GUJARAT TECHNOLOGICAL UNIVERSITY

Chandkheda , Ahmedabad

Affiliated

L.D COLLEGE OF ENGINEERING

A

Project Report

On

WEATHER REPORTING SYSTEM USING IOT

Under the subject of

Design Engineering

B.E. III , Semester – VI

Instrumentation and Control Engineering

Academic Year 2023-2024

TEAM ID :- 516731

TEAM MEMBERS:-

Vedik Rajeev Sethi – 210280117038

Vansh Rajesh Patel – 210280117064

Jiney Samir Shah – 210280117041

Harsh Abhilash Raval – 210280117067

Om Chandreshbhai Soni – 210280117042

ABSTRACT

In this report, a system is introduced to report the weather conditions in various different locations. Problem of public un-awareness is solved and even the normal public can access the weather conditions such as temperature, humidity, rainfall etc. We have used different sensors, Microcontrollers such as Arduino and Raspberry Pi for fulfilling our purposes and used the help of IoT (Internet of Things) to achieve our goal in this project. This system instantly captures data from the sensors, transmits data over IoT over display and also displays it over internet. The IoTGecko server receives data and displays the values live over the server. The user may logon to the IoTGecko system over any web browser and check live weather values anytime and anywhere.

CONTENTS

INTRODUCTION

Weather Reporting is a very important process which sometimes need instant output and speedy coordination between the teams. The general target audience for such feature is sailors , fishermen or disaster relief department and more.

However , We streamline and automates this whole procedure where system continuously receives and transfers the data over internet by using IoT i.e Internet of Things. The system uses raspberry pi, Arduino Uno Microcontrollers , several sensors for temperature, humidity , rain etc and displays it over a LCD display. There are several other components to fulfil this task also. This system provides easy-to-use quick look on the weather conditions and the easiest and the most user-friendly way.

AEIOU CANVAS

(A)ctivities are goal-directed set of actions. What are the modes People work in and the specific activities and processes does they go through?

(E)nvironment includes the place where activity is going to take place. It also refers to the environmental impact of the model.

(I)nteractions are between person/user and something else. It can be person-person , person-object , person-environment or object-environment etc. Also sometimes refers to the area where it interacts

(O)bjects are the building blocks of the environment. It refers to any non-living thing user interacts with while using the system.

(U)sers are the people who are being observed or are crucial in operating the system or the object.

AEIOU Summary of our model with a snapshot of the Canvas is shown below:-

Activities

-Measures Temperature

-Measures Humidity

-Measures AQI

-Predicts Seasonal Changes

Environment

-Compact in Size

-Eco-Friendly

-Easy to operate

-Representation on Weather Conditions

Interactions

-Lighthouse

-Vendors

-Retailers

-Research Institutions

Objects

-IoT

-Raspberry Pi

-Arduino UNO

-Sensors

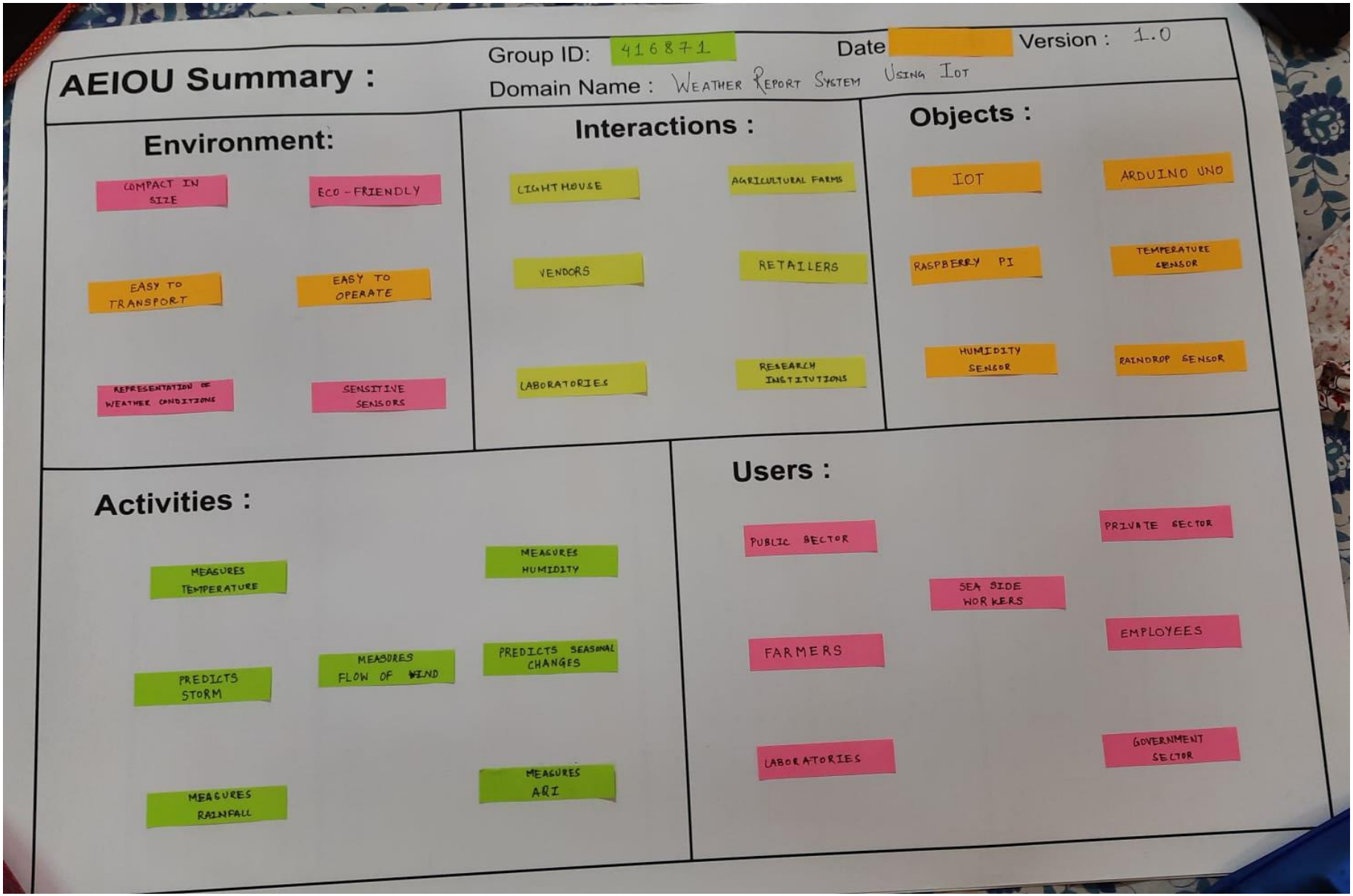
Users

-Private, Public, Government Sectors

-Employees

-Farmers

-Laboratories


AEIOU Summary :
Group ID: 416871
Date
Version : 1.0
Domain Name : Weather Report System Using Iot
Environment:
COMPACT IN SIZE
ECO - FRIENDLY
EASY TO TRANSPORT
EASY TO OPERATE
REPRESENTATION OF WEATHER CONDITIONS
SENSITIVE SENSORS
Interactions :
LIGHTHOUSE
AGRICULTURAL FARMS
VENDORS
RETAILERS
LABORATORIES
RESEARCH INSTITUTIONS
Objects :
IOT
ARDUINO UNO
RASPBERRY PI
TEMPERATURE SENSOR
HUMIDITY SENSOR
RAINDROP SENSOR
Activities :
MEASURES TEMPERATURE
MEASURES HUMIDITY
PREDICTS STORM
MEASURES FLOW OF WIND
PREDICTS SEASONAL CHANGES
MEASURES RAINFALL
MEASURES ARI
Users :
PUBLIC SECTOR
PRIVATE SECTOR
SEA SIDE WORKERS
EMPLOYEES
FARMERS
LABORATORIES
GOVERNMENT SECTOR

MIND MAPPING CANVAS

A mind map is a diagram used to visually organize information. A mind map is hierarchical and shows relationships among pieces of the whole. It is often created around a single concept, drawn as an image in the center of a blank page, to which associated representations of ideas such as images, words and parts of words are added. Major ideas are connected directly to the central concept, and other ideas branch out from those major ideas.

