FINAL VERSION

2017

# Plant.io

## A FUN PLANT GROWING COMPANION APP INTEGRATED WITH SELF-WATERING POT

SOFTWARE PROJECT MANAGEMENT DOCUMENTATION

**PRESENTED BY: GROUP 9**

| | |
|---|---|
| **AQILA FATHIMAH KARIM** | **1506734664** |
| **HAEKAL FEBRIANSYAH RAMADHAN** | **1506744280** |
| **WHISNU SAMUDRA** | **1506673706** |

FACULTY OF ENGINEERING
UNIVERSITAS INDONESIA

## PREFACE

Firstly, grateful acknowledgment is here made to everyone who have helped in the process of making this documentation, especially to our lecturer Prof. Dr. Ir. Riri Fitri Sari, M.Sc., MM. who has been guiding and giving constructive feedback during the process. This work would not have completed well and on time without those invaluable help.

This documentation presents the overall management plan and implementation of the project “Plant.io”. Development, budget, implementation, testing, and many other development principals are showed in this documentation.

This paper is addressed to engineering students, new developers, or simply anyone with interest of software project management as a guideline to see and understand the flow of development and control of a software developing process.

Lastly, the authors truly realized that this documentation still lack in many things, and thus genuinely expect critics and advices to improve this work. The authors also hope that this documentation can be useful for the readers despite its lacks.

Depok, 2017

Developer Team

## Table of Contents

## Table of Figures

# I. INTRODUCTION

This chapter will describe a brief introduction on why the title is chosen, the general purpose and goals of the project, the functions expected from the project, the tools needed and the risk analysis of the project working.

## 1.1. Introduction

Gardens play a crucial role in urban and suburban areas, from helping to protect us against flooding and extremes of temperature, to supporting wildlife and human health[1]. Gardens, even small ones in homes contribute to a better and greener environment. Unfortunately, society nowadays are so busy with their daily works that people rarely plant. Kids in schools are not taught to garden and were focused to academics. They grew up being not familiar with gardening. One of the reasons why people do not want to garden is because gardening takes a lot of time and effort. Different kinds of plants require different treatments, such as the moisture level, light intensity, temperature, and fertilizers. These factors will determine the success of growing plant, and mistakes in treatment could lead to the plant dying. Especially because watering plants need a routine schedule and not everyone can manage that schedule.

By using the fact that average people spent 24 minutes in playing mobile games a day with their smartphone[2], this project strives to develop a mobile application that promotes gardening. This mobile application will provide the information needed about growing plants, generally and specifically, and monitor the condition of the plant in an entertaining way, so that user enjoys their time gardening with the application. The application will also act as an assistant in taking care of the plant, by providing fertilizing scheduler and integrating it with a self-watering plant pot. This application

[1] https://www.rhs.org.uk/communities/campaigns/greening-grey-britain
[2] http://www.vertoanalytics.com/average-mobile-game-day/

will hopefully contribute in trying to help people take care of their plant in a much more efficient and friendly way, and as well promotes the importance of gardening to children in this modern and fast-moving generation.

## 1.2. General Purpose

The general purpose of this project is to help people in taking care of plant and to give gardening education to the users, and integrate it with an automatic self-watering system that will provide a better and more efficient way of gardening. Wi-Fi communication is used to connect between the application and the pot system.

## 1.3. Functions

The functions of this project's mobile application are:

- User can see the plant's current condition (soil moisture, light intensity, temperature)
- User can set a fertilizing schedule
- User can see plant info (ideal surroundings, scientific name, photo)
- User can see gardening tips
- Plant Pot (hardware) can automatically water plant when bad soil moisture is detected (integrated self-watering pot)

## 1.4. Tools

### 1.4.1. Hardware

- Soil moisture sensor (LM293)

- LDR sensor (Model 3190)

- temperature sensor (DHT11)
- mini hose
- water pump
- Arduino UNO R3
- Wi-Fi module (ESP8266)

### 1.4.2. Software

- Arduino IDE
- Android Studio

## 1.5. Risk Analysis

There are some risks that might happen during this project. Firstly, because this project uses self-watering system, the hardware involved in the self-watering pot can get wet and resulted in a short-circuit because of mistakes in the installation of the self-watering system. Due to this reason as well, false information might be given to the application. Mistakes in coding or designing the application or system might also resulted in the application and the hardware being not connected at all, and thus cannot exchange information.

# II. PROJECT MANAGEMENT

This chapter will describe in detail each aspects of the project management, including the project plans, time and cost estimating, human resources and communication, scope and quality verification as well as risk management.

## 2.1. Project Integration Management

### 2.1.1 Project Plan Development

In this plant-watcher mobile application integrated with self-watering system project, the project plan is divided into two parts. The first part being the plan of the mobile application itself. In this project, the targeted operating system which the application can run in is Android version 4.0 or above. The mobile application will need Wi-Fi connection to communicate with the plant pot later. This application also acts as GUI for the user in monitoring activities happening in the pot. The application will have interesting graphics and layout, so that user will enjoy using the application as much as they enjoy playing a mobile game. The application will consist of a welcoming screen, a home screen, and utility screens. The home screen will display a plant figure as its center, which the user can name. There will be information buttons on the screen showing brief information about the plant, which users can click to see more details (this leads to the utility screen). Utility screen will consist of 6 types, each having their specialized function. The first three will display detail information about soil moisture, light intensity, and temperature, using percentage for soil moisture and light intensity, and Celsius degrees for temperature. The fourth screen will display basic information about the plant, including their scientific name, real-life photo, and ideal surroundings for optimal growth. This screen will also display tips and trivial information in gardening. The fifth screen will display a calendar-like schedule which the user can fill in for fertilization schedules. This calendar will act like a to-do list that the user can check anytime for the fertilization routine. The last screen will be the Wi-

Fi settings screen, where connection settings to the Arduino pot takes place. A help button will also be included, with general user guide on how to operate the application.

Below are the concept drafts of the application’s GUI layout:

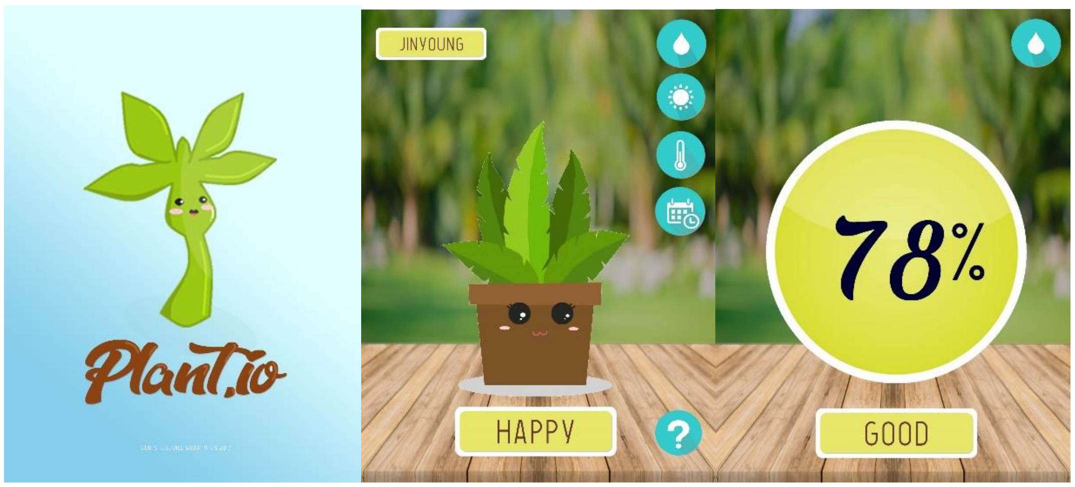


*Figure II-1 GUI Plan*

The second part of the plan is the self-watering pot. This pot will be given specific sensors to detect soil moisture, light intensity, and temperature. There will be a structure of small water hose and pump connected to Arduino as a controller. Each time the controller reads that the soil moisture is in poor condition, it will instruct the hose to water the pot to a certain level. This Arduino will also store information from all the sensors in the EPROM memory, which can be sent to the application via Wi-Fi connection, and later translated by the application as an understandable information in the GUI.  This routine will be done continuously, and information sent will be updated every time the application is opened. Below is the design draft of the hardware:

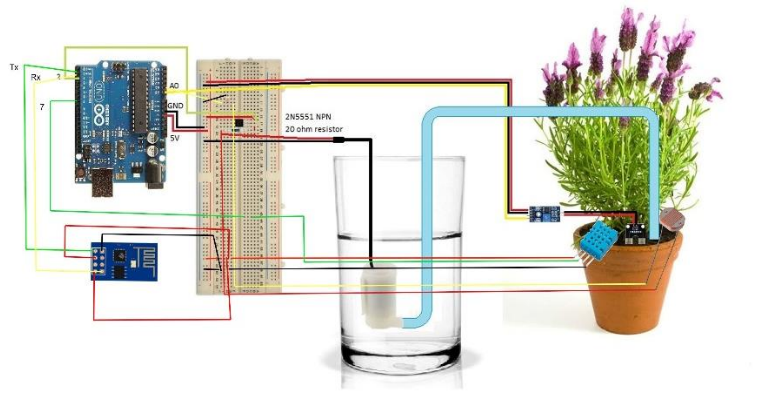


*Figure II-2 Hardware Plan*

### 2.1.2 Project Plan Execution

To reach the plan as elaborated above, there are some system requirements regarding the software used in the project. These requirements are:

- Arduino IDE (embedded system programming)
- Android Studio (mobile application building)
- Visual Paradigm (diagram designing)
- Adobe Photoshop (interface and layout designing)
- Ms. Project and GitHub (project managing)

### 2.1.3 Overall Change Control

In doing this project, the first made plan will be the goal priorities that needs to be achieved during the project development. However, if during the process of making the project there are features or changes that might enhance the performance of the application and/or is feasibly more efficient and sustainable in finishing the project, minor changes might occur. Reviews and advice from users in the testing period will also be considered for changes.

