**Nalaiya Thiran**

Batch No: B9-3A5E

ST.XAVIERS CATHOLIC COLLEGE OF ENGINEERING KANYAKUMARI

Department of Information Technology

**Smart Waste Management System For**

**Metropolitan Cities**

**Team ID: PNT2022TMID34757**

**Team Members:**

**962219205002 :** AGISHMA SINI C

**962219205031 :** PRADHISHA P

**962219205030 :** NIVETHA SAROJINI R

**962219205046 :** TEENA S

**Project Guide**:

Industry mentor: Mr. Dinesh

Faculty Mentor: Ms. N. Ansgar Mary

**ABSTRACT**

One issue that most cities and municipalities are dealing with currently, is the degradation of environmental cleanliness with reference to waste management. This is a result of improper garbage collection management. Dumping garbage onto the streets and in public areas is a common synopsis found in all developing countries and this mainly ends up affecting the environment and creating several unhygienic conditions. To avoid improper garbage management and to create a hygienic environment, the concept of automation is used in waste management system. Any city being referred to as a "smart city" is because of its orderly and tidy surroundings. But currently, many issues including those related to smart grids, smart environments, and smart living are faced. Today, cities and metropolitan areas' top priority is proper garbage management.

Traditional waste management techniques are too simplistic to create an effective and reliable waste management. The ideology put forward includes hardware and software technologies i.e. connecting Wi-Fi system to the normal dustbin in order to provide free internet facilities to the user for a particular period of time. The technology awards the user for keeping the surrounding clean and thus work hand in hand for the proper waste management in a locality. The smart bin uses multiple technologies - firstly the technology for measuring the amount of trash dumped and secondly the movement of the waste and lastly sending necessary signals and connecting the user to the WiFi system. The proposed system will function on client server model, a cause that will assure clean environment, good health, and pollution free society.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Title** | **Page No.** |
| 1 | **Introduction**   * 1. Project Overview   2. IOT in our day-to-day life   3. Purpose | 5 |
| 2 | **Literature Survey**  2.1 Existing Problem  2.2 References  2.3 Problem Statement Definition | 7 |
| 3 | **Ideation and Proposed Solution**  3.1 Empathy Map Canvas  3.2 Ideation & Brainstorming  3.3 Proposed Solution | 11 |
| 4 | **Requirement Analysis**  4.1 Functional requirements  4.2 Non-Functional requirements | 16 |
| 5 | **Project Design**  5.1 Data Flow Diagrams  5.2 Solution & Technical Architecture | 18 |
| 6 | **Project Planning and Scheduling**  6.1 Sprint Planning & Estimation  6.2 Sprint Delivery Schedule | 20 |
| 7 | **Coding and Solutioning**  7.1 Feature 1  7.2 Feature 2  7.3 Feature 3 | 23 |
| 8 | **Testing**  8.1 Test Cases  8.2 User Acceptance Testing | 36 |
| 9 | **Results**  9.1 Performance Metrics | 37 |
| 10 | **Advantages and Disadvantages** | 40 |
| 11 | **Conclusion** | 41 |
| 12 | **Future Works** | 42 |
| 13 | **Appendix**  13.1 Source Code  13.2 Project Links | 43 |

**CHAPTER 1: INTRODUCTION**

* 1. **Project Overview**

Smart waste management is an innovative approach to handling and collecting waste. Based on IOT (Internet of Things) technology, smart waste management provides data on waste generation patterns and behaviour.

Our Smart waste management solution uses sensors placed in garbage bins to measure fill levels and notifies city collection services when bins are ready to be emptied. There are load and ultrasonic sensors placed to continuously monitor the bins. This data is sent to the cloud (via a microcontroller that is connected to Wi-Fi) where it is stored after which it is processed further. When the levels exceed a certain limit, a notification is sent to the garbage collector via a web application.

Over time, historical data collected by sensors can be used to identify fill patterns, optimize driver routes and schedules, and reduce operational costs. The cost of these sensors is steadily decreasing, making IOT waste bins more feasible to implement and more attractive.

**1.2. IOT IN OUR DAY-TO-DAY LIFE**

The term IOT itself refers to the collective network of connected devices and the technology that facilitates communication between devices and the cloud, as well as between the devices themselves. Thanks to the advent of inexpensive computer chips and high bandwidth telecommunication, we now have billions of devices connected to the internet. This means everyday devices like toothbrushes, vacuums, cars, and machines can use sensors to collect data and respond intelligently to users.  The Internet of Things integrates everyday “things” with the internet. Computer Engineers have been adding sensors and processors to everyday objects since the 90s. Internet of things has been considered a misnomer because devices do not need to be connected to the public internet, they only need to be connected to a network and be individually addressable.

**1.3. Purpose**

Around 2.1 billion tonnes of municipal solid waste is generated annually around the globe. Population growth and rapid urbanization lead to a huge increase in waste generation, so the traditional methods of waste collection have become inefficient and costly. This system cannot measure the fullness levels of containers, and as a result, half-full containers can be emptied, and in contrast, pre-filled ones need to wait until the next collection period comes. Moreover, since drivers collect empty bins, predefined collection routes of the system cause waste of time, an increase in fuel consumption, and excessive use of resources.

In today’s ever-technological world, an innovative and data-driven approach is the only way forward, the waste sector needs a solution that empowers event-driven waste collection. The most efficient way this extraordinary amount of waste can be solved is through smart waste management without obsolete methods of waste collection. This empowers municipalities, cities, and waste collectors to optimize their waste operations, become more sustainable, and make more intelligent business decisions.

**CHAPTER 2: LITERATURE SURVEY**

* 1. **Existing Problem**

Around 80% of waste collections happen at the wrong time. Late waste collections lead to overflowing bins, unsanitary environments, citizen complaints, illegal dumping, and increased cleaning and collection costs. Early waste collections mean unnecessary carbon emissions, more traffic congestion, and higher running costs. The old way of doing waste management is highly inefficient. And in today’s ever-technological world, an innovative and data-driven approach is the only way forward. Traditionally, municipalities and waste management companies would operate on a fixed collection route and schedule. This means that waste collection trucks would drive the same collection route and empty every single waste container – even if the waste container did not need emptying. This means high labour and fuel costs – which residents ultimately foot the bill for.

* 1. **References**

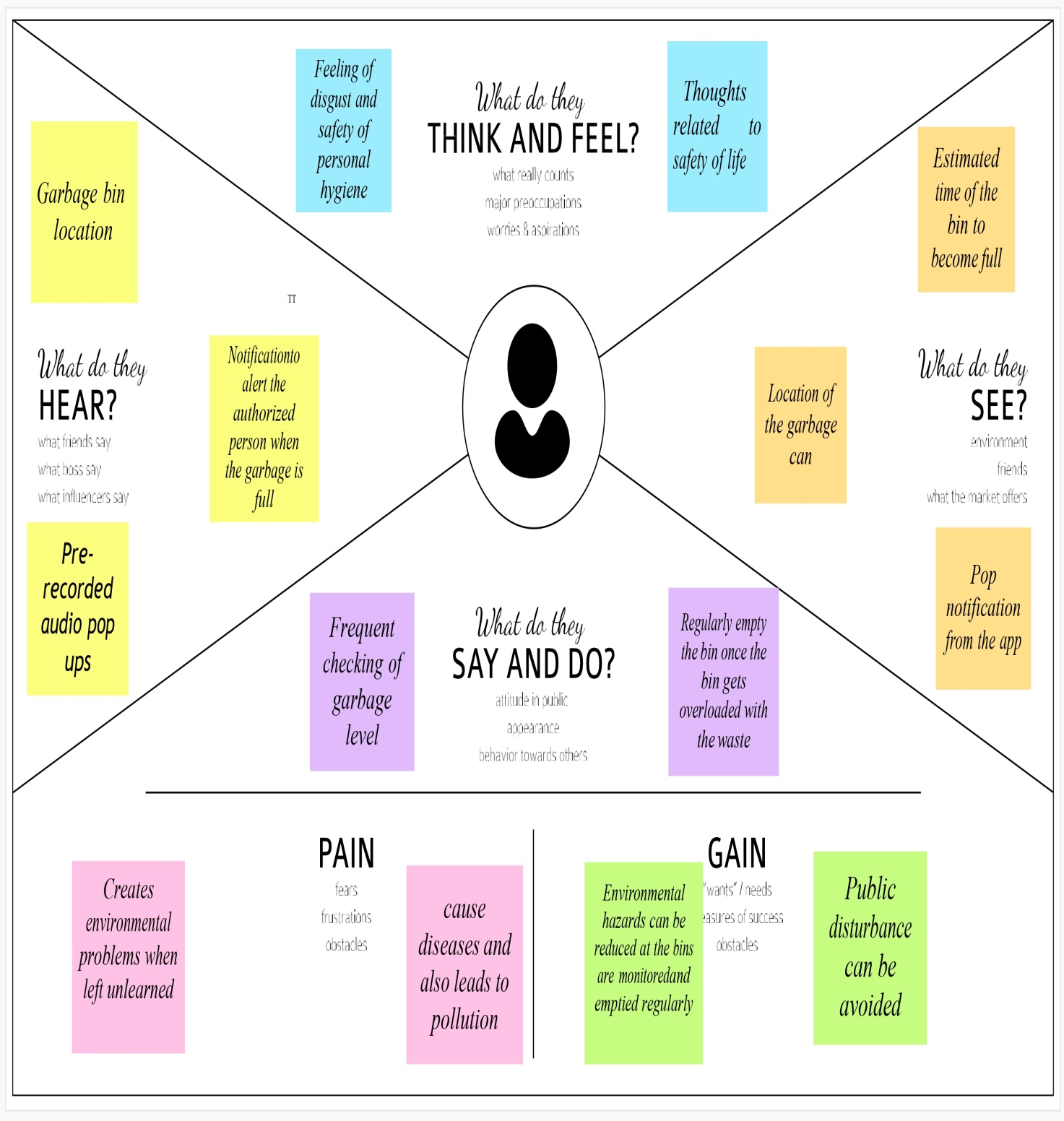
|  |  |  |
| --- | --- | --- |
| **Paper Title** | **Author** | **Outcome** |
| IoT-Enabled Solid Waste Management in Smart Cities | 1) S.Vishnu  2) S.R.Jino Ramson | This IoT based process is to check whether the dustbins all over the city are full or not. In case if the bins get filled, it detects overweight through a moisture sensor and then workers come to collect the garbage.The trash bins were filled with wastes at different levels and the corresponding unfilled levels of every trash bin were monitored through the intelligent GUI. It manages the waste in different types of boxes by using automation. In this system, it detects the dry and wet waste and separates them automatically using a Moisture sensor. |
| IoT Enabled Smart Waste Bin with Real Time Monitoring for efficient waste management in Metropolitan Cities | 1)Manju Mohan  2)Kuppan Chetty Ramanathan | In this paper, the capacitance sensor in the bin continuously monitors the level of the bin in real time and communicates to the central cloud where the bins are connected. Ultrasonic sensor is used to open and close the lid of the bin whenever the persons are nearby the bin. Such smart bins are connected to the cloud, where the bin  status is communicated, recorded  and  monitored  by  the  local  bodies  through  an  android  app  or  a centralized server. |
| Smart Waste Management System. | 1) Sanjiban Charkraborty | This Waste management is one of the serious challenges of the cities, the system now used in cities, we continue to use an old and outmoded paradigm that no longer serves the entail of municipalities, Still find over spilled waste containers giving off irritating smells causing serious health issues and atmosphere impairment. |
| Smart Solid Waste Management. | 1) Mohd Helmy Abd Wahab | At the time of trash disposal, the material to be recycled could be identified using RFID technology. |
| Smart Waste Management: Garbage Monitoring Using IoT | 1. Mrs Sarmila SS 2. Siva Kumar V | This is a Smart bin system that identifies hazardous gasses and fullness of bins. The system is designed to collect data and to deliver the data through a wireless mesh network. To collect data and to obtain bin utilization and bin daily information, with such information, wastage bin providers and cleaning contractors are able to make better decisions.  The ultrasonic sensor intimates the load of the garbage dump if it is full. If the dustbin is not cleaned in a specific time, then the record is sent to the higher authority who can take appropriate action against the concerned contractor. |
| Automation of smart waste management using IoT | 1. Shubhangi Ithape 2. Sandhya Lungase 3. Madhuri Mohare | ESP8266 WI-Fi module is used to update the status of dustbins on the mobile app. Moisture sensor is used to detect garbage is either dry or wet. Two DC motors are used; one is for moving the conveyor belt and second is for rotating the dustbin position to collect garbage in a separate dustbin. Relays are used for driving DC motors. Ultrasonic sensors are used to detect garbage level in dustbins, to determine the dustbin is full or empty. One is used to detect the garbage level of dry dustbin and second is to detect the garbage level of wet dustbin. |
| Smart Waste Management for Green Environment | 1. T. P. Fei | The system is based on Bootstrap platform. This system works on the waterfall methodology which has 4 crucial phases: planning and analysis, system design, system implementation and system testing. Using this system, operators can get the information regarding collection from trash bins. The limitations of this approach are that the resultant product has a short life and uniformity is lost after a certain period. |