Mind maps can also be drawn by hand, either as "rough notes" during a lecture, meeting or planning session, for example, or as higher quality pictures when more time is available. Mind maps are considered to be a type of spider diagram.

Some Points of our Mind Map canvas and a snapshot of it is given below:-

IOT BASED WEATHER REPORTING

→Basic Definition

-Predicts weather conditions

-Informs user about the extremes or moderate temperature

→Weather Report Type

-Checks temperature

-Checks humidity

-Predicts rainfall

→Useful Sectors

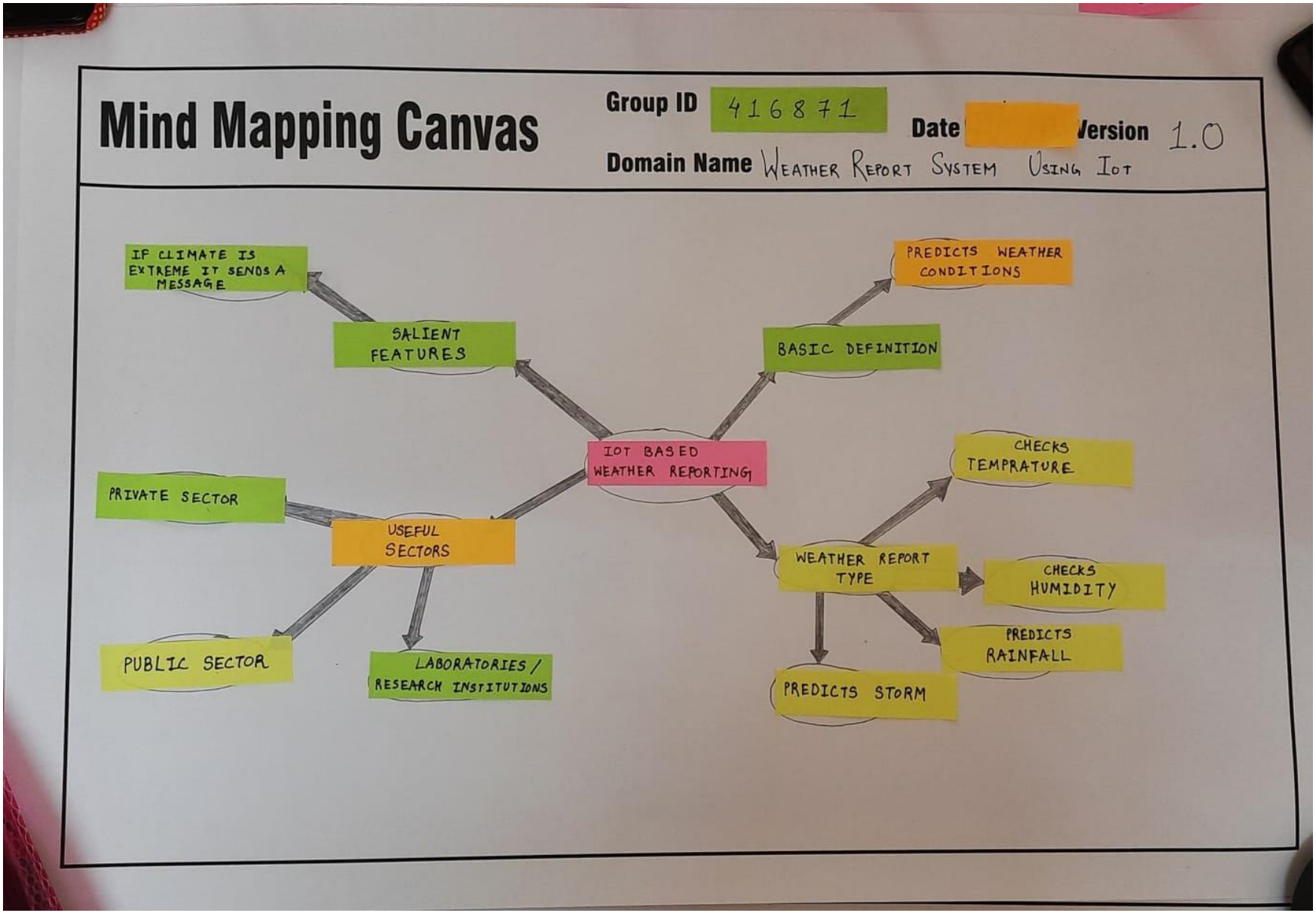
-Private sectors

-Public sectors

-Laboratories

→Salient Features

-Alerts the user in form of message if climate becomes extreme


Mind Mapping Canvas
Group ID 416871
Date
Version 1.0
Domain Name WEATHER REPORT SYSTEM USING IOT
IF CLIMATE IS EXTREME IT SENDS A MESSAGE
SALIENT FEATURES
PREDICTS WEATHER CONDITIONS
BASIC DEFINITION
IOT BASED WEATHER REPORTING
CHECKS TEMPRATURE
PRIVATE SECTOR
USEFUL SECTORS
WEATHER REPORT TYPE
CHECKS HUMIDITY
PREDICTS RAINFALL
PUBLIC SECTOR
LABORATORIES / RESEARCH INSTITUTIONS
PREDICTS STORM

IDEATION CANVAS

Ideation Canvas is a systematic arrangement of selected concepts side by side that tries to guide the mind of infinite ideators to innovative answers to specific questions.