## 2.2 Project Scope Management

### 2.2.1 Scope Definition

The goal of this project is to create an interesting mobile application to help users monitor their plant and to give convenience in watering plants. This project targets all gender in all ages that has interest in growing plant. And not only to assist users in taking care of their plant, this project also aims in giving general knowledge of gardening as well as promote green environment messages, especially for children. This project hopefully will give a fun and enjoyable experience in growing a plant, compared to growing it traditionally. As for the operating system scope, it has been stated in the previous section that the application will be aimed for Android 4.0 and above.

### 2.2.2 Scope Verification

Scope verification is a way of obtaining acceptance from users, which includes reviewing work product and result to ensure that the project reaches user's satisfactory. In this project, several documentations are required to be delivered at the end of the project in terms of scope verification. These documentations are:

- SPMP (Software Project Manager Plan)
- Design diagram
- Test plan
- Questionnaires
- User's manual

## 2.3 Project Time Management

In finishing this project, a timeline of project-planning, developing and executing, as well as testing and analysis is made with Microsoft Project and will be attached along. There might be some slight changes in reaching the milestones of the project depending on the situation during the work, but the critical milestones of the project should stand.

## 2.4 Project Cost Management

### 2.4.1 Cost Estimating and Budgeting

The cost estimated for this project is as listed below:

| | |
|---|---|
| Arduino soil moisture hygrometer sensor | Rp. 21 000 |
| LDR light sensor | Rp. 2000 |
| Arduino Uno R3 | Rp. 90 000 |
| TMP36 temperature sensor | Rp. 12 000 |
| Small water submersible pump | Rp. 30 000 |
| Small hose | Rp. 12 000 |
| Plastic plant pot | Rp. 10 000 |
| **Total** | **Rp. 177 000** |

### 2.4.2 Resource Planning

The cost estimation above might vary and change according to the actual purchases made during the project. The resources of these cost will be from project team personal money.

## 2.5 Project Quality Management

### 2.5.1 Quality Planning

The planned quality that the project will try to achieve to ensure the project design is efficient and effective in terms of performance are:

- User-friendly application
- Well working self-watering system with application

## 2.6 Project Human Resources Management

To achieve the goals of the project, team project is divided and given specific roles and responsibilities regarding the project, which are:

| **Name** | **Role** |
|---|---|
| Aqila Fathimah Karim | Overall project manager<br>Programmer<br>Interface designer |
| Haekal Febriansyah Ramadhan | Programmer<br>Hardware designer |
| Whisnu Samudra | Programmer<br>Documentation manager |

## 2.7 Project Communication Management

As all team member are students of Computer Engineering FTUI, regular meeting can be held at least once every week to discuss about the progress of the project. Additional communication can also be done through online messaging group. Along with this documentation, reports of the project will be gradually completed with each evaluations and improvements given throughout the project making.

## 2.8 Project Risk Management

### 2.8.1 Risk Identification

There are some risks that might happen during this project working, which falls to categories below:

- Programming: medium risk

Lack of skill and experience in developing mobile software, unfamiliar with the development platforms.

- Design: low risk

Interface design being not interesting and user-friendly, causing users to feel unsatisfied.

- Time shortage: medium risk

Schedule and milestones of the project not reached on time, caused by various factors.

- Sudden obstacles: low risk

External factors and sudden obstacles to team which leads to delayed works.

- User satisfaction: high risk

User being unsatisfied by the final product from the project, may resulted from bugs in software or feature failures.

### 2.8.2 Risk Response Control

To anticipate the risks that might happen as elaborated above, there are some response control moves that can be done, which are:

- Intense learning and reference searching for mobile software developing
- Reference searching and sample user questioning to achieve most likeable design
- Consistent in doing works based on timeline and focus on finishing each role’s task before helping one another
- Open two-way communication to understand each member’s concern and sudden obstacles that might arise
- Always evaluating and reviewing all steps involved in project development

# III. DESIGN DIAGRAMS

In this chapter, design diagrams will be provided to give more thorough understanding about the project. Design diagrams are used to represent a system in a more understandable way. These design diagram are represented in the standard format of UML as standardized by OMG. UML is a modeling language used to visualize a design of a system.

## 3.1. Use-Case Diagram

A use case diagram is a simple representation of a user's interaction with the system, which shows the relationship between the user and the different use cases where the user is involved. A use case drills into a lot of detail about every possibility and provides a higher-level view of the system. It is helpful and sometimes referred to " the blueprints for your system"[3].

Below is a use-case diagram which shows the interaction between users and the application developed in this project:

[3] McLaughlin et al, 2006, page 297

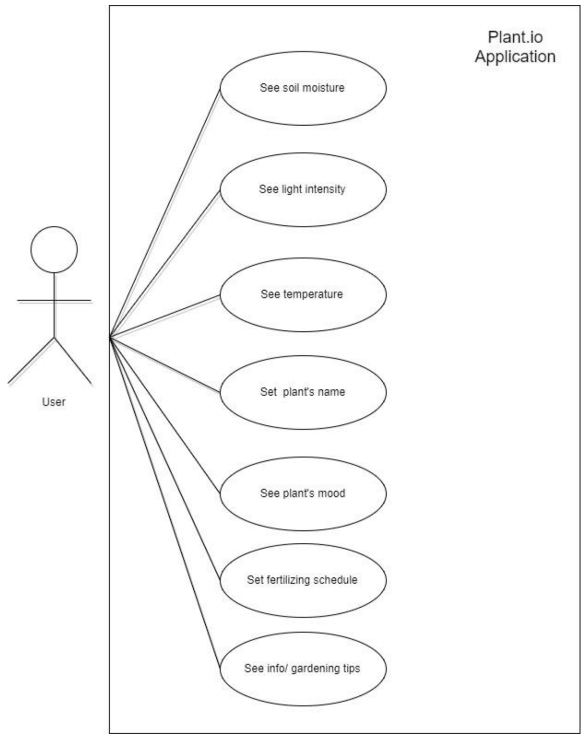


*Figure III-1 Use Case Diagram*

The diagram shows that the application interacted with one user at a time, in which the user can do a set of activities in the program. These activities include seeing the plant's current condition (soil moisture, light intensity, temperature), setting plant's name, seeing plant's mood (good, bad, etc.), plant's general info and gardening tips, and set fertilizing schedule.

### 3.2. Activity Diagram

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency[4]. In other words, activity diagram shows the overall flow of control in a design.

Below are activity diagrams used to represent the flow of control in this project:

[4] Glossary of Key Terms at McGraw-hill.com. Retrieved 20 July 2008.

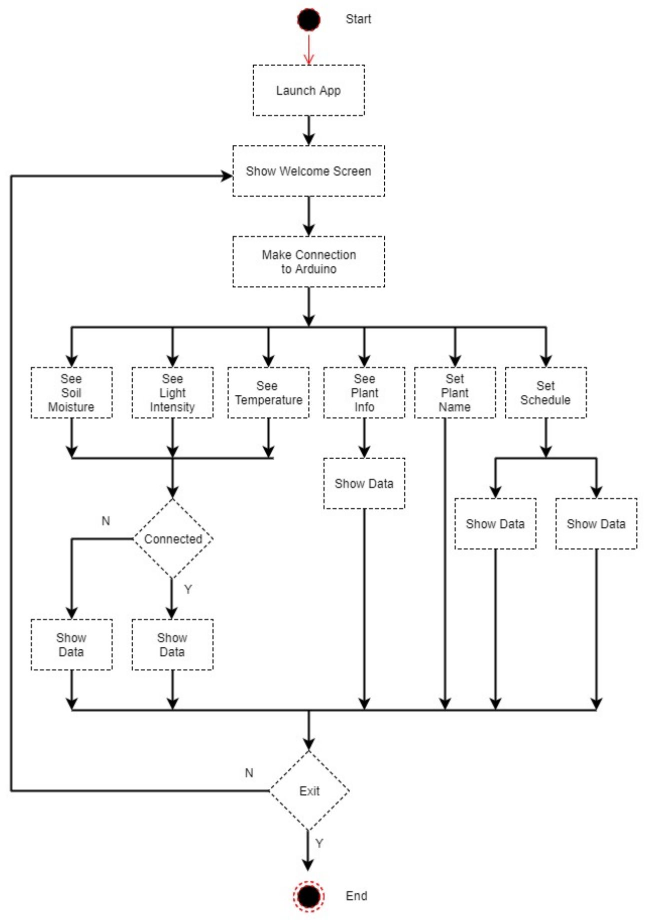


*Figure III-2 Activity Diagram 1*

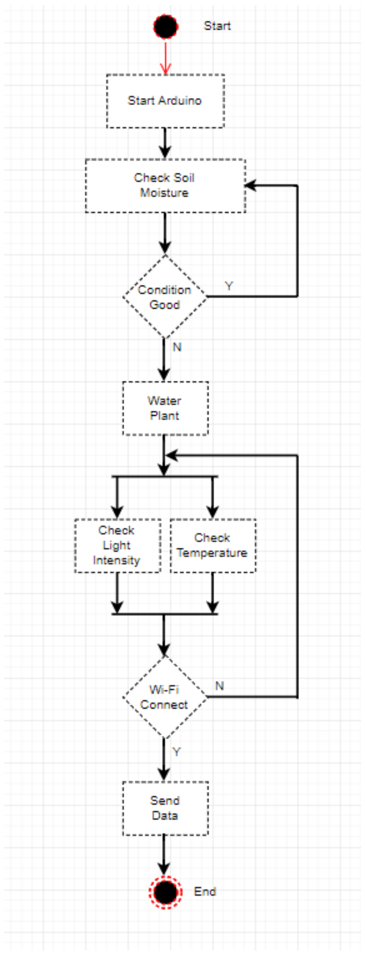


*Figure III-3 Activity Diagram 2*

Figure III-2 shows the activity diagram for the software application, in which the flow control starts from launching the app, showing the welcoming screen and redirecting to the home screen. It is shown that while loading the home screen, the application communicates with the self-watering system pot via Wi-Fi to obtain information needed about the plant. After the home screen is loaded, the flow of control is divided into choices as an input from the user, and continue executing the series of events according to the choice made (as shown above).