* 1. **Problem Statement Definition**



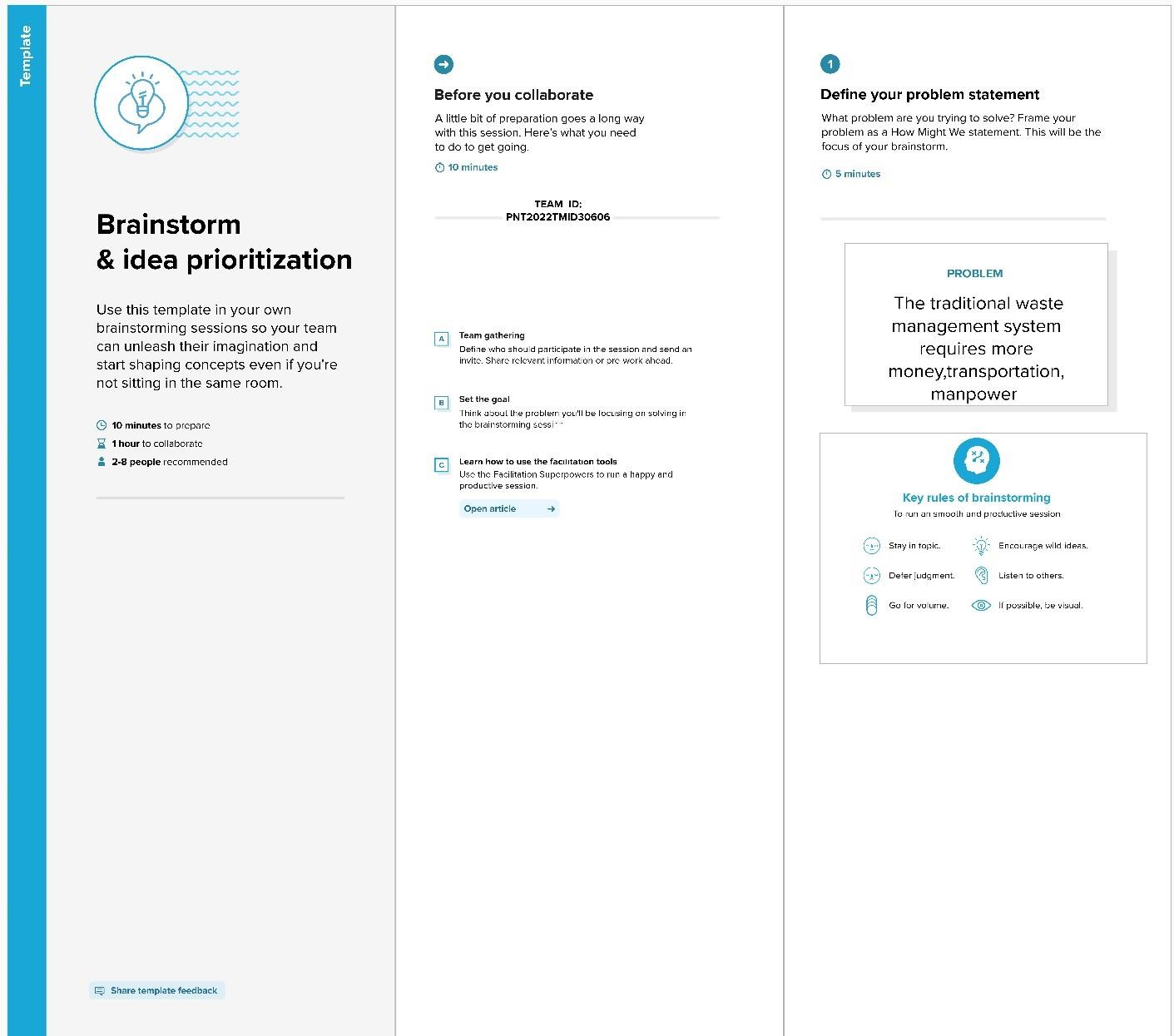
**CHAPTER 3: IDEATION & PROPOSED SOLUTION**

**3.1 Empathy Map Canvas**

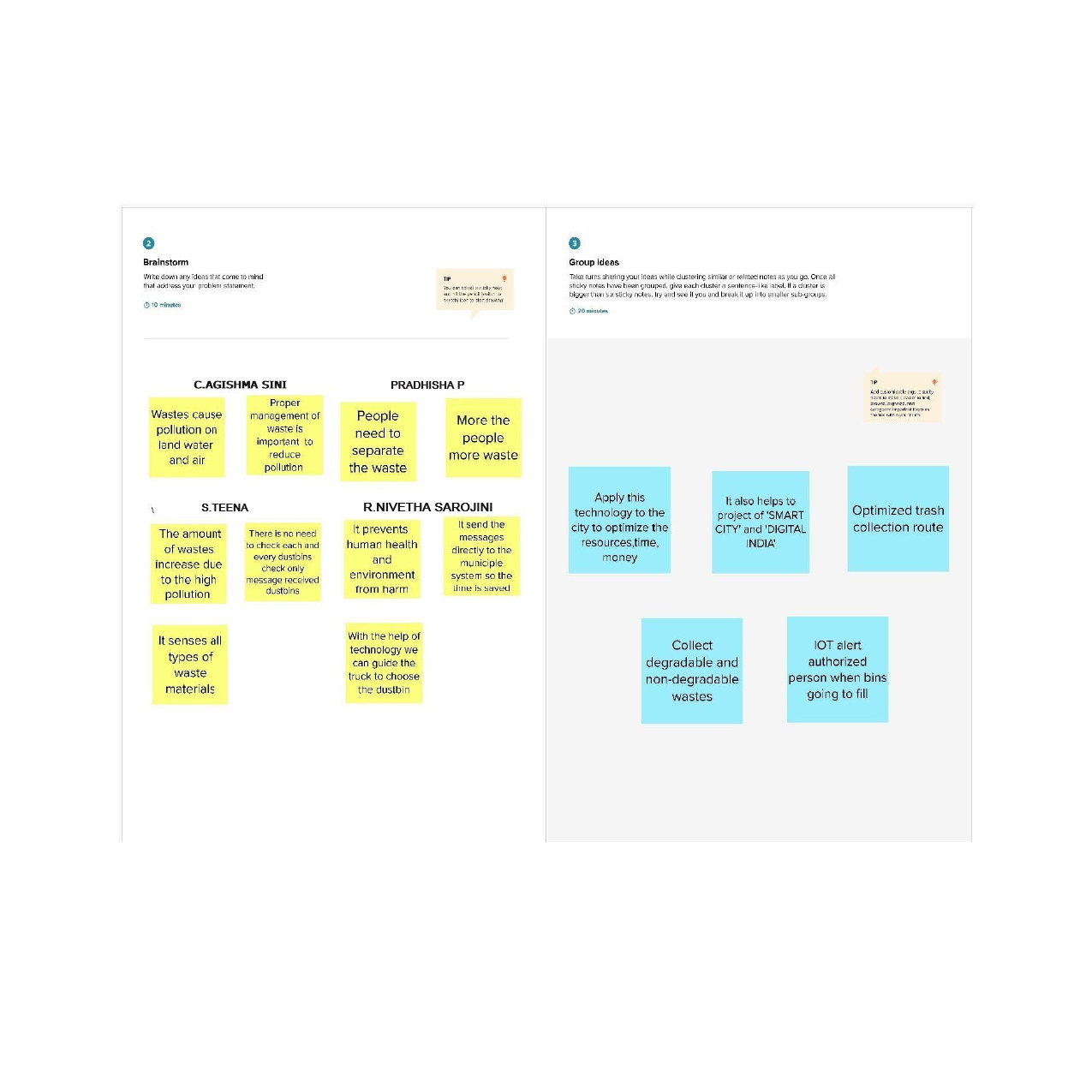
****

**3.2 Ideation & Brainstorming**

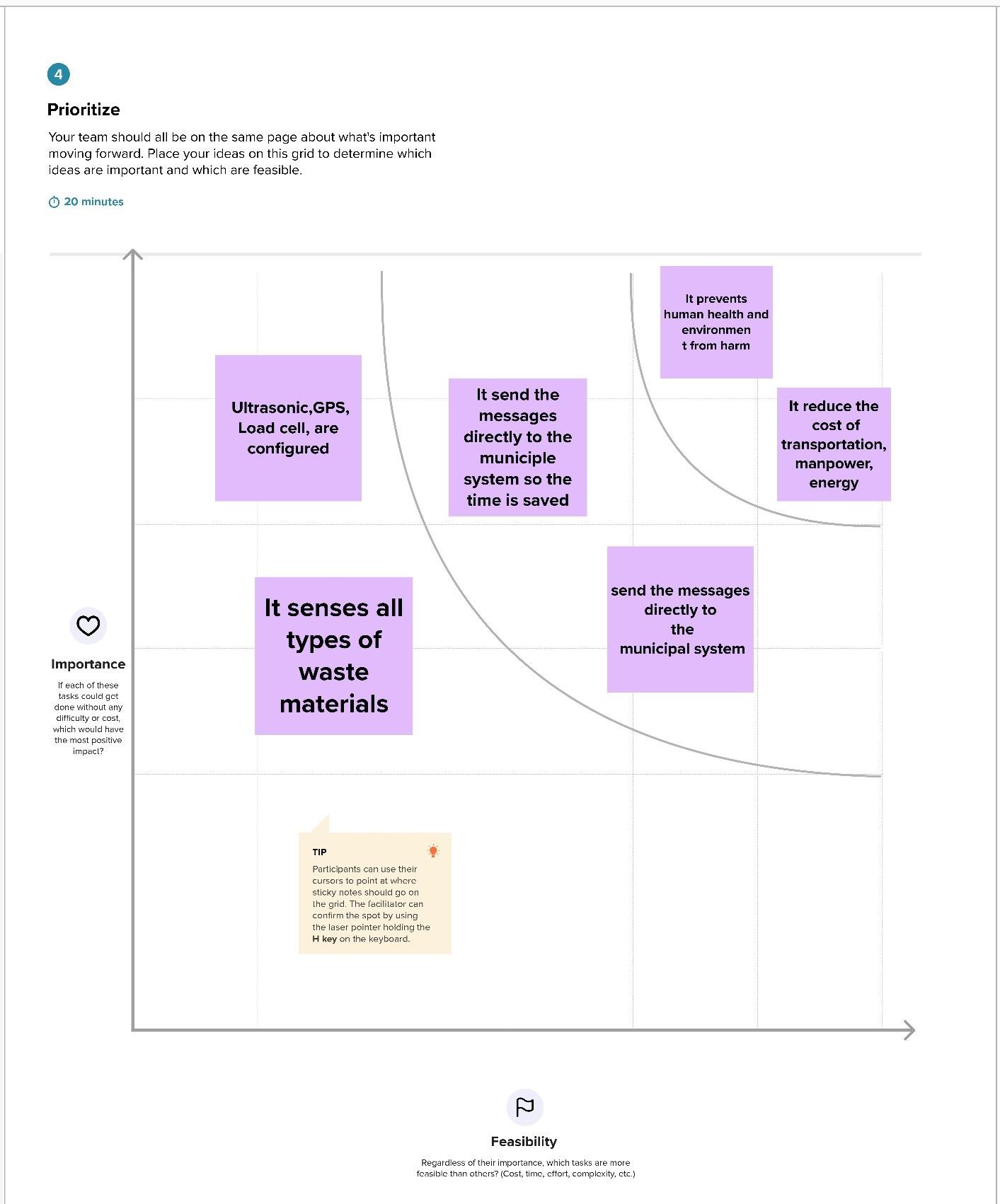
# Step-1: Team Gathering, Collaboration and Select the Problem Statement

