

Nalaiya Thiran
Batch No: B3-3M5E
Jeppiaar Engineering College
Department of Information Technology

Smart Waste Management System For Metropolitan Cities

Team ID: PNT2022TMID27134

Team Lead:

Bentyson J

Team Members:

Aineesh Golda A

Dhileepan A

Akash T

Project Guide:

Industry mentor: Mr. Dinesh

Faculty mentor: Mr. Saravanan T

INDEX

S. No.	Title	Page No.
1	Introduction 1.1 Project Overview 1.2 Purpose	4
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 3.4 Problem Solution fit	13
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 5.3 User Stories	19
6	Project Planning and Scheduling 6.1 Sprint Planning & Estimation 6.2 Sprint Delivery Schedule 6.3 Reports from JIRA	23
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	38
9	Results 9.1 Performance Metrics	39
10	Advantages and Disadvantages	41
11	Conclusion	42
12	Future Works	43

13	Appendix 13.1 Source Code 13.2 Project Links	61
----	---	----

CHAPTER 1: INTRODUCTION

1.1 Project Overview

Garbage Management System which helps in the management of waste with the least human interaction in order to maintain a clean environment . Provide route planning for the collection based on the selected fill level and priorities of each bin.

It is very much faster than manual system.

Easy and fastest record finding technique.

It is very much flexible to work.

It is very user oriented.

Data can be stored for a longer period.

1.2 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

2.1 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 datadriven 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.

2.2 References

Bharadwaj B [1] Name of the Paper Automation of smart waste management. The main motto of the application is collection of dry and wet waste separately which is placed in a conveyor belt on which the dry waste collected dust bins are placed left side and wet waste collected bins on right side. The system will get the input through the dust collecting person through switches and sends signal to the Micro controller unit using RF technology and that makes the H-bridge to rotate conveyor belt. When the belt starts rotating clockwise the dust bin's lid is automatically closed,

simultaneously the waste is dumped into the underground garbage container placed at the ground floor Published at the year 2017.

Bikram Jit Singh [2] Name of the paper Smart dustbin for smart cities. To avoid all such situations, we have implemented a project called GSM based smart dustbins for smart cities. In this, dustbin is interfaced with microcontroller-based system having IR wireless system. If the dustbin is loaded with garbage, the status will be displayed on the screen. Published at the year 2016.

Siva Nagendra Reddy [3] Name of the paper Wireless dustbin monitoring and alert system using Arduino. This project is designed for the effective garbage collection using Embedded System. The main aim of the proposed method is collecting waste into the dumping vehicles. In this method whenever dustbin filled to certain levels the module placed on the dustbin will send an alert message to server node. From the server node it again sends a message to the concerned authorities. This system also sends information about harmful gases emanation. Published at the year 2017.

Mamta Pandey [4] Name of the paper Smart dustbin (IOT). we have designed a smart dustbin using ARDUINO UNO, ultrasonic sensor which will sense the item to be thrown in the dustbin and open the lid with the help of the motor. It is an IOT based project that will bring a new and smart way of cleanliness. It is a decent gadget to make your home clean, due to practically all offspring of home consistently make it grimy and spread litter to a great extent by

electronics, rappers and various other things. Published at the year 2020.

Samruddhi Bagalkot [5] Name of the paper iot based smart dustbin system. The main focus of this paper is on the monitoring of trash levels and tracking those garbage bins which are full and need to be flushed out immediately in those respective areas. In addition to that the data is sent into workstation for real time monitoring. After reaching maximum level of garbage in the dustbin the alerts are sent directly to the municipal corporation's garbage collector through GSM module. If in case any fire occurs in the dustbin due to any flammable object, it will get alert through buzzer so that nearby people will take required action.

2.3 Problem Statement Definition

The improvement of the urban waste collection service and, in general, the achievement of a more efficient management of the waste, is one of the main challenges that the cities face, especially due to the population growth. Poor waste management contributes to climate change and air pollution, and directly affects many ecosystems and species. And make the living area unclean and leak the foul odour throughout the city and even cause unknow diseases to human beings. Thus, smart waste management is a key factor of smart cities. A smart waste collection system that allows citizens to segregate the various types of waste they want to dispose and the municipal authorities to efficiently collect the same.