Some points of Ideation Canvas and a snapshot of the canvas is given below:-

PEOPLE

-Scientists

-Researchers

-Farmers

-Sea Side Workers

ACTIVITIES

-Predicts Average Rainfall

-Indicates Humidity

-Indicates Temperature

-Indicates AQI

-Alerts About The Severity Of The Weather

SITUATION/CONTEXT/LOCATION

-Water Bodies

-Seasonal Changes

-Laboratories

-Pollution

PROPS

-IoT

-LCD Display

-Crystal Oscillator

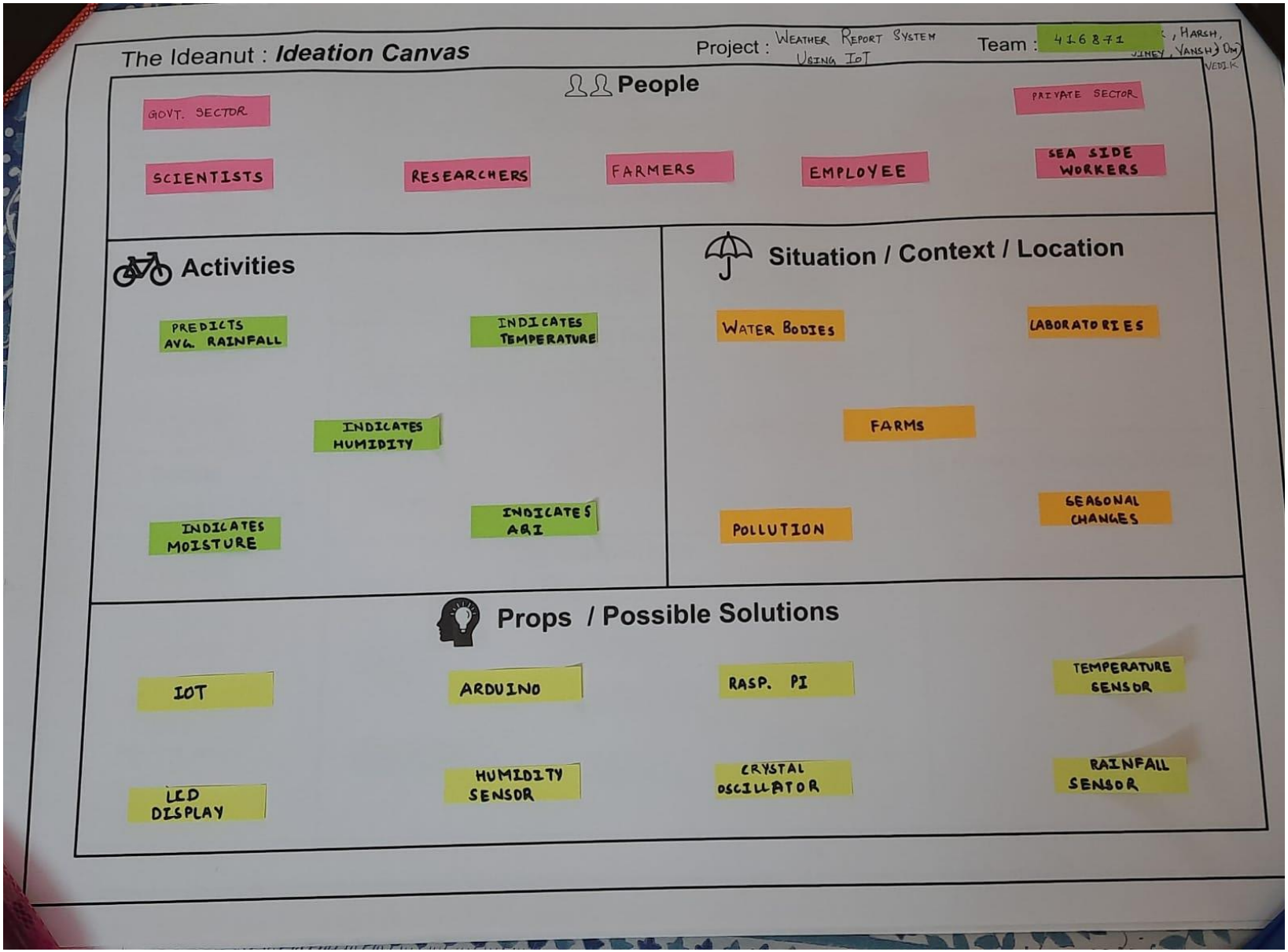
-Rainfall Sensor

-Temperature Sensor

-Arduino UNO

-Raspberry Pi

-Humidity Sensor


The Ideanut : Ideation Canvas
Project : Weather Report System Using IoT
Team : 416871
People
GOVT. SECTOR
PRIVATE SECTOR
SCIENTISTS
RESEARCHERS
FARMERS
EMPLOYEE
SEA SIDE WORKERS
Activities
PREDICTS AVG. RAINFALL
INDICATES TEMPERATURE
INDICATES HUMIDITY
INDICATES MOISTURE
INDICATES AQI
Situation / Context / Location
WATER BODIES
LABORATORIES
FARMS
POLLUTION
SEASONAL CHANGES
Props / Possible Solutions
IOT
ARDUINO
RASP. PI
TEMPERATURE SENSOR
LCD DISPLAY
HUMIDITY SENSOR
CRYSTAL OSCILLATOR
RAINFALL SENSOR

PRODUCT DEVELOP. CANVAS

Product Development Canvas Summary points and a snapshot of the canvas is given below:-

PURPOSE

-Provides AQI

-Provides Humidity Condition

-Provides Temperature Condition

PRODUCT EXPERIENCE

-Alerts the user about the extreme weather condition

-Easy to use

-User Friendly Interface

-Compact and Smooth Functioning of the Model

PRODUCT FUNCTIONS

-Makes People Aware about the weather Conditions

-Multi-tasking

PRODUCT FEATURES

-Portable

-Compact in Size

-Affordable for General Audience

-Low Maintenance

COMPONENTS

-Raspberry Pi

-Arduino UNO

-Transistors

-Sensors

-LCD Display

-Cables and Connections

CUSTOMER REVALIDATION

-Smooth Transitions

-Almost Accurate Readings

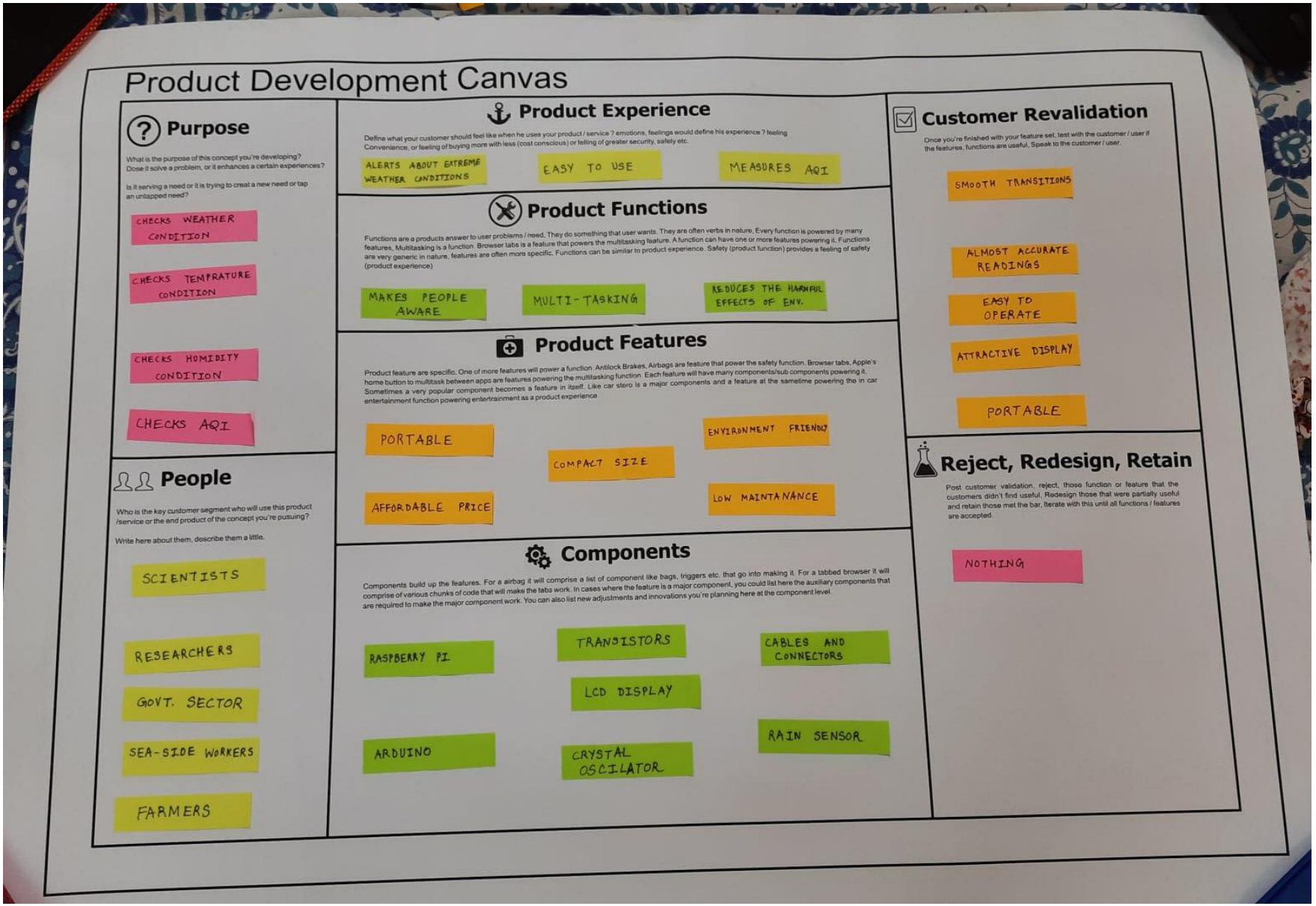
-Easy To Operate

-Attractive Display

-Easy To Carry

REJECT,REDISGN,RETAIN

-Not having any feedback in this case since we are yet unable to make this model on a larger scale.