****

**Step-2: Brainstorm, Idea Listing and Grouping**

****

**Step-3: Idea Prioritization**



**3.3 Proposed Solution**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be solved) | This project enables the organizations to meet their needs of smart garbage management systems. This system allows the authorized person to know the fill level of each garbage bin in a locality or city at all times, to give a cost-effective and time-saving route to the truck drivers. |
| 2. | Idea / Solution description | The key research objectives are as follows:     * The proposed system would be able to automate the solid waste monitoring process and management of the overall collection process using IOT. * The Proposed system consists of main subsystems namely Smart Trash System(STS) and Smart Monitoring and Controlling Hut(SMCH).      * In the proposed system, whenever the waste bin gets filled this is acknowledged by placing the circuit at the waste bin, which transmits it to the receiver at the desired place in the area or spot.      * In the proposed system, the received signal indicates the waste bin status at monitoring control system. |
| 3. | Novelty / Uniqueness | We are going to establish SWM in our college but the real hard thing is that janitor (cleaner) don’t know to operate these thing practically so here our team planned to build a wrist band to them, that indicate via light blinking when the dustbin fill and this is Uniqueness we made here beside from project constrain. |
| 4. | Social Impact / Customer Satisfaction | From the public perception as worst impacts of present solid waste disposal practices are seen direct social impacts such as neighbourhood of landfills to communities, breeding of pests and in property values. |
| 5. | Business Model (Revenue Model) | Waste Management organizes its operations into two reportable business segments: Solid Waste, comprising the Company’s waste collection, transfer, recycling and resource recovery, and disposal services, which are operated and managed locally by the Company’s various subsidiaries, which focus on distinct geographic areas; and Corporate and Other, comprising the Company’s other activities, including its development and operation of landfill gas-to energy facilities in the INDIA, and its recycling brokerage services, as well as various corporate function. |
| 6. | Scalability of the Solution | The proposed system uses sensor and communication technologies where waste data is collected from the smart bin, in real-time, and then transmitted to an online platform where citizens can access and check the availability of the compartments scattered around a city. |

**CHAPTER 4: REQUIRMENT ANALYSIS**

**4.1 Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | Detailed bin inventory. | The bins and stands which are monitored can be seen on the map, and we can also visit them at any time via the Street View feature from Google map .Bins or stands are visible on the map as green, orange circles.  We can also see the bin details in the Dashboard – last weight measurement, GPS location and collection schedule |
| FR-2 | Real time bin monitoring. | The Dashboard displays which displays all the real-time data on filling levels of bins monitored by smart sensors. Along to the percentage of fill level, based on the previous data, the tool predicts when the bin will become full, one of the functionalities that are not included even in the best waste management software. Sensors recognize picks as well; so you can check when the bin was collected last. With the help of real-time data and predictions, you can eliminate the overflowing bins and stop collecting half- empty ones. |
| FR-3 | Expensive bins. | One can help you identify bins that drove up your collection costs. The tool calculates a rating for each bin in terms of collection costs. The tool considers the average distance depo bin discharge in the area. The tool assigns bin a rating and calculates the distance from depo bin discharge. |
| FR-4 | Adjust bin distribution. | Ensure that the most optimal distribution of bins and Identify areas with either dense or sparse bin distribution. Make sure that all trash types are represented within a stand. Based on the previous data, you can adjust bin capacity or location where ever necessary. |
| FR-5 | Eliminate Un efficient picks. | Removing the collection of half-empty bins. By using real- time data on fill-levels and pick recognition, we can show you how full the bins can be collected. |
| FR-6 | Plan waste collection routes. | The tool which semi-automates the waste collection planning of route. Based on current bin fill levels and predictions of reaching full capacity, we need to be ready to respond and scheduled the waste collect. We can also compare planned vs. executed routes to identify inconsistencies. |

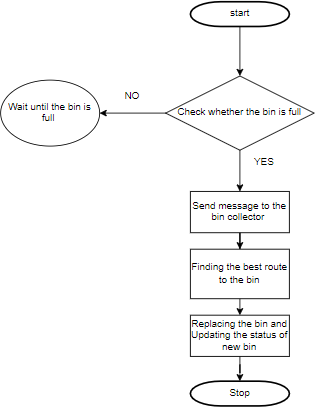
**4.2 Non-functional Requirements**

|  |  |  |
| --- | --- | --- |
| **NFR No.** | **Non-Functional Requirement** | **Description** |
| 1 | Usability | IOT device verifies that usability is a special and important perspective to analyze user requirements, which can further improve the design quality. In the design process with user experience as the core, the analysis of users’ product usability can indeed help designers better understand users’ potential needs in waste management, behaviour and experience. |
| 2 | Security | * Use of reusable bottles * Use of reusable grocery bags * Purchase wisely and recycle * Avoid single use food and drink containers |
| 3 | Reliability | Smart waste management is also about creating better working conditions for waste collectors and drivers. Instead of driving the same collection routes and servicing empty bins, waste collectors will spend their time more efficiently, taking care of bins that need servicing. |
| 4 | Performance | The Smart Sensors use ultrasound technology to measure the fill levels (along with other data) in bins several times a day. Using a variety of IOT networks (NB-IOT, GPRS), the sensors send the data to IBM Watson, that contains all the devices. Customers are hence provided data-driven decision making and optimization of waste collection routes, frequencies, and vehicle loads resulting in route reduction by at least 30%. |
| 5 | Availability | By developing & deploying resilient hardware and beautiful software we empower cities, businesses, and countries to manage waste smarter. |
| 6 | Scalability | Using smart waste bins reduce the number of bins inside town , cities coz we able to monitor the garbage 24/7 more cost effect and scalability when we moves to smarter. |