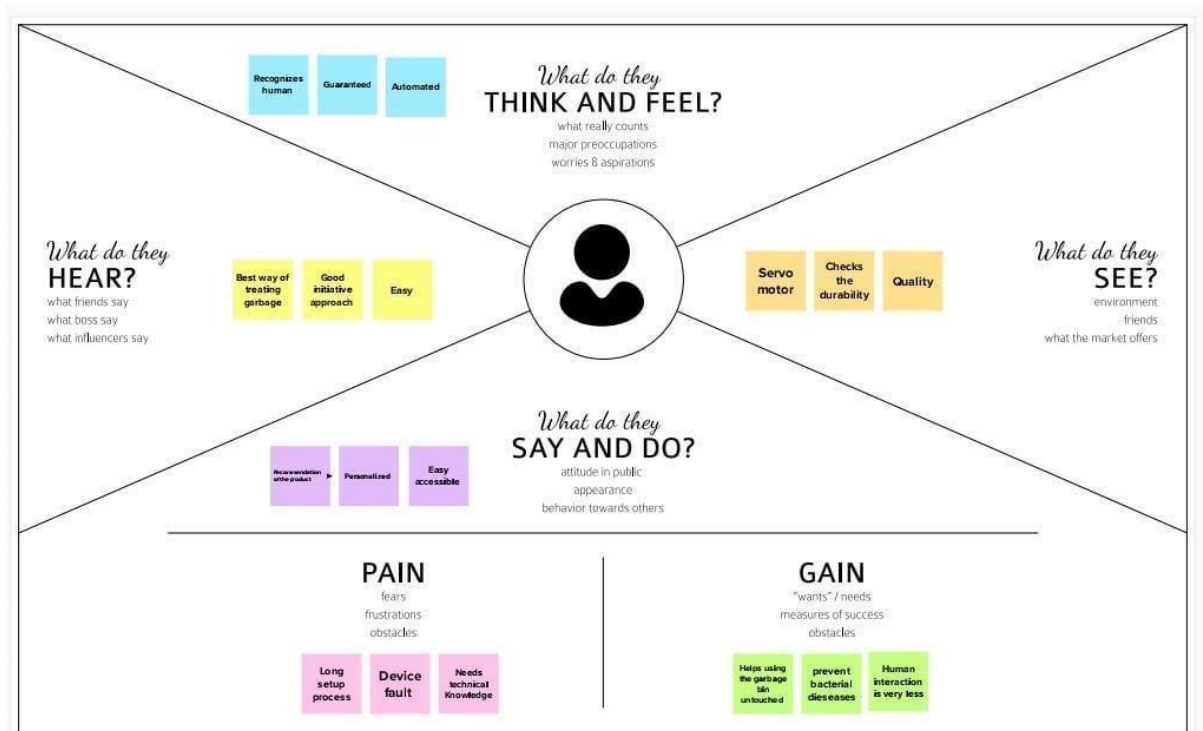
CHAPTER 3: IDEATION & PROPOSED

SOLUTION 3.1 Empathy Map Canvas Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.


It is a useful tool to help teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.






3.2 Ideation & Brainstorming


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




Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

 10 minutes to prepare
 1 hour to collaborate
 2-8 people recommended

 **Before you collaborate**

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

 10 minutes

A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.


B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →


1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

 5 minutes







PROBLEM

How might we [your problem statement]?



Key rules of brainstorming

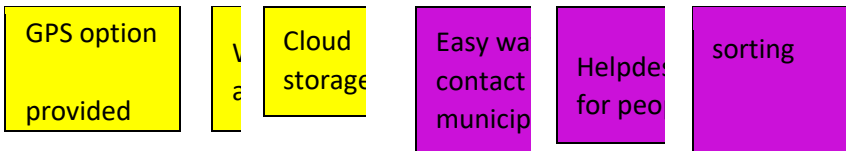
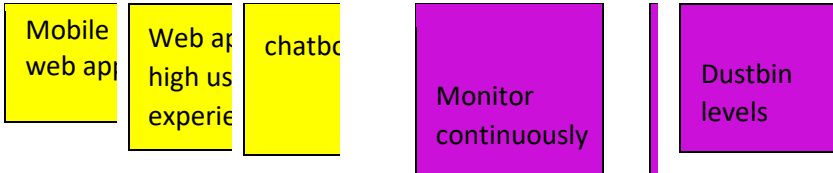
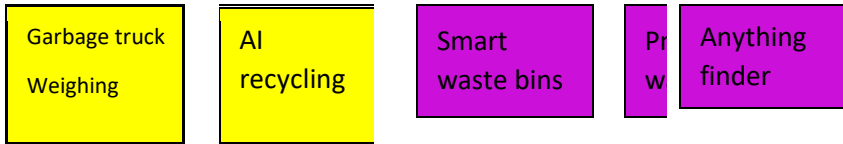
To run a smooth and productive session

 Stay in topic.	 Encourage wild ideas.
 Defer judgment.	 Listen to others.
 Go for volume.	 If possible, be visual.

Step-2: Brainstorm, Idea Listing and Groupi [Grab your reader's attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

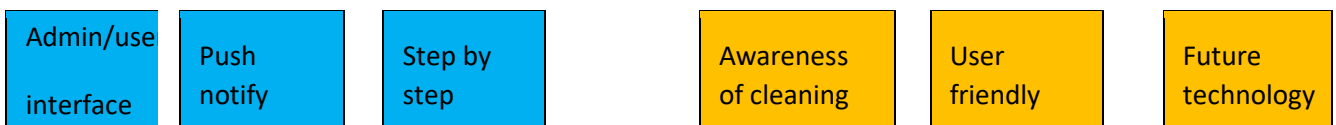
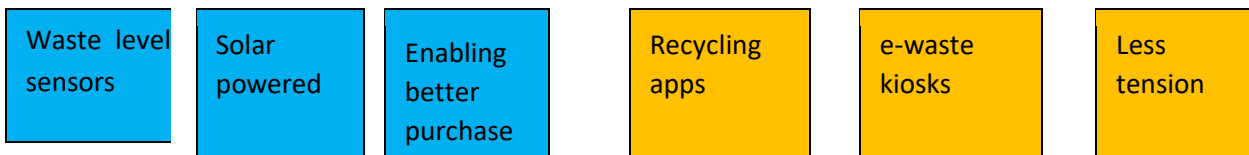
DHILEEPAN A