Figure III-3 shows the activity diagram of the self-watering pot system which goes through a loop of checking the soil moisture condition through the sensor, obtaining the information, and deciding whether to water the plant or not based on the condition.

### 3.3. Class Diagram

Class diagram is a type of static structure diagram that describes the structure of a system, consists of the system's classes, their attributes, operations (or methods), and the relationships among objects[5]. Below is the class diagram used to show the structure of this project’s system:

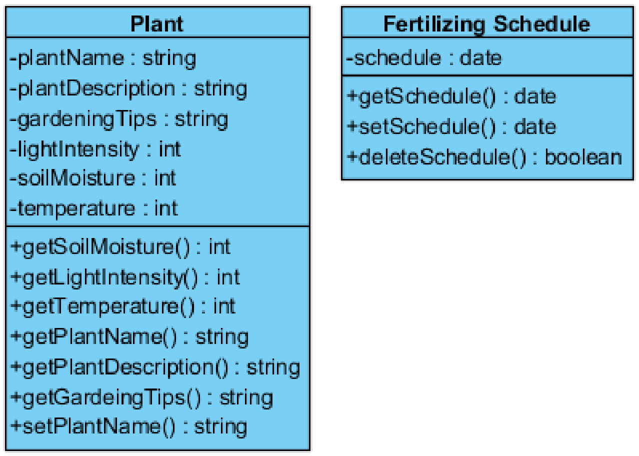


*Figure III-4 Class Diagram*

The structure for the application consists of two classes; Plant and Fertilizing Schedule in which each class contains corresponding attributes.

[5] Sparks, Geoffrey. "Database Modelling in UML". Retrieved 8 September 2011.

# IV. IMPLEMENTATION

In this chapter, the source codes of both the software and hardware will be provided to give thorough understanding about how the system is implemented. The language used to implement the system is in Java for software logic functions, XML for GUI layout, and Embedded C for hardware implementation. These languages are used because they were the most common languages for developing a mobile application, especially android, so it will be easier to find resources and it is more flexible to use for various devices.

## 4.1. Software Implementation

Android Studio is used in implementing the software. Source codes are divided into two main categories: Java codes and XML codes. Java codes consist of eight different source codes, each represents an activity in the android application.

### 4.1.1. Main Activity Java Code

```java
package id.sample.test;
//RPL Project Group 9
//Aqila Fathimah Karim 1506734664
//Whisnu Samudra 15066673706
//Haekal Febriansyah 1506744280
//2017

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Typeface;
import android.os.AsyncTask;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TextView;

import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URI;
import java.net.URISyntaxException;


public class MainActivity extends AppCompatActivity {
    //codes used to pass shared preferences (to save data user inputs
eventhough the app is closed)
    //data will only be erased if the user reinstall app or clear app data
from settings
    SharedPreferences sharedpreferences;
    public static final String mypreference = "mypref";
    public static final String Name = "nameKey";
    //define custom font styles
    Typeface dopestyle;
    Typeface lightfont;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //set custom font styles
        lightfont =
Typeface.createFromAsset(getAssets(),"fonts/TheLightFont.ttf");
        dopestyle =
Typeface.createFromAsset(getAssets(),"fonts/Dopestyle.ttf");
        TextView mood = (TextView) findViewById(R.id.Mood);
        mood.setTypeface(lightfont);

        TextView name = (TextView) findViewById(R.id.Plantname);
        //display saved name if available
        sharedpreferences = getSharedPreferences(mypreference,
                Context.MODE_PRIVATE);
        if (sharedpreferences.contains(Name)) {
            name.setText(sharedpreferences.getString(Name, ""));
        }

        //codes used to activate button for navigating between activities on
click
        ImageButton moist=(ImageButton)findViewById(R.id.moist);
        moist.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent MoistureScreen=new
Intent(getApplicationContext(),MoistureActivity.class);
                startActivity(MoistureScreen);
            }
        });


        ImageButton light=(ImageButton)findViewById(R.id.light);
        light.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent LightScreen=new
Intent(getApplicationContext(),LightActivity.class);
                startActivity(LightScreen);
            }
        });


        ImageButton temp=(ImageButton)findViewById(R.id.temp);
        temp.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent TempScreen=new
Intent(getApplicationContext(),TempActivity.class);
                startActivity(TempScreen);
            }
        });

        ImageButton sked=(ImageButton)findViewById(R.id.sched);
        sked.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent skedScreen=new
Intent(getApplicationContext(),SchedActivity.class);
                startActivity(skedScreen);
            }
        });

        TextView plant = (TextView) findViewById(R.id.Plantname);
        plant.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent infoScreen=new
Intent(getApplicationContext(),infoActivity.class);
                startActivity(infoScreen);
            }
        });

        ImageButton help=(ImageButton)findViewById(R.id.info);
        help.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent helpScreen=new
Intent(getApplicationContext(),helpActivity.class);
                startActivity(helpScreen);
            }
        });

        ImageView wifi=(ImageView) findViewById(R.id.wifi);
        wifi.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent wifiscr=new
Intent(getApplicationContext(),WifiActivity.class);
                startActivity(wifiscr);
            }
        });
        //codes to read user input for plant name, store the data to shared
preferences,
        //get the data and display it on the app on button click
        Button setname = (Button) findViewById(R.id.name);
        final EditText et = (EditText) findViewById(R.id.editText);
//EditText is defined as edittext in xml
        setname.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                TextView name = (TextView) findViewById(R.id.Plantname);
                String n = et.getText().toString();
                SharedPreferences.Editor editor = sharedpreferences.edit();
                editor.putString(Name, n);
                editor.commit();
                if (sharedpreferences.contains(Name)) {
                    name.setText(sharedpreferences.getString(Name, ""));
                }
            }
        });

    }
}
```

### 4.1.2. Schedule Activity Java Code

```
package id.sample.test;

import android.app.DatePickerDialog;
import android.content.Context;
import android.content.SharedPreferences;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;

import java.util.Calendar;

public class SchedActivity extends AppCompatActivity {
    //variables declaration
    TextView sked;
    Calendar mCurrentDate;
    int day, month, year;
    SharedPreferences prefs;
    public static final String mypreference = "mypref";
    public static final String Date = "dateKey";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sched);
        sked = (TextView) findViewById(R.id.textView4);
        //variables to retrieve date from calendar
        mCurrentDate = Calendar.getInstance();
        day = mCurrentDate.get(Calendar.DAY_OF_MONTH);
        month = mCurrentDate.get(Calendar.MONTH);
        year = mCurrentDate.get(Calendar.YEAR);
        month = month +1;
        prefs = getSharedPreferences(mypreference,
                Context.MODE_PRIVATE);
        if (prefs.contains(Date)) {
            sked.setText(prefs.getString(Date, ""));
        }
        else {
            sked.setText(day+"/"+month+"/"+year);
        }
        //code to set text view as date the user picked from calendar on
click
        sked.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                sked = (TextView) findViewById(R.id.textView4);
                DatePickerDialog datePickerDialog = new
DatePickerDialog(SchedActivity.this, new
DatePickerDialog.OnDateSetListener(){
                    @Override
                    public void onDateSet(DatePicker view, int year, int
monthOfYear, int dayOfmonth)
                    {
                        monthOfYear = monthOfYear+1;
                        String n= dayOfmonth+"/"+monthOfYear+"/"+year;
                        SharedPreferences.Editor editor = prefs.edit();
                        editor.putString(Date, n);
                        editor.apply();
                        if (prefs.contains(Date)) {
                            sked.setText(prefs.getString(Date, ""));
                        }
                    }
                }, year,month,day);
                datePickerDialog.show();
            }
        });

    }
}
```

### 4.1.3. Info Activity Java Code

```
package id.sample.test;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class infoActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_info);
    }
}
```

### 4.1.4. Light Intensity Activity Java Code

```
package id.sample.test;
```

```java
import android.content.SharedPreferences;
import android.graphics.Typeface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

import static id.sample.test.WifiActivity.RESPONSE;

public class LightActivity extends AppCompatActivity {
    //define custom font
    Typeface dopestyle;
    Typeface lightfont;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_light);
        //set custom font
        lightfont =
Typeface.createFromAsset(getAssets(),"fonts/TheLightFont.ttf");
        dopestyle =
Typeface.createFromAsset(getAssets(),"fonts/Dopestyle.ttf");
        TextView condilight = (TextView) findViewById(R.id.condilight);
        TextView conlight = (TextView) findViewById(R.id.conlight);
        condilight.setTypeface(lightfont);
        conlight.setTypeface(dopestyle);
        //get value sent by the wifi server before
        SharedPreferences pref =
getSharedPreferences("HTTP_HELPER_PREFS",MODE_PRIVATE);
        if (pref.contains(RESPONSE)) {
            conlight.setText(pref.getString(RESPONSE, ""));
        }
        else {
            conlight.setText("-");
            condilight.setText("NOT CONNECTED");
        }
    }
}
```

