SMART WASTE MANAGEMENT SYSTEM ON METROPOLITAN CITIES

NALAIYATHIRAN PROJECT BASED LEARNING
ON

PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP A PROJECT REPORT

SUBMITTED BY

Team ID : PNT2022TMID34601
Team Leader : ASHA A (962219104022)
Team member : ANNIE ASHIMA A (962219104017)
Team member : BENCY W (962219104034)
Team member : EULI HESHI C (962219104051)

CONETNT

1. INTRODUCTION

1.1 Project Overview:

With the increasing population and industrialization of nations throughout the globe, waste has become a great concern for all of us. Over years, researchers figured that only waste management is not enough for its proper treatment and disposal techniques to preserve our environment and keeping it clean in this era of globalization. With the help of technology researchers have, introduced IoT based Smart Waste Management solutions and initiatives that ensures reduced amount of time and energy required to provide waste management services and reduce the amount of waste generated. Unfortunately, developing countries are not being able to implement those existing solutions due to many factors like socio-economic environment. Therefore, in this research we have concentrated our thought on developing a smart IoT based waste management system for developing countries like INDIA that will ensure proper disposal, collection, transportation and recycling of household waste with the minimum amount of resources being available.

1.2 Purpose:

We amalgamate technology along with waste management in order to effectively create a safe and a hygienic environment. Smart waste management is about using technology and data to create a more efficient waste industry. Based on IoT (Internet of Things) technology, smart waste management aims to optimize resource allocation, reduce running costs, and increase the sustainability of waste services. This makes it possible to plan more efficient routes for the trash collectors who empty the bins, but also lowers the chance of any bin being full for over a week. A good level of coordination exists between the garbage collectors and the information supplied via technology. This makes them well aware of the existing garbage level and instigate them whenever the bins reach the threshold level. They are sent with alert messages so that they can collect the garbage on time without littering the surrounding area. The fill patterns of specific containers can be identified by historical data and managed accordingly in the long term. In addition to hardware solutions, mobile applications are used to overcome the challenges in the regular waste management system, such as keeping track of the drivers while they are operating on the field. Thus, smart waste management provides us with the most optimal way of managing the waste in an efficient manner using technology.

2. LITERATURE SURVEY:

2.1 Existing problem:

Waste management has become an alarming challenge in local towns and cities across the world. Often the local area bins are overflowing and the municipalities are not aware of it. This affects the residents of that particular area in numerous ways starting from bad odour to unhygienic and unsafe surroundings. Poor waste management - ranging from non-existing collection systems to ineffective disposal -causes air pollution, water and soil contamination. Open and unsanitary areas contribute to contamination of drinking water and can cause infection and transmit diseases. Toxic components such as Persistent Organic Pollutants (POPs) pose particularly significant risks to human health and the environment as they accumulate through the food chain. Animals eating contaminated plants have higher doses of contaminants than if they were directly exposed. Precipitation or surface water seeping through waste will absorb hazardous components from landfills, agricultural areas, feedlots, etc. and carry them into surface and groundwater. Contaminated groundwater also poses a great health risk, as it is often used for drinking, bathing and recreation, as well as in agricultural and industrial activities. Landfills and waste transfer stations can attract various pests (insects, rodents, gulls, etc.) that look for food from waste. These pests can spread diseases through viruses and bacteria (i.e., salmonella and e coli), which are a risk to human health.

2.2 References:

PAPER 1:

TITLE: IoT Based Waste Management for Smart City

AUTHOR NAME: Parkash Tambare, Prabu Venkatachalam

PUBLICATION YEAR: 2016

DESCRIPTION:

In the current situation, we frequently observe that the trash cans or dust cans that are located in public spaces in cities are overflowing due to an increase in the amount of waste produced each day. We are planning to construct "IoT Based Waste Management for Smart Cities" to prevent this from happening because it makes living conditions for people unsanitary and causes unpleasant odours in the surrounding area. There are numerous trash cans scattered throughout the city or on the campus that are part of the proposed system. Each trash can is equipped with a low-cost embedded device that tracks the level of the trash cans and an individual ID that will enable it to be tracked and identified.

PAPER 2:

AUTHOR NAME: Mohammad Aazam, Marc St-Hilaire, Chung-Horng Lung, Ioannis Lambadaris

PUBLICATION YEAR: 2016

DESCRIPTION:

Each bin in the Cloud SWAM system that Mohammad Aazam et al suggested has sensors that can detect the amount of waste inside. There are separate bins for organic, plastic/paper/bottle/glass, and metal waste. This way, each form of waste is already divided, and it is known how much and what kind of waste is collected thanks to the status. Different entities and stakeholders may benefit from the accessibility of cloud-stored data in different ways. Analysis and planning can begin as soon as garbage is collected and continue through recycling and import/export-related activities. Timely garbage collection is provided via the Cloud SWAM system. A timely and effective method of waste collection improves health, hygiene, and disposal.

PAPER 3:

TITLE: Arduino Microcontroller Based Smart Dustbins for Smart Cities

AUTHOR NAME: K. Suresh, S. Bhuvanesh and B. Krishna Devan

PUBLICATION YEAR: 2019

DESCRIPTION:

In this paper, a technique for cleaning up our surroundings and environment is described. The Indian government just began work on a smart city initiative, and in order for these towns to be smarter than they already are, the garbage collection and disposal system must be improved upon. Self-Monitoring Automated Route Trash (SMART) dustbins are intended for use in smart buildings such as colleges, hospitals, and bus stops, among other places. In this study, we have employed the PIR and Ultrasonic sensors to detect human presence, the Servomotor to open the dustbin lid, and the Ultrasonic sensor to detect the level of rubbish. Signals between two trash cans are transmitted using a communication module, and the GSM module sends the message to the operator.

