Software Requirements Specification

for

Smart Plant Monitoring System with Automated Care

Minor Project

A Project Submitted
in Partial Fulfilment of the Requirements
For the Course of
Bachelor of Technology
specialization(IOT)
In
Third year – Fifth Semester
Under the supervision of
Dr.Rohit Tanwar
By

Name	Roll No.	Sap I'd
Arpit Kansal	R2142210150	500091746

INFORMATICS CLUSTER
SCHOOL OF COMPUTER SCIENCE
UNIVERSITY OF PETROLEUM AND ENERGY STUDIES, BIDHOLI, DEHRADUN, UTTRAKHAND, INDIA

December 2023

Mentor Signature

Project Guide

1. INTRODUCTION

1.1Purpose:-

The purpose of this project is to revolutionize plant care through the creation of a Smart Plant Monitoring System with Automated Care. In a world marked by increasing urbanization and sustainability concerns, this innovative system seeks to democratize the art of nurturing plants. It aims to provide efficient, eco-friendly, and hassle-free plant care for individuals with varying levels of expertise, all while promoting resource conservation and fostering healthier plant growth. By integrating cutting-edge sensor technologies, IoT connectivity, and user-friendly interfaces, this project empowers users to connect with nature and enhance their quality of life through thriving green spaces.

1.2 Target Beneficiary:-

The target beneficiaries for this project include urban dwellers, home gardeners, and individuals seeking sustainable plant care solutions. It caters to both novice and experienced gardeners, making plant care accessible and efficient. By simplifying the process and conserving resources, it fosters healthier plant growth, enhances living spaces, and promotes environmental consciousness.

1.3 Project Scope:-

The project scope encompasses the development of a Smart Plant Monitoring System with Automated Care, incorporating soil moisture, humidity, temperature, and motion detection sensors. It will automate critical plant care tasks such as irrigation control and pest detection. The system's IoT connectivity will facilitate real-time data transmission to a cloud platform. User interfaces, including a mobile app and website, will enable remote monitoring and control. The project also aims to provide educational resources to promote informed plant care practices.

2. PROJECT DESCRIPTION

2.1 Reference Algorithm

Loop:

Read soil moisture data from sensor

Read humidity data from sensor

Read temperature data from sensor

Read motion detection data from sensor

IF soil moisture < desired moisture level:

Turn on irrigation system for a specific duration

ELSE:

Turn off irrigation system

IF humidity is too low:

Increase humidity control

ELSE IF humidity is too high:

Decrease humidity control

IF temperature is too high:

Activate cooling system

ELSE IF temperature is too low:

Activate heating system

IF motion detected:

Send alert or activate pest control

Store data in a database or cloud platform

Delay for a set interval before the next reading

2.2 SWOT Analysis

Strengths

•**Automation and Efficiency:** The project's automated plant care and resource optimization capabilities offer a significant strength, reducing the burden of manual labor and promoting resource efficiency.
•**User-Centric Design:** User-friendly interfaces empower users of all skill levels, fostering a deeper connection to plant care and promoting accessibility.
•**Sustainability:** The project aligns with the growing emphasis on sustainability and green living, contributing to environmental consciousness and resource conservation

Weaknesses

•**Initial Setup:** Complex setup and integration of sensors and IoT components may pose a learning curve for some users, potentially limiting adoption.
•**Maintenance and Upkeep:** Ongoing maintenance and troubleshooting may be required, which could deter users from seeking entirely maintenance-free solutions.

Opportunities:

•**Market Growth:** The increasing interest in home gardening, urban agriculture, and smart home technologies presents a growing market for the project's solutions.
•**Expanding Features:** Continuous development could add new features and capabilities, such as compatibility with more plant species or additional sensor types.
•**Educational Resources:** Opportunities exist to provide educational content and guides, enhancing user knowledge about plant care and IoT technology.

Threats:

•**Competition:** Competition from other smart plant care systems or traditional gardening methods could pose a threat to market share.
•**Data Privacy:** Ensuring data privacy and security is crucial, as concerns over data breaches and privacy violations continue to grow.
•**Resource Availability:** Dependence on cloud services and internet connectivity may be vulnerable to disruptions, impacting system functionality.

2.3 Assumption and Dependencies

Assumptions: This project assumes accurate sensor readings, stable internet and power supply, user engagement, consistent pest behavior, and secure data storage.

Dependencies: It relies on the availability of hardware components, proper sensor calibration, software development, dependable cloud services, reliable power infrastructure, internet connectivity, ongoing maintenance, and user participation for optimal functionality and success.