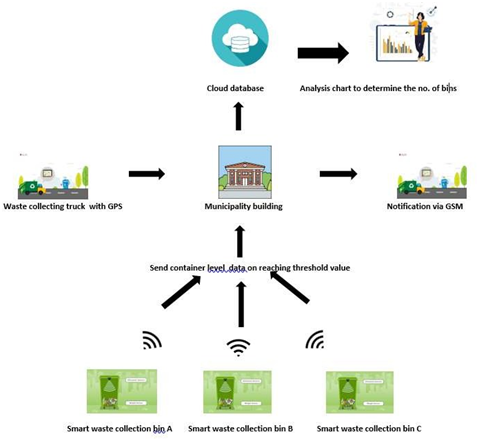
**CHAPTER 5: PROJECT DESIGN**

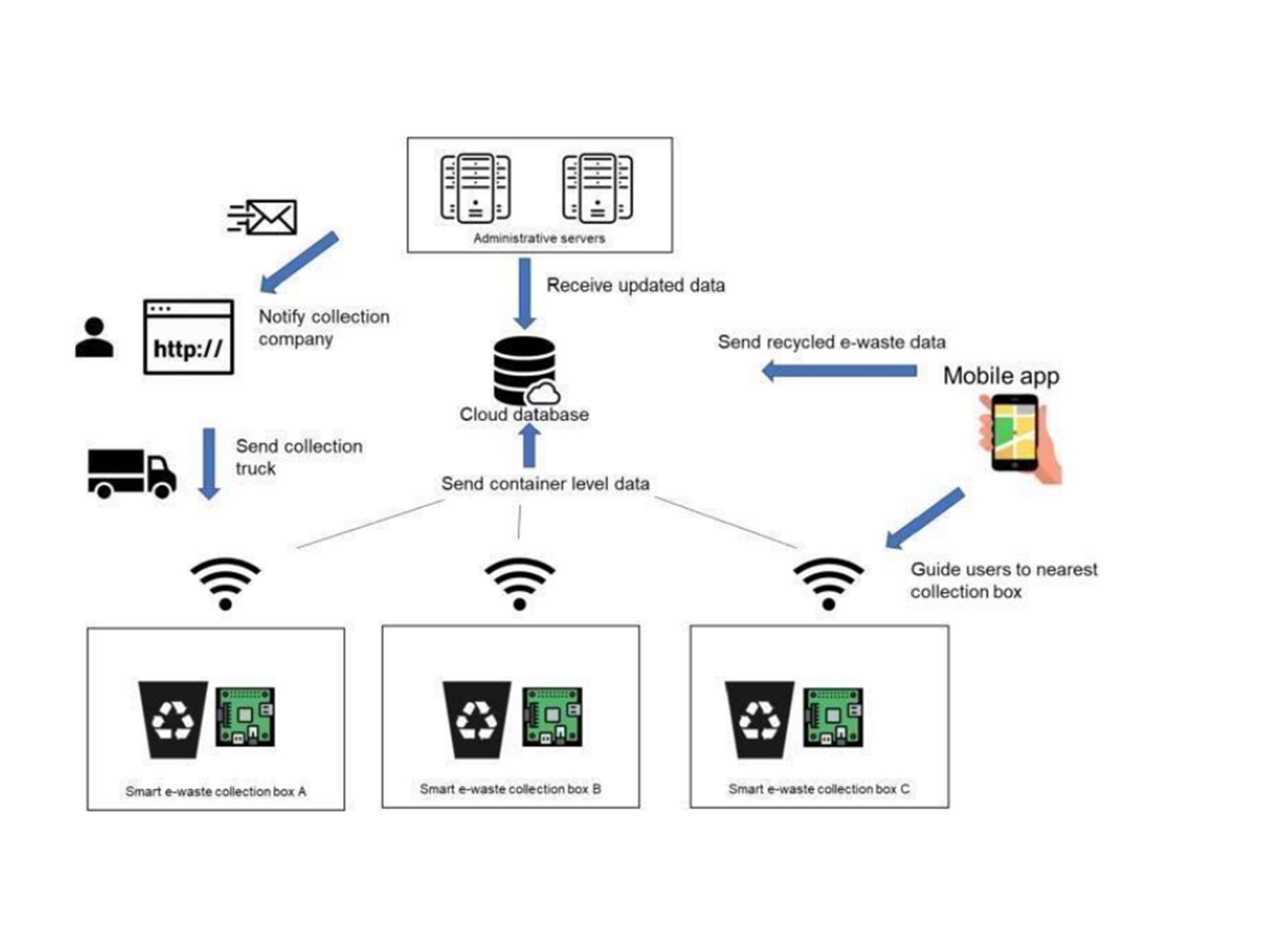
* 1. **Data Flow Diagram:**

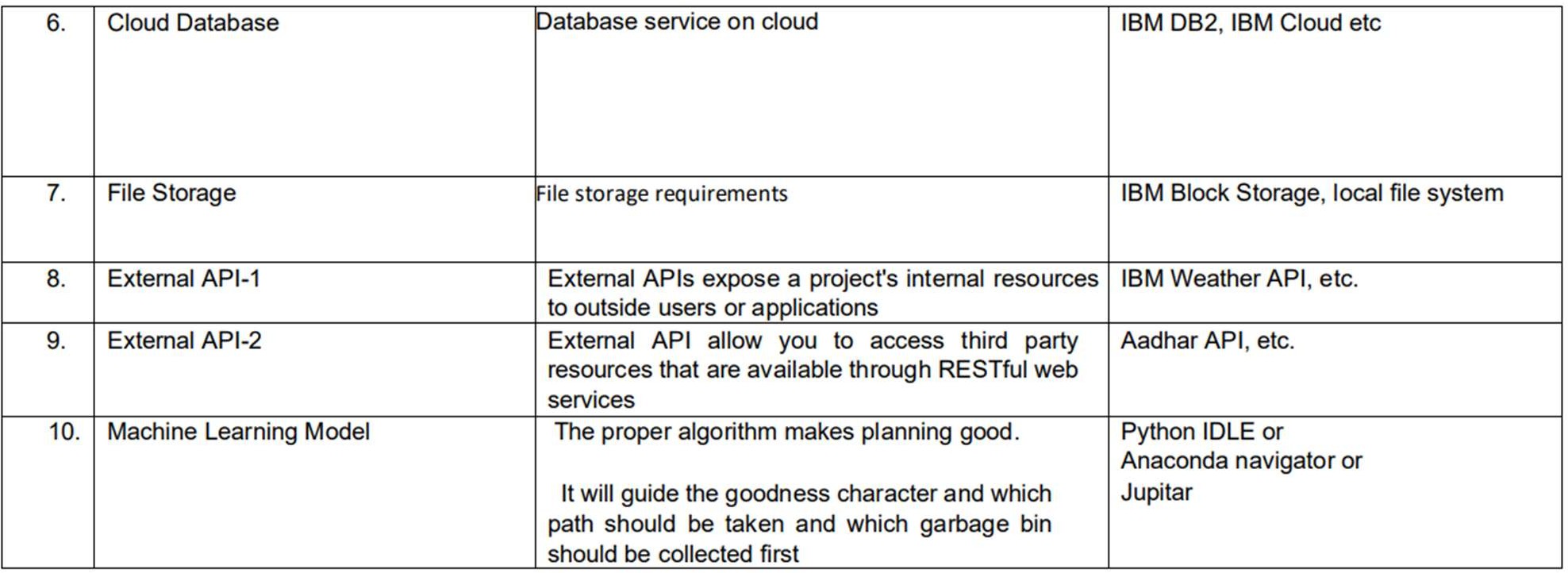
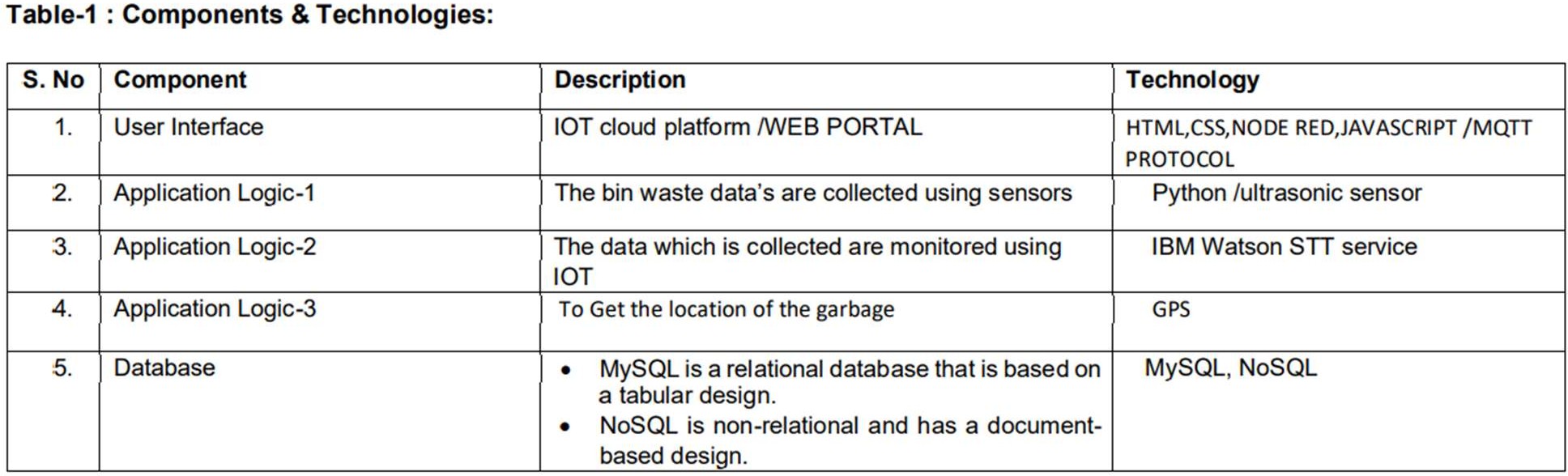
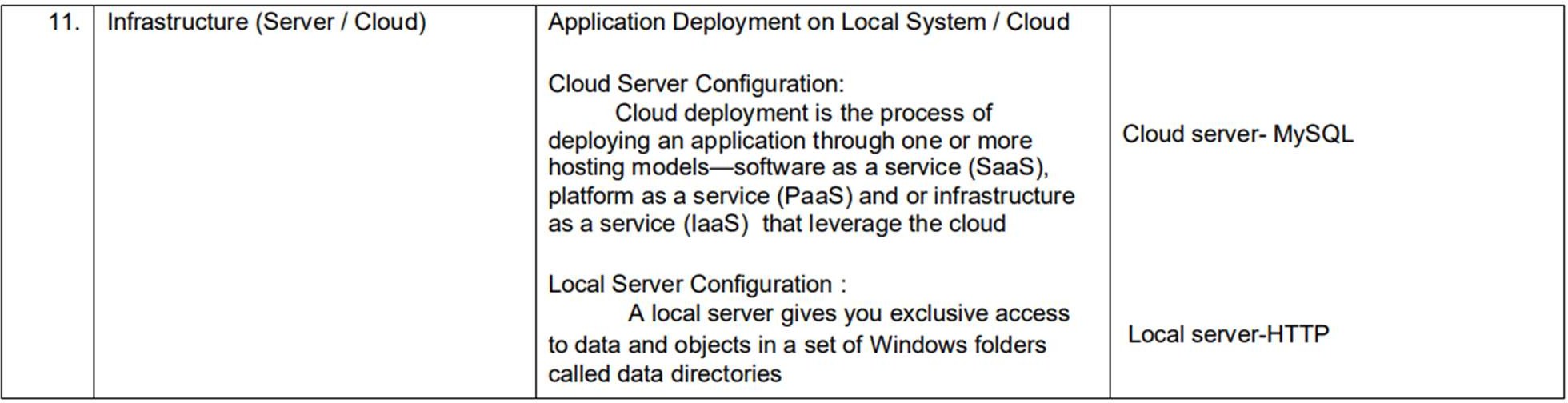
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

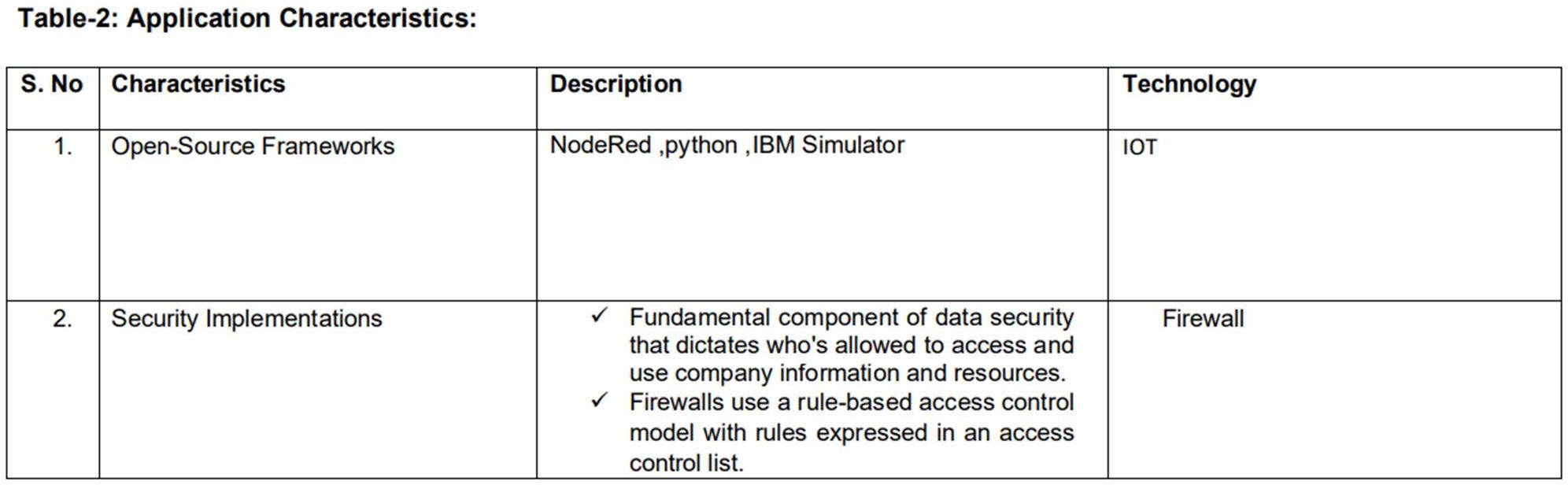


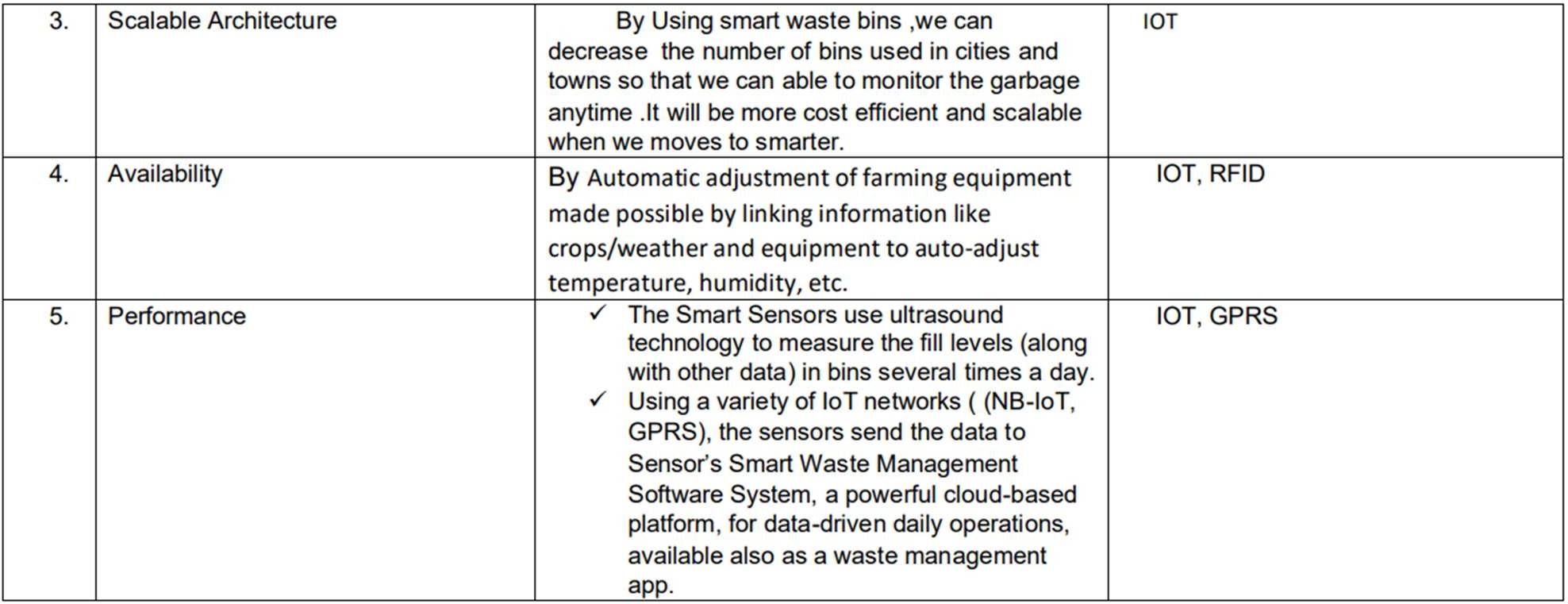
* 1. **Solution and Technical Architecture**