PAPER 4:

AUTHOR NAME: Mohd Helmy Abd Wahab, Aeslina Abdul Kadir, Mohd Razali Tomari and Mohamad Hairol Jabbar

PUBLICATION YEAR: 2014

DESCRIPTION:

Proposed a smart recycle bin that can handle the recycling of plastic, glass, paper, and aluminium cans. It generates a 3R card after automatically determining the value of the trash thrown away. The recycle system makes it possible to accumulate points for placing waste into designated recycle bins. By allowing the points to be redeemed for goods or services, such a system promotes recycling activities. The system keeps track of information on disposal procedures, materials disposed of, user identification, and points accrued by the user.

PAPER 5:

TITLE: Waste Management Initiatives in India For Human Wellbeing

AUTHOR NAME: Dr. Raveesh Agarwal, Mona Chaudhary and Jayveer Singh

PUBLICATION YEAR: 2015

DESCRIPTION:

The objective of this paper is to examine the present methods used in India for the welfare of its people in different waste management efforts. The other goal is to offer advice on how to make Indian municipalities' trash disposal procedures better. On secondary research, this essay is founded. The system is improved by looking at the reports that have already been written about waste management and the suggestions made for improvement by planners, NGOs, consultants, government accountability organisations, and important business leaders. It provides in-depth understanding of the various waste management programmes in India and identifies areas where waste management might be improved for societal benefit. The essay makes an effort to comprehend the crucial part that our nation's official waste management sector plays in the waste management process.

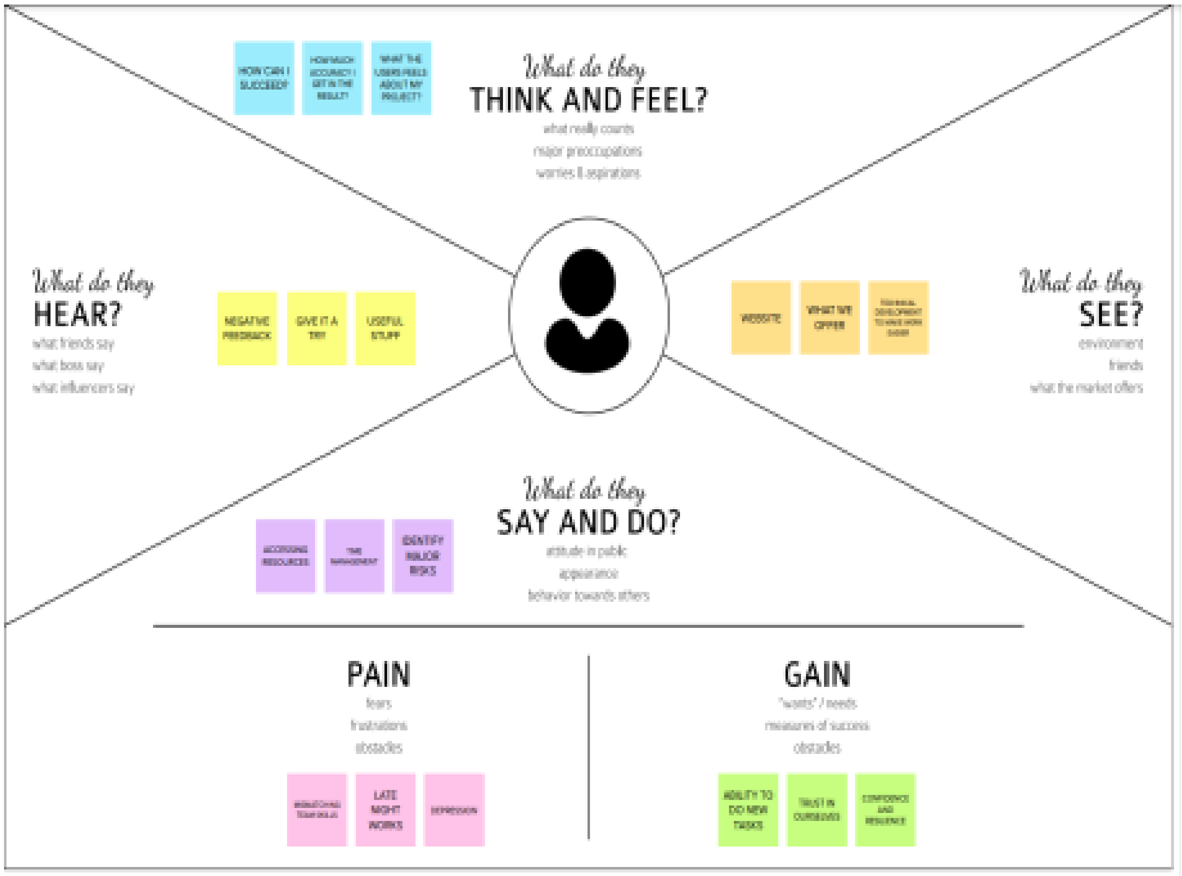
2.3 Problem Statement Definition:

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel

PS-1	Municipal corporation authority	Get notified when the trash cans are full and be made aware of where the full cans are located.	Don't have the facilities at the moment	There is no tool available to determine the level of bins.	Frustrated
PS-2	Individual working for a private limited corporation	Get rid of the example of a surplus of waste	The trash cans are always filled	I occupy a metropolitan where there is activity is invariably crowd.	Worried