BENTYSON J



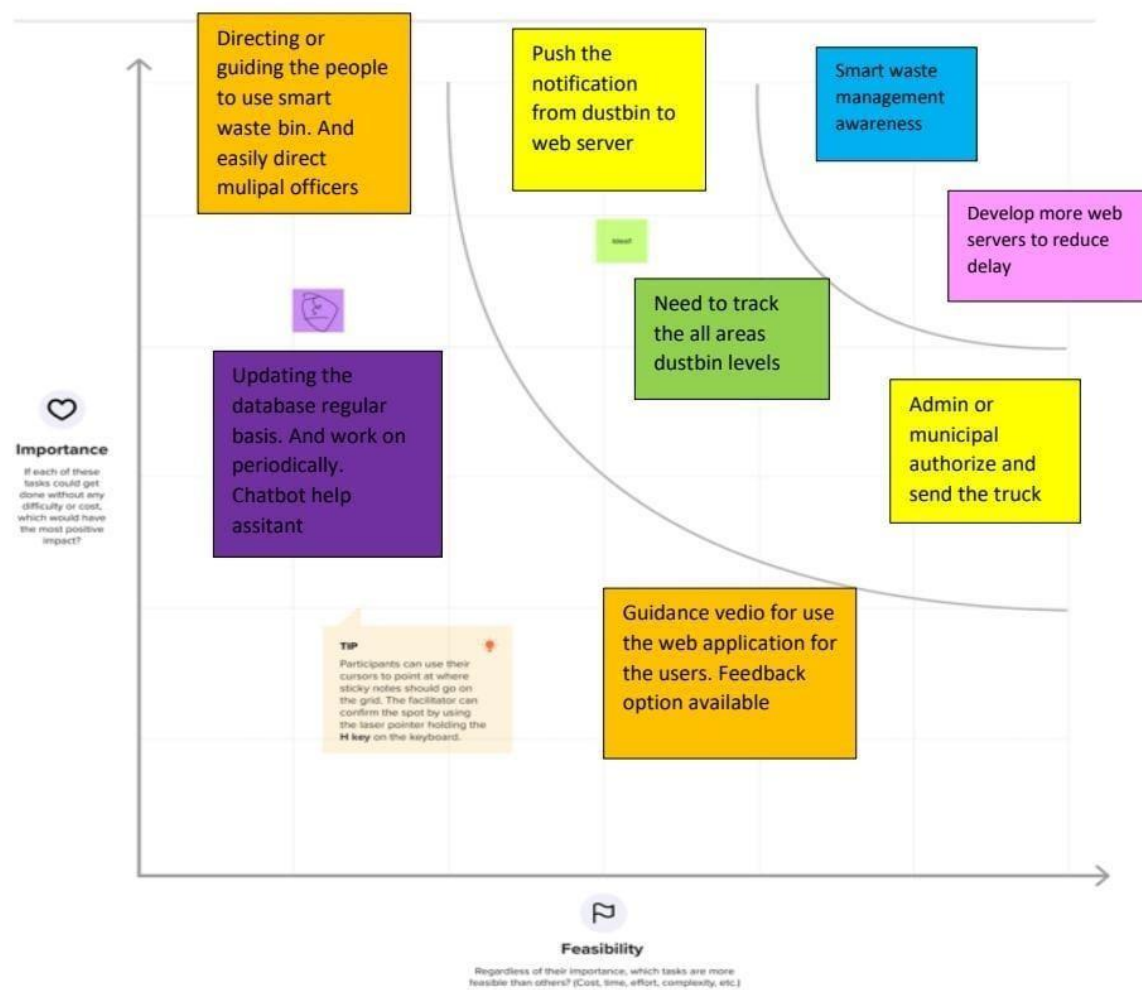
Aineesh Golda A

Akash T





Step-3: Idea Prioritization



3.3 Proposed Solution

S. No	Parameter	Description
1	Problem Statement (Problem to be solved)	The purpose of this project is to reduce the waste overflow from the dumpsters, Recycling with separation of waste and enhance the best route for collecting waste using IOT.
2	Idea / Solution description	If the waste in the dumpsters reach 3/4th of its level the sensors senses it and transmits data to the garbage trucks to initiate garbage collection.
3	Novelty / Uniqueness	The system will turn the alert on, automatically reaches the garbage collector trucks, and enhance the best route preventing dump overflow.
4	Social Impact / Customer Satisfaction	The sensor-enabled solution helps prevent the dump overflow and increases environmental air quality. The IR sensors help detect the level of the dump filled to avoid the waste overflow.
5	Business Model (Revenue Model)	Waste management is the most essential need in our daily life, it reduces man power and utilize machines of IOT in a useful manner.

