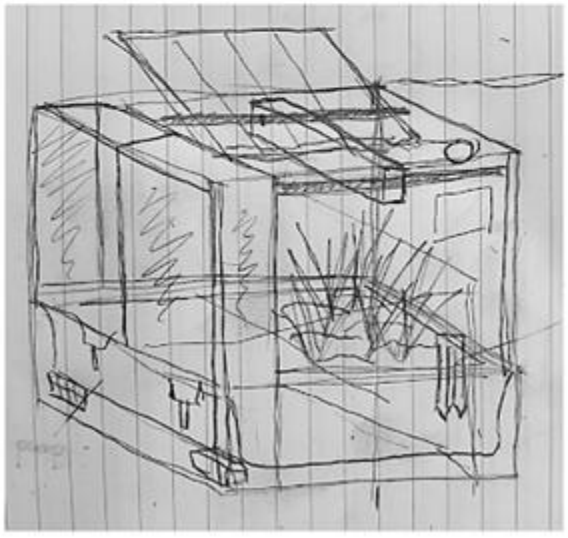
steven-dev@outlook.com (not shared) Switch account

## What is Smart Plant Caring System

An automatic plant caring system based on IoT that will detect the moisture level of the soil and water the plants automatically or manually using a web based dashboard. Moreover, the system will detect the level of sunlight indoor or outdoor to determine if UV lights are needed at that hour. Through IoT, several data could also be collected and visualize, for example,

- Soil moisture level
- Environment temperature
- Sunlight
- Humidity
- Water pump status

Sketch of the Smart Plant Caring System



## Are these information useful on the dashboard?

On the previous section, we have mentioned that the dashboard will visualise several data collected by the sensors. Users potentially use these collected information by studying the patterns and make decisions to better taking care of a plant.

In your opinion, are these collected information useful to be visualised on the dashboard?

Soil Moisture Level *

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Not Useful | ○ | ○ | ○ | ◉ | ○ | Very Useful |

Environment Temperature *

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Not Useful | ○ | ○ | ○ | ◉ | ○ | Very Useful |

Sunlight *

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Not Useful | ○ | ○ | ○ | ◉ | ○ | Very Useful |

Humidity *

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Not Useful | ○ | ○ | ○ | ◉ | ○ | Very Useful |

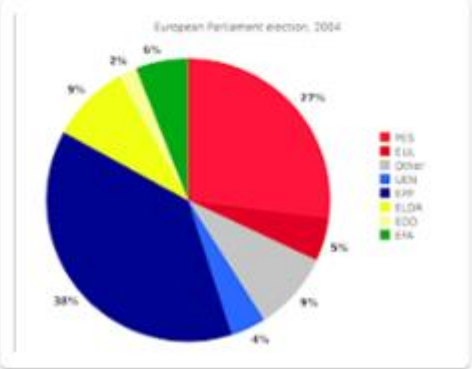
Water Pump Status *

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Not Useful | ○ | ○ | ○ | ◉ | ○ | Very Useful |

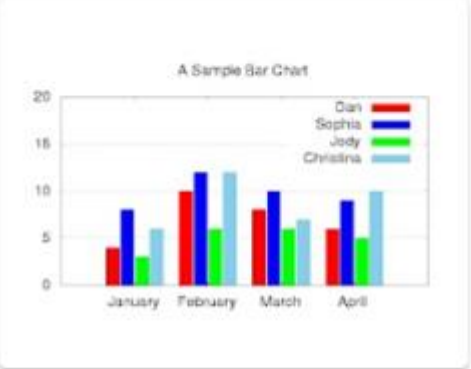
## Data Visualising Method

Data Visualisation is one of the essential core features in an IoT dashboard. It receives and process data transmitted by the sensors and visualises them on an interactive dashboard that shows valuable trends and patterns in real-time (Prolim, 2021). Data Visualisation is also a solution to support interpreting, visualising, and analysing big data, which is a term for complicated and large data sets collected by various sources such as IoT systems or devices across the Internet over the years (Lowe and Matthee, 2020). The primary functions of data visualisation are to enable users to see the data at a glance instead of the necessity of close reading.

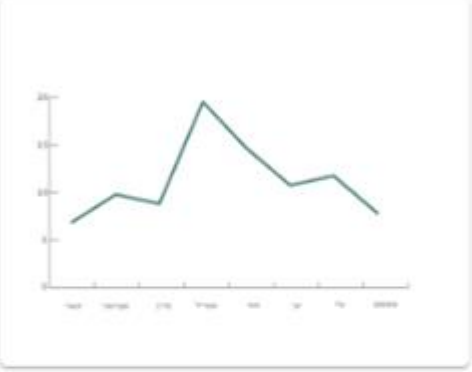
Which visualising method do you think suits you the best?




☐ Pie Chart




☐ Bar Chart

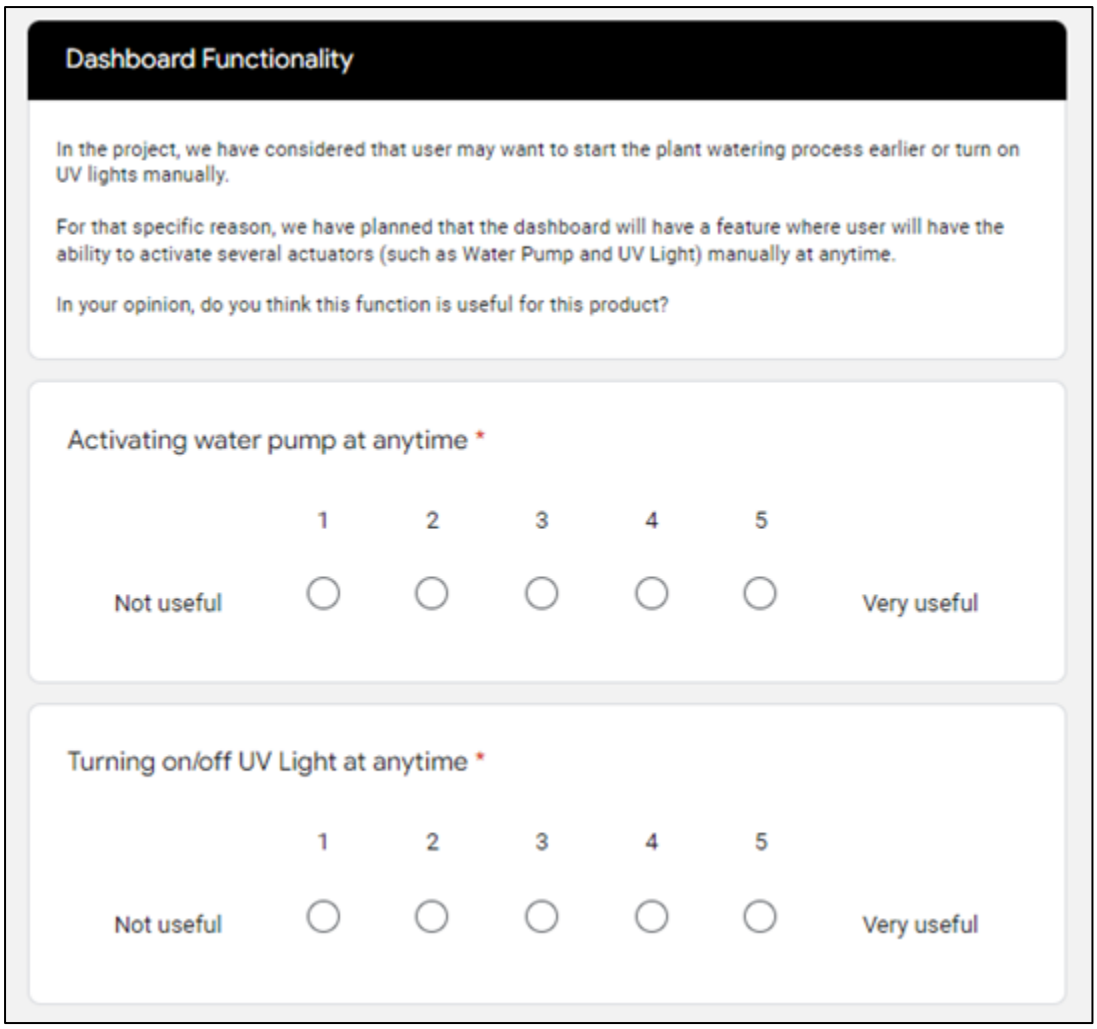


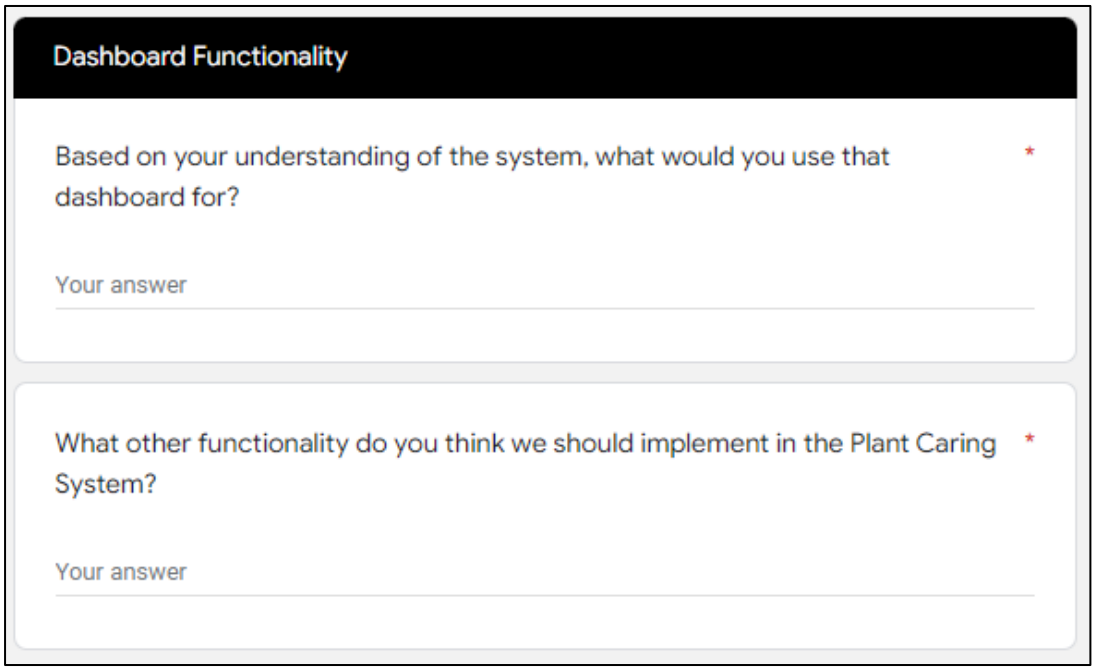
☐ Line Chart


☐ Traffic Light Face

Based on your choices, what do you like about that visualising method? *

Your answer




# Appendix 3: Participant Information Sheet

Northumbria University NEWCASTLE

Smart Plant Caring System using IoT

# Participant Information Sheet

You are being invited to take part in this software engineering project.

Before you decide it is important for you to read this leaflet so you understand why the study is being carried out and what it will involve.

Reading this leaflet, discussing it with others or asking any questions you might have will help you decide whether or not you would like to take part.

## Purpose of the Study

I will be developing a smart plant caring system based on IoT that will enable plant caring automation. The smart plant caring system is proposed to detect the moisture level of the soil and water. The plants can then be automatically watered via the IoT system, or the user can override this functionality and choose to water the plant by activating a 'water now' feature. Moreover, the system will also detect the level of sunlight indoor or outdoor to determine if UV lights are needed at that hour. Moreover, a web dashboard will then visualise all the data collected through the sensors in the form of graphs, which will enable users to easily monitor and analyse the plant's data through the dashboard.

It is essential for us to find out and understand the users' views and opinions on what attractive or convenient features could be implemented in the web dashboard to enhance the overall user experience. Hence, the study will deliver the smart plant caring system concept to several participants and then gather their thoughts and suggestions. I am conducting this study as part of my BSc in Computer Science at Northumbria University.

## Why have I been invited?

We must assess as many people as possible to collect sufficient and accurate data. You are invited to participate in this study because you have indicated that you are currently employed and interested in the topic of gardening and that you are interested in taking part in this study.

Collecting accurate data is vital to us in this project. The participant needs to be employed and be interested in the topic of gardening because one of the aims of this project is to eliminate the problem of people who are interested in adopting gardening as their hobby but are very occupied with busy working schedules.

## Do I have to take part?

No. It is up to you whether you would like to take part in the study. I am giving you this information sheet to help you make that decision. If you decide to take part, remember that you can stop being involved in the study whenever you choose, without giving us a reason. You are entirely free to decide whether to take part or to take part and then leave the study before completion. You are welcome anytime to take part in the smart plant caring system project and choose not to take part in the project. Deciding not to take part or leave the study will not affect your right to join the smart plant caring system project in the future.

## What will happen if I take part?

You will be asked to review and complete a set of questions on the prepared Google Form survey anonymously. You will be able to start answering questions prepared upon agreeing

to this participant information sheet. After completing the survey, you may choose to withdraw your data if you wish. It is estimated that the total time to complete this study will be 10-20 minutes. This survey will be informal, and you will be able to complete it at any time that suits you best.

By taking part in the study, you will be participating in the smart plant caring system project, which will hopefully provide you with information regarding the primary purpose and aims of the project. Furthermore, by taking part and providing us with your suggestions of the smart plant caring system, you will be helping to develop a better user experience.

With your permission, I would use the data collected to implement the functions for the smart plant caring system dashboard and documentation. Your name will not be written on any of the data we collect; the written information you provide will have an ID number, not your name. Your name will not be written or appear in any reports or documents resulting from this study. The consent form you have signed will be stored separately from your other data. Rest assured that all the data you have filled in will remain confidential and anonymous, and you will still have the right to withdraw at any point. The only exception to this confidentiality is if the researcher feels that you or others may be harmed if the information is not shared.