2.4 Code

```
#include <ESP8266_Lib.h>
/* Connections
Relay. D3
Btn.   D7
Soil.  A0
PIR.   D5
SDA.   D2
SCL.   D1
Temp.  D4 */
//Include the library files
#include <LiquidCrystal_I2C.h>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
```

```
#include <DHT.h>
//Initialize the LCD display
LiquidCrystal_I2C lcd(0x3F, 16, 2);
char auth[] = "_epQMfmTnR1k2TdUX-DSYuvxZScphp7I";  //Enter your Blynk Auth token
char ssid[] = "Arpit";  //Enter your WIFI SSID
char pass[] = "Arpit3222";  //Enter your WIFI Password
DHT dht(D4, DHT11);//(DHT sensor pin,sensor type)  D4 DHT11 Temperature Sensor
BlynkTimer timer;
//Define component pins
#define soil A0     //A0 Soil Moisture Sensor
#define PIR D5      //D5 PIR Motion Sensor
int PIR_ToggleValue;
void checkPhysicalButton();
int relay1State = LOW;
int pushButton1State = HIGH;
#define RELAY_PIN_1       D3   //D3 Relay
#define PUSH_BUTTON_1     D7   //D7 Button
#define VPIN_BUTTON_1    V12
//Create three variables for pressure
double T, P;
char status;
void setup() {
  Serial.begin(9600);
  lcd.begin();
  lcd.backlight();
  pinMode(PIR, INPUT);
 pinMode(RELAY_PIN_1, OUTPUT);
 digitalWrite(RELAY_PIN_1, LOW);
  pinMode(PUSH_BUTTON_1, INPUT_PULLUP);
  digitalWrite(RELAY_PIN_1, relay1State);
  Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
```

```
  dht.begin();
  lcd.setCursor(0, 0);
  lcd.print("  Initializing  ");
  for (int a = 5; a <= 10; a++) {
    lcd.setCursor(a, 1);
    lcd.print(".");
    delay(500);
  }
  lcd.clear();
  lcd.setCursor(11, 1);
  lcd.print("W:OFF");
  //Call the function
  timer.setInterval(100L, soilMoistureSensor);
  timer.setInterval(100L, DHT11sensor);
  timer.setInterval(500L, checkPhysicalButton); }
//Get the DHT11 sensor values
void DHT11sensor() {
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return; }
  Blynk.virtualWrite(V0, t);
  Blynk.virtualWrite(V1, h);
  lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print(t);
  lcd.setCursor(8, 0);
  lcd.print("H:");
  lcd.print(h); }
//Get the soil moisture values
```

```
void soilMoistureSensor() {
  int value = analogRead(soil);
  value = map(value, 0, 1024, 0, 100);
  value = (value - 100) * -1;
  Blynk.virtualWrite(V3, value);
  lcd.setCursor(0, 1);
  lcd.print("S:");
  lcd.print(value);
  lcd.print(" "); }
//Get the PIR sensor values
void PIRsensor() {
  bool value = digitalRead(PIR);
  if (value) {
    Blynk.logEvent("pirmotion","WARNNG! Motion Detected!"); //Enter your Event Name
    WidgetLED LED(V5);
    LED.on();
  } else {
    WidgetLED LED(V5);
    LED.off();
  }
  }
BLYNK_WRITE(V6)
{
 PIR_ToggleValue = param.asInt();
}
BLYNK_CONNECTED() {
  // Request the latest state from the server
  Blynk.syncVirtual(VPIN_BUTTON_1);
}
BLYNK_WRITE(VPIN_BUTTON_1) {
  relay1State = param.asInt();
```

```
  digitalWrite(RELAY_PIN_1, relay1State);
}
void checkPhysicalButton()
{
  if (digitalRead(PUSH_BUTTON_1) == LOW) {
    // pushButton1State is used to avoid sequential toggles
    if (pushButton1State != LOW) {
      // Toggle Relay state
      relay1State = !relay1State;
      digitalWrite(RELAY_PIN_1, relay1State);
      // Update Button Widget
      Blynk.virtualWrite(VPIN_BUTTON_1, relay1State);
    }
    pushButton1State = LOW;
  } else {
    pushButton1State = HIGH;
  }
}
void loop() {
    if (PIR_ToggleValue == 1)
    {
    lcd.setCursor(5, 1);
    lcd.print("M:ON ");
      PIRsensor();
      }
     else
     {
    lcd.setCursor(5, 1);
    lcd.print("M:OFF");
    WidgetLED LED(V5);
    LED.off();
```

```
    }
if (relay1State == HIGH)
{
  lcd.setCursor(11, 1);
  lcd.print("W:ON ");
  }
  else if (relay1State == LOW)
  {
    lcd.setCursor(11, 1);
    lcd.print("W:OFF");
    }
  Blynk.run();//Run the Blynk library
  timer.run();//Run the Blynk timer
  }
```