6	Scalability of the Solution	The result of this project is that the sensor detects the level of the waste collected in the dumpsters, before the overflow starts the garbage trucks are meant to collect it. This is an efficient method as the trucks automatically segregates the different wastes so that it can be dumped directly to its corresponding places, ensuring less man power, increased air quality and an efficient waste management.
---	-----------------------------	--

3.4 Problem Solution fit

Define CS, fit into CL	1. CUSTOMER SEGMENT(S) CS The industrialists who promote recycling waste from the large dump yards	6. CUSTOMER LIMITATIONS <small>EG. BUDGET, DEVICES</small> CL High budget in installing other products make them to move far from modern technologies.	5. AVAILABLE SOLUTIONS <small>PLUSES & MINUSES</small> AS The monitoring and controlling of the leakage could be done by the sensors. Hence the manpower is reduced. Thus the environmental pollution is decreased and as well as the workers health is also increased.	Explore AS, differentiate
	2. PROBLEMS / PAINS <small>+ ITS FREQUENCY</small> PR <ul style="list-style-type: none">Decrease in air quality due to the unawareness of proper waste management.Having no proper system for controlling or monitoring the waste overflow from garbage dumpsters.Facing budget problems for installing the modern technology for waste management	9. PROBLEM ROOT / CAUSE RC When the sensors failed to monitor properly, the waste can overflow from the garbage dumpster.	7. BEHAVIOR <small>+ ITS INTENSITY</small> BE The manpower is reduced making away from direct contact with the waste as it is machine processed.	
Focus on PR, tap into BE, understand RC	3. TRIGGERS TO ACT TR The heavy damages or higher health issues due to the direct contact of waste by human urges them to find out a solution as soon as they could possible.	10. YOUR SOLUTION SL Develop an efficient system & an application that can monitor and alert the workers.	8. CHANNELS of BEHAVIOR CH Promoting through social media. With the help of social media entrepreneurs/influencer.	Extract online & offline CH of BE
	4. EMOTIONS <small>BEFORE / AFTER</small> EM Before: The heavy leakage due to the overflow made us feel of guilt due to the workers were in direct contact with the waste. After: Increased the level of confidence and feel secured and healthy.		OFFLINE Through newspaper advertisements.	
Identify strong TR & EM				

CHAPTER 4: REQUIRMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR.NO	Functional Requirements	Sub Requirement (Story/Sub-Task)
FR-1	User registration	Registration through e-mail id & Mobile number
FR-2	User confirmation	Confirmation via email Confirmation via OTP
FR-3	Web application	Node-Red -Service
FR-4	Configure to Device	IBM Watson IOT Platform
FR-5	Database	Detailed database of bins and stands
FR-6	Python Script	IBM IOT Platform

4.2 Non Functional Requirements

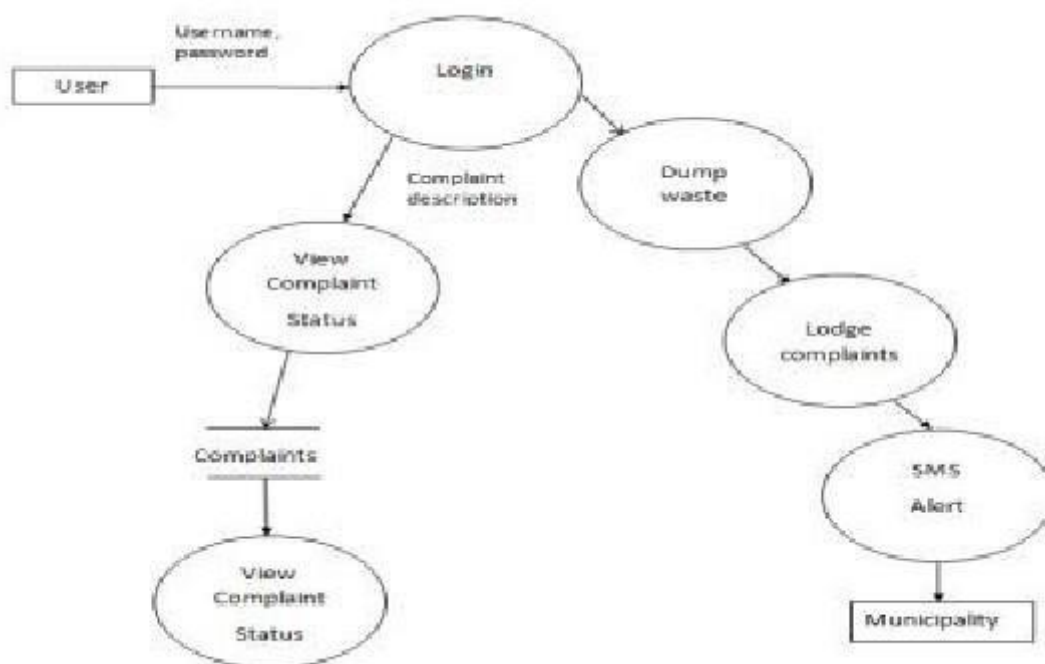
Following are the non-functional requirements of the proposed solution

FR.NO	Non-Functional Requirement	Description
NFR-1	Usability	The reduction of waste
NFR-2	Security	Prediction in bin fulness
NFR-3	Reliability	Effective waste disposal
NFR-4	Performance	Optimize source allocation, reduce running costs and increase sustainability of waste services
NFR-5	Availability	Available for the allocated time by the municipality or the private companies
NFR-6	Scalability	This is very effective in managing waste in big city. Here priority system is used to clean the city all the time without any overflowing dumpsters