### 4.1.5. Soil Moisture Activity Java Code

```java
package id.sample.test;

import android.content.SharedPreferences;
import android.graphics.Typeface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

import static id.sample.test.WifiActivity.RESPONSE;

public class MoistureActivity extends AppCompatActivity {
    //define custom font
    Typeface dopestyle;
    Typeface lightfont;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_moisture);
        //set custom font
        lightfont =
Typeface.createFromAsset(getAssets(),"fonts/TheLightFont.ttf");
        dopestyle =
Typeface.createFromAsset(getAssets(),"fonts/Dopestyle.ttf");
        TextView condimoist = (TextView) findViewById(R.id.condimoist);
        TextView conmoist = (TextView) findViewById(R.id.conmoist);
        condimoist.setTypeface(lightfont);
        conmoist.setTypeface(dopestyle);
        //get value sent by the wifi server before
        SharedPreferences pref =
getSharedPreferences("HTTP_HELPER_PREFS",MODE_PRIVATE);
        if (pref.contains(RESPONSE)) {
            conmoist.setText(pref.getString(RESPONSE, ""));
        }
        else {
            conmoist.setText("-");
            condimoist.setText("NOT CONNECTED");
        }
    }
}
```

### 4.1.6. Splash Activity Java Code

```
package id.sample.test;

import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;

public class SplashActivity extends AppCompatActivity {
    //set time interval
    private final int SPLASH_DISPLAY_LENGTH = 2000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        new Handler().postDelayed(new Runnable(){
            @Override
            public void run() {
                /* Create an Intent that will start the Menu-Activity. */
                Intent mainIntent = new
Intent(SplashActivity.this,MainActivity.class);
                SplashActivity.this.startActivity(mainIntent);
                SplashActivity.this.finish();
            }
        }, SPLASH_DISPLAY_LENGTH);

    }

}
```

### 4.1.7. Temperature Activity Java Code

```java
package id.sample.test;

import android.content.SharedPreferences;
import android.graphics.Typeface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

import static id.sample.test.WifiActivity.RESPONSE;

public class TempActivity extends AppCompatActivity {
    //define custom font
    Typeface dopestyle;
    Typeface lightfont;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_temp);
        //set custom font
        lightfont =
Typeface.createFromAsset(getAssets(),"fonts/TheLightFont.ttf");
        dopestyle =
Typeface.createFromAsset(getAssets(),"fonts/Dopestyle.ttf");
        TextView conditemp = (TextView) findViewById(R.id.conditemp);
        TextView contemp = (TextView) findViewById(R.id.contemp);
        conditemp.setTypeface(lightfont);
        contemp.setTypeface(dopestyle);
        //get value sent by the wifi server before
        SharedPreferences pref =
getSharedPreferences("HTTP_HELPER_PREFS",MODE_PRIVATE);
        if (pref.contains(RESPONSE)) {
            contemp.setText(pref.getString(RESPONSE, ""));
        }
        else {
            contemp.setText("-");
            conditemp.setText("NOT CONNECTED");
        }
    }
}
```

### 4.1.8. Main Activity Layout XML Code

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="id.sample.test.MainActivity"
    android:background="@drawable/main2"
       >

    <Button
        android:id="@+id/name"
        style="@style/Widget.AppCompat.Button.Colored"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/temp"
        android:layout_marginBottom="28dp"
        android:layout_toEndOf="@+id/hi"
        android:layout_toRightOf="@+id/hi"
        android:backgroundTint="#e6e768"
        android:text="NAME"
        android:textColor="#7c5327" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/name"
        android:layout_alignBottom="@+id/name"
        android:layout_toLeftOf="@+id/name"
        android:layout_toStartOf="@+id/name"
        android:ems="10"
        android:hint="Name your plant"
        android:inputType="textPersonName"
        tools:layout_editor_absoluteX="85dp"
        tools:layout_editor_absoluteY="242dp" />

    <ImageButton
        android:id="@+id/temp"
        android:layout_width="65dp"
        android:layout_height="65dp"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/moist"
        android:background="@drawable/temp"
        android:contentDescription="Suhu" />

    <ImageButton
        android:id="@+id/moist"
        android:layout_width="65dp"
        android:layout_height="65dp"
        android:layout_alignParentRight="true"
        android:background="@drawable/moist"
        android:contentDescription="moist" />

    <ImageButton
        android:id="@+id/light"
        android:layout_width="65dp"
        android:layout_height="65dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/temp"
        android:background="@drawable/light"
        android:contentDescription="light" />

    <ImageButton
        android:id="@+id/sched"
        android:layout_width="65dp"
        android:layout_height="65dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/light"
        android:background="@drawable/sked"
        android:contentDescription="sched" />

    <ImageButton
        android:id="@+id/info"
        android:layout_width="65dp"
        android:layout_height="65dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_marginBottom="23dp"
        android:background="@drawable/info"
        android:contentDescription="info" />

    <TextView
        android:id="@+id/Mood"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/info"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="16dp"
        android:text="HAPPY"
        android:textColor="#7c5327"
        android:textSize="24dp" />

    <TextView
        android:id="@+id/Plantname"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/temp"
        android:layout_centerHorizontal="true"
        android:text="(I don't have any name yet)"
        android:textColor="#ffffff"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/hi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/Plantname"
        android:layout_centerHorizontal="true"
        android:text="Hi, I'm"
        android:textColor="#ffffff"
        android:textStyle="bold" />

    <ImageView
        android:id="@+id/wifi"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/Mood"
        android:layout_marginEnd="57dp"
        android:layout_marginRight="57dp"
        android:layout_toLeftOf="@+id/Plantname"
        android:layout_toStartOf="@+id/Plantname"
        android:adjustViewBounds="false"
        android:cropToPadding="true"
        app:srcCompat="@drawable/wifi3" />
```

```
</RelativeLayout>
```

### 4.1.9. Android Manifest XML Code

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="id.sample.test">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"></activity>
        <activity
            android:name=".SplashActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MoistureActivity" />
        <activity android:name=".TempActivity" />
        <activity android:name=".LightActivity" />
        <activity android:name=".SchedActivity" />
        <activity android:name=".infoActivity" />
        <activity android:name=".helpActivity" />
        <activity android:name=".WifiActivity"></activity>
    </application>

</manifest>
```

### 4.1.10. Info Activity Layout XML Code

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="id.sample.test.infoActivity"
    android:background="@drawable/plantinfo">

    <TextView
        android:id="@+id/textView"
        android:layout_width="166dp"
        android:layout_height="46dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="38dp"
        android:layout_marginStart="38dp"
        android:layout_marginTop="41dp"
        android:text="Plant Info and Gardening Tips"
        android:textColor="#ffffffff"
        android:textSize="16dp"
        android:textStyle="bold|italic"
        tools:layout_editor_absoluteX="41dp"
        tools:layout_editor_absoluteY="50dp" />

</RelativeLayout>
```

### 4.1.11. Schedule Activity Layout XML Code

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="id.sample.test.SchedActivity"
    android:background="@drawable/bg_info">

    <TextView
        android:id="@+id/porttextview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/calendarView"
        android:layout_alignStart="@+id/calendarView"
        android:layout_below="@+id/calendarView"
        android:layout_marginLeft="28dp"
        android:layout_marginStart="28dp"
        android:layout_marginTop="28dp"
        android:text="To-do Fertilizing Schedule:   "
        android:textColor="#ffffff"
        android:textSize="16dp"
        android:textStyle="bold|italic"
        tools:layout_editor_absoluteX="42dp"
        tools:layout_editor_absoluteY="409dp" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/porttextview"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="31dp"
        android:text="date"
        android:textColor="#ffffff"
        android:textSize="24dp"
        android:textStyle="bold"
        tools:layout_editor_absoluteX="168dp"
        tools:layout_editor_absoluteY="447dp" />

    <CalendarView
        android:id="@+id/calendarView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="51dp"
        tools:layout_editor_absoluteX="18dp"
        tools:layout_editor_absoluteY="43dp" />

</RelativeLayout>
```

### 4.1.12. Moisture Activity Layout XML Code

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="id.sample.test.MoistureActivity"
    android:background="@drawable/moist_screen2">

    <TextView
        android:id="@+id/conmoist"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/condimoist"
        android:layout_centerHorizontal="true"
        android:layout_gravity="center_vertical"
        android:layout_marginBottom="128dp"
        android:text="-"
        android:textColor="#000033"
        android:textSize="110sp" />

    <TextView
        android:id="@+id/condimoist"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="61dp"
        android:text="not connected"
        android:textColor="#7c5327"
        android:textSize="24dp" />

</RelativeLayout>
```

### 4.1.13. Temperature Activity Layout XML Code

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="id.sample.test.TempActivity"
    android:background="@drawable/suhu_screen2">

    <TextView
        android:id="@+id/contemp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/conditemp"
        android:layout_centerHorizontal="true"
        android:layout_gravity="center_vertical"
        android:layout_marginBottom="128dp"
        android:text="-"
        android:textColor="#000033"
        android:textSize="110sp" />

    <TextView
        android:id="@+id/conditemp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="61dp"
        android:text="not connected"
        android:textColor="#7c5327"
        android:textSize="24dp" />


</RelativeLayout>
```

### 4.1.14. Splash Activity Layout XML Code

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="id.sample.test.SplashActivity"
    >

    <ImageView
        android:id="@+id/SplashView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:srcCompat="@drawable/welcomenu"
        android:scaleType="fitXY"/>

    <ProgressBar
        android:id="@+id/progressBar"
        style="@style/Widget.AppCompat.ProgressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="49dp"
        android:progressTint="#66cc33" />


</RelativeLayout>
```

### 4.1.15. Wi-Fi Activity Java Code

```java
package id.sample.test;

import android.content.Context;
import android.content.SharedPreferences;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.os.AsyncTask;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URI;
import java.net.URISyntaxException;

public class WifiActivity extends AppCompatActivity implements
View.OnClickListener {

    public final static String PREF_IP = "PREF_IP_ADDRESS";
    public final static String PREF_PORT = "PREF_PORT_NUMBER";
    public final static String RESPONSE = "SERVER_RESPONSE";
    private EditText editTextIPAddress, editTextPortNumber;
    private Button connect;