3. IDEATION PHASE

3.1 Empathy Map Canvas



3.2 Proposed Solution

SI.No	Parameter	Description

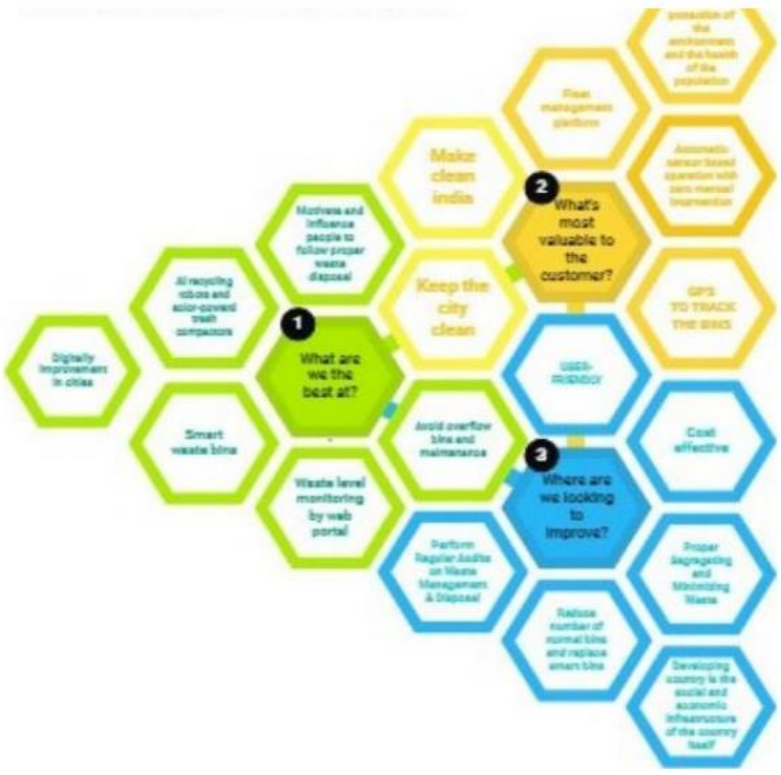
1.	Problem Statement (Problem to be solved)	Proposed Solution means the combination of software, hardware, other products or equipment, and any and all services (including any installation, implementation, training, maintenance and support services) necessary to implement the solution described by Vendor in its Proposal. Smart City technology evolved together with the developments in wireless sensor networks (WSN) and the Internet of Things (IOT). Smart cities essentially combine the use of ICT to provide services for better living conditions inside cities. It is a diverse topic of discussion with several application areas .
2.	Idea / Solution description	The current process of waste management starts with the waste being created by people in the cities and disposed in trash bins near its creation point. The disposed trash is collected by municipality or private company trucks at the predefined times and transferred to temporary collection centers. The trash at the collection centers is then sent for recycling.
3.	Novelty / Uniqueness	Smart waste management companies have recently developed solutions based on ultrasonic distance measurement. Some companies prefer to approach the problem with an alternative solution using image processing and camera as a passive sensor. However, the majority of these solutions use ultrasonic sensor for measurement of the distance.

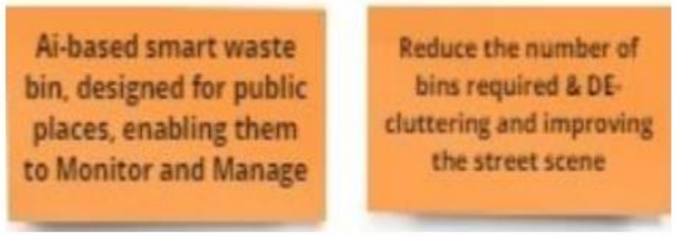
4.	Social Impact / Customer Satisfaction	Recycling of used products could be simpler and cheaper through the establishment of a system for automated management of product lifecycle, whereby, the technical information about the product must be incorporated in itself during production.

5.	Business Model (Revenue Model)	Product management at the end of life-cycle could become widespread thanks to the technological improvements in the identification and the use of the Internet. There are several innovations for the automatic identification of products, among which RFID tags have shown great potential in applications such as the management of the products [7]. Waste management is another promising application area of RFID technology.
6.	Scalability of the Solution	In the system advocated above, the fusion of sensors, identification technology, and internet connectivity will lead to a uniquely smart disposal trash bin. Together with the cloud, these trash bins would become irreplaceable elements in the waste management cycle where the collection, transportation, storage, and recycling of waste could be automated. The use of RFID technology in waste collection services not only increases the efficiency of waste management through automation but also increases environmental responsibility which is one of the pillars of the Smart City.