CHAPTER 5: PROJECT DESIGN

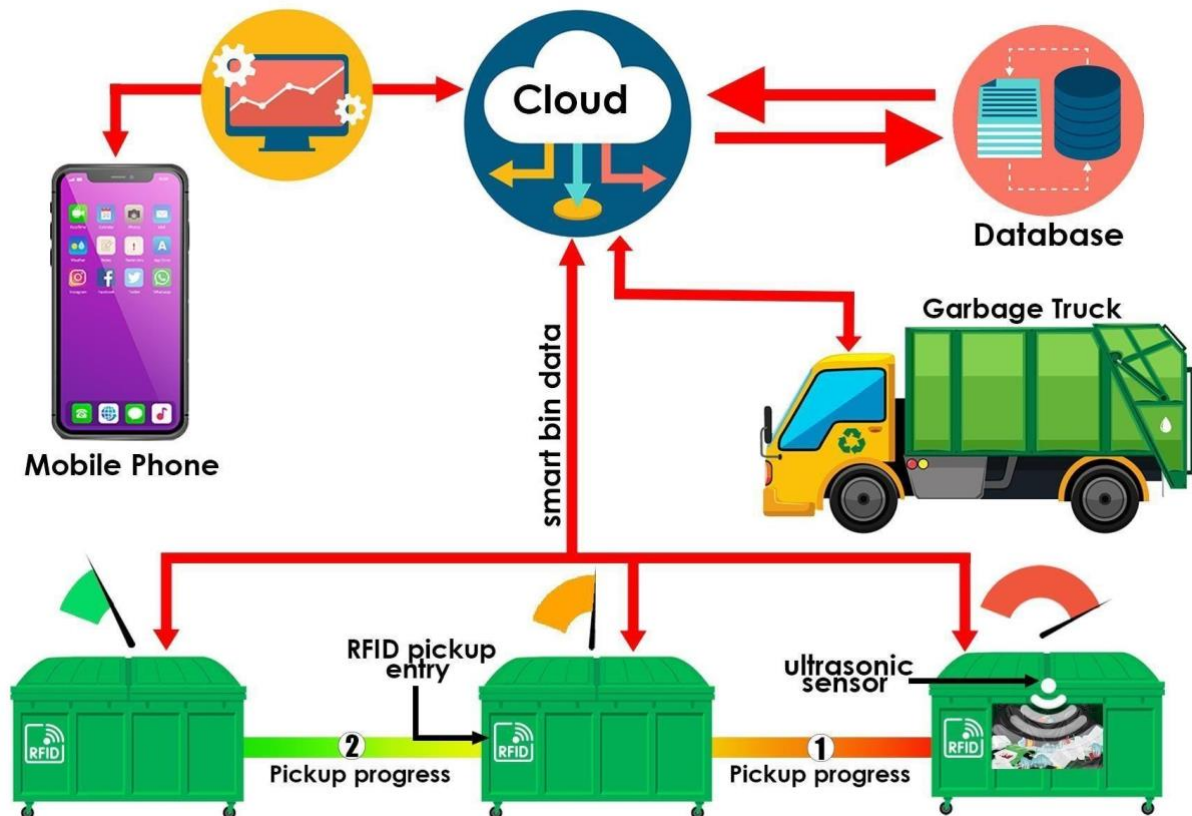
5.1 Data Flow Diagram:

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.

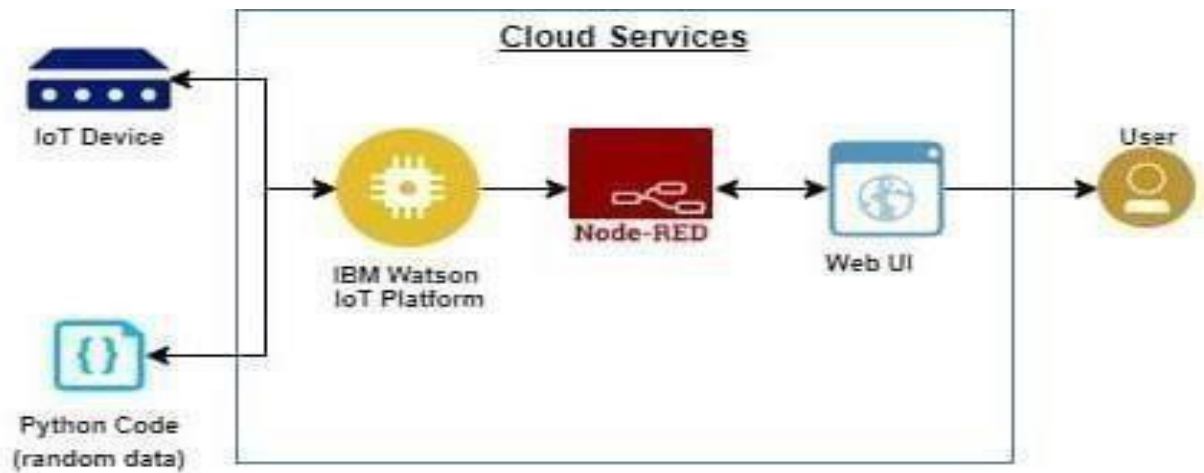


5.2 Solution and Technical Architecture

- Ultrasonic Sensor detects the garbage level of every Bin.
- RFID Sensor takes the entry of each garbage pickups by the employee.
- The pickup progress can be tracked and intimated to users and trash collectors.