    // shared preferences objects used to save the IP address and port so
that the user doesn't have to
    // type them next time he/she opens the app.
    SharedPreferences sharedPreferences;
    SharedPreferences.Editor editor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_wifi);
        // declare buttons and text inputs
        connect = (Button) findViewById(R.id.connect);
        editTextIPAddress = (EditText) findViewById(R.id.ipedit);
        editTextPortNumber = (EditText) findViewById(R.id.portedit);
        sharedPreferences =
getSharedPreferences("HTTP_HELPER_PREFS",MODE_PRIVATE);
        editor = sharedPreferences.edit();
        // set button listener (this class)
        connect.setOnClickListener(this);
        // get the IP address and port number from the last time the user
used the app,
        // put an empty string "" is this is the first time.
        editTextIPAddress.setText(sharedPreferences.getString(PREF_IP,""));

editTextPortNumber.setText(sharedPreferences.getString(PREF_PORT,""));
    }

    @Override
    public void onClick(View view) {
        // get the ip address
        String ipAddress = editTextIPAddress.getText().toString().trim();
        // get the port number
        String portNumber = editTextPortNumber.getText().toString().trim();

        // save the IP address and port for the next time the app is used
        editor.putString(PREF_IP,ipAddress); // set the ip address value to
save
        editor.putString(PREF_PORT,portNumber); // set the port number to
save
        editor.commit(); // save the IP and PORT

        // execute HTTP request
        if(ipAddress.length()>0 && portNumber.length()>0) {
            new HttpRequestAsyncTask(view.getContext(),ipAddress,
portNumber).execute();
        }
    }
    /**
     * Description: Send an HTTP Get request to a specified ip address and
port.
     * @param ipAddress the ip address to send the request to
     * @param portNumber the port number of the ip address
     * @return The ip address' reply text, or an ERROR message if it fails
to receive one
     */
    public String sendRequest(String ipAddress, String portNumber) {
        String serverResponse = "Oops! Can't connect to pot right now :(";
        try {

            HttpClient httpclient = new DefaultHttpClient(); // create an
HTTP client
            // define the URL e.g. http://myIpaddress:myport/?pin=13 (to
toggle pin 13 for example)
            URI website = new URI("http://"+ipAddress+":"+portNumber);
            HttpGet getRequest = new HttpGet(); // create an HTTP GET object
            getRequest.setURI(website); // set the URL of the GET request
            HttpResponse response = httpclient.execute(getRequest); //
execute the request
            // get the ip address server's reply
            InputStream content = null;
            content = response.getEntity().getContent();
            BufferedReader in = new BufferedReader(new InputStreamReader(
                    content
            ));
            serverResponse = in.readLine();
            // Close the connection
            content.close();
        } catch (ClientProtocolException e) {
            // HTTP error
            serverResponse = e.getMessage();
            e.printStackTrace();
        } catch (IOException e) {
            // IO error
            serverResponse = e.getMessage();
            e.printStackTrace();
        } catch (URISyntaxException e) {
            // URL syntax error
            serverResponse = e.getMessage();
            e.printStackTrace();
        }
        editor.putString(RESPONSE,serverResponse);
        // return the server's reply/response text
        return serverResponse;
    }
    /**
     * An AsyncTask is needed to execute HTTP requests in the background so
that they do not
     * block the user interface.
     */
    private class HttpRequestAsyncTask extends AsyncTask <Void, Void, Void>
{

        // declare variables needed
        private String requestReply,ipAddress, portNumber;
        private Context context;
        private AlertDialog alertDialog;
        private String parameter;
        private String parameterValue;

        /**
         * Description: The asyncTask class constructor. Assigns the values
used in its other methods.
         * @param context the application context, needed to create the
dialog
         * @param ipAddress the ip address to send the request to
         * @param portNumber the port number of the ip address
         */
        public HttpRequestAsyncTask(Context context,String ipAddress, String
portNumber)
        {
            this.context = context;

            alertDialog = new AlertDialog.Builder(this.context)
                    .setTitle("The Plant.io pot says:")
                    .setCancelable(true)
                    .create();

            this.ipAddress = ipAddress;
            this.portNumber = portNumber;
        }
        /**
         * Name: doInBackground
         * Description: Sends the request to the ip address
         * @param voids
         * @return
         */
        @Override
        protected Void doInBackground(Void... voids) {
            alertDialog.setMessage("Waiting for Plant.io pot to notice
you...");
            if(!alertDialog.isShowing())
            {
                alertDialog.show();
            }
            requestReply = sendRequest(ipAddress,portNumber);
            return null;
        }

        /**
         * Name: onPostExecute
         * Description: This function is executed after the HTTP request
returns from the ip address.
         * The function sets the dialog's message with the reply text from
the server and display the dialog
         * if it's not displayed already (in case it was closed by
accident);
         * @param aVoid void parameter
         */
        @Override
        protected void onPostExecute(Void aVoid) {
            alertDialog.setMessage(requestReply);
            if(!alertDialog.isShowing())
            {
                alertDialog.show(); // show dialog
            }
        }

        /**
         * Name: onPreExecute
         * Description: This function is executed before the HTTP request is
sent to ip address.
         * The function will set the dialog's message and display the
dialog.
         */
        @Override
        protected void onPreExecute() {
            alertDialog.setMessage("Trying to talk with the pot, please
wait...");
            if(!alertDialog.isShowing())
            {
                alertDialog.show();
            }
        }

    }
}
```

### 4.1.16. Wi-Fi activity XML code

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="id.sample.test.WifiActivity"
    android:background="@drawable/bg_tes">

    <TextView
        android:id="@+id/iptextview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/iptextview"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="49dp"
        android:text="IP Address:" />

    <TextView
        android:id="@+id/porttextview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/portedit"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="36dp"
        android:text="Port Number:" />

    <EditText
        android:id="@+id/ipedit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_marginTop="94dp"
        android:hint="e.g. 192.168.0.10" />

    <EditText
        android:id="@+id/portedit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:ems="10"
        android:hint="e.g. 80"
        android:inputType="number" />

    <Button
        android:id="@+id/connect"
        style="@style/Widget.AppCompat.Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/portedit"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="29dp"
        android:text="connect" />


</RelativeLayout>
```

### 4.2. Hardware Implementation

Arduino IDE is used in implementing the hardware code. Source code consists of a single .ino code for the main functions of the Arduino, written in embedded C. There are several header files used for additional library other than the default library given by the Arduino IDE.