Product Development Canvas
Purpose
What is the purpose of this concept you're developing? Dose it solve a problem, or it enhances a certain experiences?
Is it serving a need or it is trying to creat a new need or tap an untapped need?
CHECKS WEATHER CONDITION
CHECKS TEMPRATURE CONDITION
CHECKS HUMIDITY CONDITION
CHECKS AQI
People
Who is the key customer segment who will use this product /service or the end product of the concept you're pusuing?
Write here about them, describe them a little.
SCIENTISTS
RESEARCHERS
GOVT. SECTOR
SEA-SIDE WORKERS
FARMERS
Product Experience
Define what your customer should feel like when he uses your product / service ? emotions, feelings would define his experience ? feeling Convenience, or feeling of buying more with less (cost conscious) or felling of greater security, safety etc.
ALERTS ABOUT EXTREME WEATHER CONDITIONS
EASY TO USE
MEASURES AQI
Product Functions
Functions are a products answer to user problems / need. They do something that user wants. They are often verbs in nature. Every function is powered by many features. Multitasking is a function. Browser tabs is a feature that powers the multitasking feature. A function can have one or more features powering it. Functions are very generic in nature, features are often more specific. Functions can be similar to product experience. Safety (product function) provides a feeling of safety (product experience)
MAKES PEOPLE AWARE
MULTI-TASKING
REDUCES THE HARMFUL EFFECTS OF ENV.
Product Features
Product feature are specific. One of more features will power a function. Antilock Brakes, Airbags are feature that power the safety function. Browser tabs, Apple's home button to multitask between apps are features powering the multitasking function. Each feature will have many components/sub components powering it. Sometimes a very popular component becomes a feature in itself. Like car stero is a major components and a feature at the sametime powering the in car entertainment function powering entertrainment as a product experience
PORTABLE
COMPACT SIZE
ENVIRONMENT FRIENDLY
AFFORDABLE PRICE
LOW MAINTANANCE
Components
Components build up the features. For a airbag it will comprise a list of component like bags, triggers etc. that go into making it. For a tabbed browser it will comprise of various chunks of code that will make the tabs work. In cases where the feature is a major component, you could list here the auxiliary components that are required to make the major component work. You can also list new adjustments and innovations you're planning here at the component level.
RASPBERRY PI
TRANSISTORS
CABLES AND CONNECTORS
LCD DISPLAY
RAIN SENSOR
ARDUINO
CRYSTAL OSCILATOR
Customer Revalidation
Once you're finished with your feature set, test with the customer / user if the features, functions are useful. Speak to the customer / user.
SMOOTH TRANSITIONS
ALMOST ACCURATE READINGS
EASY TO OPERATE
ATTRACTIVE DISPLAY
PORTABLE
Reject, Redesign, Retain
Post customer validation, reject, those function or feature that the customers didn't find useful, Redesign those that were partially useful and retain those met the bar, Iterate with this until all functions / features are accepted.
NOTHING

EMPATHY CANVAS

USERS

It refers to the people who are involved in using this model or are indirectly related to the model.

STAKEHOLDERS

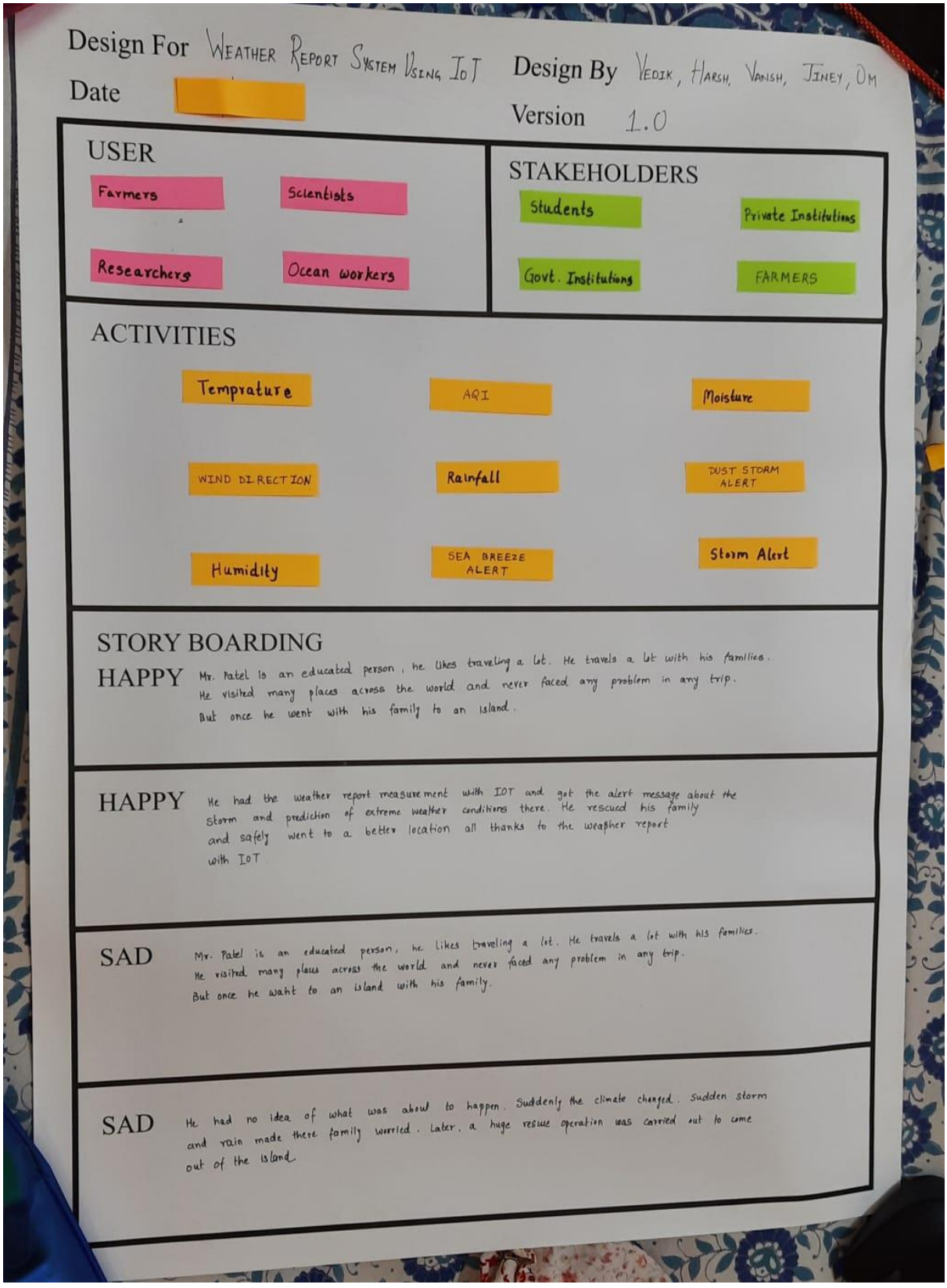
It refers to the people who are interested in this model in investing the time or funds or hold the ownership of the model.

ACTIVITIES

Activities indirectly or directly related to the stakeholders or the model.

STORY BOARDING

It refers to the happy and sad stories related to our model/project which shows the problem tolerated by the users/stakeholders.



Design For Weather Report System Using IoT

Design By Vedik, Harsh, Vanish, Jiney, Om

Date

Version 1.0

USER

- Farmers
- Scientists
- Researchers
- Ocean workers

STAKEHOLDERS

- Students
- Private Institutions
- Govt. Institutions
- FARMERS

ACTIVITIES

- Temprature
- AQI
- Moisture
- WIND DIRECTION
- Rainfall
- DUST STORM ALERT
- Humidity
- SEA BREEZE ALERT
- Storm Alert

STORY BOARDING

HAPPY Mr. Patel is an educated person, he likes traveling a lot. He travels a lot with his families. He visited many places across the world and never faced any problem in any trip. But once he went with his family to an island.