Technical Architecture:



The Deliverable shall include the architectural diagram as below and the information as the table 1&table 2

Table-1: Components & Technologies:

S.NO	Components	Description	Technology
1.	User interface	User interact with application using form, login, request notification	Python/HTML/MYSQL/JAVA
2.	Registration	User register in the application to connect bank account	Python/HTML/MYSQL/JAVA
3.	Verification	Verification in the application to connect bank account	Python/HTML/MYSQL/JAVA
4.	Sensor (IOT device)	A device that responds to a physical stimulus (such as heat, light, sound, pressure, magnetism, or a particular motion) and transmit a resulting impulse.	Raspberry pi/Arduino UNO/Temperature sensor/ultrasonic sensor
5.	Sends notification	Sends the notification to the cloud database	IBM Cloud
6.	Cloud Database (Node Red)	Database service on cloud	Node Red
7.	Application	A computer software package that performs a specific function directly for an end user	IBM Watson STT service

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
------	-----------------	-------------	------------

1.	Open-Source Framework	Open source is a term denoting that a product includes permission to use its source code, design documents or content.	Bootstrap
2.	Scalable Architecture	It connected with scalable architecture	IBM Watson
3.	Availability	This application access is available at the work time of the workers according to their corporation or municipality.	Python
4.	Performance	Record resource requests and save registered information. Availability of application.	IBM Watson

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Admin (who manages web server)	Login	USN-1	As an admin, I gave user ID and password for every worker and manage them.	I can manage my account / dashboard	High	Sprint-1
Co admin	Login	USN-2	As a co admin, I will manage garbage level monitor. If garbage get filled I will send alert and will post location and garbage ID to trash truck.	I can manage monitoring	High	Sprint-1
Truck driver	Login	USN-3	As a Truck driver, I will follow the route sent by co admin to reach the filled garbage.	I can drive to reach the garbage filled route in shortest route given	Medium	Sprint-2
Local garbage collector	Login	USN-4	As a Waste collector, I will collect all the trash from garbage and load into garbage truck and send them to landfill.	I can collect trash and load it to truck and send off	Medium	Sprint-2
Municipality	Login	USN-5	As a Municipality, I will check the process if they are happening in disciplined manner without any issues.	I can manage all these process if going good	High	Sprint-1

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 a Administrator, I need to give user id and passcode forever workers over there in municipality	10	High	Bentyson J
Sprint-1	Login	USN-2	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	Aineesh Golda A
Sprint-2	Dashboard	USN-3	As a Truck Driver, I'll follow Co Admin's Instruction to reach the filling bin in short roots and save time	20	High	Dhileepan A
Sprint-1	Dashboard	USN-4	As a Local Garbage Collector, I'll gather all the waste from the garbage, load it onto a garbage truck, and deliver it to Landfills	20	High	Akash T
Sprint-1	Dashboard	USN-5	As a Municipality officer, I'll make sure everything is proceeding as planned and without any problems	20	High	Bentyson J & Aineesh Golda A

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	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 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	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

6.3 Reports from JIRA:

Jira Software Your work Projects Filters Dashboards People Apps Create

Does your team need more from Jira? Get a free trial of our Standard plan.

Smart Waste Manage... Software project

PLANNING

- Roadmap
- Board

DEVELOPMENT

- Code
- Project pages
- Add shortcut
- Project settings

You're in a team-managed project. Learn more

Projects / Smart Waste Management System

SWMS board

TO DO

+ Create issue

IN PROGRESS

DONE 14 ISSUES

- create the web app html part
SPRINT 1
SWMS-2
- adding js part for live location
SPRINT 1
SWMS-3
- getting the live location
SPRINT 1
SWMS-4

Python sprint - weight of the bin

GROUP BY: None

Quickstart

Jira Software Your work Projects Filters Dashboards People Apps Create

Smart Waste Manage... Software project

PLANNING

- Roadmap
- Board

DEVELOPMENT

- Code
- Project pages
- Add shortcut
- Project settings

Projects / Smart Waste Management System

Roadmap

RB

Status category Epic

		NOV
SWMS-1 Sprint 1	DONE	
SWMS-2 create the web app ...	DONE	
SWMS-3 adding js part for li...	DONE	
SWMS-4 getting the live loca...	DONE	
SWMS-5 Sprint 2	DONE	
SWMS-6 Sprint 3	DONE	
SWMS-7 Sprint 4	DONE	
SWMS-19 reports	DONE	