## How will my data be stored, and how long will it be stored for?

All data, including the survey questionnaires, the survey's typed-up results, and your consent forms, will be kept in a password-protected hard drive. All the data will be stored following university guidelines and the Data Protection Act (2018).

The personally identifiable data will be stored until the end of the BSc in Computing Science individual computing project module. The data collected will not be accessible on any data repository and will remain anonymous throughout the module.

## What categories of personal data will be collected and processed in this study?

Under the EU's new General Data Protection Regulation, if we are collecting personal data indirectly, we will have to be transparent about what categories of personal data will be collected and processed in the study. In this study, the participant's name and email is the only personal data necessary to be collected for smoother processing and analysing of the personal data. Participants' names and email are meant only for us and will always remain confidential, and they will not appear or be written in any reports or documentation.

## What is the legal basis for processing personal data?

According to GDPR, we must be transparent about the legal basis for undertaking research that will collect and process personal data. The legal basis chosen for this university project will be *Article 6(1)(e) "processing is necessary for the performance of a task carried out in the public interest or in the exercise of official authority vested in the controller".*

## What will happen to the results of the study, and could personal data collected be used in future research?

The general findings might be reported in a scientific journal or presented at a research conference. However, the data will be anonymised, and you or the data you have provided will not be personally identifiable unless we have asked for your specific consent for this beforehand. We can provide you with a summary of the findings from the study if you email the researcher at the address listed below.

## Who is Organising the study?

Northumbria University Newcastle

## Who has reviewed this study?

The research project, submission reference [39925], has been approved in Northumbria University's Ethics Online system. It has been reviewed to safeguard your interests and have granted approval to conduct the study.

## What are my rights as a participant in this study?

**Your right to get copies of your data**
You have the right to ask the organisation whether we are using or storing your personal information. You can also ask them for copies of your personal information, verbally or in writing.

To get a copy of your data, you should submit a *Subject Access Request* through the link. (https://www.northumbria.ac.uk/about-us/leadership-governance/vice-chancellors-office/legal-services-team/gdpr/gdpr---rights-of-the-individual/right-to-subject-access/)

**Your right to get your data corrected**
You can challenge the accuracy of personal data held about you by an organisation and ask for it to be corrected or deleted. This is known as the 'right to rectification'. If your data is incomplete, you can ask the organisation to complete it by adding more details.

To get your data corrected, you should submit a request through the contact email listed below.

**Your right to get your data deleted**
You have the right to get your data deleted, also known as the 'right to erasure'. You can ask any organisation that holds data about you to delete that data. In some circumstances, they must then do so. You may sometimes hear this called the 'right to be forgotten'. However, this right only applies in the following circumstances:

- *The organisation no longer needs your data for the original reason they collected or used it for.*
- *You initially consented to the organisation using your data but have now withdrawn your consent.*
- *You have objected to using your data, and your interests outweigh those of the organisation using it.*
- *You have objected to the use of your data for direct marketing purposes.*
- *We have collected or used your data unlawfully.*
- *The organisation has a legal obligation to erase your data.*
- *The data was collected from you as a child for an online service.*

To get your data deleted, you should submit a request through the contact email listed below.

**Your right to raise a concern with an organisation**
You have the right to be confident that organisations handle your personal information responsibly and in line with good practice. If you have a concern about the way we are handling your information; if it:

- *It is not keeping your information secure.*
- *Holds inaccurate information about you.*
- *Has disclosed information about you.*
- *Is keeping information about you for longer than is necessary; or*
- *Has collected information for one reason and is using it for something else.*

To raise a concern, you should submit a request through the contact email listed below.

For more information, please see *the ICO website* (https://ico.org.uk/).

Contact for further information:

Researcher:
Teck Xun Tan w20003691@northumbria.ac.uk

Supervisor:
Kay Rogage k.rogage@northumbria.ac.uk

Records and Information Officer at Northumbria University:
Duncan James dp.officer@northumbria.ac.uk

You can find out more about how we use your information at:
www.northumbria.ac.uk/about-us/leadership-governance/vice-chancellors-office/legal-services-team/gdpr/gdpr---privacy-notices/ or by contacting a member of the project team

# Appendix 4: Consent Form

## CONSENT FORM

Northumbria University NEWCASTLE

**Project Title:** Smart Plant Caring System using IoT
**Student Name:** Teck Xun Tan
**Student ID No. (if applicable):** W20003691

**If you would like to take part in this study, please read and tick the boxes below.**

| | |
|---|---|
| I have carefully read and understood the Participant Information Sheet. | ☐ |
| I have had an opportunity to ask questions and discuss this study, and I have received satisfactory answers. | ☐ |
| I understand I am free to withdraw from the study at any time, without giving a reason for withdrawing and without prejudice. | ☐ |
| I agree to take part in this study.<br>I also consent to the retention of this data under the condition that any subsequent use is restricted to research projects that have gained ethical approval from Northumbria University. | ☐ |
| I am aware that my name will be kept confidential and will not affect my treatment/education/care. | ☐ |
| I agree to provide information to the investigator and understand that my contribution will remain anonymous and confidential. | ☐ |
| I agree to the University of Northumbria at Newcastle recording and processing this information about me. I understand that this information will be used only for the purpose(s) set out in the information sheet supplied to me. My consent is conditional upon the University complying with its duties and obligations under the Data Protection Act 2018, which incorporates General Data Protection Regulations (GDPR). You can find out more about how we use your information here - Privacy Notices | ☐ |

## Participant

Name/Signature of participant: _____

Date: _____

---

***This section below should only be signed by the investigator***

## Investigator

Name of investigator: Teck Xun Tan

Signature:

_____

Date: 28 October 2021

# Appendix 5: Survey Responses from Participants

## Survey Questionnaires Collected Response

**IMPORTANT!**

Please note that the survey result in this document has removed the participant's name and email collected in accordance with the Data Protection Act (2018) and one of the guideline written in the Participant Information Sheet as shown below:

*"Under the EU's new General Data Protection Regulation, if we are collecting personal data indirectly, we will have to be transparent about what categories of personal data will be collected and processed in the study. In this study, the participant's name and email is the only personal data necessary to be collected for smoother processing and analysing of the personal data. Participants' names and email are meant only for us and will always remain confidential, and they will not appear or be written in any reports or documentation."*

The responses have been extracted from the Google Form Response and formatted into this document.

The survey questionnaires could be found in the Word Document located within the Survey Questionnaire folder or by using the link: https://forms.gle/HL4CR49uBqPERuxk8

### Page 3: Are these information useful on the dashboard?

**(1 = Not Useful, 5 = Very Useful)**

| Participants | Soil Moisture Level | Environment Temperature | Light Intensities | Humidity | Water Pump Status |
|---|---|---|---|---|---|
| Participant 1 | 5 | 4 | 3 | 5 | 4 |
| Participant 2 | 5 | 5 | 5 | 5 | 3 |
| Participant 3 | 5 | 5 | 5 | 5 | 2 |
| Participant 4 | 5 | 5 | 5 | 5 | 1 |
| Participant 5 | 5 | 4 | 5 | 5 | 2 |

**Page 4: Which visualising method do you think suits you the best?**

| Participants | Selected Visualising Method |
|---|---|
| Participant 1 | Traffic Light Face |
| Participant 2 | Traffic Light Face |
| Participant 3 | Traffic Light Face |
| Participant 4 | Traffic Light Face |
| Participant 5 | Traffic Light Face |

**Page 4: Based on your choices, what do you like about that visualising method?**

| Participants | User Feedback |
|---|---|
| Participant 1 | I feel like I can just look at the image to know what is happening to my plant, so I think it's a really good method |
| Participant 2 | The image is kind of cute in my opinion, I really like the style it is using, and I think I can easily read the status of my plant from the faces |
| Participant 3 | I think most of the user will choose the Face visual method because user can understand the status of the plant easily by just looking at the image shown by the dashboard rather than looking at the graph which may take a second before understanding the status in my opinion |
| Participant 4 | It is really easy to read |
| Participant 5 | I really like the images better because I feel like they can let me know my plant's status easily without taking a long time |

**Page 5: In your opinion, do you think this function is useful for this product?**

| Participants | Activating water pump at anytime | Turning on/off UV light at anytime |
|---|---|---|
| Participant 1 | 5 | 4 |
| Participant 2 | 4 | 2 |
| Participant 3 | 1 | 5 |
| Participant 4 | 3 | 5 |
| Participant 5 | 1 | 5 |

## Page 5: Based on your understanding of the system, what would you use that dashboard for?

| Participants | User Feedback |
|---|---|
| Participant 1 | I will use that dashboard to learn the status of my plants and keep track on my plant health, really good idea |
| Participant 2 | I feel like I will be using that dashboard mostly on learning my plant status. I like the idea of users being able to activating or deactivating watering process and UV light, but I don't really think they are necessary if they are going to be automated. Finally, I think all the data that will be collected and shown in the dashboard are useful and are mostly critical to taking care of a plant. |
| Participant 3 | I have looking for a product that could both automatic water my plant and have a quick way to show me current status of the plant and I think this product really suits me well. I will use this dashboard to view all the necessary data that can help me in further taking care of my plant. Although all of the data selected has been really useful, the only data that I find not that useful is the pump status since that is not a useful information on plant caring. Other than that, I find that user have the ability to turn on or off the UV light manually is an interesting addition since not everyone would like the UV light to be turned on 24/7. Lastly, I think ability to turn on watering process is not really necessary since its automated and it will most likely cause overwatering if its controlled manually by the user. |
| Participant 4 | It is really easy to read |
| Participant 5 | I like the idea of the dashboard because I can learn how my plants are doing by looking at the data easily. |

**Page 5: What other functionality do you think we should implement in the Plant Caring System?**

| Participants | User Feedback |
|---|---|
| Participant 1 | It will be great if we could upload a custom photo of our plant or change the name and description of the plant for further personalisation |
| Participant 2 | I think the data collected are sufficient to show the status of the plant, however it will be great if we could see the current weather of the location we are currently on, so that we could further study or change the way we take care of our plants under different weather condition. |
| Participant 3 | I have no idea for another functionality to be honest, but I sure hope to see that the dashboard is easy to learn. |
| Participant 4 | Not really a function but I think users' data should be well protected |
| Participant 5 | Maybe we could have an option where we could see the raw data collected such as their time and date collected and the exact value |

# Appendix 6: Functional Requirements

## Functional Requirements

### System Dashboard

| # | Requirement | Priority | Implemented? |
|---|---|---|---|
| F1 | User should be able to login into the dashboard | Must | Yes |
| F2 | User should be able to register for a new account | Would | No |
| F3 | User should be able to manage (add, edit, remove) thrit plant | Would | No |
| F4 | User should be able to view the raw data | Should | Yes |
| F5 | User should be able to turn on UV light through dashboard | Must | Yes |
| F6 | User should be able to turn on watering process through dashboard | Would | No |
| F7 | User should be able to view the sensor captured data through dashboard | Must | Yes |
| F8 | System should be able to retrieve sensor data through implemented API | Must | Yes |
| F9 | System should be able to display weather forecast | Should | Yes |
| F10 | System should be able to retrieve weather forecast through external API | Should | Yes |
| F11 | User should be able to customise the plant information | Should | Yes |
| F12 | User should be able to upload a new image in plant information | Should | Yes |
| F13 | User should be able to logout from their account | Must | Yes |
| F14 | User should be able to change their password | Could | No |
| F15 | System should be able to authenticate a user | Must | Yes |
| F16 | System should be able to generate token through JSON Web Token | Must | Yes |
| F17 | System should be able to retrieve data from database | Must | Yes |
| F18 | System should be able to access the session storage | Must | Yes |
| F19 | System should be able to handle API request | Must | Yes |

### System Hardware

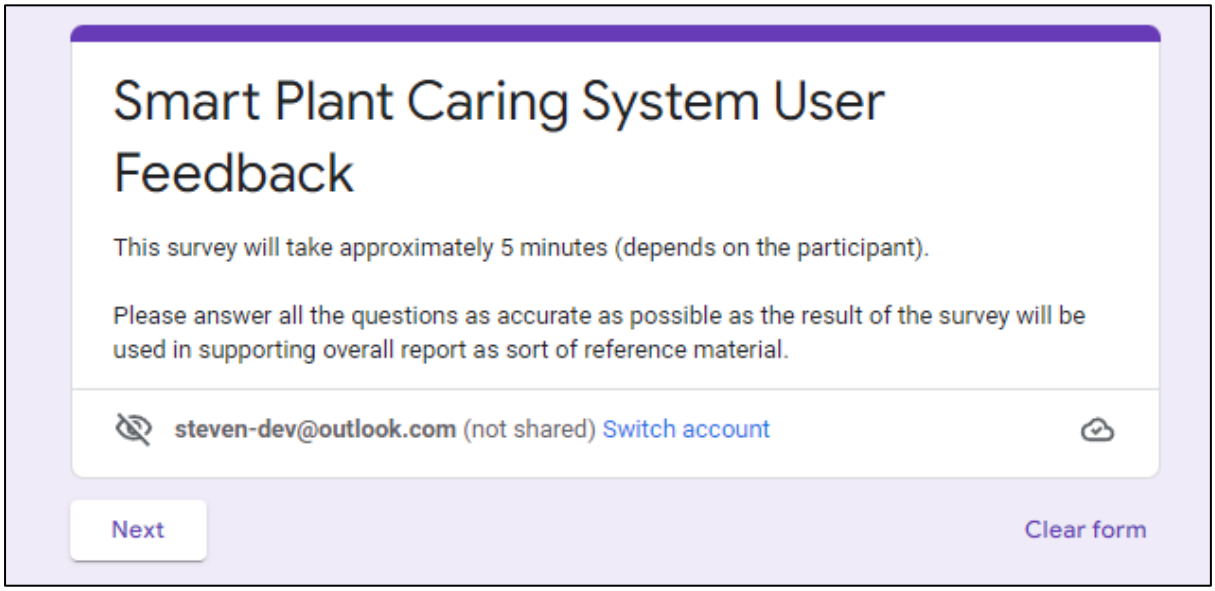
| # | Requirement | Priority | Implemented? |
|---|---|---|---|
| F20 | System should be able to capture soil moisture level with sensor | Must | Yes |
| F21 | System should be able to capture light intensity level with sensor | Should | Yes |
| F22 | System should be able to capture humidity with sensor | Could | Yes |
| F23 | System should be able to capture temperature with sensor | Could | Yes |
| F24 | System should be able to turn on UV light on manually | Should | Yes |
| F25 | System should be able to turn on plant watering process manually | Would | No |
| F26 | System should be able to publish sensor data | Must | Yes |
| F27 | System should be able to subscibe sensor data | Must | Yes |
| F28 | System should be able to connect to the internet | Must | Yes |
| F29 | System should be able to connect to MQTT Broker | Must | Yes |
| F30 | Sensors should be connected to the microcontroller | Must | Yes |
| F31 | Actuators should be connected to the microcontroller | Must | Yes |