#### 4.2.1. Arduino Source Code

```
#include <DHT.h>
#include <SoftwareSerial.h>

DHT DHT;
SoftwareSerial esp8266(0, 1);      // RX, TX for ESP8266

bool DEBUG = true;    //show more logs
int responseTime = 10; //communication timeout

#define DHT11_PIN 7 //input for temperature
#define WATERPIN 2 //output for water pump
#define READSOILPIN A0 //input for soil moisture
#define MAXDRYNESS 700 //define upper limit of soil moisture
#define WATERDELAY 750 // define delay for pre water pump
#define WATERPOSTDELAY 3000 //define delay for post water pump
#define SensorPin A1 //input for light

void setup()
{
  Serial.begin(9600);
  pinMode(READSOILPIN, INPUT);
  pinMode(DHT11_PIN, INPUT);
  pinMode(SensorPin, INPUT);
  pinMode(WATERPIN, OUTPUT);
  esp8266.begin(9600);
  sendCommand("AT+RST\r\n",2000,DEBUG); // reset module
  sendCommand("AT+CWMODE=1\r\n",1000,DEBUG); // configure as access point
  sendCommand("AT+CWJAP=\"Plantio\",\"Rplgroup9\"\r\n",3000,DEBUG); //set
SSID as Plantio and password as Rplgroup9
  delay(10000);
  sendCommand("AT+CIFSR\r\n",1000,DEBUG); // get ip address
  sendCommand("AT+CIPMUX=1\r\n",1000,DEBUG); // configure for multiple
connections
  sendCommand("AT+CIPSERVER=1,80\r\n",1000,DEBUG); // turn on server on port
80 (http)
}

void loop()
{
  int chk = DHT.read11(DHT11_PIN);
  Serial.print("\nTemperature\t= ");
  Serial.println(DHT.temperature);
  delay(1000);
  int SensorValueSoil = ((950-analogRead(READSOILPIN))/950)*100;
  Serial.print("Soil Moisture\t= ");
  Serial.println(SensorValueSoil);
  Serial.print("%");
  if(SensorValueSoil >= MAXDRYNESS)
  {
    Serial.print("Soil Dry, Start Watering");
    digitalWrite(WATERPIN, HIGH);
    delay(WATERDELAY);
    digitalWrite(WATERPIN, LOW);
    delay(WATERPOSTDELAY);
  }
  delay(50);

  int SensorValueLDR;
  SensorValueLDR = (analogRead(SensorPin)/1000)*100;
  Serial.print("Light Intensity\t= ");
  Serial.println(SensorValueLDR);
  Serial.print("%");
  delay(100);

  if(esp8266.available())
  {
     // get the connection id so that we can then disconnect
     int connectionId = esp8266.read()-48; // subtract 48 because the read()
function returns
                                            // the ASCII decimal value and 0
(the first decimal number) starts at 48

     String content;
     content += SensorValueSoil;
     sendHTTPResponse(connectionId,content);
     delay(1000);

     content += DHT.temperature;
     sendHTTPResponse(connectionId,content);
     delay(1000);

     content += SensorValueLDR;
     sendHTTPResponse(connectionId,content);
     delay(1000);

     // make close command
     String closeCommand = "AT+CIPCLOSE=";
     closeCommand+=connectionId; // append connection id
     closeCommand+="\r\n";

     sendCommand(closeCommand,1000,DEBUG); // close connection
  }
}

/*
* Name: sendData
* Description: Function used to send data to ESP8266.
* Params: command - the data/command to send; timeout - the time to wait for
a response; debug - print to Serial window?(true = yes, false = no)
* Returns: The response from the esp8266 (if there is a reponse)
*/
```

```
String sendData(String command, const int timeout, boolean debug)
{
    String response = "";

    int dataSize = command.length();
    char data[dataSize];
    command.toCharArray(data,dataSize);

    esp8266.write(data,dataSize); // send the read character to the esp8266
    if(debug)
    {
      Serial.println("\r\n====== HTTP Response From Arduino ======");
      Serial.write(data,dataSize);
      Serial.println("\r\n========================================");
    }

    long int time = millis();

    while( (time+timeout) > millis())
    {
      while(esp8266.available())
      {

        // The esp has data so display its output to the serial window
        char c = esp8266.read(); // read the next character.
        response+=c;
      }
    }

    if(debug)
    {
      Serial.print(response);
    }

    return response;
}

/*
* Name: sendHTTPResponse
* Description: Function that sends HTTP 200, HTML UTF-8 response
*/
void sendHTTPResponse(int connectionId, String content)
{

     // build HTTP response
     String httpResponse;
     String httpHeader;
     // HTTP Header
     httpHeader = "HTTP/1.1 200 OK\r\nContent-Type: text/html; charset=UTF-
8\r\n";
     httpHeader += "Content-Length: ";
     httpHeader += content.length();
     httpHeader += "\r\n";
     httpHeader +="Connection: close\r\n\r\n";
     httpResponse = httpHeader + content + " "; // There is a bug in this
code: the last character of "content" is not sent, I cheated by adding this
extra space
     sendCIPData(connectionId,httpResponse);
}

/*
* Name: sendCIPDATA
* Description: sends a CIPSEND=<connectionId>,<data> command
*
*/
void sendCIPData(int connectionId, String data)
{
   String cipSend = "AT+CIPSEND=";
   cipSend += connectionId;
   cipSend += ",";
   cipSend +=data.length();
   cipSend +="\r\n";
   sendCommand(cipSend,1000,DEBUG);
   sendData(data,1000,DEBUG);
}

/*
* Name: sendCommand
* Description: Function used to send data to ESP8266.
* Params: command - the data/command to send; timeout - the time to wait for
a response; debug - print to Serial window?(true = yes, false = no)
* Returns: The response from the esp8266 (if there is a reponse)
*/
String sendCommand(String command, const int timeout, boolean debug)
{
    String response = "";

    esp8266.print(command); // send the read character to the esp8266

    long int time = millis();

    while( (time + timeout) > millis())
    {
      while(esp8266.available())
      {

        // The esp has data so display its output to the serial window
        char c = esp8266.read(); // read the next character.
        response+=c;
      }
    }

    if(debug)
    {
      Serial.print(response);
    }

    return response;
}
```

# V. TESTING ANALYSIS

This chapter documents and tracks the necessary information required to effectively define the testing of the project's product. The test plan document is created during the planning phase of the project. Unit testing are done by team member who does not act as the unit code author. System testing will be done by customer using standard Software Usability Scale questionnaire, and will be done after all the unit testing have been determined pass.

## 5.1. Compatibility Testing

### 5.1.1. Test Risk / Issues

Plant.io is an application developed using Android Studio, a free and official platform of android application development provided by Google and Intellij. In this testing plan, Plant.io are tested in multiple android devices with various Android OS version.

### 5.1.2. Items to be Tested

| Item to Test | Test Description | Test Date | Responsibility |
|---|---|---|---|
| Application | Does it run well in various android devices and various android OS | November 21st, 2017 | Whisnu Samudra<br>Aqila F. Karim |

### 5.1.3. Test Approach

This test goal is to find if there are any operating system or device not compatible with the application and to find as many errors and bugs as possible to repair and retest it before the final release of the application.

### 5.1.4. Test Regulatory / Mandate Criteria

To perform the test, the only pre-condition needed is the application installed on tester's device. Multiple devices type with different android OS version is needed to check the application's compatibility.

### 5.1.5. Test Pass / Fail Criteria

Compatibility test is determined passed if the application runs well in the multiple operating system in which the Plant.io is designed to be compatible to.

Compatibility test is determined pass if all the graphics, interface, and functions are working correctly and on the right places. This test is determined failed if the graphics and text is not where they should be, and/or the functions intended are not working according to plan.

### 5.1.6. Test Entry / Exit Criteria

The test can be conducted if:

- The Android device has Plant.io installed

The test can be stopped when:

- All the program features and functions run well on the operating system intended

### 5.1.7. Test Deliverables

The result of this test is delivered with a documentation, including device's specification, running-apps screenshot and error analysis.

The result is as follows:

| Tester Number | Device's Name | Device's OS | Passed or Failed |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 1 | Samsung Galaxy S6 Edge | Android Nougat, API 24 | Passed |
| 2 | Samsung Galaxy A5 | Android Marshmallow, API 23 | Passed |
| 3 | Asus Zenfone 3 Max | Android Marshmallow, API 23 | Passed |

#### 5.1.8. Test Suspension

The test will be suspended if there are any obstacles, such as the availability of the devices that supports the test, readiness of the features which will be tested, and the team member's availability. The test will be resumed if all the requirements needed is fulfilled.

#### 5.1.9. Test Environmental

The environment of test is the supporting programs to create the application, and the operating system used to run the program. No special trainings are needed to run the test, but all responsible team members should learn Java programming language, since it is the programming language used in Plant.io.

### 5.2. Functional Testing

#### 5.2.1. Test Risks / Issues

It is possible that some internal algorithm issues may be faced since the test will be performed without further insight to the code itself. Also, human error in determining the test failed or passed is a common factor of error in this test. To minimize the error, we will perform more than one test and re-examine the algorithm each time after the function test is done.

### 5.2.2. Items to Test

| Items to Test | Test Description | Test Date | Test Responsibility |
|---|---|---|---|
| Graphics | Does all the image used shows correctly on the screen | 22/11/2017 | Whisnu Samudra |
| Text | Does all the text and explanation show correctly for each activity | | Haekal F.R |
| Hardware | Does the hardware work like it should be | | Aqila F. Karim |
| Scheduling | Does the scheduling system work and the date are saved | | Aqila F. Karim |
| Plant's Condition | Does the data from hardware delivered to the app and showed correctly | | Aqila F. Karim Whisnu Samudra |
| Plant's Name | Does the plant's name that user use saved and showed correctly | | Haekal F. R |

### 5.2.3. Test Approach(s)

This test's main goal is to compare the application's functionality with what the application was planned to be, described by the documentation. Our main testing is to synchronize data from hardware and the application, application functionality such as plant's condition and scheduling, and navigation between screens.

### 5.2.4. Test Regulatory / Mandate Criteria

The minimum criteria for running this test is that the hardware and application have passed the compatibility test.

### 5.2.5. Test Pass/ Fail Criteria

The test succeeded if the application works as expected, and all the screen such as main screen, plant's condition screen, scheduling screen works as expected.

### 5.2.6. Test Entry / Exit Criteria

The testing begins when the application runs in the device, and will be stopped when all the test conditions works correctly, such as scheduling can be saved, and all the screen and button works correctly. The test also stopped when the tester found a button or a function that does not work correctly.

### 5.2.7. Test Deliverables

This test will deliver testing documentation such as how many trials the problems encountered and the solution for each problem.

| Tester Number | How Many Trials | Problems Encountered | Solution | Passed or Failed |
|---|---|---|---|---|
| 1 | 5 | Splash screen won't show, connection cannot be established | Working on the splash screen activity code and connection code | 5 out of 6 test items passed |

| | | | | |
|---|---|---|---|---|
| 2 | 6 | Scheduling screen won’t show the calendar, connection cannot be established | Working on the scheduling screen activity code and connection code | 5 out of 6 test items passed |
| 3 | 4 | Plant’s name can’t be saved, connection cannot be established | Working on the main screen activity code and connection code | 5 out of 6 test items passed |

#### 5.2.8. Test Suspension / Resumption Criteria

The application functionality testing will be suspended if there are any errors or bugs in the application that needs to be fixed before resuming test, or when the team have a higher priority task to do.

#### 5.2.9. Test Environmental / Staffing / Training Needs

The requirement for the tester are knowledge of the application’s purposes, usage of each button, and expected results of each test condition.

### 5.3. Performance Testing

#### 5.3.1. Test Risk / Issues

It is very common in performance testing that the application will crash when tester try to measure its capability of handling data because of RAM limits of the device.

### 5.3.2. Items to Test

| Item to Test | Test Description | Test Date | Responsibility |
|---|---|---|---|
| Loading Speed | Determining the speed of this application and time needed for the tester set something | 22/11/2017 | Haekal F. R |
| Bug | Determining if there is some bug or error that this application have | | Aqila F. Karim<br>Whisnu Samudra |

### 5.3.3. Test Approach(s)

The tested application will be measured by the amount of time needed to load screen and features. The time will be compared with other devices in the same test.

### 5.3.4. Test Regulatory / Mandate Criteria

The regulations are that the device have already installed the application and passed compatibility test.

### 5.3.5. Test Pass / Fail Criteria

The test is determined passed if the application and each feature loads in maximum of two seconds and the refresh rate is not longer than 5 seconds periodically. This numbers can be changed depends on what type of mobile phone that is used for testing.

### 5.3.6. Test Entry / Exit Criteria

The test started when the criteria is fulfilled, and stopped when the application passed or failed the test after three trials.