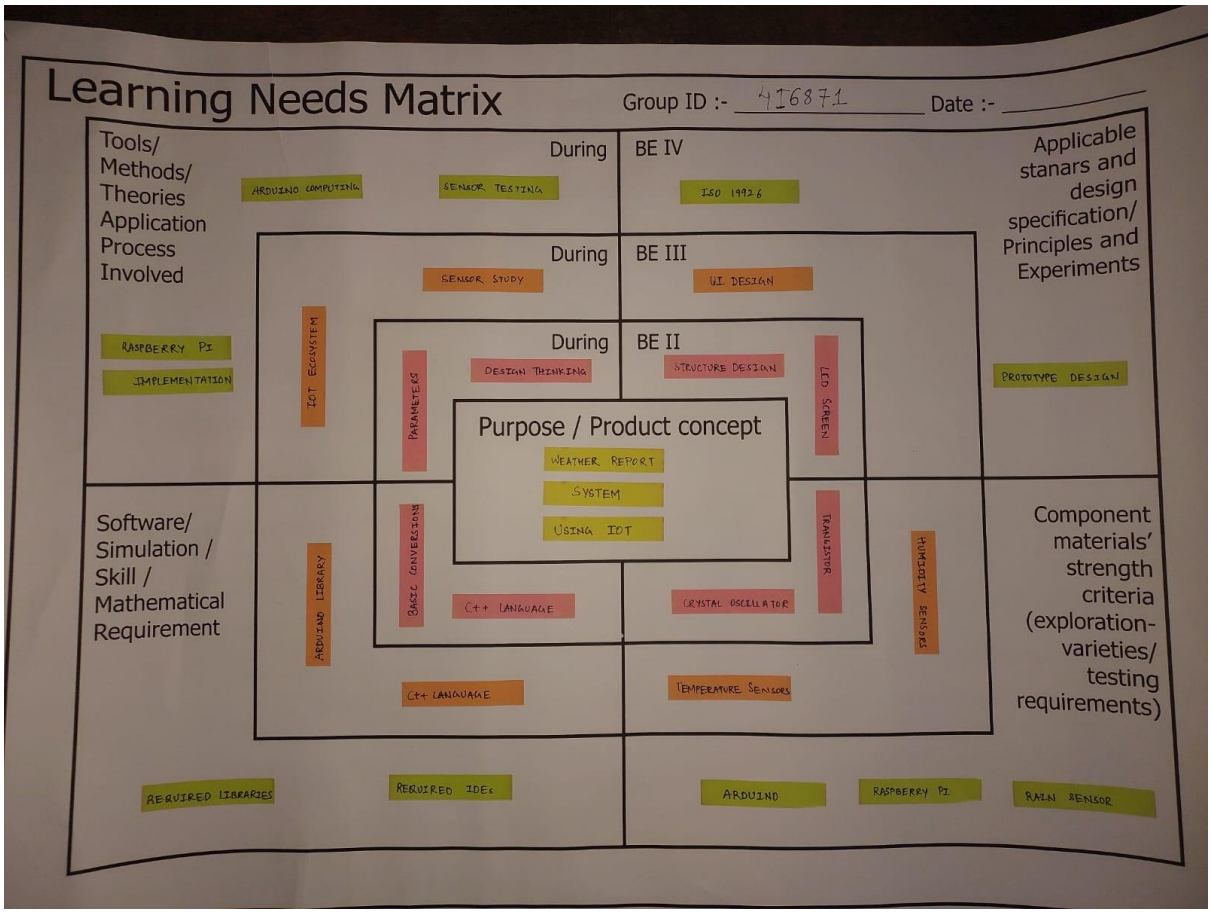
HAPPY He had the weather report measurement with IOT and got the alert message about the storm and prediction of extreme weather conditions there. He rescued his family and safely went to a better location all thanks to the weather report with IOT.

SAD Mr. Patel is an educated person, he likes traveling a lot. He travels a lot with his families. He visited many places across the world and never faced any problem in any trip. But once he waht to an island with his family.

SAD He had no idea of what was about to happen. Suddenly the climate changed. Sudden storm and rain made there family worried. Later, a huge rescue operation was carried out to come out of the island.

L.N.M CANVAS

In this section , we summarised and tracked our progress and needs in our semesters of B.E in making this Weather Report system using IoT project . For example, Theories involved, Design Specifications , Components , materials , Software and Skill Requirements in the respect semesters



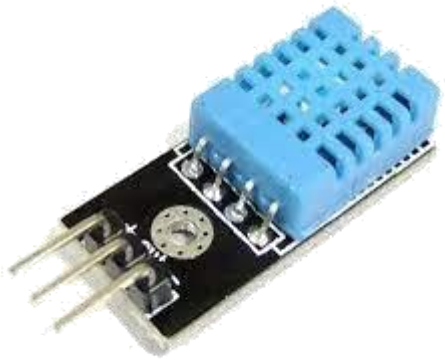
REQUIREMENT SPECIFICATION

Some of the important components along with a picture is shown below.