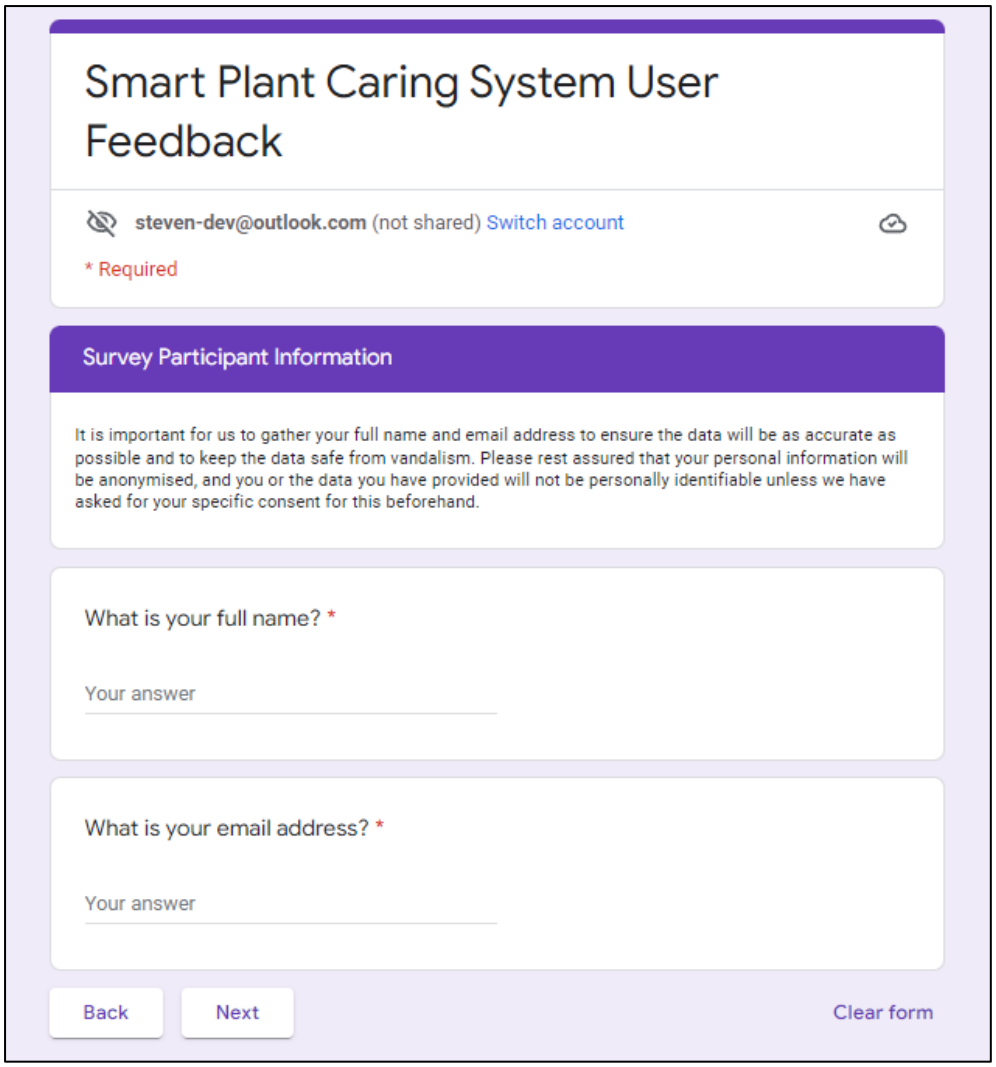
# Appendix 7: Non-Functional Requirements

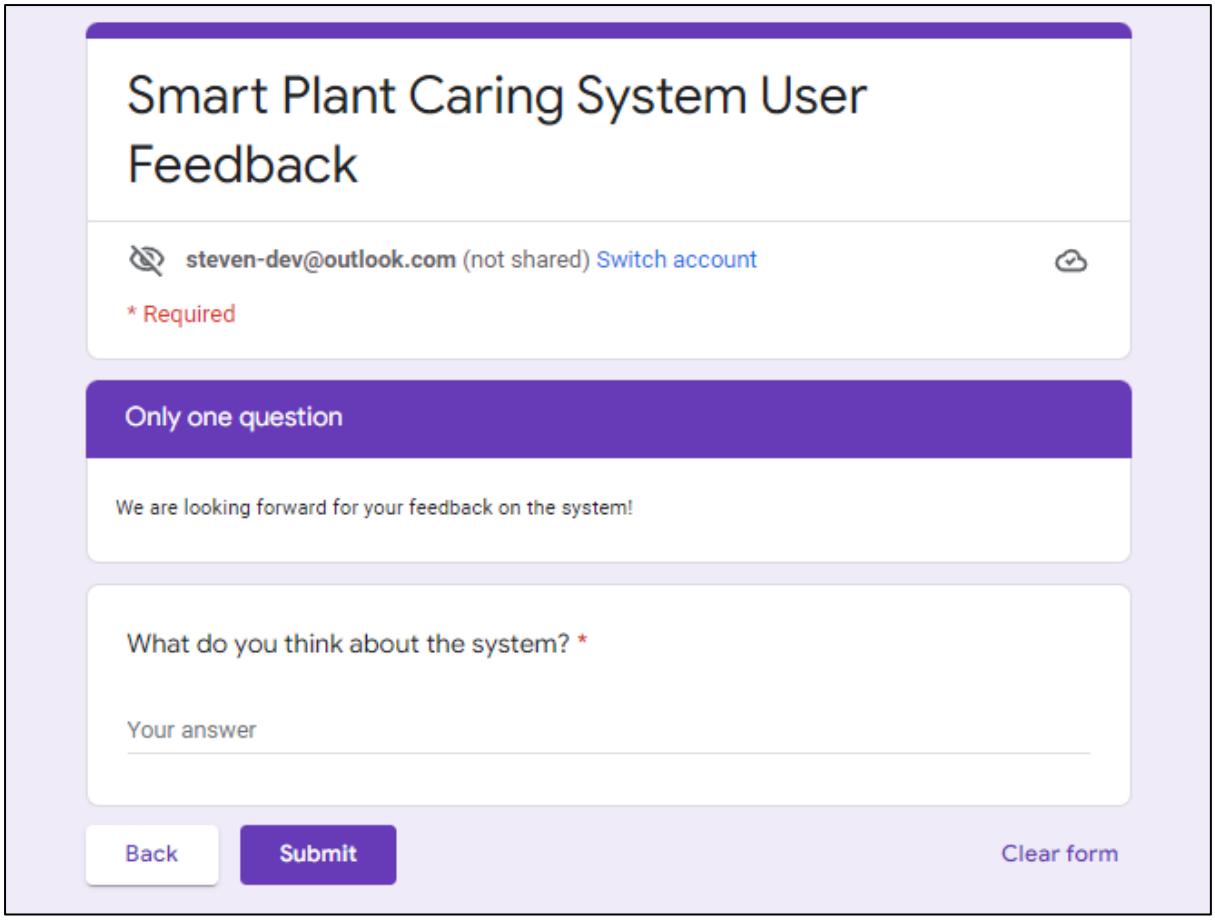
## Non-Functional Requirements

| # | Requirement | Heuristic(s) Applied | Priority | Implemented? |
|---|---|---|---|---|
| NF1 | User Interface should be consistent across different pages | UH4 | Must | Yes |
| NF2 | Dashboard should handle error appropriately | UH5 | Must | Yes |
| NF3 | Dashboard webpage should be in human readable language | UH2, UH5, UH9 | Must | Yes |
| NF4 | User Interface shouls keep the design as simple as possible | UH8 | Should | Yes |
| NF5 | Dasshboard should have a responsive design | UH7 | Should | Yes |
| NF6 | Dashboard should handle cancellation action appropriately | UH3 | Must | Partially |
| NF7 | Dashboard should be well performed | UH1 | Must | Yes |
| NF8 | Dashboard should have an API documentation | UH10 | Would | No |
| NF9 | Dashboard should have a design that is easy to recognise | UH6 | Should | Yes |
| NF10 | System should handle user data securely | - | Must | Yes |
| NF11 | System should be easy to learn to improve learnability | - | Must | Yes |
| NF12 | System should be user friendly towards people with disabilities | - | Should | Yes |
| NF13 | System should be safe to use by user | - | Must | Yes |

# Appendix 8: User Feedback Survey Form

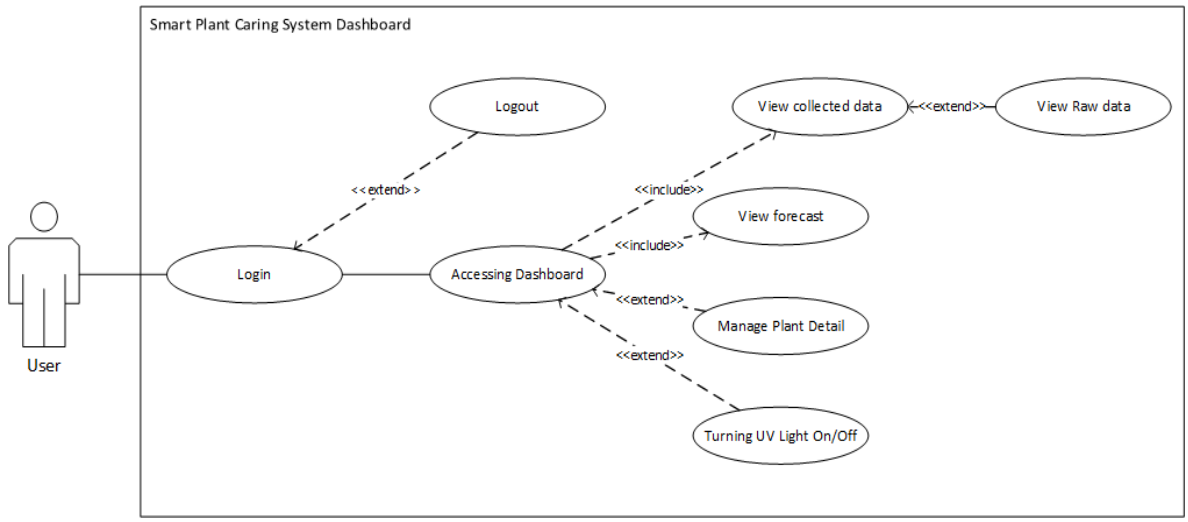


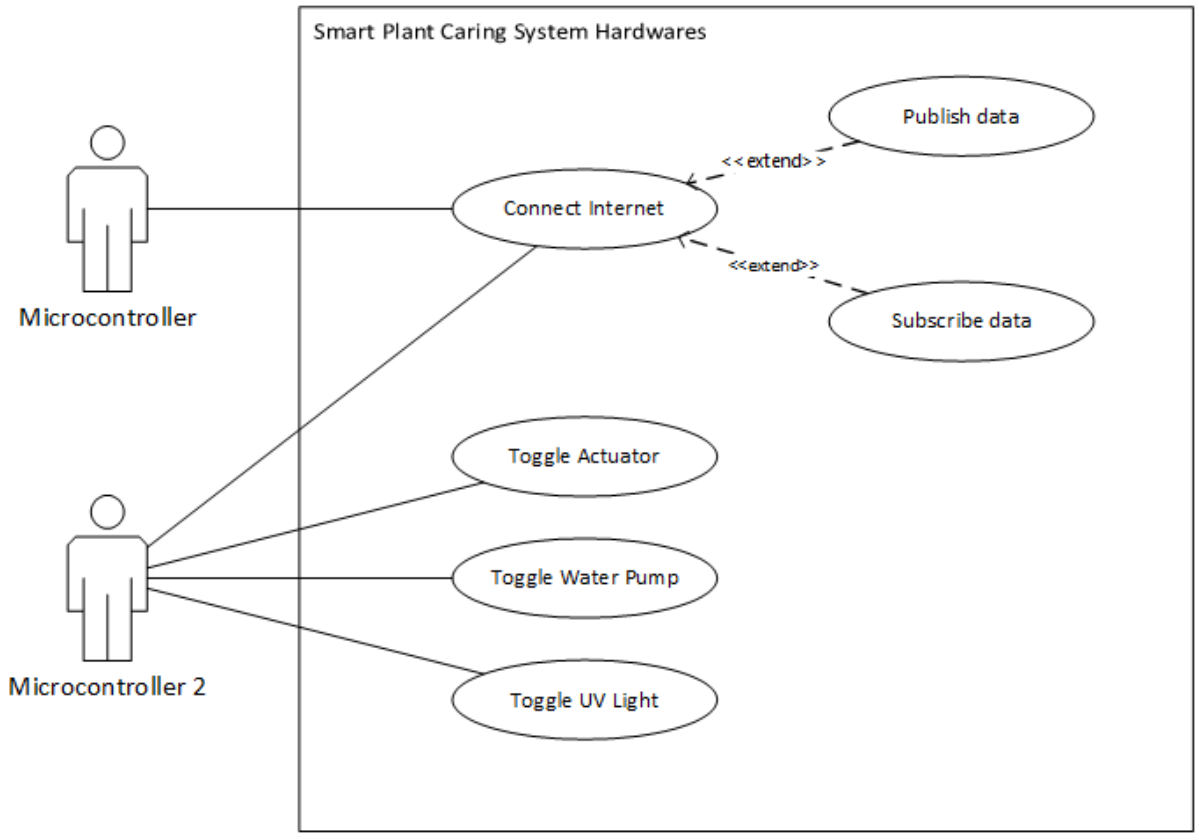

Smart Plant Caring System User Feedback
steven-dev@outlook.com (not shared) Switch account
* Required
Only one question
We are looking forward for your feedback on the system!
What do you think about the system? *
Your answer
Back
Submit
Clear form