### 5.3.7. Test Deliverables

The deliverables in this test are tables of each trial made on each device and their average of time taken to load the app.

| Test Device | Average of Loading Time (second) | Bug Found | Passed or Failed |
|---|---|---|---|
| 1 | 1 | Nothing | Passed |
| 2 | 2 | Nothing | Passed |
| 3 | 2 | Nothing | Passed |

### 5.3.8. Test Suspension / Resumption Criteria

The test stopped when all devices has passed the second trial in loading time or if a bug is found.

### 5.3.9. Test Environmental / Staffing / Training Needs

There are no special requirements needed, but tester should know the application’s purposes and expected results of each test condition.

# VI. USER MANUAL

This chapter will explain about user's manual of this project.

## 6.1. USER MANUAL

### 6.1.1. General Information

Plant.io is a plant-monitoring mobile application integrated with self-watering pot system that aims to help people in taking care of plants and to give gardening education to the users.

### 6.1.2. Project References

This application is developed by learning from website references below:

- http://www.developer.android.com
- http://www.okedroid.com
- http://allaboutee.com/2015/01/20/esp8266-android-application-for-arduino-pin-control/
- https://github.com/cihadguzel/android-nodemcu-connection
- https://www.arduino.cc/reference/en/
- https://github.com/esp8266/Arduino/tree/4897e0006b5b0123a2fa31f67b14a3fff65ce561

### 6.1.3. Authorized Use Permission

All the contents in this application can be accessed by the user.

## 6.2. Point of Contact

### 6.2.1. Information

This section provides a list of the points of organizational contact (POCs) that may be needed by document user for informational and troubleshooting purposes. The POCs includes type of contact, contact name, telephone number, and email address (if applicable). Points of contact may include, but are not limited to, help desk POC, development/maintenance POC, and operations POC.

**Aqila Fathimah Karim**

As Project Manager, Interface Designer, and Programmer

- Email : aqila.fathimah@ui.ac.id
- Phone number : +62 838 970 66584

### 6.2.2. Help Desk

This section provides help desk information including responsible developer's phone numbers for emergency assistance.

1. **Haekal Febriansyah Ramadhan** -as Programmer and Hardware Designer

- Email : haekal.febriansyah@ui.ac.id
- Phone number : +62 812 967 1767

2. **Whisnu Samudra** - as Programmer and Documentation Manager

- Email : whisnu.samudra@ui.ac.id
- Phone number : +62 857 188 48415

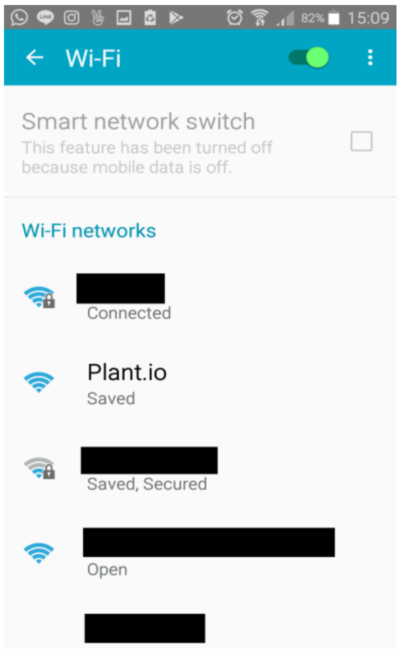
## 6.3. Getting Started

### 6.3.1. Installation

Plant.io is a mobile application which is developed by using Android Studio and Arduino IDE. It is designed to run in full compatibility in Android version 4.0 and above. To install this application, simply download and run the application's apk file.

### 6.3.2. Wi-Fi Connection

Before launching the application, to be able to receive data from the pot, the phone must be connected to pot's Wi-Fi. To do this, go to user's phone network settings and turn on Wi-Fi connection. The pot's Wi-Fi will be discovered as "Plant.io". Connect to this network.



Smart network switch
This feature has been turned off because mobile data is off.

Wi-Fi networks

[redacted]
Connected

Plant.io
Saved

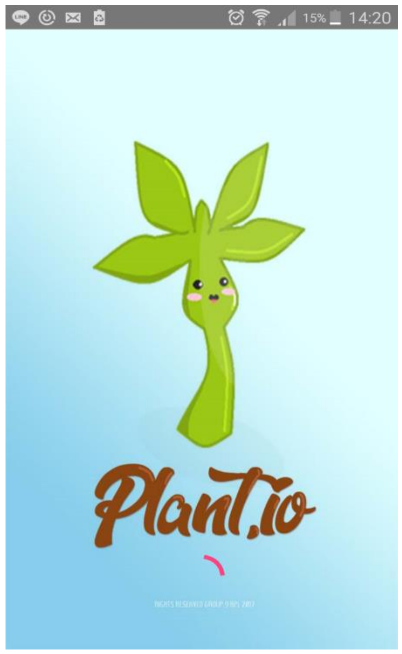
[redacted]
Saved, Secured

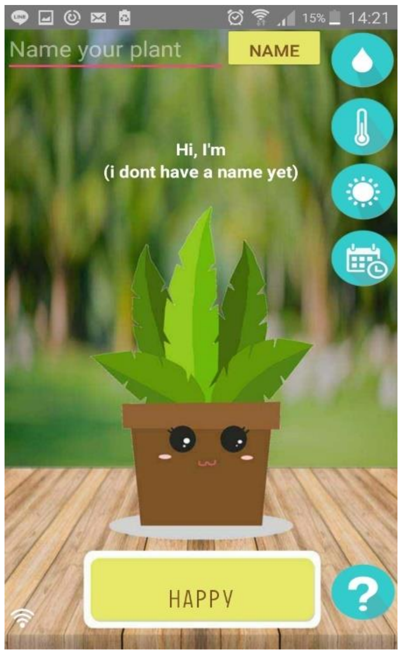
[redacted]
Open

[redacted]

### 6.3.3. Application Launch

Run the downloaded application and the application will launch with a splash screen, followed by the home screen as following:

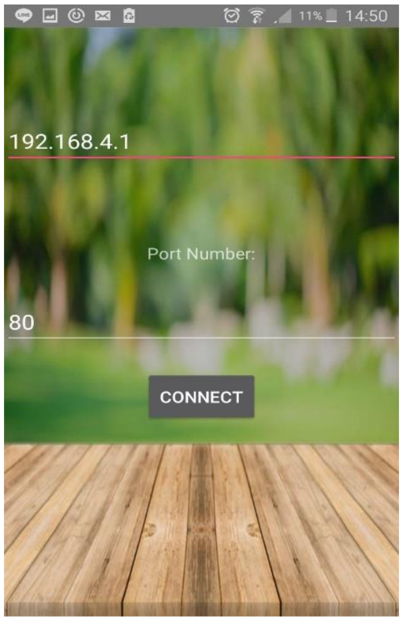




The main screen will show a figure of plant and its mood (happy or sad). Its condition is based on the data collected from the pot. If two or more aspects (e.g. soil moisture and light intensity) is detected to be in bad condition, the plant's mood will change to sad. When opening it for the first time and connection is not yet established with the pot, it will show the mood as no connection. If a connection has been made before and user exits and launches the application again (without reattempting connection), the preferences from the last connection will remain. When the application is launched without connection to the pot, an alert message on the bottom of the main screen will notice the user that the pot is not connected.

### 6.3.4. Pot Connection Page

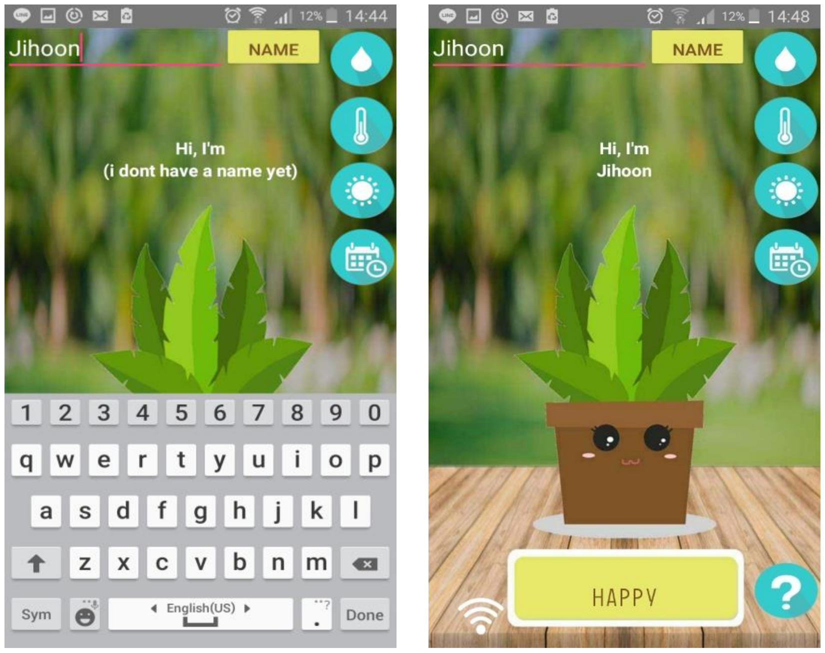
Click on the Wi-Fi icon on the bottom left corner of the main screen to navigate to the connection page. If this is the first time connecting, user is required to input the IP address and port for the pot connection (default IP address and port are 192.168.4.1 and 80). An alert dialog will show if a connection is established successfully.



Click on phone’s back button to navigate back to home screen.