1. Arduino Esp32

2. LCD Display

4. Temperature Sensor

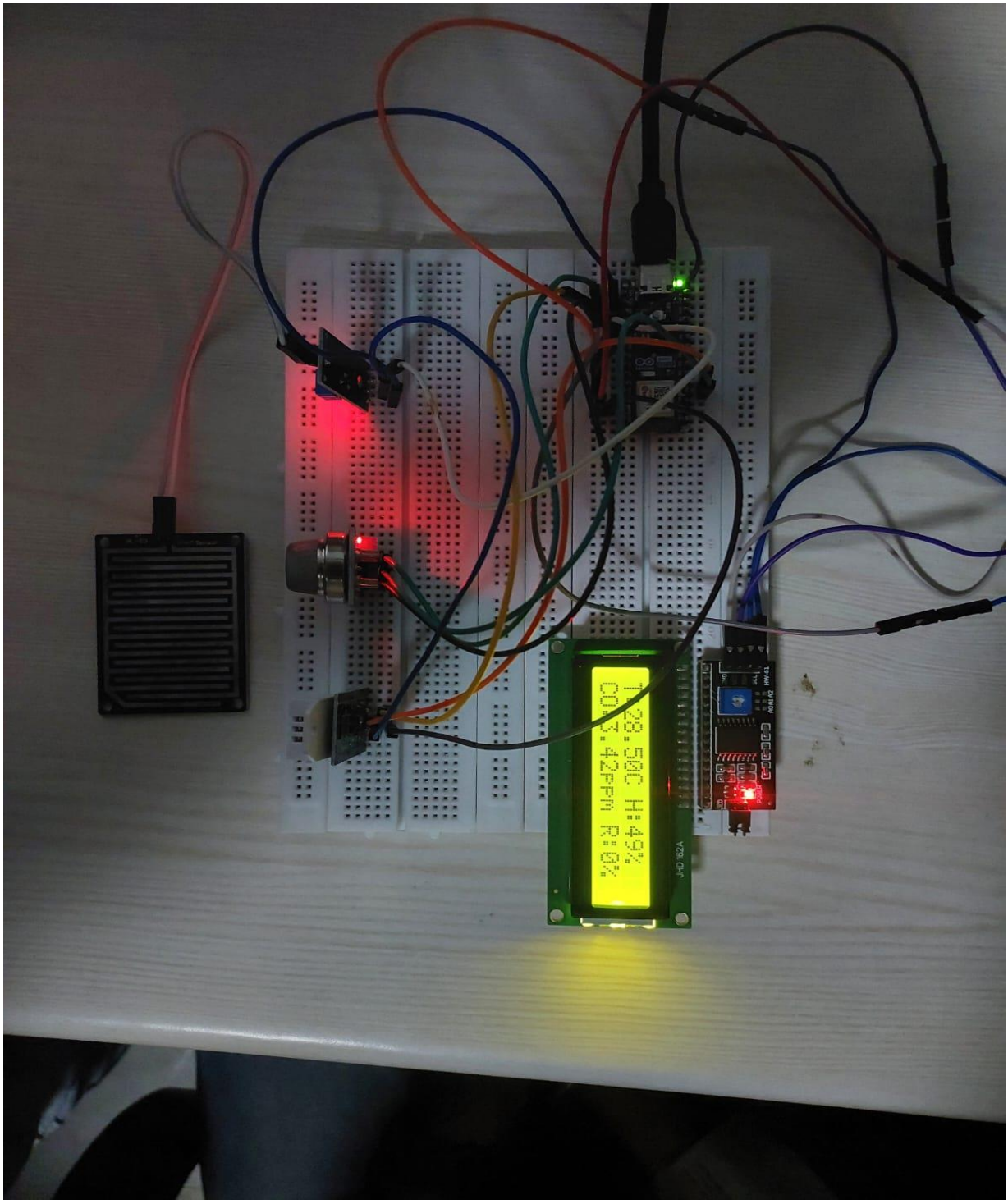


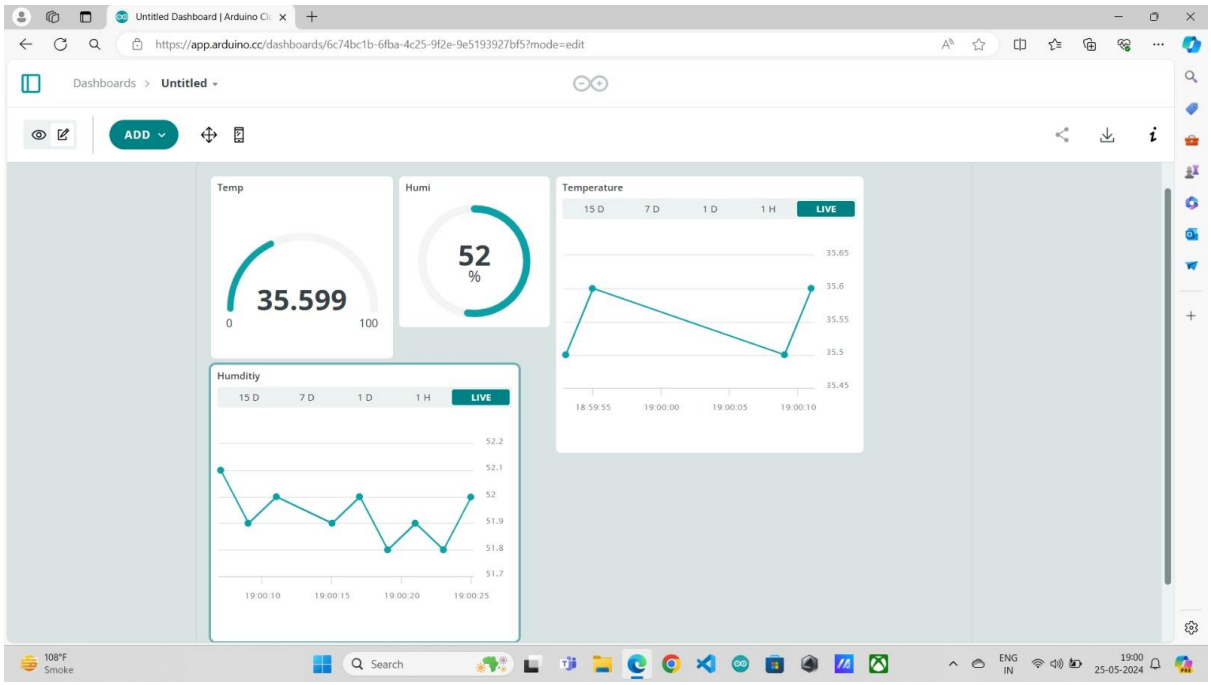
5. Humidity Sensor

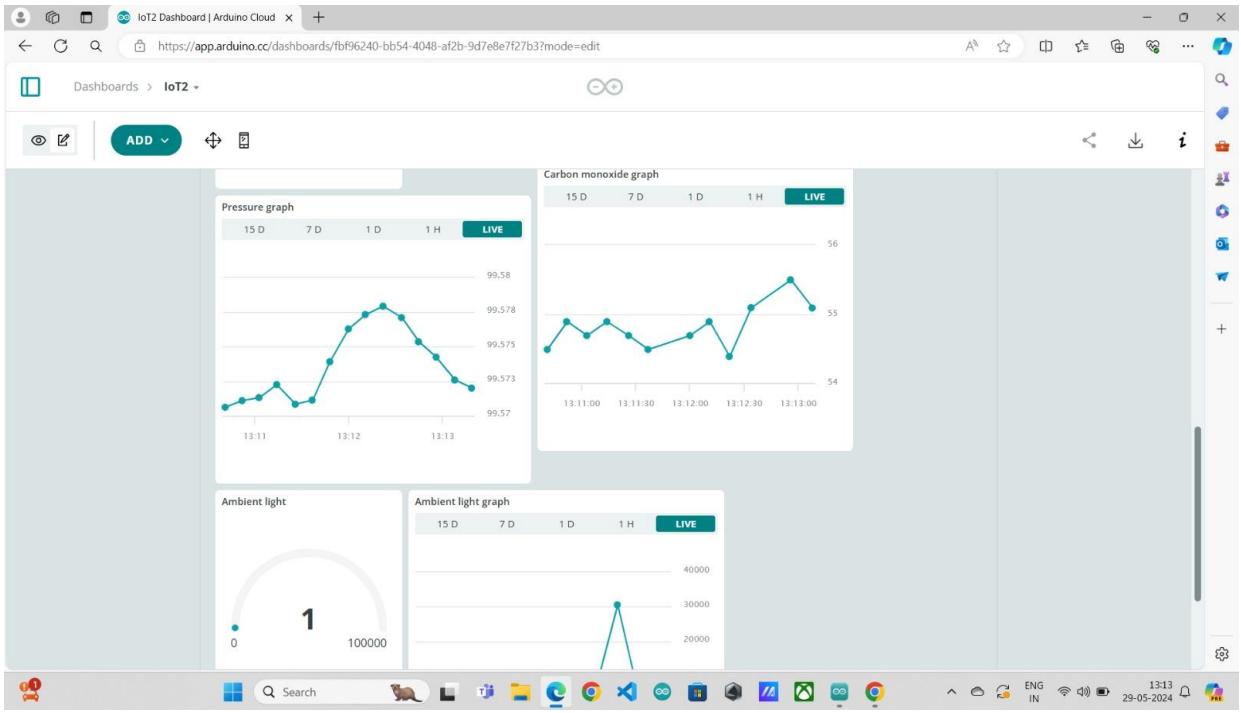
6. Rain Sensor

PROTOTYPE

Prototype of our main model is shown below:-




Untitled Dashboard | Arduino Cl
https://app.arduino.cc/dashboards/6c74bc1b-6fba-4c25-9f2e-9e5193927bf5?mode=edit
Dashboards > Untitled
ADD
Temp
35.599
0
100
Humi
52
%
Temperature
15 D
7 D
1 D
1 H
LIVE
35.65
35.6
35.55
35.5
35.45
18:59:55
19:00:00
19:00:05
19:00:10
Humditiy
52.2
52.1
52
51.9
51.8
51.7
19:00:10
19:00:15
19:00:20
19:00:25
108°F
Smoke
Search
ENG
IN
19:00
25-05-2024


IoT2 Dashboard | Arduino Cloud
https://app.arduino.cc/dashboards/fbf96240-bb54-4048-af2b-9d7e8e7f27b3?mode=edit
Dashboards > IoT2
ADD
Carbon monoxide graph
15 D
7 D
1 D
1 H
LIVE
56
55
54
13:11:00
13:11:30
13:12:00
13:12:30
13:13:00
Pressure graph
99.58
99.578
99.575
99.573
99.57
13:11
13:12
13:13
Ambient light
1
0
100000
Ambient light graph
40000
30000
20000
Search
ENG
IN
13:13
29-05-2024