****

****

****

****

**CHAPTER 6: PROJECT PLANNING & SCHEDULING**

**6.1 Sprint Planning & Estimation:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| Sprint-1 | Login | USN-1 | As an Administrator, I need to give user id and pass code for ever workers over there in municipality. | 10 | High | Agishma Sini C  Pradhisha P  Teena S  Nivetha Sarojini R |
| Sprint-1 | Login | USN-1 | As a Co-Admin, I’ll control the waste level by monitoring them via real time web portal. Once the filling happens, I’ll notify trash truck with location of bin with bin ID. | 10 | High | Agishma Sini C  Pradhisha P  Teena S  Nivetha Sarojini R |
| Sprint-2 | Dashboard | USN-2 | As a Truck Driver, I’ll follow Co-Admin’s instruction to reach the filling bin in short roots and save time. | 20 | Low | Agishma Sini C  Pradhisha P  Teena S  Nivetha Sarojini R |
| Sprint-3 | Dashboard | USN-3 | As a Local Garbage Collector, I’II gather all the waste from the garbage, load it onto a garbage truck, and deliver it to landfills. | 20 | Medium | Agishma Sini C  Pradhisha P  Teena S  Nivetha Sarojini R |
| Sprint-4 | Dashboard | USN-4 | As a Municipality officer, I'll make sure everything is proceeding as planned and without any problems. | 20 | High | Agishma Sini C  Pradhisha P  Teena S  Nivetha Sarojini R |

**6.2 Sprint Delivery Schedule:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Total Story Points** | **Duration** | **Sprint Start Date** | **Sprint End Date (Planned)** | **Story Points**  **Completed (as on Planned End Date)** | **Sprint Release Date (Actual)** |
| Sprint-1 | 20 | 6 Days | 24 Oct 2022 | 31 Oct 2022 | 20 | 29 Oct 2022 |
| Sprint-2 | 20 | 6 Days | 01 Nov 2022 | 05 Nov 2022 | 20 | 05 Nov 2022 |
| Sprint-3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 20 | 12 Nov 2022 |
| Sprint-4 | 20 | 6 Days | 12 Nov 2022 | 19 Nov 2022 | 20 | 16 Nov 2022 |

**CHAPTER 7: CODING AND SOLUTIONING**

**7.1 Feature 1:**

The main and first feature of the smart waste management is to get the live location of anyone who access the website for putting out a request for garbage collection in their locality. The live location is obtained as a result of the below code.

**Web Application to get the Live location:**

**index.html:**

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>Garbage Management System</title>

<link rel="icon" type="image/x-icon" href="/Images/DUMPSTER.png">

<link href="style.css" rel="stylesheet" type="text/css" />

<script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-app.js"></script>

<script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-database.js"></script>

<script>

var firebaseConfig =

{

apiKey: "AIzaSyB9ysbnaWc3IyeCioh-aJQT\_UCMd5CBFeU",

authDomain: "fir-test-923b4.firebaseapp.com",

databaseURL: "https://fir-test-923b4-default-rtdb.firebaseio.com",

projectId: "fir-test-923b4",

storageBucket: "fir-test-923b4.appspot.com",

messagingSenderId: "943542145393",

appId: "1:943542145393:web:9b5ec7593e6a3cbd7966d0",

measurementId: "G-BN7JNX1Q7B"

};

firebase.initializeApp(firebaseConfig)

</script>

<script defer src="database.js"></script>

</head>

<body style="background-color:#1F1B24;">

<script src="map.js"></script>

<div id="map\_container">

<h1 id="live\_location\_heading" >LIVE LOCATION</h1>

<div id="map"></div>

<div id="alert\_msg">ALERT MESSAGE!</div>

</div>

</div>

<center><a href="https://goo.gl/maps/G9XET5mzSw1ynHQ18"

type="button" class="btn btn-dark">DUMPSTER</a></center>

<script

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBBLyWj-3FWtCbCXGW3ysEiI2fDfrv2v0Q&callback=myMap"></script></div>

</body>

</html>

**db.js:**

const cap\_status = document.getElementById('cap\_status');

const alert\_msg = document.getElementById('alert\_msg');

var ref = firebase.database().ref();

ref.on("value", function(snapshot)

{

snapshot.forEach(function (childSnapshot) {

var value = childSnapshot.val();

const alert\_msg\_val = value.alert;

const cap\_status\_val = value.distance\_status;

alert\_msg.innerHTML= `${alert\_msg\_val}`;

});

}, function (error) {

console.log("Error: " + error.code);

});

**maps.js:**

const database = firebase.database();

function myMap()

{

var ref1 = firebase.database().ref();

ref1.on("value", function(snapshot)

{

snapshot.forEach(function (childSnapshot) {

var value = childSnapshot.val();

const latitude = value.latitude;

const longitude = value.longitude;

var latlong = { lat: latitude, lng: longitude}

var mapProp =

{

center: new google.maps.LatLng(latlong),

zoom: 10,

};

var map = new google.maps.Map(document.getElementById("map"), mapProp);

var marker = new google.maps.Marker({ position: latlong });

marker.setMap(map);

});

}, function (error) {

console.log("Error: " + error.code);

});

}

**replit.nix**

**{** pkgs }: {

deps = [

pkgs.nodePackages.vscode-langservers-extracted

pkgs.nodePackages.typescript-language-server

];

}

**7.2 Feature 2:**

In this part, the filled level of the bin is measured with the help of IBM IOT Watson platform devices, IBM Cloud interface and Node-RED is used for creating the dashboard nodes that helps us create a UI to display the distance, that is, the fill level of the bin. It also intimates the location of the bin with the fill level and alerts the collection authority if the fill level goes beyond a threshold value.

**Code to evaluate the level of the garbage in bin:**

**bin1.py:**

import requests

import json

import ibmiotf.application

import ibmiotf.device

import time

import random

import sys

# watson device details

organization = "73ffyv"

devicType = "BIN1"

deviceId = "BIN1ID"

authMethod= "token"

authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)

def myCommandCallback(cmd):

global a

print("command recieved is:%s" %cmd.data['command'])

control=cmd.data['command']

print(control)

try:

deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-method":authMethod,"auth-token":authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

except Exception as e:

print("Exception while connecting device %s" %str(e))

sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for every 10 seconds

deviceCli.connect()

while True:

distance= random.randint(10,70)

loadcell= random.randint(5,15)

data= {'dist':distance,'load':loadcell}

if loadcell < 13 and loadcell > 15:

load = "90 %"

elif loadcell < 8 and loadcell > 12:

load = "60 %"

elif loadcell < 4 and loadcell > 7:

load = "40 %"

else:

load = "0 %"

if distance < 15:

dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'

elif distance < 40 and distance >16:

dist = 'Risk warning:' 'garbage is above 60%'

elif distance < 60 and distance > 41:

dist = 'Risk warning:' '40 %'

else:

dist = 'Risk warning:' '17 %'

if load == "90 %" or distance == "90 %":

warn = 'alert :' ' Garbage level is high, collection time :)'

elif load == "60 %" or distance == "60 %":

warn = 'alert :' 'garbage is above 60%'

else :

warn = 'alert :' 'Levels are low, collection not needed '

def myOnPublishCallback(lat=11.035081,long=77.014616):

print("Peelamedu, Coimbatore")

print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat = %s" %lat)

print(load)

print(dist)

print(warn)

time.sleep(10)

success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on\_publish= myOnPublishCallback)

success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on\_publish= myOnPublishCallback)

if not success:

print("not connected to ibmiot")

time.sleep(30)

deviceCli.commandCallback=myCommandCallback

#disconnect the device

deviceCli.disconnect()

**bin2.py:**

import requests

import json

import ibmiotf.application

import ibmiotf.device

import time

import random

import sys

# watson device details

organization = "73ffyv"

devicType = "BIN2"

deviceId = "BIN2ID"

authMethod= "token"

authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)

def myCommandCallback(cmd):

global a

print("command recieved is:%s" %cmd.data['command'])

control=cmd.data['command']

print(control)

try:

deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-method":authMethod,"auth-token":authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

except Exception as e:

print("Exception while connecting device %s" %str(e))

sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for every 10 seconds

deviceCli.connect()

while True:

distance= random.randint(10,70)

loadcell= random.randint(5,15)

data= {'dist':distance,'load':loadcell}

if loadcell < 13 and loadcell > 15:

load = "90 %"

elif loadcell < 8 and loadcell > 12:

load = "60 %"

elif loadcell < 4 and loadcell > 7:

load = "40 %"

else:

load = "0 %"

if distance < 15:

dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'

elif distance < 40 and distance >16:

dist = 'Risk warning:' 'garbage is above 60%'

elif distance < 60 and distance > 41:

dist = 'Risk warning:' '40 %'

else:

dist = 'Risk warning:' '17 %'

if load == "90 %" or distance == "90 %":

warn = 'alert :' ' Garbage level is high, collection time :)'

elif load == "60 %" or distance == "60 %":

warn = 'alert :' 'garbage is above 60%'

else :

warn = 'alert :' 'Levels are low, collection not needed '

def myOnPublishCallback(lat=11.068774,long=77.092978):

print("PSG iTech, Coimbatore")

print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat = %s" %lat)

print(load)

print(dist)

print(warn)

time.sleep(10)

success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on\_publish= myOnPublishCallback)

success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on\_publish= myOnPublishCallback)

if not success:

print("not connected to ibmiot")

time.sleep(30)

deviceCli.commandCallback=myCommandCallback

#disconnect the device

deviceCli.disconnect()