 **Smart Waste Manage...**
Software project

Projects / Smart Waste Management System

Roadmap

RB





Status category ▾

Epic ▾

▼ **PLANNING**

 Roadmap

 Board

DEVELOPMENT

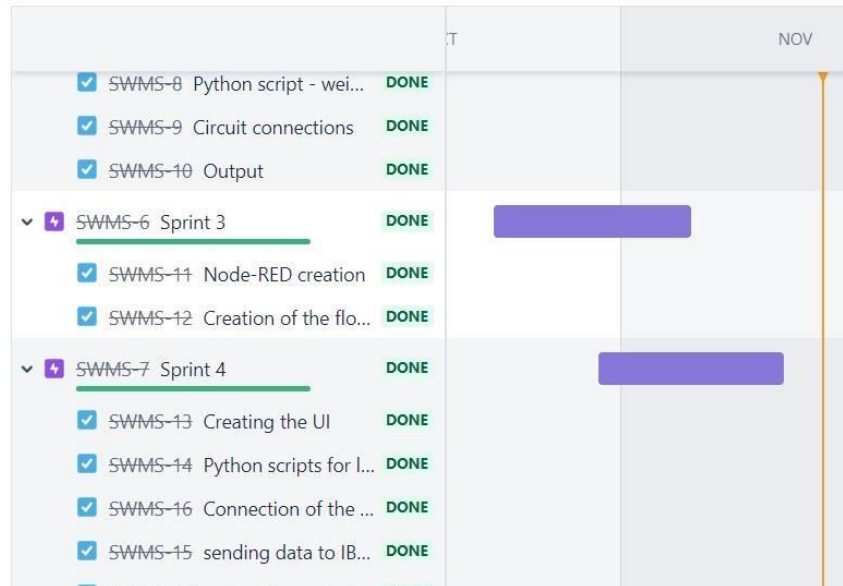
 Code

 Project pages

 Add shortcut

 Project settings

You're in a team-managed project



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>Smart Waste Management System</title>

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

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

<script src="https://www.gstatic.com/firebasejs/9.14.0/firebase-app.js"></script>

<script src="https://www.gstatic.com/firebasejs/9.14.0/firebase-analytics.js"></script>

<script src="https://www.gstatic.com/firebasejs/9.14.0/firebase-database.js"></script>

<script>    var

firebaseConfig =

{

    apiKey: "AlzaSyCcZk7b1CLOGviwUpthRDLotrmFX0MFuTs",

    authDomain: "swms-3840.firebaseio.com",

    projectId: "swms-3840",
```



```

        storageBucket: "swms-3840.appspot.com",
        messagingSenderId: "479902726304",

        appId: "1:479902726304:web:3d822880d1275ee57a71c5",
        measurementId: "G-MHP4N77MTP"

    };

    firebase.initializeApp(firebaseConfig)
</script>

<script defer src="db.js"></script>
</head>

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

    <script src="maps.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=AlzaSyBBLyWj3FWtCbCXGW3ysEil2f
Dfrv2v0Q&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);
        }
    );
}

```

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
random            variables
(temperature&humidity)
def
myCommandCallback(cm):
global a

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

```

```

print(control)

try:
    deviceOptions={"org": organization, "type": devicType,"id":
        deviceId,"authmethod":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 = "73ffv" devicType
=          "BIN2" deviceId
=          "BIN2ID" authMethod=
"token" authToken=
"123456789"