CODES

1. Arduino

```
#include <Wire.h>
// Include the Wire library for I2C communication

#include <LiquidCrystal_I2C.h>
// Include the LiquidCrystal_I2C library for LCD display

#include <DHT.h>
// Include the DHT library for DHT22 sensor

#include <LTR390.h>
// Include the LTR390 library for the light and UV sensor

#include <Adafruit_BMP280.h>
// Include the Adafruit BMP280 library for pressure and
altitude sensor

#define DHTPIN 2
// Digital pin connected to the DHT sensor

#define DHTTYPE DHT22      // DHT sensor type
DHT dht(DHTPIN, DHTTYPE);  // Initialize DHT sensor

const int MQ9_PIN = A0;
// Analog pin connected to the MQ-9 sensor

const int RAIN_PIN = A1;
// Analog pin connected to the Rain Drop sensor
```

```
const int RAIN_THRESHOLD = 700;
// Define the rain threshold value

const int RAIN_MAX_VALUE = 4095;
// Maximum value of the rain sensor

#define I2C_LCD_ADDRESS 0x27
// I2C address for the LCD

#define I2C_LTR390_ADDRESS 0x53
// I2C address for the LTR390 sensor

#define TCAADDR 0x70
// TCA9548A I2C address

LiquidCrystal_I2C lcd(I2C_LCD_ADDRESS, 16, 2);
// Set the LCD address, columns, and rows

LTR390 ltr390(I2C_LTR390_ADDRESS);
// Initialize the LTR390 sensor

Adafruit_BMP280 bmp280;
// Initialize the BMP280 sensor

void tcaSelect(uint8_t i) {
  if (i > 7) return;
  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}

float calculateDewPoint(float temperature, float humidity) {
```

```
  // Constants for the Magnus formula
  const float a = 17.27;
  const float b = 237.7;

  // Magnus formula
  float alpha = ((a * temperature) / (b + temperature)) +
log(humidity / 100.0);
  float dewPoint = (b * alpha) / (a - alpha);
  return dewPoint;
}

void setup() {
  Serial.begin(9600);  // Start serial communication
  Wire.begin();        // Initialize I2C communication

  // Start sensor initializations
  dht.begin();  // Start DHT sensor

  // Select the channel for LCD
  tcaSelect(1);     // Assume LCD is on channel 1
  lcd.init();       // Initialize LCD
  lcd.backlight();  // Turn on backlight

  // Select the channel for LTR390
  tcaSelect(0);  // Assume LTR390 is on channel 0
  if (!ltr390.init()) {
    Serial.println("LTR390 not connected!");
  }
  ltr390.setMode(LTR390_MODE_ALS);
  ltr390.setGain(LTR390_GAIN_3);
  ltr390.setResolution(LTR390_RESOLUTION_18BIT);

  // Select the channel for BMP280
```

```
  tcaSelect(2);  // Assume BMP280 is on channel 2
  if (!bmp280.begin(0x76)) {
    Serial.println("BMP280 not connected!");
  }

bmp280.setSampling(Adafruit_BMP280::MODE_NORMAL,
/* Operating Mode. */
                   Adafruit_BMP280::SAMPLING_X2,     /*
Temp. oversampling */
                   Adafruit_BMP280::SAMPLING_X16,    /*
Pressure oversampling */
                   Adafruit_BMP280::FILTER_X16,      /*
Filtering. */
                   Adafruit_BMP280::STANDBY_MS_500); /*
Standby time. */
  // End sensor initializations
}

void loop() {
  delay(1000);  // Delay between readings

  // Read data from analog and digital pins
  float temperature = dht.readTemperature();  // Read
temperature
  int humidity = dht.readHumidity();          // Read humidity

  float mq9Value = analogRead(MQ9_PIN);         // Read
value from MQ-9 sensor
  float voltage = mq9Value * (3.3 / 4095.0);    // Convert
analog value to voltage
  float mq9_ppm = (voltage - 0.1) * 100 / 0.8;  // Convert
voltage to ppm (Parts Per Million)
```

```
  int rainValue = analogRead(RAIN_PIN);                    //
Read value from Rain Drop sensor
  int rainPercentage = map(rainValue, 0,
RAIN_MAX_VALUE, 100, 0);  // Map the rain value to a
percentage

  if (isnan(temperature) || isnan(humidity)) {  // Check if any
DHT reading failed
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  float dewPoint = calculateDewPoint(temperature, humidity);

  // Read data from LTR390 using TCA9548A
  tcaSelect(0);  // Select channel 0 for LTR390
  float lux = 0;
  float uvi = 0;
  if (ltr390.newDataAvailable()) {
    if (ltr390.getMode() == LTR390_MODE_ALS) {
      lux = ltr390.getLux();
      ltr390.setGain(LTR390_GAIN_18);              //
Recommended for UVI - x18
      ltr390.setResolution(LTR390_RESOLUTION_20BIT);  //
Recommended for UVI - 20-bit
      ltr390.setMode(LTR390_MODE_UVS);
      delay(100);  // Small delay to allow mode switch
      uvi = ltr390.getUVI();
      ltr390.setGain(LTR390_GAIN_3);               //
Recommended for Lux - x3
      ltr390.setResolution(LTR390_RESOLUTION_18BIT);  //
Recommended for Lux - 18-bit
      ltr390.setMode(LTR390_MODE_ALS);
```

```
    }
  }

  // Read data from BMP280 using TCA9548A
  tcaSelect(2);  // Select channel 2 for BMP280
  float pressure = bmp280.readPressure() / 1000;  // Convert to
hPa
  float altitude = bmp280.readAltitude(1013.25);    // Calculate
altitude with a baseline pressure

  // Print data to Serial
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C\t");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  Serial.print("CO PPM: ");
  Serial.print(mq9_ppm);
  Serial.println(" ppm");

  Serial.print("Rain Drop Value: ");
  Serial.println(rainValue);
  Serial.println(" %");

  Serial.print("Ambient Light: ");
  Serial.print(lux);
  Serial.println(" Lux");

  Serial.print("UV Index: ");
  Serial.println(uvi);
```

```
  Serial.print("Pressure: ");
  Serial.print(pressure);
  Serial.println(" kPa");

  Serial.print("Altitude: ");
  Serial.print(altitude);
  Serial.println(" m");

  Serial.print("Dew: ");
  Serial.print(dewPoint);
  Serial.println(" °C");

  // Select the channel for LCD
  tcaSelect(1);  // Select channel 1 for LCD

  // Display data on LCD
  lcd.clear();  // Clear the LCD display

  lcd.setCursor(0, 0);  // Set cursor to the first column and first
row
  lcd.print("Temp:");
  lcd.print(temperature);
  lcd.print("C");

  lcd.setCursor(0, 1);  // Set cursor to the first column and
second row
  lcd.print("Humi:");
  lcd.print(humidity);
  lcd.print("%");

  delay(2000);

  lcd.clear();
```

```
  lcd.setCursor(0, 0);  // Set cursor to the first column and first row
  lcd.print("CO:");
  lcd.print(mq9_ppm);
  lcd.print("ppm");

  lcd.setCursor(0, 1);  // Set cursor to the first column and second row
  lcd.print("Rain:");
  lcd.print(rainPercentage);
  lcd.print("%");

  delay(2000);

  lcd.clear();
  lcd.setCursor(0, 0);  // Set cursor to the first column and first row
  lcd.print("Lux:");
  lcd.print(lux);
  lcd.print("lux");

  lcd.setCursor(0, 1);  // Set cursor to the first column and second row
  lcd.print("UV Index:");
  lcd.print(uvi);

  delay(2000);

  lcd.clear();
  lcd.setCursor(0, 0);  // Set cursor to the first column and first row
  lcd.print("Press:");
  lcd.print(pressure);
```

```
  lcd.print("kPa");

  lcd.setCursor(0, 1);  // Set cursor to the first column and
second row
  lcd.print("Alt:");
  lcd.print(altitude);
  lcd.print("m");

  delay(2000);

  lcd.clear();
  lcd.setCursor(0, 0);  // Set cursor to the first column and first
row
  lcd.print("Dew:");
  lcd.print(dewPoint);
  lcd.print("C");

  delay(2000);
}
```