# Appendix 9: Use Case Diagram

## Use Case: User to SPCS Dashboard




## Use Case: Microcontroller to SPCS Hardware



# Appendix 10: Use Case Descriptions Documentation

## Use Case: User to SPCS Dashboard

**Use Case Description 1**

| | |
|---|---|
| **Use Case** | Login |
| **Use Case Description** | User may login into the system to access the dashboard to their plant. |
| **Actor** | User |
| **Precondition** | - |
| **Basic Flows** | • The user clicks on the Dashboard Portal button<br>• The system prompted user with a login page<br>• The user enters their credentials<br>• The system validates the credentials<br>• The system generates an authentication token<br>• The user is directed into the dashboard page |
| **Alternative Path** | **1. The user entered an incorrect credential**<br>The system displays an error message, "Incorrect Username/Password".<br><br>**2. The system fails to generate an authentication token**<br>The user is redirected into an error page with error 500.<br><br>**3. The user is already logged in**<br>The user is redirected into the dashboard page. |

**Use Case Description 2**

| | |
|---|---|
| **Use Case** | Accessing Dashboard |
| **Use Case Description** | User may access the dashboard to view all the collected sensor data uploaded by the microcontroller including soil moisture, light intensity, water sprinkler status and UV light status |
| **Actor** | User |
| **Precondition** | The user must be logged in |
| **Basic Flows** | • The user clicked on the Dashboard Portal button<br>• The user is directed into the dashboard page<br>• The system verifies if an authentication token is presented<br>• The system displays the dashboard page<br>• The user views the collected data displayed on the dashboard |
| **Alternative Path** | **1. The system fails to find an authenticated token**<br>The user is redirected into an error page with error 401<br><br>**2. The system detects that the token has expired**<br>The user is redirected into an error page with error 401..<br><br>**3. The system fails to display the dashboard page**<br>The user is redirected into an error page with error 500. |

**Use Case Description 3**

| | |
|---|---|
| **Use Case** | View Collected Data |
| **Use Case Description** | User may view all the collected sensor data uploaded by the microcontroller within the dashboard |
| **Actor** | User |
| **Precondition** | The user must be logged in |
| **Basic Flows** | • The user clicked on the Dashboard Portal button<br>• The system displays the dashboard page<br>• The user views the collected data displayed on the dashboard |
| **Alternative Path** | **1. The system fails to display the dashboard page**<br>The user is redirected into an error page with error 500.<br><br>**2. The system fails to retrieve the collected data**<br>The user is redirected into an error page with error 500. |

**Use Case Description 4**

| | |
|---|---|
| **Use Case** | View Forecast |
| **Use Case Description** | User may view the forecast retrieved from OpenWeatherAPI |
| **Actor** | User |
| **Precondition** | The user must be logged in |
| **Basic Flows** | • The user clicked on the Dashboard Portal button<br>• The system displays the dashboard page<br>• The user views the current forecast |
| **Alternative Path** | **1. The system fails to display the dashboard page**<br>The user is redirected into an error page with error 500.<br><br>**2. The system fails to retrieve the forecast**<br>The user is redirected into an error page with error 500. |

**Use Case Description 5**

| | |
|---|---|
| **Use Case** | Manage Plant Detail |
| **Use Case Description** | User may personalise their plant by changing the information such as plant name, description, and image |
| **Actor** | User |
| **Precondition** | The user must be logged in |

| | |
|---|---|
| **Basic Flows** | • The user clicked on the Dashboard Portal button<br>• The system displays the dashboard page<br>• The user clicks on the "Change Plant Information" button<br>• The system displays a "Change Plant Detail" modal<br>• The system request API endpoint to retrieve plant information<br>• The system populates the modal with the retrieve data<br>• The user entered the plant name and description<br>• The user uploads a new image<br>• The user clicks on "Save Information"<br>• The system updates the database<br>• The system refreshes the page<br>• The user is presented with an updated plant detail |
| **Alternative Path** | **1. The system fails to display the dashboard page**<br>The user is redirected into an error page with error 500.<br><br>**2. The user leaves an empty field**<br>The system displays an error message, "Please fill in this field".<br><br>**3. The user uploaded a non-image file**<br>The system displays an error message, "File extension not supported! Please try again!".<br><br>**4. The system fails to update the plant information**<br>The user is redirected into an error page with error 500. |

**Use Case Description 6**

| | |
|---|---|
| **Use Case** | Turning UV Light On/Off |
| **Use Case Description** | User may toggle the UV light directly through the dashboard |
| **Actor** | User |
| **Precondition** | The user must be logged in |
| **Basic Flows** | • The user clicked on the Dashboard Portal button<br>• The system displays the dashboard page<br>• The user clicks on the "UV Light Switch" toggle button<br>• The system calls the update UV status API endpoint<br>• The system turns on the UV light |
| **Alternative Path** | **1. The system fails to display the dashboard page**<br>The user is redirected into an error page with error 500.<br><br>**2. The system API endpoint fail to execute**<br>The user is redirected into an error page with error 500. |

**Use Case Description 7**

| | |
|---|---|
| **Use Case** | View Raw data |
| **Use Case Description** | User may view the data collected in raw format by accessing this feature |
| **Actor** | User |
| **Precondition** | The user must be logged in |
| **Basic Flows** | • The user clicked on the Dashboard Portal button<br>• The system displays the dashboard page<br>• The user clicks on the "View Raw Data" button<br>• The system displays a "Raw Data" modal<br>• The system request API endpoint to retrieve raw data<br>• The system populates the modal with retrieved data<br>• The user views the raw data |
| **Alternative Path** | **1. The system fails to display the modal**<br>The user is redirected into an error page with error 500.<br><br>**2. The system API endpoint fail to execute**<br>The user is redirected into an error page with error 500. |

**Use Case Description 8**

| | |
|---|---|
| **Use Case** | Logout |
| **Use Case Description** | User may logout their account from the system |
| **Actor** | User |
| **Precondition** | The user must be logged in |
| **Basic Flows** | • The user clicks on the "Logout" button<br>• The system request API endpoint to logout<br>• The user is directed into the homepage<br>• The system displays the homepage |
| **Alternative Path** | **1. The system API endpoint fail to execute**<br>The user is redirected into an error page with error 500. |

## Use Case: Microcontroller to SPCS Hardware

**Use Case Description 9**

| | |
|---|---|
| **Use Case** | Connect Internet |
| **Use Case Description** | The microcontroller startup and attempt to connect to the Internet |
| **Actor** | Microcontroller |
| **Precondition** | The microcontroller must be powered on |
| **Basic Flows** | • The microcontroller switched on<br>• The microcontroller attempts to connect to the internet using the specified network SSID and Password<br>• The microcontoller connected to the Internet |
| **Alternative Path** | **1. The microcontroller fails to connect to the Internet**<br>The microcontroller retries the process until it is connected |

**Use Case Description 10**

| | |
|---|---|
| **Use Case** | Publish data |
| **Use Case Description** | The microcontroller publishes the collected sensor data to MQTT |
| **Actor** | Microcontroller |
| **Precondition** | The microcontroller must be connected to the Internet |
| **Basic Flows** | • The microcontroller connected to MQTT<br>• The microcontroller collects sensor data<br>• The microcontroller publishes the sensor data |
| **Alternative Path** | **1. The microcontroller fails to connect to the MQTT**<br>The microcontroller retries the process until it is connected to MQTT<br><br>**2. The microcontroller fails to publish data to the MQTT**<br>The microcontroller skips the current data publish process, collect a new data, and retries the publish process. |

**Use Case Description 11**

| | |
|---|---|
| **Use Case** | Subscribe data |
| **Use Case Description** | The microcontroller subscribes a certain topic to retrieve the latest collected sensor data from MQTT |
| **Actor** | Microcontroller |
| **Precondition** | The microcontroller must be connected to the Internet |

| | |
|---|---|
| **Basic Flows** | • The microcontroller connected to MQTT<br>• The microcontroller subscribes to the sensor data topic<br>• The microcontroller retrieves the latest sensor data |
| **Alternative Path** | **1. The microcontroller fails to connect to the MQTT**<br>The microcontroller retries the process until it is connected to MQTT<br><br>**2. The microcontroller fails to retrieve data from MQTT**<br>The microcontroller skips the current data and retries the subscribe process. |

**Use Case Description 12**

| | |
|---|---|
| **Use Case** | Toggle actuator |
| **Use Case Description** | The microcontroller can toggle the actuator by turning it on or off |
| **Actor** | Microcontroller |
| **Precondition** | The actuator must be connected to the microcontroller |
| **Basic Flows** | • The microcontroller switched on<br>• The microcontroller retrieves data<br>• The microcontroller compares the data with a prefix value<br>• The microcontroller activates the actuator |
| **Alternative Path** | **1. The actuator is already turned on**<br>The microcontroller skips the turning on process<br><br>**2. The actuator is already turned off**<br>The microcontroller skips the turning off process |

**Use Case Description 13**

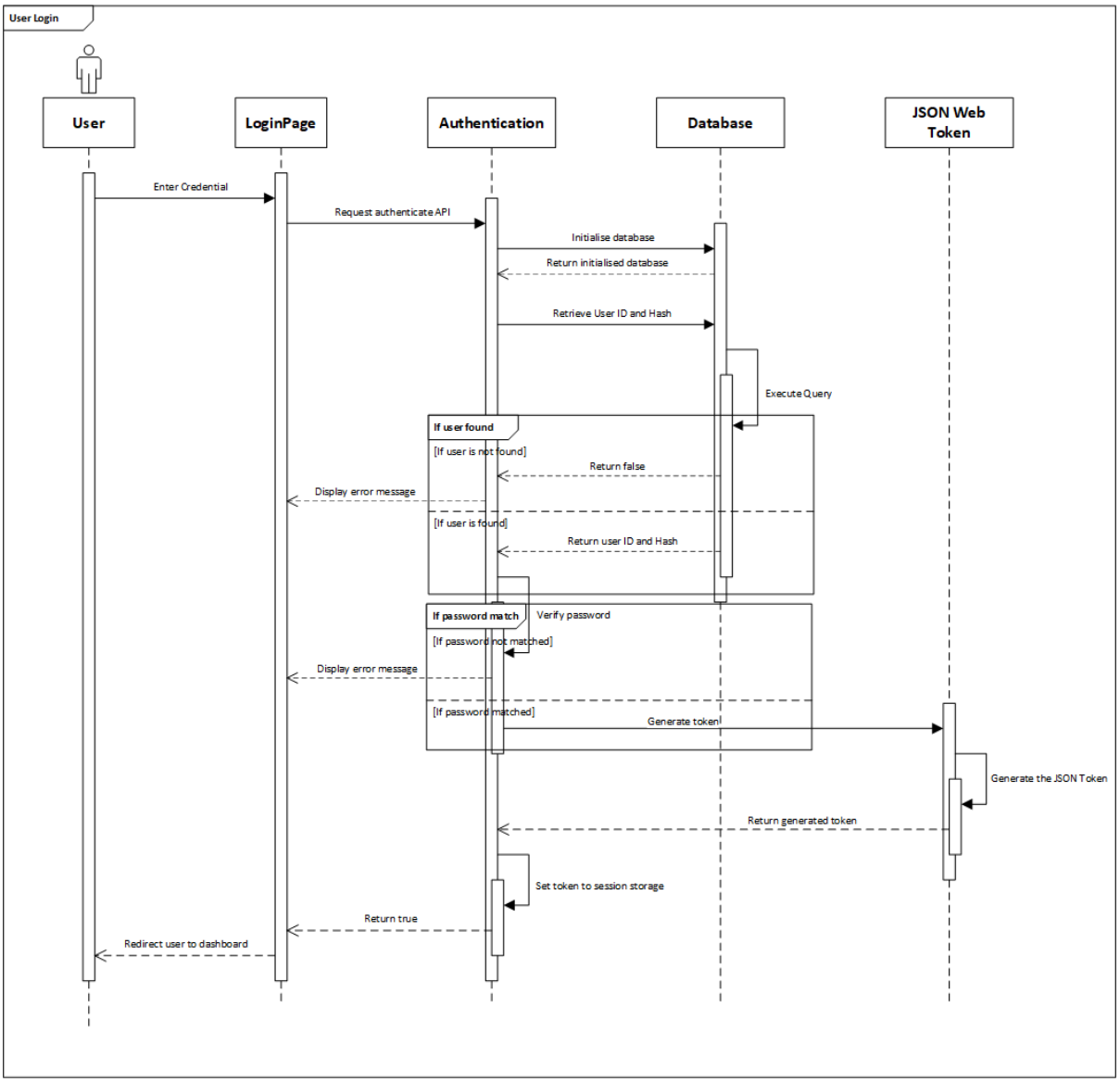
| | |
|---|---|
| **Use Case** | Toggle water pump |
| **Use Case Description** | The microcontroller can toggle the water pump by turning it on or off |
| **Actor** | Microcontroller |
| **Precondition** | The water pump must be connected to the microcontroller |
| **Basic Flows** | • The microcontroller switched on<br>• The microcontroller retrieves data<br>• The microcontroller compares the data with a prefix value<br>• The microcontroller activates the water pump |
| **Alternative Path** | **1. The water pump is already turned on**<br>The microcontroller skips the turning on process<br><br>**2. The water pump is already turned off**<br>The microcontroller skips the turning off process |