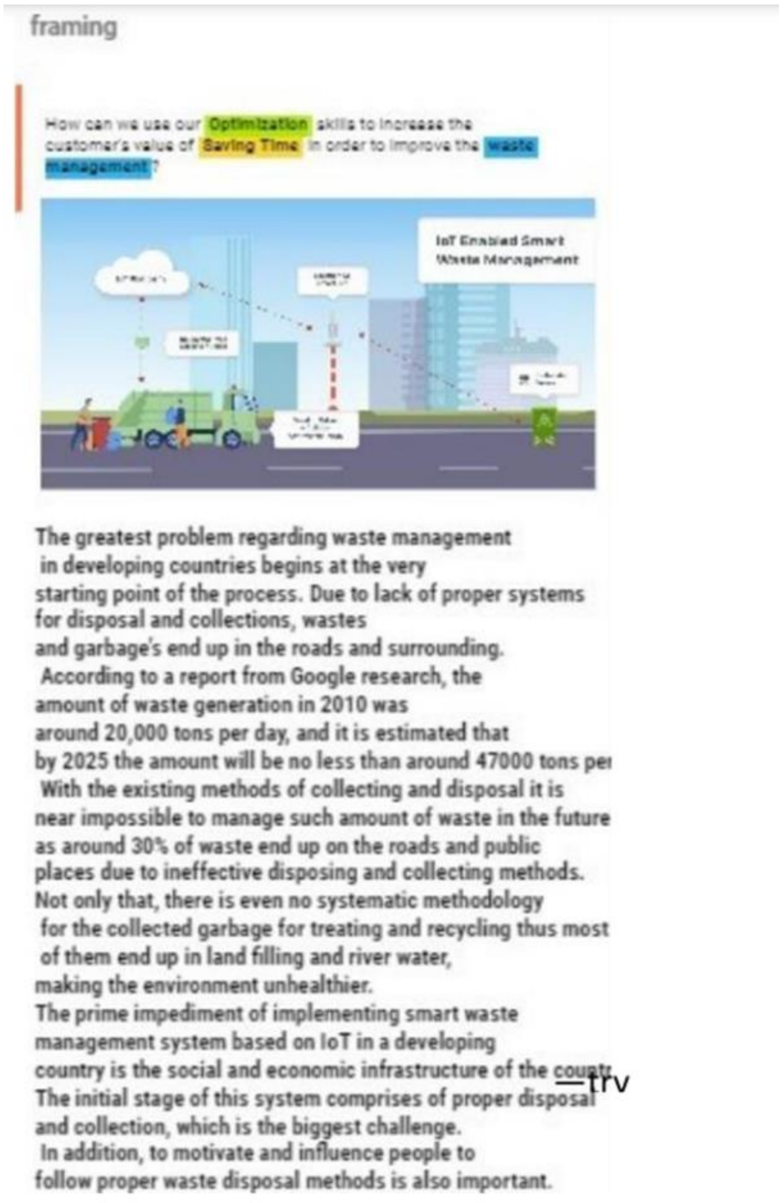
3.3 Problem Solution Fit

PNT2022TMID34601


Digitally improvement in cities
AI recycling robots and solar-powered trash compactors
Smart waste bins
Motivate and influence people to follow proper waste disposal
1
What are we the best at?
Waste level monitoring by web portal
Make clean india
Keep the city clean
Avoid overflow bins and maintenance
Perform Regular Audits on Waste Management & Disposal
Fleet management platform
2
What's most valuable to the customer?
USER-FRIENDLY
3
Where are we looking to improve?
Reduce number of normal bins and replace smart bins
protection of the environment and the health of the population
Automatic sensor based operation with zero manual intervention
GPS TO TRACK THE BINS
Cost effective
Proper Segregating and Minimizing Waste
Developing country is the social and economic infrastructure of the country itself

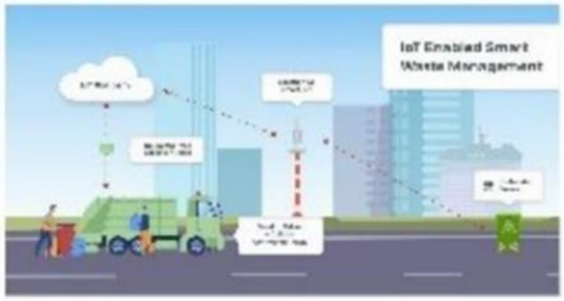

Ai-based smart waste bin, designed for public places, enabling them to Monitor and Manage

Reduce the number of bins required & DE-cluttering and improving the street scene



framing

How can we use our Optimization skills to increase the customer's value of Saving Time in order to improve the waste management?



The greatest problem regarding waste management in developing countries begins at the very starting point of the process. Due to lack of proper systems for disposal and collections, wastes and garbage's end up in the roads and surrounding. According to a report from Google research, the amount of waste generation in 2010 was around 20,000 tons per day, and it is estimated that by 2025 the amount will be no less than around 47000 tons per With the existing methods of collecting and disposal it is near impossible to manage such amount of waste in the future as around 30% of waste end up on the roads and public places due to ineffective disposing and collecting methods. Not only that, there is even no systematic methodology for the collected garbage for treating and recycling thus most of them end up in land filling and river water, making the environment unhealthier.
The prime impediment of implementing smart waste management system based on IoT in a developing country is the social and economic infrastructure of the countr —trv
The initial stage of this system comprises of proper disposal and collection, which is the biggest challenge.
In addition, to motivate and influence people to follow proper waste disposal methods is also important.

4.REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Detailed bin inventory.	All monitored bins and stands can be seen on the map, and you can visit them at any time via the Street View feature from Google. Bins or stands are visible on the map as green, orange or red circles. You can see bin details in the Dashboard – capacity, waste type, last measurement, GPS location and collection schedule or pick recognition.
FR-2	Real time bin monitoring.	The Dashboard displays real-time data on fill-levels of bins monitored by smart sensors. In addition to the % of fill-level, based on the historical 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 last collected. With real-time data and predictions, you can eliminate the overflowing bins and stop collecting half-empty ones.
FR-3	Expensive bins.	We help you identify bins that drive 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 (1-10) and calculates distance from depo-bin discharge.

FR-4	Adjust bin distribution.	Ensure the most optimal distribution of bins. Identify areas with either dense or sparse bin distribution. Make sure all trash types are represented within a stand. Based on the historical data, you can adjust bin capacity or location where necessary.
FR-5	Eliminate inefficient picks.	Eliminate the collection of half-empty bins. The sensors recognize picks. By using real-time data on fill-levels and pick recognition, we can show you how full the bins you collect are. The report shows how full the bin was when picked. You immediately see any inefficient picks below 80% full.
FR-6	Plan waste collection routes.	The tool semi-automates waste collection route planning. Based on current bin fill-levels and predictions of reaching full capacity, you are ready to respond and schedule waste collection. You can compare planned vs. executed routes to identify any inconsistencies.