**bin3.py:**

import requests

import json

import ibmiotf.application

import ibmiotf.device

import time

import random

import sys

# watson device details

organization = "73ffyv"

devicType = "BIN3"

deviceId = "BIN3ID"

authMethod= "token"

authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)

def myCommandCallback(cmd):

global a

print("command recieved is:%s" %cmd.data['command'])

control=cmd.data['command']

print(control)

try:

deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-method":authMethod,"auth-token":authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

except Exception as e:

print("Exception while connecting device %s" %str(e))

sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for every 10 seconds

deviceCli.connect()

while True:

distance= random.randint(10,70)

loadcell= random.randint(5,15)

data= {'dist':distance,'load':loadcell}

if loadcell < 13 and loadcell > 15:

load = "90 %"

elif loadcell < 8 and loadcell > 12:

load = "60 %"

elif loadcell < 4 and loadcell > 7:

load = "40 %"

else:

load = "0 %"

if distance < 15:

dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'

elif distance < 40 and distance >16:

dist = 'Risk warning:' 'garbage is above 60%'

elif distance < 60 and distance > 41:

dist = 'Risk warning:' '40 %'

else:

dist = 'Risk warning:' '17 %'

if load == "90 %" or distance == "90 %":

warn = 'alert :' ' Garbage level is high, collection time :)'

elif load == "60 %" or distance == "60 %":

warn = 'alert :' 'garbage is above 60%'

else :

warn = 'alert :' 'Levels are low, collection not needed '

def myOnPublishCallback(lat=11.007403,long=76.963439):

print("Kattoor, Coimbatore")

print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat = %s" %lat)

print(load)

print(dist)

print(warn)

time.sleep(10)

success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on\_publish= myOnPublishCallback)

success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on\_publish= myOnPublishCallback)

if not success:

print("not connected to ibmiot")

time.sleep(30)

deviceCli.commandCallback=myCommandCallback

#disconnect the device

deviceCli.disconnect()

**bin4.py:**

import requests

import json

import ibmiotf.application

import ibmiotf.device

import time

import random

import sys

# watson device details

organization = "73ffyv"

devicType = "BIN4"

deviceId = "BIN4ID"

authMethod= "token"

authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)

def myCommandCallback(cmd):

global a

print("command recieved is:%s" %cmd.data['command'])

control=cmd.data['command']

print(control)

try:

deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-method":authMethod,"auth-token":authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

except Exception as e:

print("Exception while connecting device %s" %str(e))

sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for every 10 seconds

deviceCli.connect()

while True:

distance= random.randint(10,70)

loadcell= random.randint(5,15)

data= {'dist':distance,'load':loadcell}

if loadcell < 13 and loadcell > 15:

load = "90 %"

elif loadcell < 8 and loadcell > 12:

load = "60 %"

elif loadcell < 4 and loadcell > 7:

load = "40 %"

else:

load = "0 %"

if distance < 15:

dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'

elif distance < 40 and distance >16:

dist = 'Risk warning:' 'garbage is above 60%'

elif distance < 60 and distance > 41:

dist = 'Risk warning:' '40 %'

else:

dist = 'Risk warning:' '17 %'

if load == "90 %" or distance == "90 %":

warn = 'alert :' ' Garbage level is high, collection time :)'

elif load == "60 %" or distance == "60 %":

warn = 'alert :' 'garbage is above 60%'

else :

warn = 'alert :' 'Levels are low, collection not needed '

def myOnPublishCallback(lat=11.453306,long=77.426024):

print("Seethammal Colony, Gobichittipalayam")

print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat = %s" %lat)

print(load)

print(dist)

print(warn)

time.sleep(10)

success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on\_publish= myOnPublishCallback)

success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on\_publish= myOnPublishCallback)

if not success:

print("not connected to ibmiot")

time.sleep(30)

deviceCli.commandCallback=myCommandCallback

#disconnect the device

deviceCli.disconnect()

**7.3 Feature 3:**

An additional feature added to the smart waste management system is to measure the weight of the bin using hx711 load cell. The weight of the bin is the output of the below code.

**Measuring the weight of the garbage bin:**

**main.py:**

from hx711 import HX711

hx = HX711(5,4,64)

print(1)

while True:

    hx.tare()

    read = hx.read()

    value=hx.read\_average()

    print(value,"#")

**hx711.py:**

from machine import Pin, enable\_irq, disable\_irq, idle

class HX711:

    def \_\_init\_\_(self, dout, pd\_sck, gain=128):

*self*.pSCK = Pin(pd\_sck , mode=Pin.OUT)

*self*.pOUT = Pin(dout, mode=Pin.IN, pull=Pin.PULL\_DOWN)

*self*.pSCK.value(False)

*self*.GAIN = 0

*self*.OFFSET = 0

*self*.SCALE = 1

*self*.time\_constant = 0.1

*self*.filtered = 0

*self*.set\_gain(gain);

    def set\_gain(self, gain):

        if gain is 128:

*self*.GAIN = 1

        elif gain is 64:

*self*.GAIN = 3

        elif gain is 32:

*self*.GAIN = 2

*self*.read()

*self*.filtered = *self*.read()

        print('Gain & initial value set')

    def is\_ready(self):

        return *self*.pOUT() == 0

    def read(self):

        # wait for the device being ready

        while *self*.pOUT() == 1:

            idle()

        # shift in data, and gain & channel info

        result = 0

        for j in range(24 + *self*.GAIN):

            state = disable\_irq()

*self*.pSCK(True)

*self*.pSCK(False)

            enable\_irq(state)

            result = (result << 1) | *self*.pOUT()

        # shift back the extra bits

        result >>= *self*.GAIN

        # check sign

        if result > 0x7fffff:

            result -= 0x1000000

        return result

    def read\_average(self, times=3):

        s = 0

        for i in range(times):

            s += *self*.read()

        ss=(s/times)/210

        return '%.1f' %(ss)

    def read\_lowpass(self):

*self*.filtered += *self*.time\_constant \* (*self*.read() - *self*.filtered)

        return *self*.filtered

    def get\_value(self, times=3):

        return *self*.read\_average(times) - *self*.OFFSET

    def get\_units(self, times=3):

        return *self*.get\_value(times) / *self*.SCALE

    def tare(self, times=15):

        s = *self*.read\_average(times)

*self*.set\_offset(s)

    def set\_scale(self, scale):

*self*.SCALE = scale

    def set\_offset(self, offset):

*self*.OFFSET = offset

    def set\_time\_constant(self, time\_constant = None):

        if time\_constant is None:

            return *self*.time\_constant

        elif 0 < time\_constant < 1.0:

*self*.time\_constant = time\_constant

    def power\_down(self):

*self*.pSCK.value(False)

*self*.pSCK.value(True)

    def power\_up(self):

*self*.pSCK.value(False)

**CHAPTER 8: TESTING**

**8.1 Test cases:**

**Unit testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case no.** | **Sensor/Stage** | **Input** | **Expected output** | **Obtained output** | **Status** |
| 1. | Ultrasonic | Garbage level in bin  i)Null  ii)Full  iii)Range in % | Correct level or distance | As expected | Pass |
| 2. | ESP – 32 | Microcontroller to process the input data | To collect the data from sensor | As expected | Pass |
| 3. | Load cell | To measure mechanical force | Calculate the force due to the bin weight | As expected | Pass |
| 4. | Gauge | To display the tares | Display the level for collection | As expected | Pass |
| 5. | HX710 | Weight of the bin  (in kg) | Measure the weight | As expected | Pass |

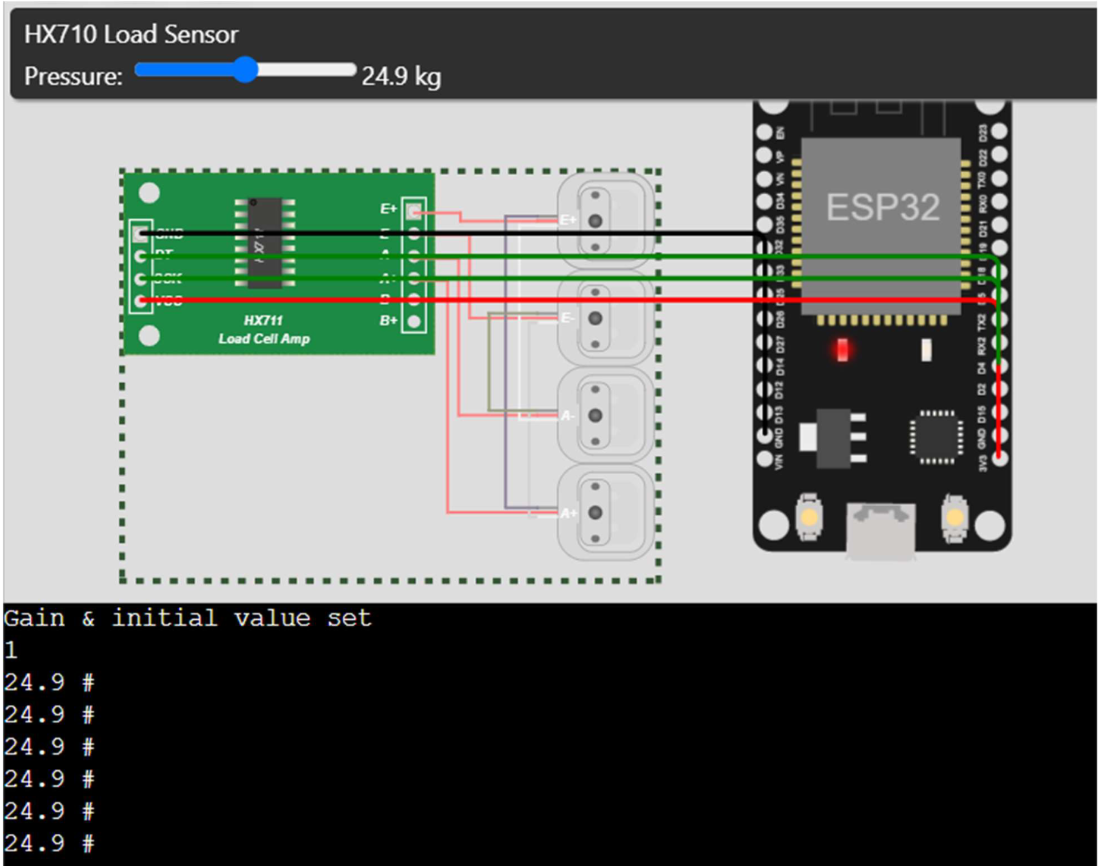
**8.2 User Acceptance testing**

Acceptance testing - is the final phase of product testing prior to public launch. A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system’s compliance with the business requirements and assess whether it is acceptable for delivery.