**Use Case Description 14**

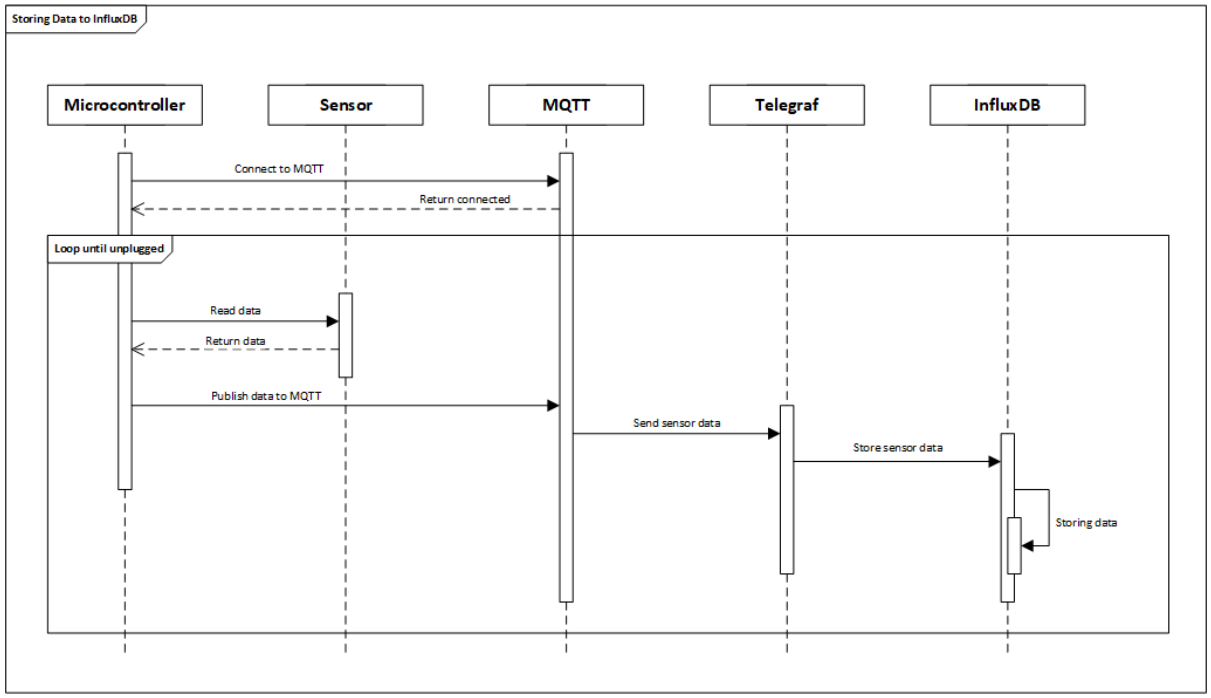
| | |
|---|---|
| **Use Case** | Toggle UV light |
| **Use Case Description** | The microcontroller can toggle the UV light by turning it on or off |
| **Actor** | Microcontroller |
| **Precondition** | The UV light must be connected to the microcontroller |
| **Basic Flows** | • The microcontroller switched on<br>• The microcontroller retrieves data<br>• The microcontroller compares the data with a prefix value<br>• The microcontroller activates the UV light |
| **Alternative Path** | **1. The UV light is already turned on**<br>The microcontroller skips the turning on process<br><br>**2. The UV light is already turned off**<br>The microcontroller skips the turning off process |