4.2 Non-Functional requirements Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	**Non-Functional Requirement**	**Description**

NFR-1	**Usability**	IoT device verifies that usability is a special and important perspective to analyse 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.
NFR-2	**Security**	Use a reusable bottles Use reusable grocery bags Purchase wisely and recycle Avoid single use food and drink containers.
NFR-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.
NFR-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 Sensoneo's Smart Waste Management Software System, a powerful cloud-based platform, for data driven daily operations, available also as a waste management app. 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%.

NFR-5	**Availability**	By developing & deploying resilient hardware and beautiful software we empower cities, businesses, and countries to manage waste smarter.
NFR-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

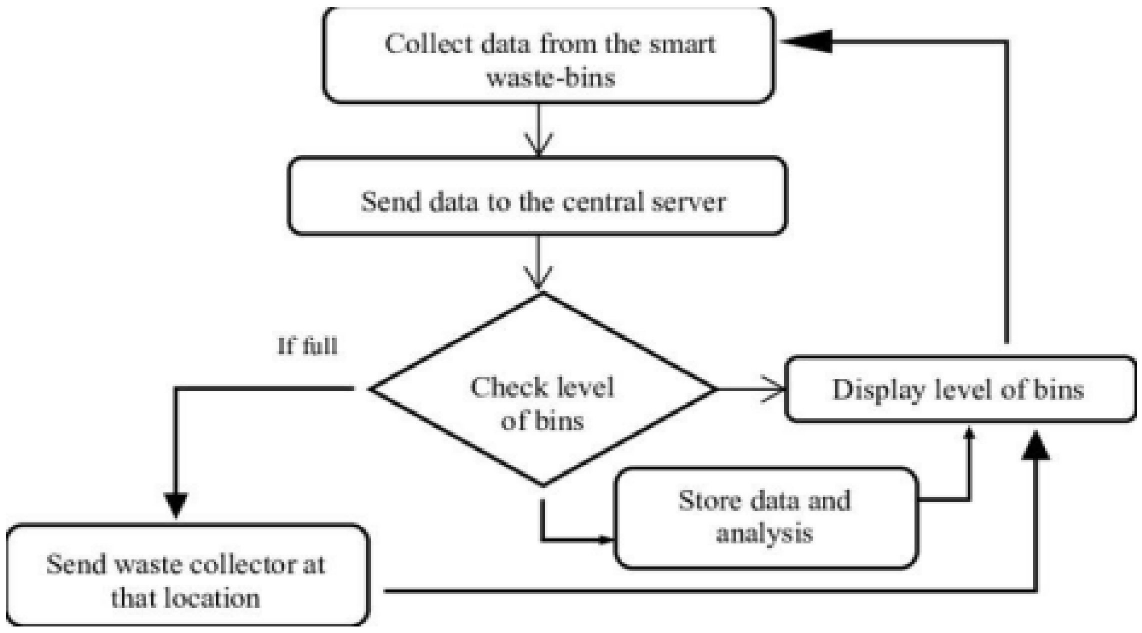
5.PROJECT DESIGN

5.1Data Flow Diagrams

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. A smart waste management platform uses analytics to translate the data gather in your bins into actionable insights to help you improve your waste services. You can receive data on metric such as:

- The first test conducted is the situation where the garbage bin is empty or its garbage level is very low
- Then, the bin is filled with more garbage until its level has surpassed the first threshold value, which is set to 80%then the first warning SMS is being sent, as depicted.
- The first notification SMS sent by the system, once the waste reaches the level of 85% full • The second notification SMS sent by the system, indicating that bin is at least 95% full and the garbage needs to be collected immediately
- Locations prone to overflow
- The number of bins needed to avoid overflowing waste
- The number of collection services that could be saved
- The amount of fuel that could be saved
- The driving distance that could be saved

Flow Diagram

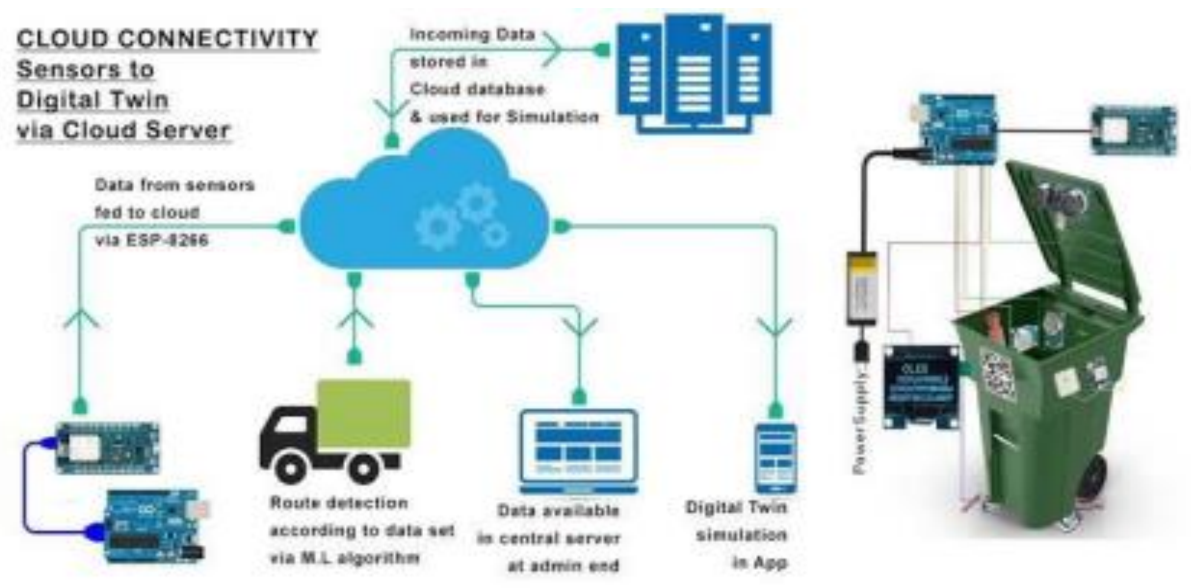

Collect data from the smart waste-bins
Send data to the central server
If full
Check level of bins
Display level of bins
Store data and analysis
Send waste collector at that location