### 6.3.5. Naming the Plant

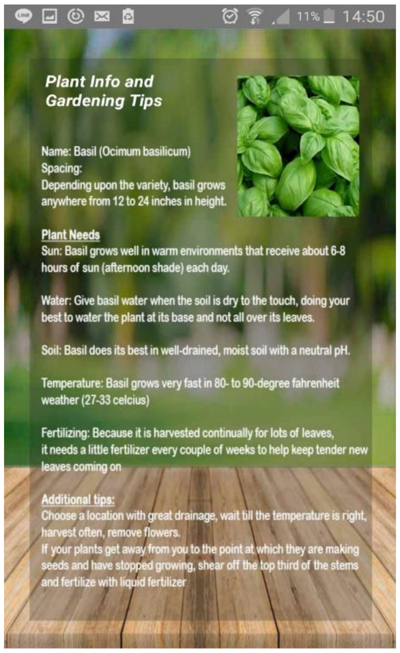
On the upper left corner of the main screen is provided a text field where user can type in desired plant’s name. Click on the NAME button to apply the name.



Click on phone’s back button to navigate back to home screen.

### 6.3.6. Plant Information Page

Click on the plant’s name to see the plant information page, containing a real-life picture of the plant, general info, and ideal surroundings of the plant for gardening guide.



*Plant Info and Gardening Tips*

Name: Basil (Ocimum basilicum)
Spacing:
Depending upon the variety, basil grows anywhere from 12 to 24 inches in height.

Plant Needs
Sun: Basil grows well in warm environments that receive about 6-8 hours of sun (afternoon shade) each day.

Water: Give basil water when the soil is dry to the touch, doing your best to water the plant at its base and not all over its leaves.

Soil: Basil does its best in well-drained, moist soil with a neutral pH.

Temperature: Basil grows very fast in 80- to 90-degree fahrenheit weather (27-33 celcius)

Fertilizing: Because it is harvested continually for lots of leaves, it needs a little fertilizer every couple of weeks to help keep tender new leaves coming on

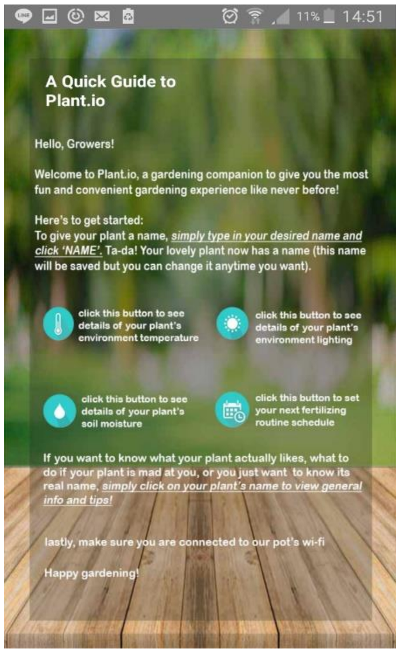
Additional tips:
Choose a location with great drainage, wait till the temperature is right, harvest often, remove flowers.
If your plants get away from you to the point at which they are making seeds and have stopped growing, shear off the top third of the stems and fertilize with liquid fertilizer

Click on phone’s back button to navigate back to home screen.

### 6.3.7. Help Page

Click on the question mark icon on the bottom right corner of the main screen to navigate to the help page, containing a user guide to run the application.

Click on phone’s back button to navigate back to home screen.



11% 14:51

A Quick Guide to Plant.io

Hello, Growers!

Welcome to Plant.io, a gardening companion to give you the most fun and convenient gardening experience like never before!

Here's to get started:
To give your plant a name, *simply type in your desired name and click 'NAME'.* Ta-da! Your lovely plant now has a name (this name will be saved but you can change it anytime you want).

click this button to see details of your plant's environment temperature

click this button to see details of your plant's environment lighting

click this button to see details of your plant's soil moisture

click this button to set your next fertilizing routine schedule

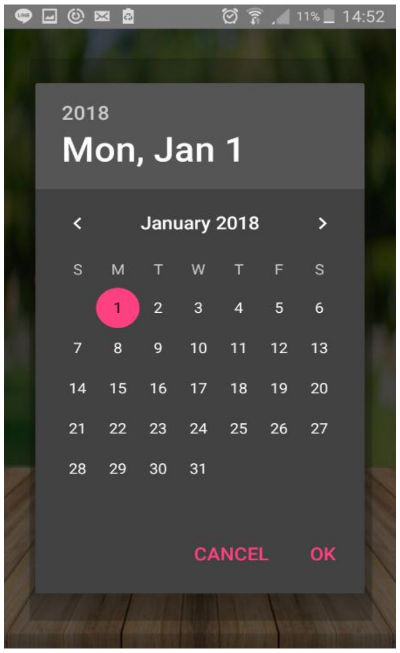
If you want to know what your plant actually likes, what to do if your plant is mad at you, or you just want to know its real name, *simply click on your plant's name to view general info and tips!*

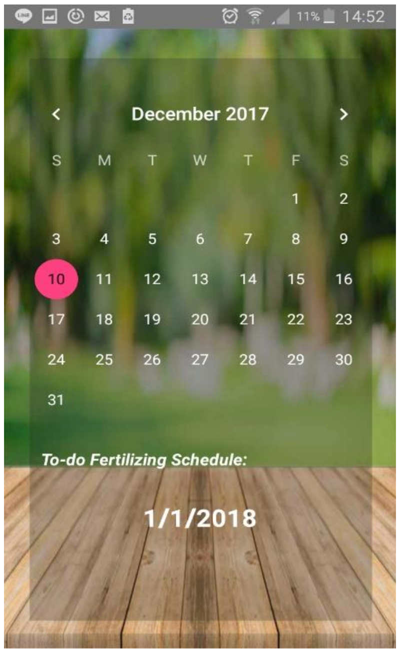
lastly, make sure you are connected to our pot's wi-fi

Happy gardening!

### 6.3.8. Fertilizing Schedule Page

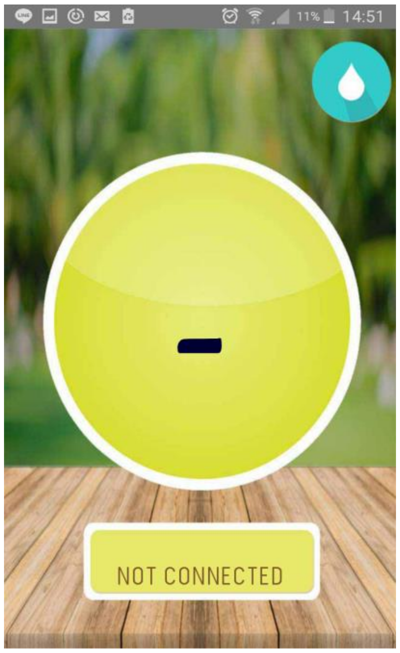
On the upper right side of the main screen, there are a set of four buttons, each with its own function. Click on the calendar icon (the fourth button from the top) to navigate to the fertilizing schedule page. The phone's current date will be shown by a calendar and a to-do fertilizing date is shown below the calendar. Click on the to-do fertilizing date to set a new date.





### 6.3.9. Soil Moisture Page

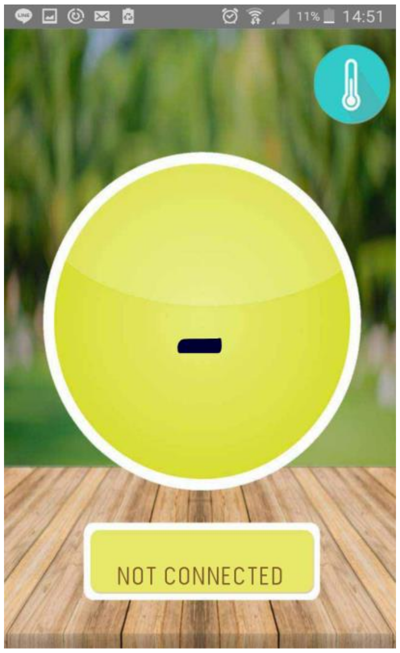
Click on the water drop icon (first button on the upper right side) to navigate to the soil moisture page. Detail of the soil moisture level as well as the condition (good, too dry, or too wet) will be shown on the page. It will show as not connected when connection has not been established with the pot.



Click the phone’s back button to navigate back to home screen.

### 6.3.10. Temperature Page

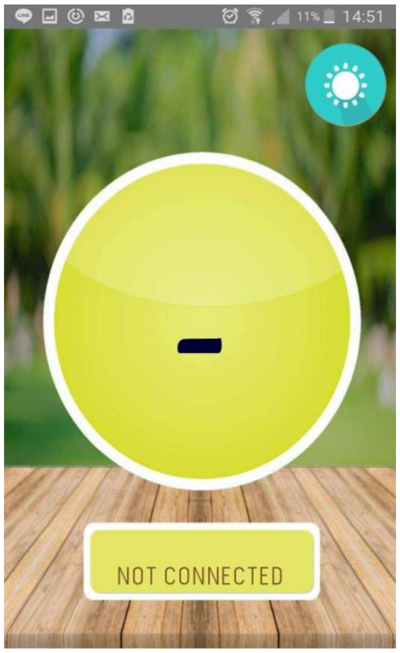
Click on the thermometer icon (second button on the upper right side) to navigate to the surrounding’s temperature page. Detail of the soil temperature in Celsius degrees as well as the condition (good, too hot, or too cold) will be shown on the page. It will show as not connected when connection has not been established with the pot.



Click the phone's back button to navigate back to home screen.

### 6.3.11. Light Intensity Page

Click on the sun icon (third button on the upper right side) to navigate to the surrounding's light intensity page. Detail of the light intensity level as well as the condition (good, too dark, or too bright) will be shown on the page. It will show as not connected when connection has not been established with the pot.



Click the phone's back button to navigate back to home screen.

### 6.3.12. Maintenance Capabilities

Any other automation or improvements could be implemented in the future, based on the suggestions and results of testing form and questionnaire.