# Appendix 11: Sequence Diagram

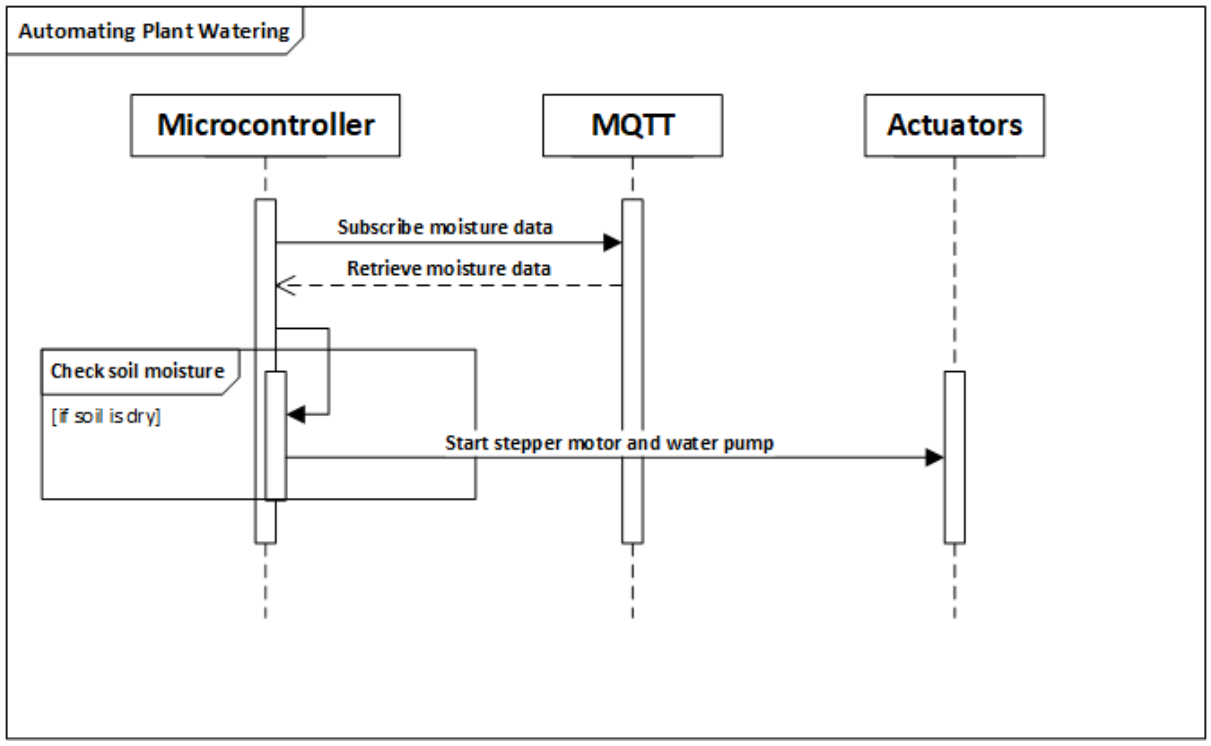
## User Login



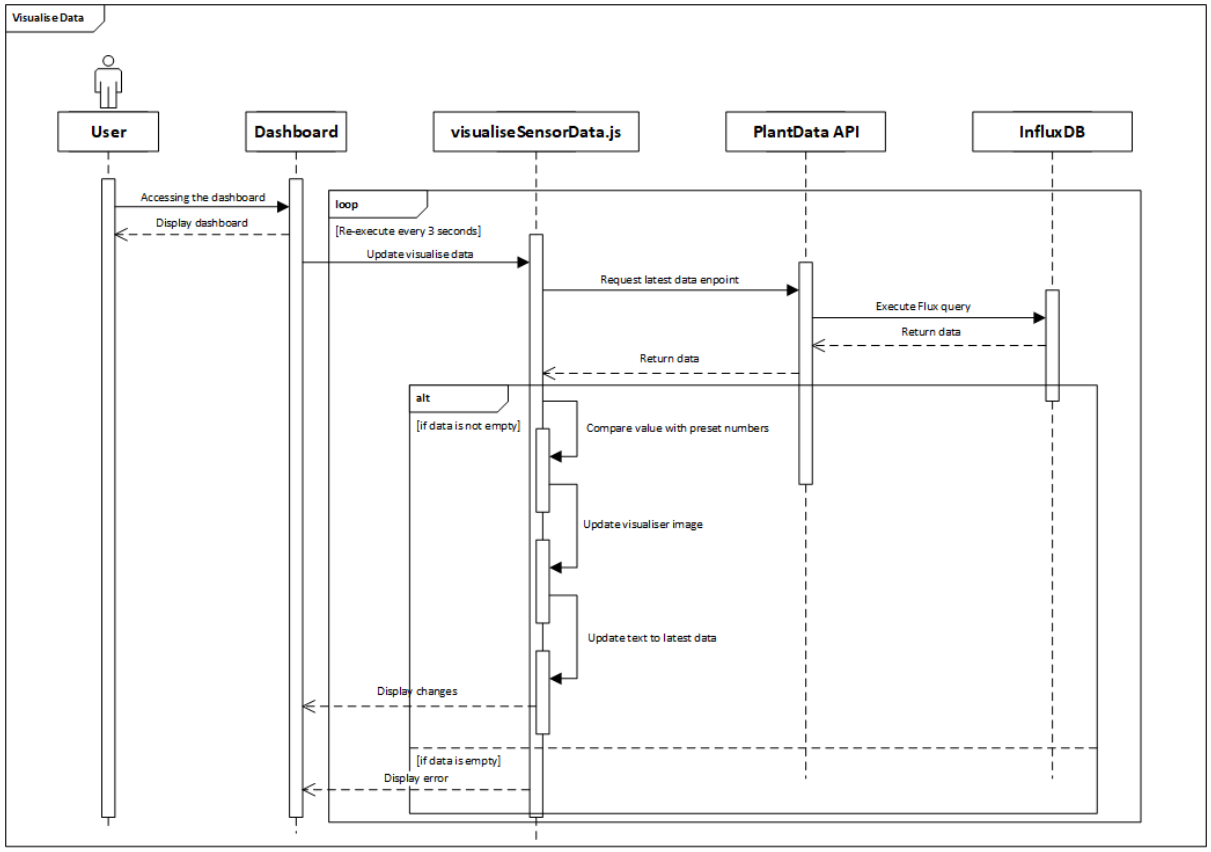
## Storing Data to InfluxDB



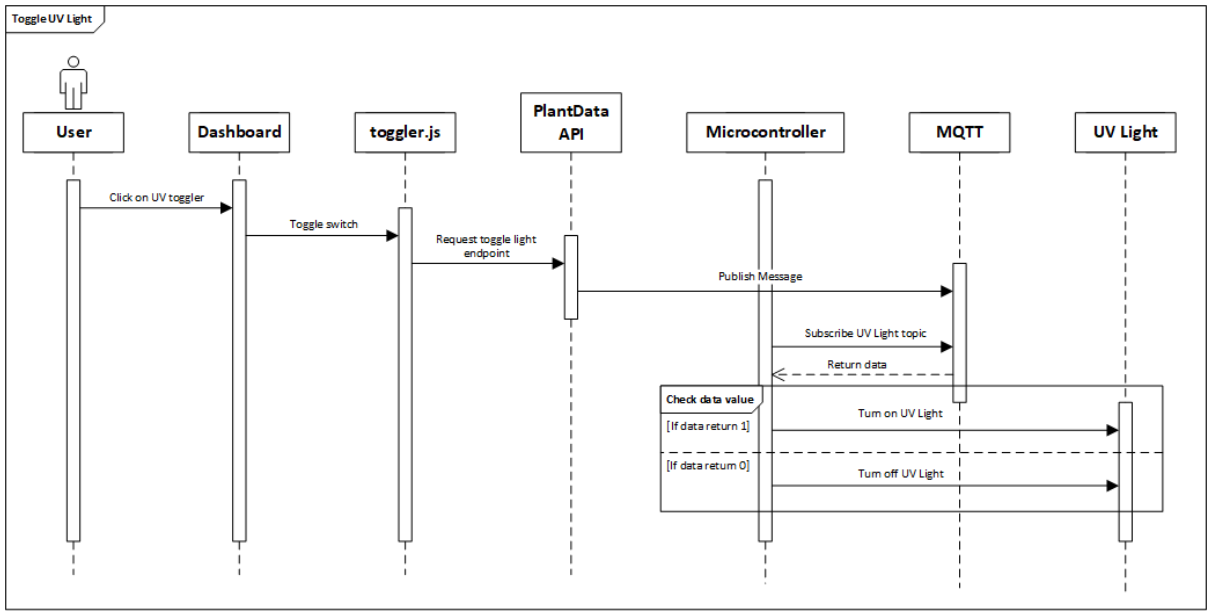

## Automate Watering Process



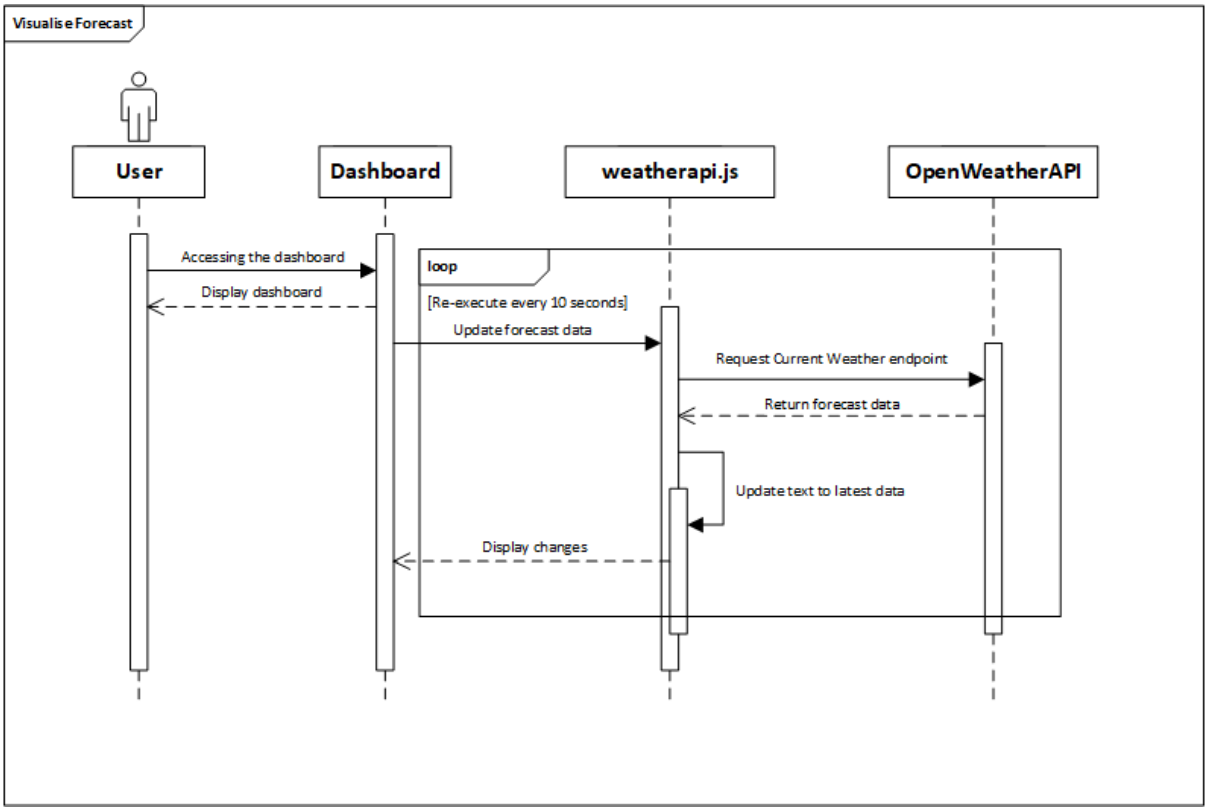
## Visualise Data



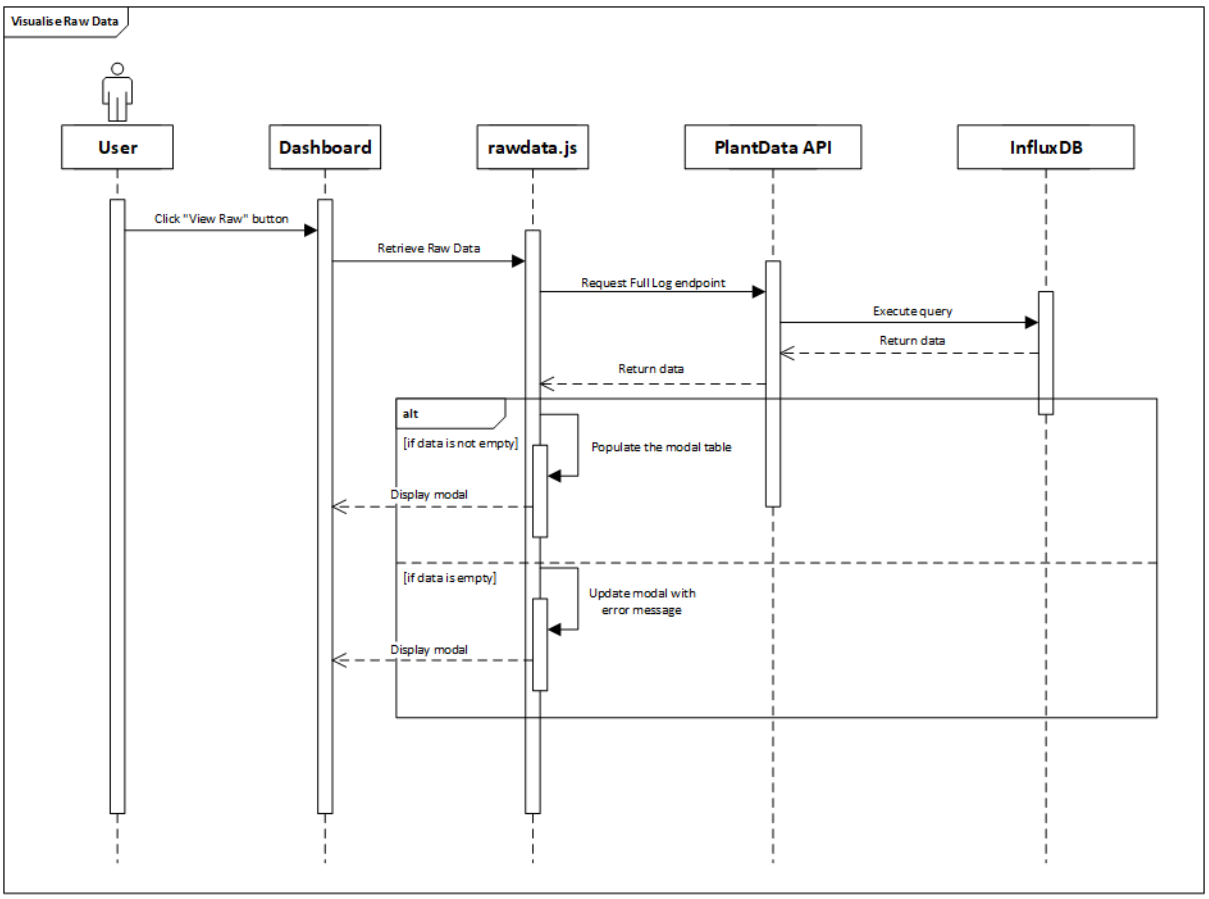