5.2 Solution Architecture

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

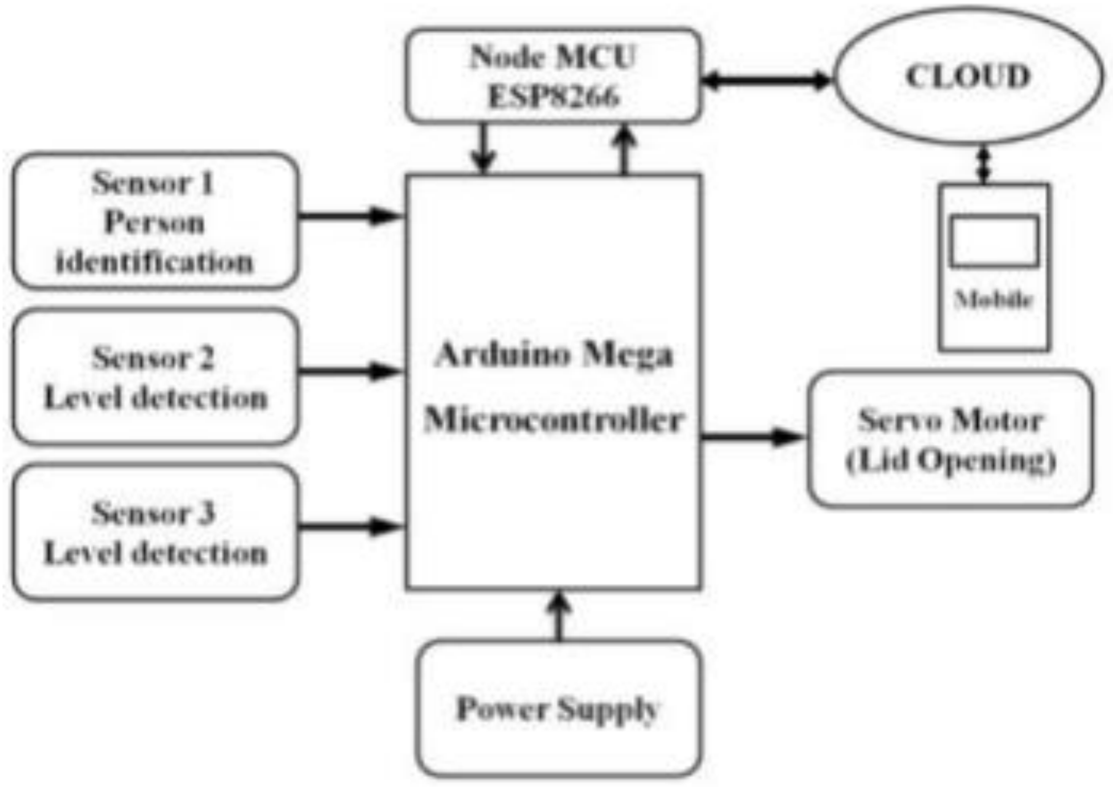
- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behaviour, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Diagram:



Architecture of Smart Waste Management System

5.3 Technology Architecture



Components & Technologies:

SI. No	**Component**	**Description**	**Technology**
1.	User Interface	Mobile Application	HTML, CSS, JavaScript.
2.	Application Logic	Logic for a process in the application	Java script
3.	Database	Data Type, Configurations etc.	Firebase, IBM cloud
4.	Cloud Database	Database Service on Cloud	IBM Cloud

5.	File Storage	File storage requirements	Local Filesystem and IBM cloud
6.	Infrastructure (Server / Cloud)	Application Deployment on Cloud Local Server Configuration	Local and Cloud Foundry

Application Characteristics:

S.no	**Characteristics**	**Description**	**Technology**
1.	Open-Source Frameworks	GitHub	Internet hosting service
2.	Security Implementations	Application security: Veracode.	Network automation
3.	Scalable Architecture	It provides the room for expansion more database of smart bins added additionally can be updated.	Cloud storage
4.	Availability	As the system control is connected to web server itis available 24*7 and can be accessed whenever needed.	Server, Appleixe, reple
5.	Performance	Performance is high it uses 5mb caches	Wireless Sensor Network