2. Arduino IoT Cloud

```
#include "thingProperties.h"#include <Wire.h>
//Include the Wire library for I2C communication

#include <LiquidCrystal_I2C.h>
// Include the LiquidCrystal_I2C library for LCD display

#include <DHT.h>
// Include the DHT library for DHT22 sensor

#include <LTR390.h>
// Include the LTR390 library for the light and UV sensor
```

```
#include <Adafruit_BMP280.h>
// Include the Adafruit BMP280 library for pressure and
altitude sensor

#define DHTPIN 2
// Digital pin connected to the DHT sensor

#define DHTTYPE DHT22
// DHT sensor type

DHT dht(DHTPIN, DHTTYPE);
// Initialize DHT sensor

const int MQ9_PIN = A0;
// Analog pin connected to the MQ-9 sensor

const int RAIN_PIN = A1;
// Analog pin connected to the Rain Drop sensor

const int RAIN_THRESHOLD = 700;
// Define the rain threshold value

const int RAIN_MAX_VALUE = 4095;
// Maximum value of the rain sensor

#define I2C_LCD_ADDRESS 0x27
// I2C address for the LCD

#define I2C_LTR390_ADDRESS 0x53
// I2C address for the LTR390 sensor

#define TCAADDR 0x70
```

```
// TCA9548A I2C address

LiquidCrystal_I2C lcd(I2C_LCD_ADDRESS, 16, 2);
// Set the LCD address, columns, and rows

LTR390 ltr390(I2C_LTR390_ADDRESS);
// Initialize the LTR390 sensor

Adafruit_BMP280 bmp280;
// Initialize the BMP280 sensor

void tcaSelect(uint8_t i) {
  if (i > 7) return;
  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}

float calculateDewPoint(float temperature, float humidity) {
  // Constants for the Magnus formula
  const float a = 17.27;
  const float b = 237.7;

  // Magnus formula
  float alpha = ((a * temperature) / (b + temperature)) +
log(humidity / 100.0);
  float dewPoint = (b * alpha) / (a - alpha);
  return dewPoint;
}
void setup() {
  // Initialize serial and wait for port to open:

 Serial.begin(9600);
```

```
  // This delay gives the chance to wait for a Serial Monitor
without blocking if none is found
  delay(1500);

  Wire.begin();        // Initialize I2C communication

  // Start sensor initializations
  dht.begin();  // Start DHT sensor

  // Select the channel for LCD
  tcaSelect(1);     // Assume LCD is on channel 1
  lcd.init();       // Initialize LCD
  lcd.backlight();  // Turn on backlight

  // Select the channel for LTR390
  tcaSelect(0);  // Assume LTR390 is on channel 0
  if (!ltr390.init()) {
    Serial.println("LTR390 not connected!");
  }
  ltr390.setMode(LTR390_MODE_ALS);
  ltr390.setGain(LTR390_GAIN_3);
  ltr390.setResolution(LTR390_RESOLUTION_18BIT);

  // Select the channel for BMP280
  tcaSelect(2);  // Assume BMP280 is on channel 2
  if (!bmp280.begin(0x76)) {
    Serial.println("BMP280 not connected!");
  }

bmp280.setSampling(Adafruit_BMP280::MODE_NORMAL,
/* Operating Mode. */
                   Adafruit_BMP280::SAMPLING_X2,     /*
Temp. oversampling */
```

```
                Adafruit_BMP280::SAMPLING_X16,    /*
Pressure oversampling */
                Adafruit_BMP280::FILTER_X16,      /*
Filtering. */
                Adafruit_BMP280::STANDBY_MS_500); /*
Standby time. */
  // End sensor initializations

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more
information
     related to the state of network and IoT Cloud connection
and errors
     the higher number the more granular information you'll
get.
     The default is 0 (only errors).
     Maximum is 4
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();

    delay(1000);  // Delay between readings
```

```
  // Read data from analog and digital pins
  float temperature = dht.readTemperature();  // Read
temperature
  int humidity = dht.readHumidity();          // Read humidity

  float mq9Value = analogRead(MQ9_PIN);         // Read
value from MQ-9 sensor
  float voltage = mq9Value * (3.3 / 4095.0);    // Convert
analog value to voltage
  float mq9_ppm = (voltage - 0.1) * 100 / 0.8;  // Convert
voltage to ppm (Parts Per Million)

  int rainValue = analogRead(RAIN_PIN);                    //
Read value from Rain Drop sensor
  int rainPercentage = map(rainValue, 0,
RAIN_MAX_VALUE, 100, 0);  // Map the rain value to a
percentage

// Check if any DHT reading failed
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  float dewPoint = calculateDewPoint(temperature, humidity);

  // Read data from LTR390 using TCA9548A
  tcaSelect(0);  // Select channel 0 for LTR390
  float lux = 0;
  float uvi = 0;
  if (ltr390.newDataAvailable()) {
    if (ltr390.getMode() == LTR390_MODE_ALS) {
      lux = ltr390.getLux();
```

```
      ltr390.setGain(LTR390_GAIN_18);
// Recommended for UVI - x18
      ltr390.setResolution(LTR390_RESOLUTION_20BIT);
// Recommended for UVI - 20-bit
      ltr390.setMode(LTR390_MODE_UVS);
      delay(100);  // Small delay to allow mode switch
      uvi = ltr390.getUVI();
      ltr390.setGain(LTR390_GAIN_3);
// Recommended for Lux - x3
      ltr390.setResolution(LTR390_RESOLUTION_18BIT);
// Recommended for Lux - 18-bit
      ltr390.setMode(LTR390_MODE_ALS);
    }
  }

  // Read data from BMP280 using TCA9548A
  tcaSelect(2);
// Select channel 2 for BMP280
  float pressure = bmp280.readPressure() / 1000;
// Convert to hPa
  float altitude = bmp280.readAltitude(1013.25);
 // Calculate altitude with a baseline pressure

  // Print data to Serial
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" °C\t");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  Serial.print("CO PPM: ");
  Serial.print(mq9_ppm);
```

```
Serial.println(" ppm");

Serial.print("Rain Drop Value: ");
Serial.println(rainValue);

Serial.print("Ambient Light: ");
Serial.print(lux);
Serial.println(" Lux");

Serial.print("UV Index: ");
Serial.println(uvi);

Serial.print("Pressure: ");
Serial.print(pressure);
Serial.println(" kPa");

Serial.print("Altitude: ");
Serial.print(altitude);
Serial.println(" m");

Serial.print("Dew: ");
Serial.print(dewPoint);
Serial.println(" °C");
Serial.println("");

ambient_light = lux;
co = mq9_ppm;
humi = humidity;
press = pressure;
temp = temperature;

// Select the channel for LCD
tcaSelect(1);  // Select channel 1 for LCD
```

```
  // Display data on LCD
  lcd.clear();  // Clear the LCD display

  lcd.setCursor(0, 0);  // Set cursor to the first column and first
row
  lcd.print("Temp:");
  lcd.print(temperature);
  lcd.print("C");

  lcd.setCursor(0, 1);
// Set cursor to the first column and second row
  lcd.print("Humi:");
  lcd.print(humidity);
  lcd.print("%");

  delay(2000);

  lcd.clear();
  lcd.setCursor(0, 0);
// Set cursor to the first column and first row
  lcd.print("CO:");
  lcd.print(mq9_ppm);
  lcd.print("ppm");

  lcd.setCursor(0, 1);
// Set cursor to the first column and second row
  lcd.print("Rain:");
  lcd.print(rainPercentage);
  lcd.print("%");

  delay(2000);
```

```
  lcd.clear();
  lcd.setCursor(0, 0);
// Set cursor to the first column and first row
  lcd.print("Lux:");
  lcd.print(lux);
  lcd.print("lux");

  lcd.setCursor(0, 1);
 // Set cursor to the first column and second row
  lcd.print("UV Index:");
  lcd.print(uvi);

  delay(2000);

  lcd.clear();
  lcd.setCursor(0, 0);
// Set cursor to the first column and first row
  lcd.print("Press:");
  lcd.print(pressure);
  lcd.print("kPa");

  lcd.setCursor(0, 1);
// Set cursor to the first column and second row
  lcd.print("Alt:");
  lcd.print(altitude);
  lcd.print("m");

  delay(2000);

  lcd.clear();
  lcd.setCursor(0, 0);
// Set cursor to the first column and first row
  lcd.print("Dew:");
```

```
  lcd.print(dewPoint);
  lcd.print("C");

  delay(2000);

}

/*
  Since Temp is READ_WRITE variable, onTempChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onTempChange()  {
  // Add your code here to act upon Temp change
}

/*
  Since Humi is READ_WRITE variable, onHumiChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onHumiChange()  {
  // Add your code here to act upon Humi change
}

/*
  Since Co is READ_WRITE variable, onCoChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onCoChange()  {
  // Add your code here to act upon Co change
}

/*
```

```
  Since AmbientLight is READ_WRITE variable,
onAmbientLightChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onAmbientLightChange()  {
  // Add your code here to act upon AmbientLight change
}

/*
  Since Pressure is READ_WRITE variable,
onPressureChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onPressureChange()  {
  // Add your code here to act upon Pressure change
}


/*
  Since Press is READ_WRITE variable, onPressChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onPressChange()  {
  // Add your code here to act upon Press change
}
```

CONCLUSION

We are able to understand the need of easy-to-use and user-friendly interface to display the weather conditions outside our mobile phones and create awareness regarding our environment with its advantages mentioned below.

1. Compact in size and portable so user doesn't have to stick to one place to access the weather report.
2. Data can be transferred to the internet for display purposes via IoT.
3. Instant Data Capturing from the sensors.
4. User-Friendly Interface for the ease of the users instead of a tech-heavy interface.

FUTURE SCOPE

Our project which aims to provide the weather reports to the user quickly without compromising the accuracy and efficiency of the model.

It will help in various sectors of a country such as :-

- Private Sectors, For Example :- Laboratories, Research Facilities etc.
- Government Sectors, For Example:- Government Offices, Disaster relief departmental offices etc.
- Public Sectors, For Example:- Houses, Flats etc.

Some other users:-

- Sea Side Workers
- Fisherman
- Sailors

REFERENCES

1. https://nevonprojects.com/iot-weather-reporting-system-using-adruino-and-ras-pi/
2. https://www.techtarget.com/iotagenda/definition/Internet -of-Things-IoT
3. https://www.raspberrypi.com/documentation/