## Toggle UV Light



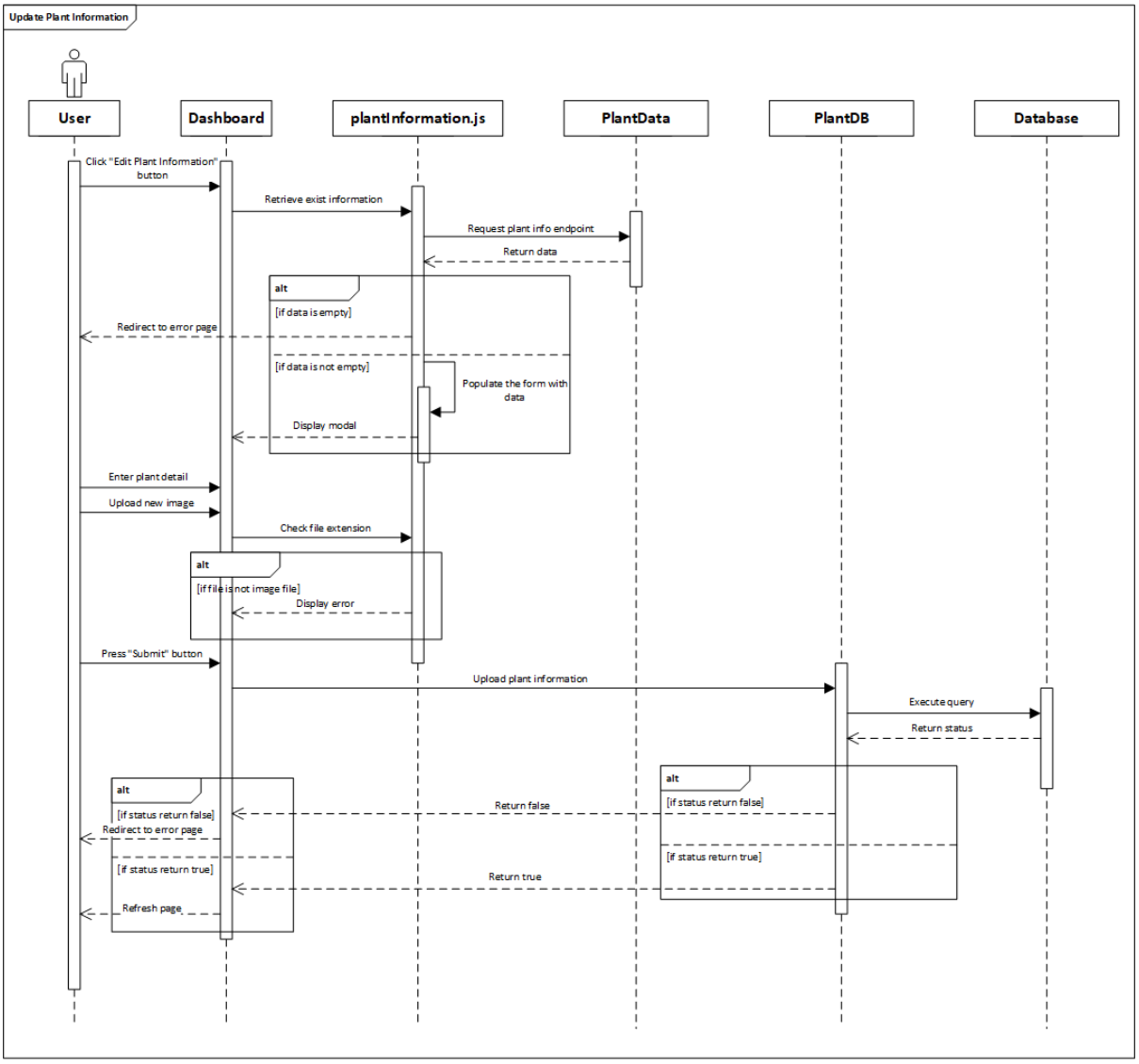
## Visualise Forecast



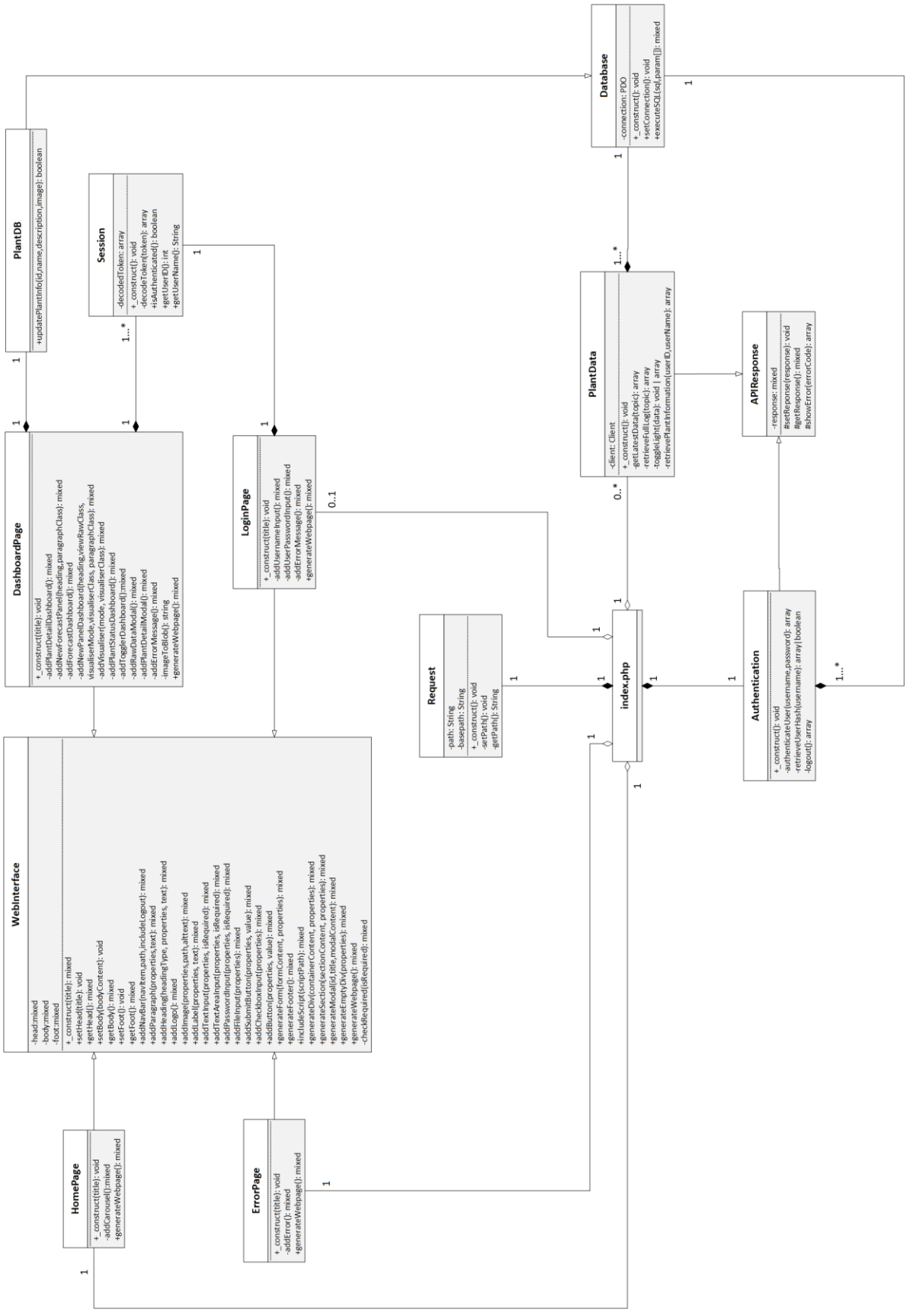

## Visualise Raw Data



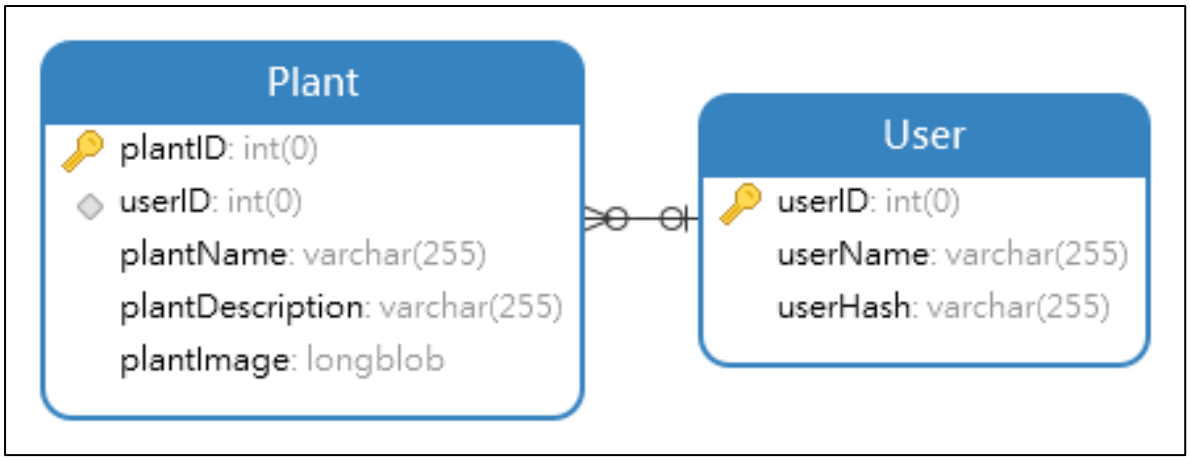
# Update Plant Information



# Appendix 12: Class Diagram



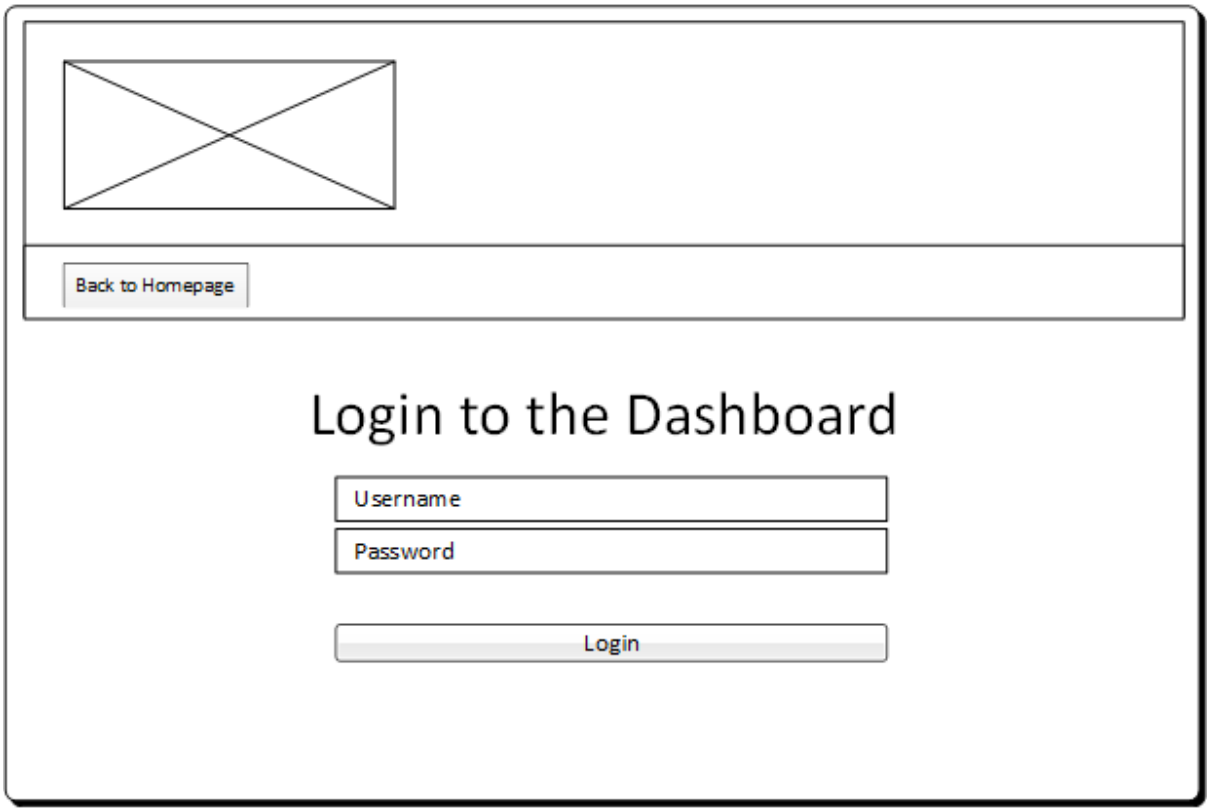
## Appendix 13: Entity Relationship Diagram (ERD)