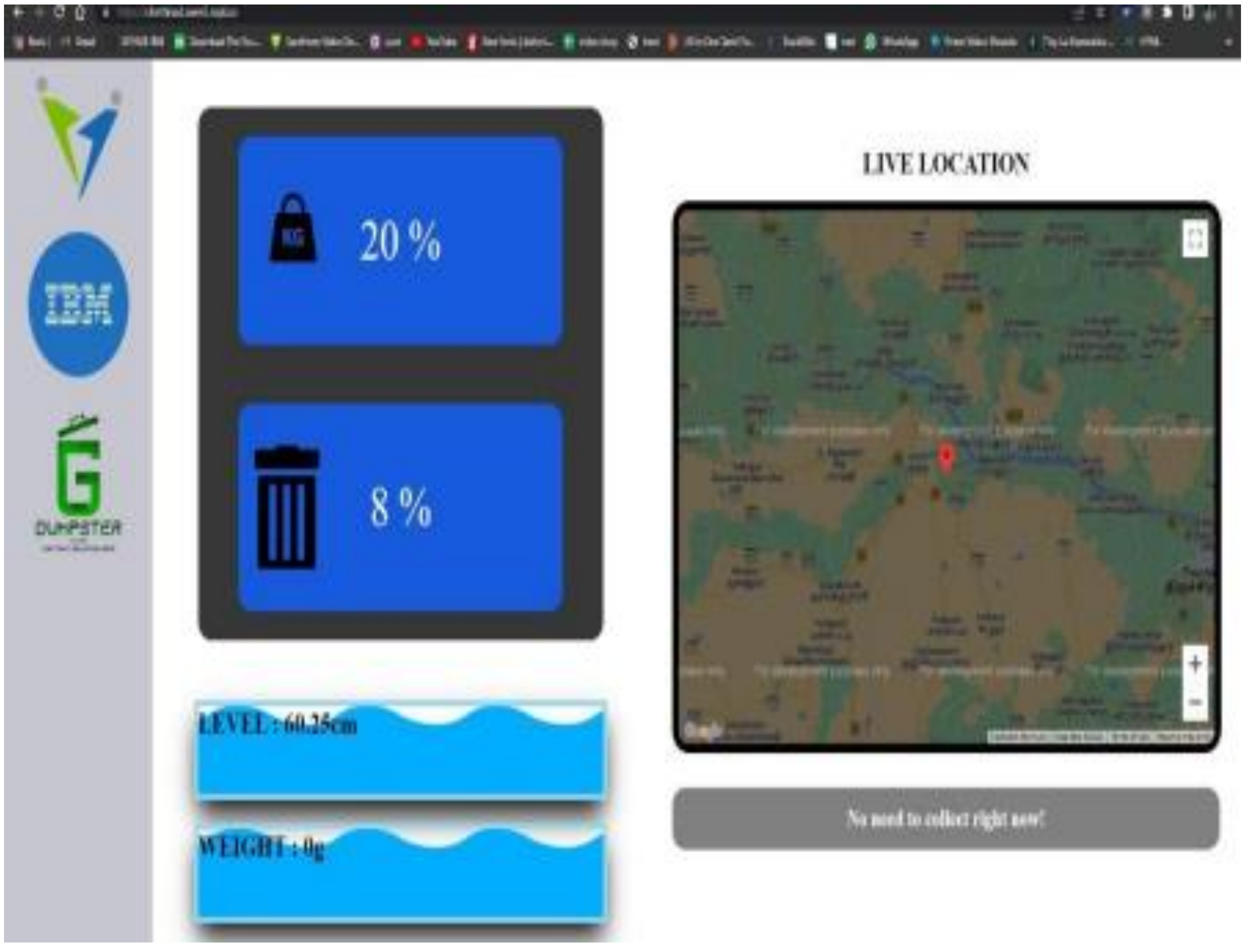
5.4 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Admin	Login	USN-1	As an administrator, I assigned user names and passwords to each employee and managed them.	I can control my online account and dashboard.	Medium	Sprint-1
Co-Admin	Login	USN-2	As a Co-Admin, I'll control the waste level monitor. If a garbage filling alert occurs, I will notify the trash truck of the location and rubbish ID.	I can handle the waste collection.	High	Sprint-1
Truck Driver	Login	USN-3	As a Truck Driver, I'll follow Co Admin's instruction to reach the filled garbage.	I can take the shortest path to reach the waste filled route specified.	Medium	Sprint-2
Local Garbage Collector	Login	USN-4	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	I can collect the trach, pull it to the truck, and send it out.	Medium	Sprint-3

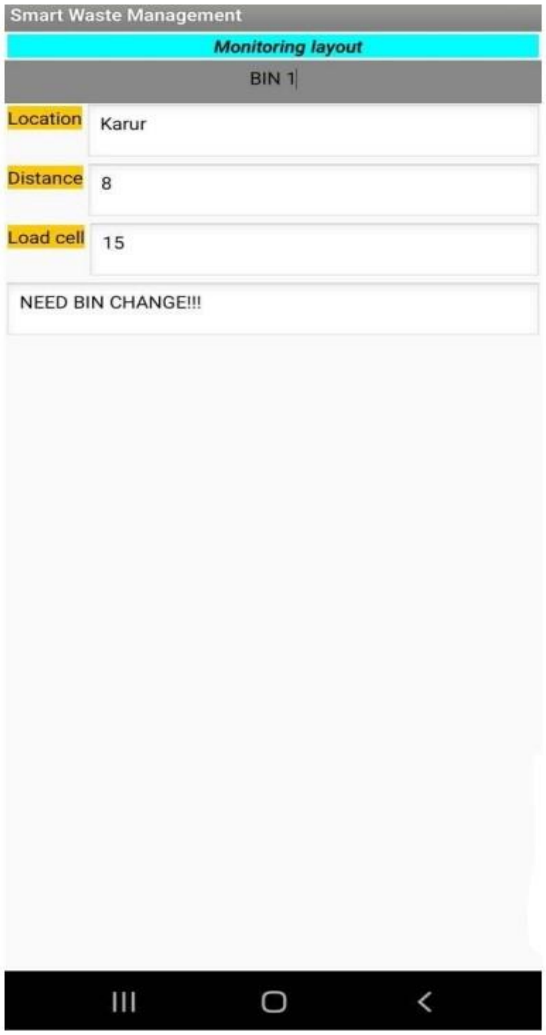
Municipality officer	Login	USN-5	As a Municipality officer, I'll make sure everything is proceeding as planned and without any problems.	All of these processes are under my control.	High	Sprint-4