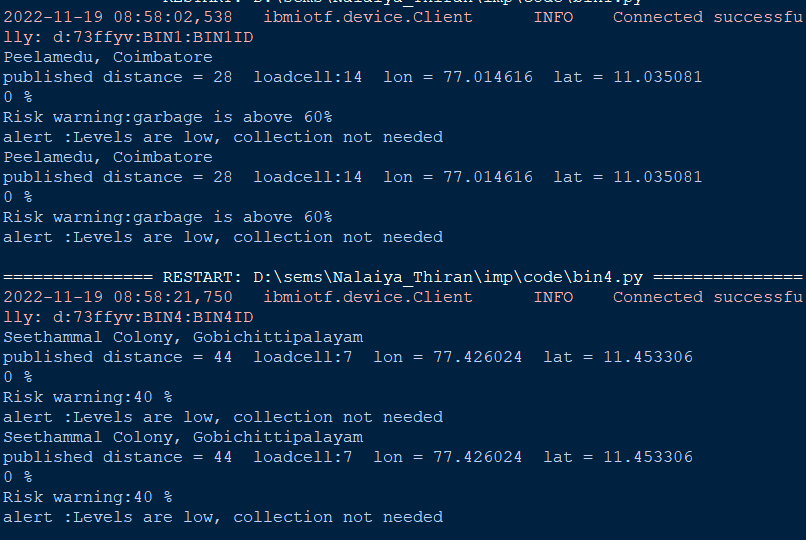
**CHAPTER 9: RESULTS**

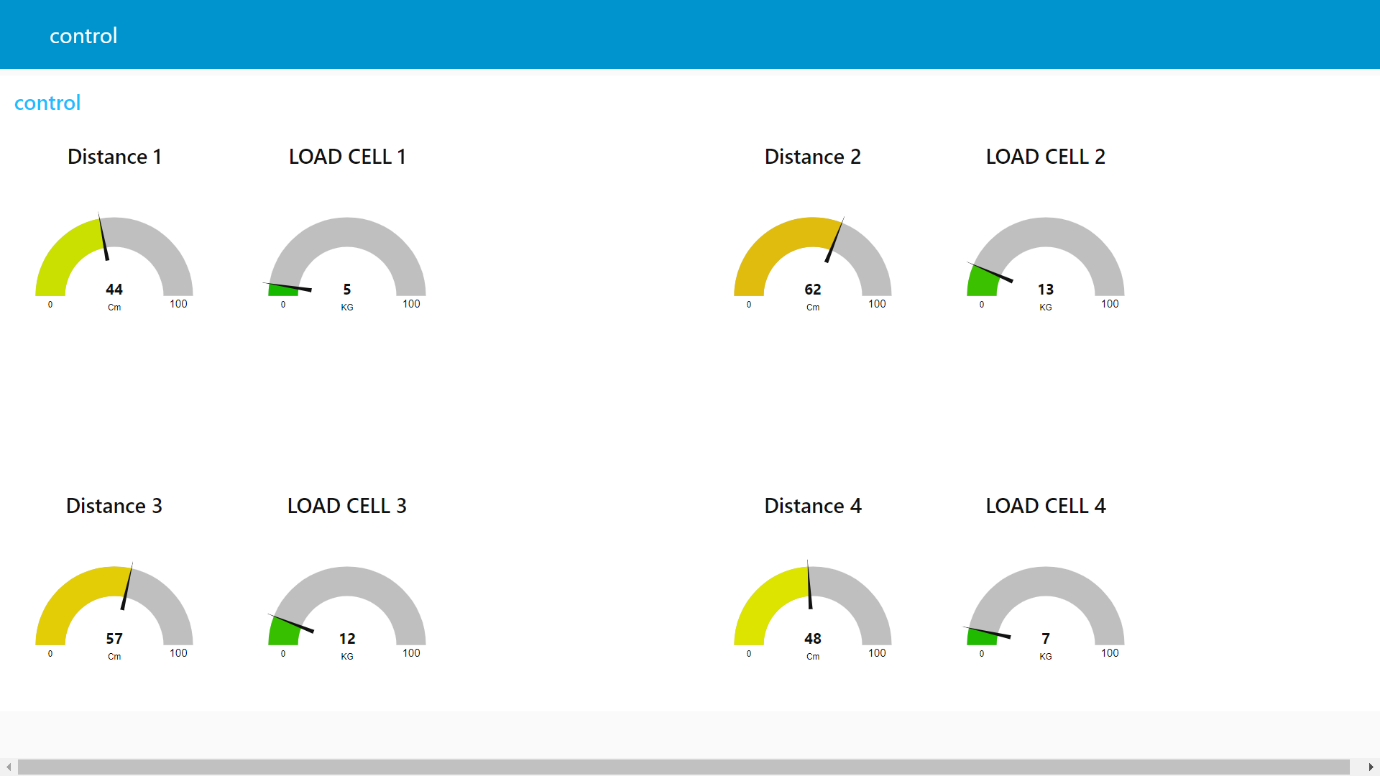
**Outputs:**

Calculating the weight of the bin using hx711 load sensor:



Measuring the weight of the bin and displaying the output using the nodes dashboard UI and python script, using IBM Watson as the intermediate to connect the devices:



****

**CHAPTER 10: ADVANTAGES AND DISADVANTAGES**

**10.1 Advantages:**

* Intelligent compaction of waste by monitoring fill level in real-time using sensors.
* It keeps our surroundings clean and keeps free from bad odour.
* Reduces manpower requirement to handle the garbage collection
* Emphasizes of healthy environment and keep the cities cleaner and more beautiful.
* It reduces infrastructure, operating and maintenance costs by upto 30%.
* Increases recycling rate of waste.

**10.2 Disadvantages:**

* Initial large-scale implementation takes cost.
* System requires more number waste bins for separate waste collection.
* Wireless technologies used should have proper connections as they have shorter range and lower data speed
* Training programs should be provided to people involving in the ecosystem of smart waste management.
* Sensors may encounter damage so it should be kept under protective ambience to prevent the damage.
* Replacement of sensors require knowledgeable people and thus acknowledgement of malfunction of sensor.

**CHAPTER 11: CONCLUSION**

Improper disposal and improper maintenance of domestic waste create issues in public health and environment pollution thus this paper attempts to provide practical solution towards managing the waste collaborating it with the use of IOT. by using the smart waste management system, we can manage waste properly we are also able to sort the Bio-degradable and non-Biodegradable waste properly which reduces the pollution in the environment. Various waste management initiatives taken for human well-being and to improve the TWM practices were broadly discussed in this chapter. The parameters that influence the technology and economic aspects of waste management were also discussed clearly. Different types of barriers in TWM, such as economic hitches, political issues, legislative disputes, informative and managerial as well as solutions and success factors for implementing an effective management of toxic organic waste within a globular context, were also discussed giving some real examples. The effect of urbanization on the environmental degradation and economic growth was also discussed. The proposed system will help to overcome all the serious issues related to waste and keep the environment clean.

**CHAPTER 12: FUTURE WORK**

Based on the real-time and historical data collected and stored in the cloud waste collection schedules and routes can be optimized. Predictive analytics could be used to make decisions ahead of time and offers insight into waste bin locations. Graph theory optimization algorithms can be used to manage waste collection strategies dynamically and efficiently. Every day, the workers can receive the newly calculated routes in their navigation devices. The system can be designed to learn from experience and to make decisions not only on the daily waste level status but also on future state forecast, traffic congestion, balanced cost-efficiency functions, and other affecting factors that a priori humans cannot foresee.

Garbage collectors could access the application on their mobile phone/tablets using the internet. Real-time GPS assistance can be used to direct them to the pre-decided route. As they go collecting the garbage from the containers, the management is also aware of the progress as the vehicle, as well as the garbage containers, are traced in real-time. The management staff gets their own personalized administration panel over a computer/tablet which gives them a bird eye view over the entire operations.

An alternative solution using image processing and camera as a passive sensor could be used. But, the cost of those image processing cameras is higher as compared to the ultrasonic sensors, which leads to high solution implementation cost.

**CHAPTER 13: APPENDIX**

**13.1 Source Code:**

**Web Application to get the Live location:**

**index.html:**

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

<meta charset="utf-8">

<meta name="viewport" content="width=device-width">

<title>Garbage Management System</title>

<link rel="icon" type="image/x-icon" href="/Images/DUMPSTER.png">

<link href="style.css" rel="stylesheet" type="text/css" />

<script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-app.js"></script>

<script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-database.js"></script>

<script>

var firebaseConfig =

{

apiKey: "AIzaSyB9ysbnaWc3IyeCioh-aJQT\_UCMd5CBFeU",

authDomain: "fir-test-923b4.firebaseapp.com",

databaseURL: "https://fir-test-923b4-default-rtdb.firebaseio.com",

projectId: "fir-test-923b4",

storageBucket: "fir-test-923b4.appspot.com",

messagingSenderId: "943542145393",

appId: "1:943542145393:web:9b5ec7593e6a3cbd7966d0",

measurementId: "G-BN7JNX1Q7B"

};

firebase.initializeApp(firebaseConfig)

</script>

<script defer src="database.js"></script>

</head>

<body style="background-color:#1F1B24;">

<script src="map.js"></script>

<div id="map\_container">

<h1 id="live\_location\_heading" >LIVE LOCATION</h1>

<div id="map"></div>

<div id="alert\_msg">ALERT MESSAGE!</div>

</div>

</div>

<center><a href="https://goo.gl/maps/G9XET5mzSw1ynHQ18"

type="button" class="btn btn-dark">DUMPSTER</a></center>

<script

src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBBLyWj-3FWtCbCXGW3ysEiI2fDfrv2v0Q&callback=myMap"></script></div>

</body>

</html>

**db.js:**

const cap\_status = document.getElementById('cap\_status');

const alert\_msg = document.getElementById('alert\_msg');

var ref = firebase.database().ref();

ref.on("value", function(snapshot)

{

snapshot.forEach(function (childSnapshot) {

var value = childSnapshot.val();

const alert\_msg\_val = value.alert;

const cap\_status\_val = value.distance\_status;

alert\_msg.innerHTML= `${alert\_msg\_val}`;

});

}, function (error) {

console.log("Error: " + error.code);

});

**maps.js:**

const database = firebase.database();

function myMap()

{

var ref1 = firebase.database().ref();

ref1.on("value", function(snapshot)

{

snapshot.forEach(function (childSnapshot) {

var value = childSnapshot.val();

const latitude = value.latitude;

const longitude = value.longitude;

var latlong = { lat: latitude, lng: longitude}

var mapProp =

{

center: new google.maps.LatLng(latlong),

zoom: 10,

};

var map = new google.maps.Map(document.getElementById("map"), mapProp);

var marker = new google.maps.Marker({ position: latlong });

marker.setMap(map);

});

}, function (error) {

console.log("Error: " + error.code);

});

}

**replit.nix**

{ pkgs }: {

deps = [

pkgs.nodePackages.vscode-langservers-extracted

pkgs.nodePackages.typescript-language-server

];

}

**style.css:**

html, body

{

height: 100%;

margin: 0px;

padding:0px;

}

#container

{

display: flex;

flex-direction: row;

height: 100%;

width: 100%;

position: relative;

}

#logo\_container

{

height: 100%;

width: 12%;

background-color: #C5C6D0;

display: flex;

flex-direction: column;

vertical-align: text-bottom;

}

.logo

{

width:70%;

margin: 5% 15%;

/\* border-radius: 50%; \*/

}

#logo\_3

{

vertical-align: text-bottom;

}

#data\_container

{

height: 100%;

width: 20%;

margin-left: 1%;

margin-right: 1%;

display: flex;

flex-direction: column;

}

#data\_status

{

height:60%;

width:8%;

margin:7%;

background-color: #691F6E;

display: flex;

flex-direction: column;

border-radius:20px;

}

#load\_status

{

background-image: url("/Images/KG.png");

background-repeat: no-repeat;

background-size: 170px;

background-position: left center;

}

#cap\_status

{

background-image: url("/Images/dust.png");

background-repeat: no-repeat;

background-size: 150px;

background-position: left center;

}

.status

{

width: 80%;

height: 40%;

margin:5% 10%;

background-color:#185adc;

border-radius:20px;

display: flex;

justify-content: center;

align-items: center;

color: white;

font-size: 60px;

}

.datas

{

width:86%;

margin:2.5% 7%;

height:10%;

background: url(water.png);

background-repeat: repeat-x;

animation: datas 10s linear infinite;

box-shadow: 0 0 0 6px #98d7eb, 0 20px 35px rgba(0,0,0,1);

}

#map\_container

{

height: 100%;

width: 100%;

display: flex;

flex-direction: column;

}

#live\_location\_heading

{

margin-top:10%;

text-align: center;

color: GREY;

}

#map

{

height: 70%;

width: 90%;

margin-left: 4%;

margin-right:4%;

border: 10px solid white;

border-radius: 25px;