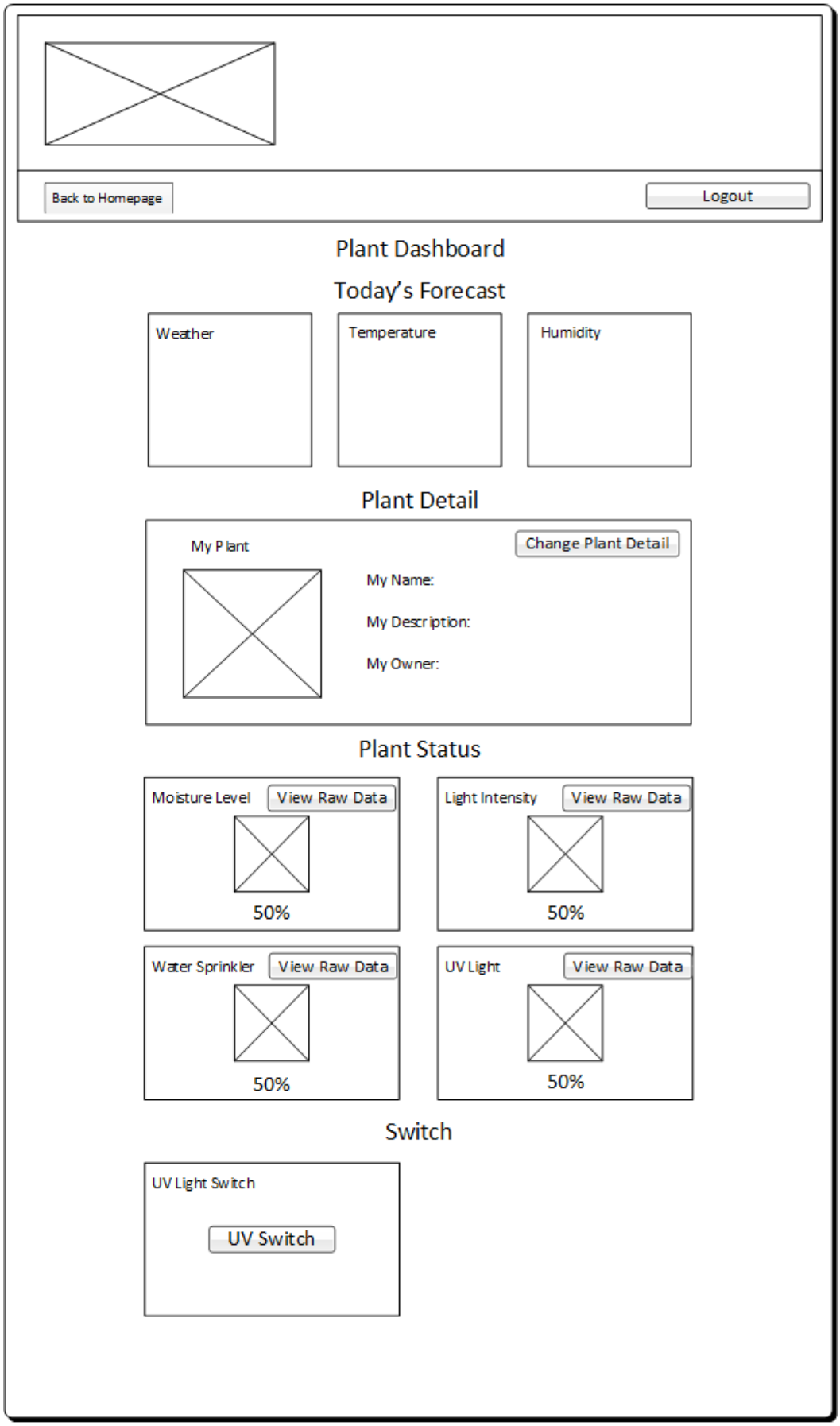

## Appendix 14: Low Fidelity

### Homepage

**Login Page**


Back to Homepage
Login to the Dashboard
Username
Password
Login

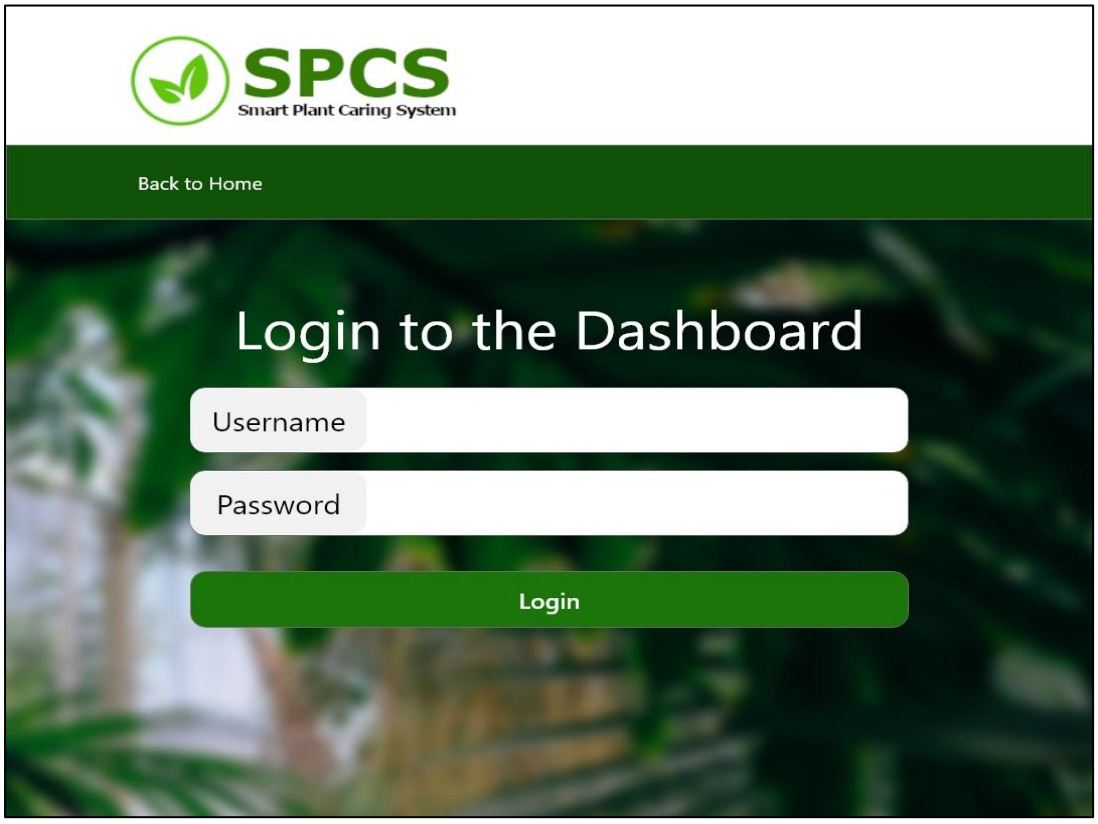
## Dashboard Page



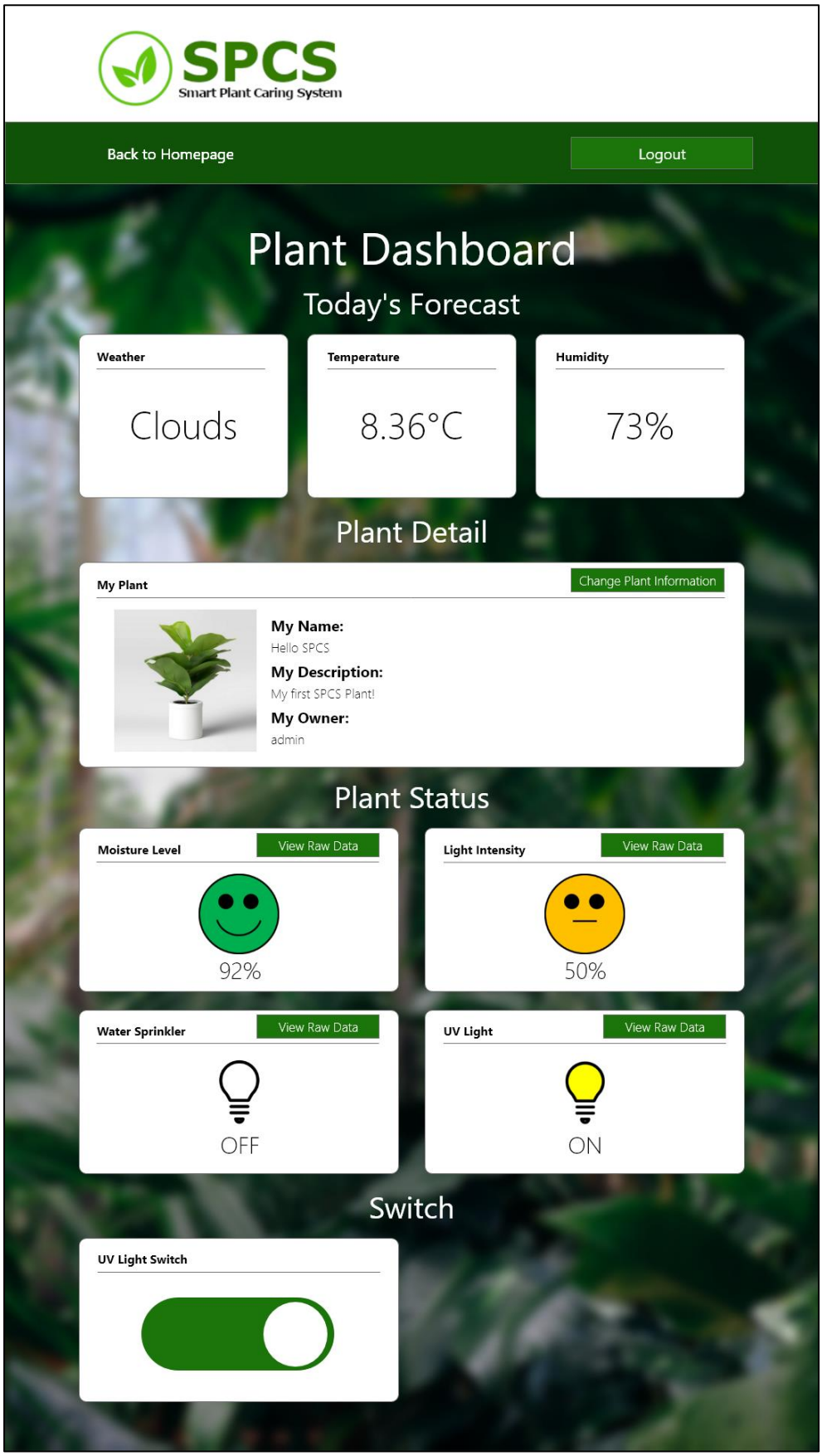
# Appendix 15: High Fidelity

## Homepage


## Login Page



## Dashboard Page



# Appendix 16: Functionality Testing Test Case and Result

**Functionality Testing (Use Case Testing)**

**Dashboard**

| Test Case ID | Use Case Description | Test Description | Test Input | Expected Outcome | Actual Outcome | Pass/Fail |
|---|---|---|---|---|---|---|
| 1 | 1 | Accessing the login screen | Click on 'Dashboard Portal' button | Redirect to login page | Page redirected successfully | Pass |
| 2 | 1 | Username is black | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass |
| 3 | 1 | Password is blank | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass |
| 4 | 1 | Username is incorrect | Username: **test** | Error message "Incorrect Username/Password. Please try again" is displayed | Error message displayed | Pass |
| 5 | 1 | Password is incorrect | Password: **test** | Error message "Incorrect Username/Password. Please try again" is displayed | Error message displayed | Pass |
| 6 | 1 | Correct credentials | Username: **admin** Password: **admin123** | Redirect to Dashboard page | Page redirected successfully | Pass |
| 7 | 1 | User is already logged in | Click on 'Dashboard Portal' button | Redirect to Dashboard page | Page redirected successfully | Pass |
| 8 | 2 | User is not logged in/No valid token found | http://192.168.0.216/index.php/dashboard | Redirect to login page | Redirected to the dashboard page | Fail |
| 9 | 2 | Token is expired | Click on 'Dashboard Portal' button | Redirect to login page | Page redirected successfully | Pass |
| 10 | 3 | Display collected data | - | All sensor data are displayed | Data displayed correctly | Pass |
| 11 | 4 | Display current forecast | - | Current forecasts are displayed | Forecast displayed correctly | Pass |
| 12 | 5 | Accessing the "Change plant information" function | Click on 'Change Plant Information' button | Plant Detail Modal is displayed | Modal is not opening correctly | Fail |
| 13 | 5 | Display exists plant information data | - | Plant data are displayed | Data displayed correctly | Pass |
| 14 | 5 | Plant name is blank | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass |
| 15 | 5 | Plant description is blank | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass |
| 16 | 5 | Upload file | File: **image.jpg** | Input accepts the file | File accepted correctly | Pass |
| 17 | 5 | Upload incorrect extension file | File: **image.exe** | Error message "File extension not supported! Please try again!" is displayed | Error message displayed | Pass |
| 18 | 5 | Update the information | Click on 'Save Information' | Information updated and refresh page | Information updated and page was refreshed | Pass |
| 19 | 6 | Toggling UV Light | Click on "UV Light Switch" toggle button | Update the toggle state | Toggle state updated | Pass |
| 20 | 6 | Updating UV Light Status | Click on "UV Light Switch" toggle button | Change state on UV Light panel | UV Light panel state changed | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| 21 | 7 | Accessing the "View Raw Data" function | Click on 'View Raw Data' button | Raw Data Modal is displayed | Modal displayed | Pass |
| 22 | 7 | Display raw data | - | Raw data are displayed | Raw data displayed correctly | Pass |
| 23 | 8 | User logout from the dashboard | Click on 'Logout' button | User logged out and redirect to home page | User logged out and page redirected successfully | Pass |

**Hardware**

| | | | | | | |
|---|---|---|---|---|---|---|
| 24 | 9 | Connecting to the internet | - | Connected to the internet | Connected to the internet successfully | Pass |
| 25 | 10 | Collect the sensor data | - | Sensor data collected | Sensor data collected successfully | Pass |
| 26 | 10 | Publishing data to MQTT | - | Data published to MQTT | Data published to MQTT successfully | Pass |
| 27 | 11 | Subscribe data from MQTT | - | Data retrieved from MQTT | Data retrieved from MQTT successfuly | Pass |
| 28 | 12 | Toggling actuator | - | Actuator toggled | Actuator toggled successfully | Pass |
| 29 | 13 | Toggling water pump | - | Water pump toggled | Water pump toggled successfully | Pass |
| 30 | 14 | Toggling UV light | - | UV light toggled | UV light toggled successfully | Pass |

**Functionality Testing (API Testing)**

| Test Case ID | API Endpoint | Test Description | Test Input | Expected Outcome | Actual Outcome | Pass/Fail |
|---|---|---|---|---|---|---|
| 31 | api/plant | User not authenticated | - | Error 401 is displayed | Error code displayed correctly | Pass |
| 32 | api/plant?latest | Parameter is not specified | - | Error 400 is displayed | Error code displayed correctly | Pass |
| 33 | api/plant?latest | Retrieve the latest data | Parameter: **topic=soil-moisture** | Latest data retrieved | Latest data retrieved successfully | Pass |
| 34 | api/plant?latest | Return null data | Parameter: **topic=testnull** | Error 204 is displayed | Error code displayed correctly | Pass |
| 35 | api/plant?full-log | Parameter is not specified | - | Error 400 is displayed | Error code displayed correctly | Pass |
| 36 | api/plant?full-log | Retrieve the full data log | Parameter: **topic=soil-moisture** | Full log data retrieved | Full log data retrieved successfully | Pass |
| 37 | api/plant?full-log | Return null data | Parameter: **topic=testnull** | Error 204 is displayed | Error code displayed correctly | Pass |
| 38 | api/plant?plant-info | Retrieve the full data log | - | Plant info retrieved | Plant info retrieved correctly | Pass |
| 39 | api/plant?toggle-light | Parameter is not specified | - | Error 400 is displayed | Error code displayed correctly | Pass |
| 40 | api/plant?toggle-light | Toggle UV light | Parameter: **switch=0** | UV light state updated | UV light state updated correctly | Pass |
| 41 | api/plant | Incorrect request method | Method: **POST** | Error 405 is displayed | Error code displayed correctly | Pass |
| 42 | api/authentication?authenticate | Parameter is not specified | - | Error 400 is displayed | Error code displayed correctly | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| 43 | api/authentication?authenticate | Username parameter is not specified | Body: **password=admin123** | Error 400 is displayed | Error code displayed correctly | Pass |
| 44 | api/authentication?authenticate | Password parameter is not specified | Body: **username=admin** | Error 400 is displayed | Error code displayed correctly | Pass |
| 45 | api/authentication?authenticate | Incorrect credential specified | Body: **username=test password=test123** | Error 204 is displayed | Error code displayed correctly | Pass |
| 46 | api/authentication?authenticate | Incorrect password specified | Body: **username=admin password=test123** | Error 401 is displayed | Error code displayed correctly | Pass |
| 47 | api/authentication?authenticate | Authenticate the user | Body: **username=admin password=admin123** | Session created and token generated | Session created and token generated successfully | Pass |
| 48 | api/authentication?logout | Logout the user | - | Session destroyed | Session destroyed successfully | Pass |

# Appendix 17: Cross Browser Testing Test Case and Result

**Cross Browser Testing (Use Case Testing)**

Dashboard

| Test Case ID | Use Case Description | Test Description | Test Input | Expected Outcome | Actual Outcome | Pass/Fail (Safari) | Pass/Fail (Firefox) |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Accessing the login screen | Click on 'Dashboard Portal' button | Redirect to login page | Page redirected successfully | Pass | Pass |
| 2 | 1 | Username is black | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass | Pass |
| 3 | 1 | Password is blank | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass | Pass |
| 4 | 1 | Username is incorrect | Username: **test** | Error message "Incorrect Username/Password. Please try again" is displayed | Error message displayed | Pass | Pass |
| 5 | 1 | Password is incorrect | Password: **test** | Error message "Incorrect Username/Password. Please try again" is displayed | Error message displayed | Pass | Pass |
| 6 | 1 | Correct credentials | Username: **admin** Password: **admin123** | Redirect to Dashboard page | Page redirected successfully | Pass | Pass |
| 7 | 1 | User is already logged in | Click on 'Dashboard Portal' button | Redirect to Dashboard page | Page redirected successfully | Pass | Pass |
| 8 | 2 | User is not logged in/No valid token found | http://192.168.0.216/index.php/dashboard | Redirect to login page | Redirected to the dashboard page | Pass | Pass |
| 9 | 2 | Token is expired | Click on 'Dashboard Portal' button | Redirect to login page | Page redirected successfully | Pass | Pass |
| 10 | 3 | Display collected data | - | All sensor data are displayed | Data displayed correctly | Pass | Pass |
| 11 | 4 | Display current forecast | - | Current forecasts are displayed | Forecast displayed correctly | Pass | Pass |
| 12 | 5 | Accessing the "Change plant information" function | Click on 'Change Plant Information' button | Plant Detail Modal is displayed | Modal is not opening correctly | Pass | Pass |
| 13 | 5 | Display exists plant information data | - | Plant data are displayed | Data displayed correctly | Pass | Pass |
| 14 | 5 | Plant name is blank | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass | Pass |
| 15 | 5 | Plant description is blank | - | Pop up error message "Please fill in this field" is displayed | Popup error message displayed | Pass | Pass |
| 16 | 5 | Upload file | File: **image.jpg** | Input accepts the file | File accepted correctly | Pass | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 17 | 5 | Upload incorrect extension file | File: **image.exe** | Error message "File extension not supported! Please try again!" is displayed | Error message displayed | Pass | Pass |
| 18 | 5 | Update the information | Click on 'Save Information' | Information updated and refresh page | Information updated and page was refreshed | Pass | Pass |
| 19 | 6 | Toggling UV Light | Click on "UV Light Switch" toggle button | Update the toggle state | Toggle state updated | Pass | Pass |
| 20 | 6 | Updating UV Light Status | Click on "UV Light Switch" toggle button | Change state on UV Light panel | UV Light panel state changed | Pass | Pass |
| 21 | 7 | Accessing the "View Raw Data" function | Click on 'View Raw Data' button | Raw Data Modal is displayed | Modal displayed | Pass | Pass |
| 22 | 7 | Display raw data | - | Raw data are displayed | Raw data displayed correctly | Pass | Pass |
| 23 | 8 | User logout from the dashboard | Click on 'Logout' button | User logged out and redirect to home page | User logged out and page redirected successfully | Pass | Pass |

# Appendix 18: User Feedback Reponse

## User Feedback Response

**IMPORTANT!**

Please note that the survey result in this document has removed the participant's name and email collected in accordance with the Data Protection Act (2018) and one of the guideline written in the Participant Information Sheet as shown below:

*"Under the EU's new General Data Protection Regulation, if we are collecting personal data indirectly, we will have to be transparent about what categories of personal data will be collected and processed in the study. In this study, the participant's name and email is the only personal data necessary to be collected for smoother processing and analysing of the personal data. Participants' names and email are meant only for us and will always remain confidential, and they will not appear or be written in any reports or documentation."*

The responses have been extracted from the Google Form Response and formatted into this document.

The survey questionnaires could be found in the Word Document located within the Survey Questionnaire folder or by using the link: https://forms.gle/DNxjeonqFNqgkJseA

### Page 2: What do you think about the system?

| Participant | Response |
|---|---|
| Participant 6 | I think you have done a great job in developing the system because all the features listed on the guideline provided could be completed without any problems, there are no visible error on the functions themselves. Moving on to the interface, I think it is very well done in terms of design because I can navigate through the webpage without any problem or feeling overwhelmed by the interface. Next, I can see that you have design all the functionalities in different length which I think it's a great idea because I was able to find the exact functions I am looking for by just recognising the length instead of keep looking for a specific title. And I think finally, when I was attempting the login feature listed in the guideline provided, I have accidentally made a typo on the username category, however when I tried to login it shows me an error message saying that my username or password is incorrect. I think this is a really great implement because now I would not have to pull my hair out just to know what has gone wrong, but instead I can just take a look at the error and let me make decision quickly. Plus, I think the error messsage is very helpful because it specifies exactly what is wrong without sending the users to a wild goose chase. So overall, I think your system is really good and well designed. |