```

```

#generate random values for random 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,"authmethod":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 random 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,"authmethod":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 = "73ffv"
devicType = "BIN4"
devicId = "BIN4ID"
authMethod = "token"
authToken = "123456789"

#generate random values for random 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":
        devicId,"authmethod":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)
        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

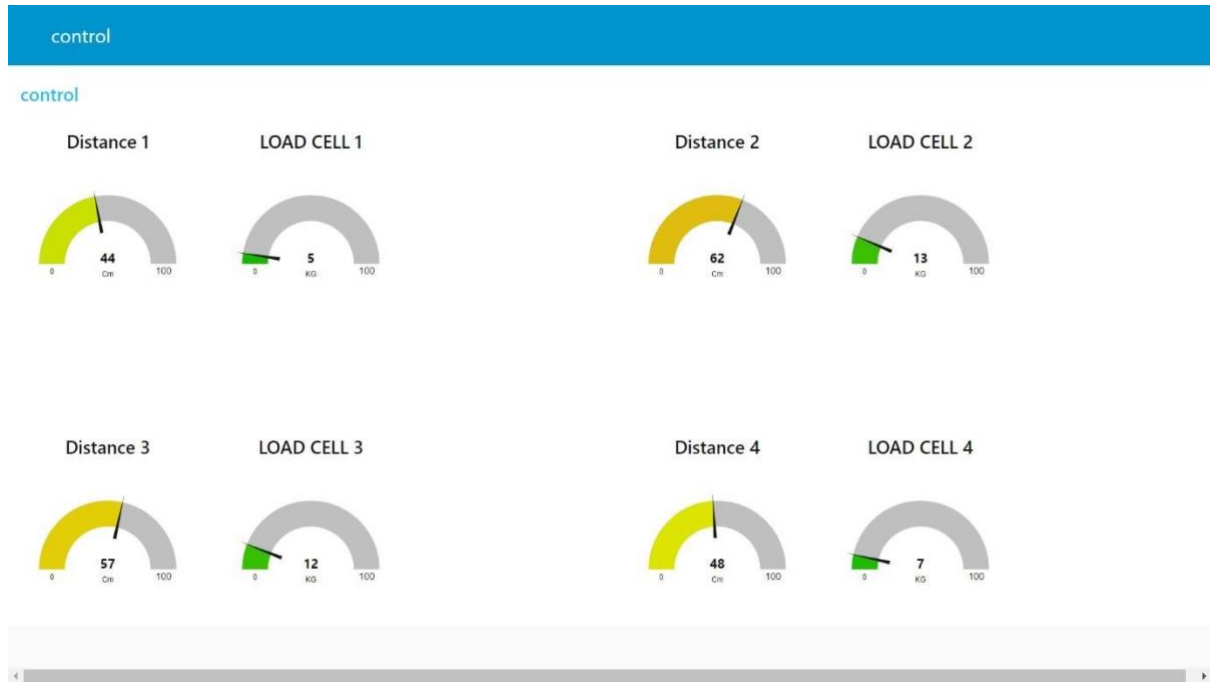
Test case no.	Sensor/Stage	Input	Expected output	Obtained output	Status
1.	Ultrasonic	Garbage level in bin i) Null ii) Full iii) Range in %	Correct level or distance	As expected	Pass
2.	ESP – 32	Microcontroller to process the input data	To collect the data from sensor	As expected	Pass
3.	Load cell	To 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

Sample output:



Before implementation of Smart waste management system

		Ground truth	
		Emptying	Nonemptying
Predicted	Emptying	911	68
	Nonemptying	990	6041

After implementation of smart waste management system

		Ground truth	
		Emptying	Nonemptying
Predicted	Emptying	1720	90
	Nonemptying	181	6091

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 Biodegradable 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>Smart Waste Management System</title>

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

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

  <script src="https://www.gstatic.com/firebasejs/9.14.0/firebase-app.js"></script>

  <script src="https://www.gstatic.com/firebasejs/9.14.0/firebase-analytics.js"></script>
<script src="https://www.gstatic.com/firebasejs/9.14.0/firebase-database.js"></script>

  <script>    var
firebaseConfig =
  {
    apiKey: "AlzaSyCcZk7b1CLOGviwUpthRDLotrmFX0MFuTs",
    authDomain: "swms-3840.firebaseio.com",    projectId:
"swms-3840",    storageBucket: "swms-
3840.appspot.com",    messagingSenderId: "479902726304",    appId:
"1:479902726304:web:3d822880d1275ee57a71c5",    measurementId:
"G-MHP4N77MTP"
  };
```

```

        firebase.initializeApp(firebaseConfig)
    </script>
    <script defer src="db.js"></script>
</head>

<body style="background-color:#1F1B24;">
    <script src="maps.js"></script>
    <div id="map_container">
        <h1 id="live_location_heading" >LIVELOCATION</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 btndark">
            DUMPSTER
        </a>
    </center>
    <script

src="https://maps.googleapis.com/maps/api/js?key=AlzaSyBBLyWj3FWtCbCXGW3ysEil2fDfrv2
v0Q&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);

    }

);

}

```

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;
flexdirection:
column;  vertical-
align: textbottom;

}.logo { width:
70%; margin:
5% 15%;

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

#logo_3 { vertical-align:
text-bottom;

}

```

```

#data_container { height:
    100%; width: 20%;
    margin-
left: 1%; marginright:
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("/imgs/KG.png");
backgroundrepeat: no-repeat;
background-size: 170px; background-
position: left center;

} #cap_status { background-image:
url("/imgs/dust.png");
backgroundrepeat: 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;
flexdirection:
column; }

#live_location_heading { margin-top:
10%; text-align:
center; color: GREY;
}

#map { height: 70%;
width:      90%;
marginleft:  4%;
marginright: 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;
fontsize:
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 random 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,"authmethod":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 myOnPublishCallback) ("IoTSensor","json",warn,qos=0,on_publish=

success=deviceCli.publishEvent myOnPublishCallback) ("IoTSensor","json",data,qos=0,on_publish=

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
= "73ffv" devicType =
"BIN2" deviceId = "BIN2ID"
authMethod=
"token" authToken=
"123456789"

#generate random values for random 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": deviceType,"id":
deviceId,"authmethod":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 myOnPublishCallback) ("IoTSensor","json",warn,qos=0,on_publish=
success=deviceCli.publishEvent myOnPublishCallback) ("IoTSensor","json",data,qos=0,on_publish=

    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      =
"73ffv" devicType =
"BIN3" deviceId =

```

```

"BIN3ID"
authMethod=
"token" authToken=
"123456789"

#generate random values for random 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,"authmethod":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 myOnPublishCallback) ("IoTSensor","json",warn,qos=0,on_publish=

success=deviceCli.publishEvent myOnPublishCallback) ("IoTSensor","json",data,qos=0,on_publish=

    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,"authmethod":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 myOnPublishCallback) ("IoTSensor","json",warn,qos=0,on_publish=
success=deviceCli.publishEvent myOnPublishCallback) ("IoTSensor","json",data,qos=0,on_publish=

```

```
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:

Git Ripo Link: <https://github.com/IBM-EPBL/IBM-Project-1010-1658334748>
Demo link: https://youtu.be/n_XVwK6HOXw