}

#alert\_msg

{

width:92%;

height:20%;

margin:4%;

background-color:grey;

border-radius: 20px;

display: flex;

justify-content: center;

align-items: center;

color: #41af7f;

font-size: 25px;

font-weight: bold;

}

.lat

{

margin: 0px;

font-size:0px;

}

@keyframes datas{

0%

{

background-position: -500px 100px;

}

40%

{

background-position: 1000px -10px;

}

80% {

background-position: 2000px 40px;

}

100% {

background-position: 2700px 95px;

}

}

**Code to evaluate the level of the garbage in bin and intimate the collection authority with the location of the bin:**

**bin1.py:**

import requests

import json

import ibmiotf.application

import ibmiotf.device

import time

import random

import sys

# watson device details

organization = "73ffyv"

devicType = "BIN1"

deviceId = "BIN1ID"

authMethod= "token"

authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)

def myCommandCallback(cmd):

global a

print("command recieved is:%s" %cmd.data['command'])

control=cmd.data['command']

print(control)

try:

deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-method":authMethod,"auth-token":authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

except Exception as e:

print("Exception while connecting device %s" %str(e))

sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for every 10 seconds

deviceCli.connect()

while True:

distance= random.randint(10,70)

loadcell= random.randint(5,15)

data= {'dist':distance,'load':loadcell}

if loadcell < 13 and loadcell > 15:

load = "90 %"

elif loadcell < 8 and loadcell > 12:

load = "60 %"

elif loadcell < 4 and loadcell > 7:

load = "40 %"

else:

load = "0 %"

if distance < 15:

dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'

elif distance < 40 and distance >16:

dist = 'Risk warning:' 'garbage is above 60%'

elif distance < 60 and distance > 41:

dist = 'Risk warning:' '40 %'

else:

dist = 'Risk warning:' '17 %'

if load == "90 %" or distance == "90 %":

warn = 'alert :' ' Garbage level is high, collection time :)'

elif load == "60 %" or distance == "60 %":

warn = 'alert :' 'garbage is above 60%'

else :

warn = 'alert :' 'Levels are low, collection not needed '

def myOnPublishCallback(lat=11.035081,long=77.014616):

print("Peelamedu, Coimbatore")

print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat = %s" %lat)

print(load)

print(dist)

print(warn)

time.sleep(10)

success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on\_publish= myOnPublishCallback)

success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on\_publish= myOnPublishCallback)

if not success:

print("not connected to ibmiot")

time.sleep(30)

deviceCli.commandCallback=myCommandCallback

#disconnect the device

deviceCli.disconnect()

**bin2.py:**

import requests

import json

import ibmiotf.application

import ibmiotf.device

import time

import random

import sys

# watson device details

organization = "73ffyv"

devicType = "BIN2"

deviceId = "BIN2ID"

authMethod= "token"

authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)

def myCommandCallback(cmd):

global a

print("command recieved is:%s" %cmd.data['command'])

control=cmd.data['command']

print(control)

try:

deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-method":authMethod,"auth-token":authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

except Exception as e:

print("Exception while connecting device %s" %str(e))

sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for every 10 seconds

deviceCli.connect()

while True:

distance= random.randint(10,70)

loadcell= random.randint(5,15)

data= {'dist':distance,'load':loadcell}

if loadcell < 13 and loadcell > 15:

load = "90 %"

elif loadcell < 8 and loadcell > 12:

load = "60 %"

elif loadcell < 4 and loadcell > 7:

load = "40 %"

else:

load = "0 %"

if distance < 15:

dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'

elif distance < 40 and distance >16:

dist = 'Risk warning:' 'garbage is above 60%'

elif distance < 60 and distance > 41:

dist = 'Risk warning:' '40 %'

else:

dist = 'Risk warning:' '17 %'

if load == "90 %" or distance == "90 %":

warn = 'alert :' ' Garbage level is high, collection time :)'

elif load == "60 %" or distance == "60 %":

warn = 'alert :' 'garbage is above 60%'

else :

warn = 'alert :' 'Levels are low, collection not needed '

def myOnPublishCallback(lat=11.068774,long=77.092978):

print("PSG iTech, Coimbatore")

print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat = %s" %lat)

print(load)

print(dist)

print(warn)

time.sleep(10)

success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on\_publish= myOnPublishCallback)

success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on\_publish= myOnPublishCallback)

if not success:

print("not connected to ibmiot")

time.sleep(30)

deviceCli.commandCallback=myCommandCallback

#disconnect the device

deviceCli.disconnect()

**bin3.py:**

import requests

import json

import ibmiotf.application

import ibmiotf.device

import time

import random

import sys

# watson device details

organization = "73ffyv"

devicType = "BIN3"

deviceId = "BIN3ID"

authMethod= "token"

authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)

def myCommandCallback(cmd):

global a

print("command recieved is:%s" %cmd.data['command'])

control=cmd.data['command']

print(control)

try:

deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-method":authMethod,"auth-token":authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

except Exception as e:

print("Exception while connecting device %s" %str(e))

sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for every 10 seconds

deviceCli.connect()

while True:

distance= random.randint(10,70)

loadcell= random.randint(5,15)

data= {'dist':distance,'load':loadcell}

if loadcell < 13 and loadcell > 15:

load = "90 %"

elif loadcell < 8 and loadcell > 12:

load = "60 %"

elif loadcell < 4 and loadcell > 7:

load = "40 %"

else:

load = "0 %"

if distance < 15:

dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'

elif distance < 40 and distance >16:

dist = 'Risk warning:' 'garbage is above 60%'

elif distance < 60 and distance > 41:

dist = 'Risk warning:' '40 %'

else:

dist = 'Risk warning:' '17 %'

if load == "90 %" or distance == "90 %":

warn = 'alert :' ' Garbage level is high, collection time :)'

elif load == "60 %" or distance == "60 %":

warn = 'alert :' 'garbage is above 60%'

else :

warn = 'alert :' 'Levels are low, collection not needed '

def myOnPublishCallback(lat=11.007403,long=76.963439):

print("Kattoor, Coimbatore")

print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat = %s" %lat)

print(load)

print(dist)

print(warn)

time.sleep(10)

success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on\_publish= myOnPublishCallback)

success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on\_publish= myOnPublishCallback)

if not success:

print("not connected to ibmiot")

time.sleep(30)

deviceCli.commandCallback=myCommandCallback

#disconnect the device

deviceCli.disconnect()

**bin4.py:**

import requests

import json

import ibmiotf.application

import ibmiotf.device

import time

import random

import sys

# watson device details

organization = "73ffyv"

devicType = "BIN4"

deviceId = "BIN4ID"

authMethod= "token"

authToken= "123456789"

#generate random values for randomo variables (temperature&humidity)

def myCommandCallback(cmd):

global a

print("command recieved is:%s" %cmd.data['command'])

control=cmd.data['command']

print(control)

try:

deviceOptions={"org": organization, "type": devicType,"id": deviceId,"auth-method":authMethod,"auth-token":authToken}

deviceCli = ibmiotf.device.Client(deviceOptions)

except Exception as e:

print("Exception while connecting device %s" %str(e))

sys.exit()

#connect and send a datapoint "temp" with value integer value into the cloud as a type of event for every 10 seconds

deviceCli.connect()

while True:

distance= random.randint(10,70)

loadcell= random.randint(5,15)

data= {'dist':distance,'load':loadcell}

if loadcell < 13 and loadcell > 15:

load = "90 %"

elif loadcell < 8 and loadcell > 12:

load = "60 %"

elif loadcell < 4 and loadcell > 7:

load = "40 %"

else:

load = "0 %"

if distance < 15:

dist = 'Risk warning:' 'Garbage level is high, collection time :) 90 %'

elif distance < 40 and distance >16:

dist = 'Risk warning:' 'garbage is above 60%'

elif distance < 60 and distance > 41:

dist = 'Risk warning:' '40 %'

else:

dist = 'Risk warning:' '17 %'

if load == "90 %" or distance == "90 %":

warn = 'alert :' ' Garbage level is high, collection time :)'

elif load == "60 %" or distance == "60 %":

warn = 'alert :' 'garbage is above 60%'

else :

warn = 'alert :' 'Levels are low, collection not needed '

def myOnPublishCallback(lat=11.453306,long=77.426024):

print("Seethammal Colony, Gobichittipalayam")

print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s " %long,"lat = %s" %lat)

print(load)

print(dist)

print(warn)

time.sleep(10)

success=deviceCli.publishEvent ("IoTSensor","json",warn,qos=0,on\_publish= myOnPublishCallback)

success=deviceCli.publishEvent ("IoTSensor","json",data,qos=0,on\_publish= myOnPublishCallback)

if not success:

print("not connected to ibmiot")

time.sleep(30)

deviceCli.commandCallback=myCommandCallback

#disconnect the device

deviceCli.disconnect()

**Measuring the weight of the garbage bin:**

**main.py:**

from hx711 import HX711

hx = HX711(5,4,64)

print(1)

while True:

    hx.tare()

    read = hx.read()

    #average=hx.read\_average()

    value=hx.read\_average()

    print(value,"#")

**hx711.py:**

from machine import Pin, enable\_irq, disable\_irq, idle

class HX711:

    def \_\_init\_\_(self, dout, pd\_sck, gain=128):

*self*.pSCK = Pin(pd\_sck , mode=Pin.OUT)

*self*.pOUT = Pin(dout, mode=Pin.IN, pull=Pin.PULL\_DOWN)

*self*.pSCK.value(False)

*self*.GAIN = 0

*self*.OFFSET = 0

*self*.SCALE = 1

*self*.time\_constant = 0.1

*self*.filtered = 0

*self*.set\_gain(gain);

    def set\_gain(self, gain):

        if gain is 128:

*self*.GAIN = 1

        elif gain is 64:

*self*.GAIN = 3

        elif gain is 32:

*self*.GAIN = 2

*self*.read()

*self*.filtered = *self*.read()

        print('Gain & initial value set')

    def is\_ready(self):

        return *self*.pOUT() == 0

    def read(self):

        # wait for the device being ready

        while *self*.pOUT() == 1:

            idle()

        # shift in data, and gain & channel info

        result = 0

        for j in range(24 + *self*.GAIN):

            state = disable\_irq()

*self*.pSCK(True)

*self*.pSCK(False)

            enable\_irq(state)

            result = (result << 1) | *self*.pOUT()

        # shift back the extra bits

        result >>= *self*.GAIN

        # check sign

        if result > 0x7fffff:

            result -= 0x1000000

        return result

    def read\_average(self, times=3):

        s = 0

        for i in range(times):

            s += *self*.read()

        ss=(s/times)/210

        return '%.1f' %(ss)

    def read\_lowpass(self):

*self*.filtered += *self*.time\_constant \* (*self*.read() - *self*.filtered)

        return *self*.filtered

    def get\_value(self, times=3):

        return *self*.read\_average(times) - *self*.OFFSET

    def get\_units(self, times=3):

        return *self*.get\_value(times) / *self*.SCALE

    def tare(self, times=15):

        s = *self*.read\_average(times)

*self*.set\_offset(s)

    def set\_scale(self, scale):

*self*.SCALE = scale

    def set\_offset(self, offset):

*self*.OFFSET = offset

    def set\_time\_constant(self, time\_constant = None):

        if time\_constant is None:

            return *self*.time\_constant

        elif 0 < time\_constant < 1.0:

*self*.time\_constant = time\_constant

    def power\_down(self):

*self*.pSCK.value(False)

*self*.pSCK.value(True)

    def power\_up(self):

*self*.pSCK.value(False)

**13.2 Project Links:**

Github Link: https://github.com/IBM-EPBL/IBM-Project-44489-1660724897