6. CODING & SOLUTIONING

6.1FEATURE 1- L OCATION TRACKER



6.2 FEATURE 2-LIVE UPDATE ON COLLECTED DATA



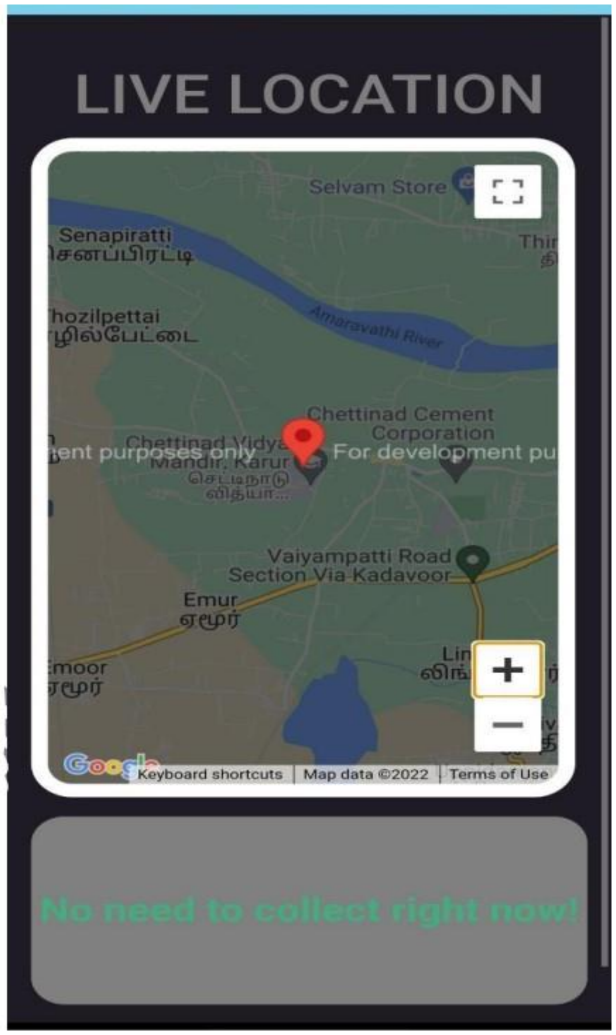
Smart Waste Management

Monitoring layout

BIN 1

Location	Karur
Distance	8
Load cell	15

NEED BIN CHANGE!!!



LIVE LOCATION

Selvam Store
Senapiratti
Thozilpettai
Amaravathi River
Chettinad Cement Corporation
Chettinad Vidya Mandir, Karur
For development purposes only
Vaiyampatti Road Section Via Kadavoor
Emur
Emoor
Keyboard shortcuts
Map data ©2022
Terms of Use

No need to collect right now!

7.ADVANTAGES &DISADVANTAGES

ADVANTAGES:

- Reduction in Collection Cost
- No Missed Pickups
- Reduced Overflows
- Waste Generation Analysis
- CO2 Emission Reduction

DISADVANTAGES:

- System requires a greater number of waste bins for separate waste collection as per populationin the city.
- This results into high initial cost due to expensive smart dustbins compare to other methods.
- Sensor nodes used in the dustbins have limited memory size.

8.CONCLUSION

A Smart Waste Management system that is more effective than the one in use now is achievable by using sensors to monitor the filling of bins. Our conception of a "smart waste management system" focuses on monitoring waste management, offering intelligent technology for waste systems, eliminating human intervention, minimizing human time and effort, and producing a healthy and trash- free environment. The suggested approach can be implemented in smart cities where residents have busy schedules that provide little time for garbage management. If desired, the bins might be put into place in a metropolis where a sizable container would be able to hold enough solid trash for a single unit. The price might be high.

9.FUTURE SCOPE

There are several future works and improvements for the proposed system, including the following:

1.Change the system of user authentication and atomic lock of bins, which would aid in protecting the bin from damage or theft.

2.The concept of green points would encourage the involvement of residents or end users, making the idea successful and aiding in the achievement of collaborative waste management efforts, thus fulfilling the idea of Swachh Bharath.

3.Having case study or data analytics on the type and times waste is collected on different days or seasons, making bin filling predictable and removing the reliance on electronic components, and fixing the coordinates.

10.APPENDIX

10.1 Source Code

Main .py

```
import time
c=1
for I in range(1,2):
while True:
if c == 1:
import distance
d=distance.distancesensor()
c = 2
elif c == 2:
import load
w = int(load.loop())
c = 3
else:
import database as db
if w < 5000 and w > 4000:
load = “90 %”
elif w < 4000 and w > 3000:
load = “60 %”
elif w < 3000 and w > 100:
load = “40 %”
else:
load = “0 %”
if d > 30:
```

```
distance = “90 %”
elif d < 30 and d >20:
distance = “60 %”
elif d < 20 and d > 5:
distance = “40 %”
else:
distance = “7 %”
if load == “90 %” or distance == “90 %”:
m = “Risk Warning: Dumpster poundage getting high, Time to collect 😊”
elif load == “60 %” or distance == “60 %”:
m =”dumpster is above 60%”
else :
m = “    “
db.database(d,w,m,load,distance)
print(“data pushed”)
c = 1
break
```

Load.py

```
import time
import sys

EMULATE_HX711=False

referenceUnit = 1

if not EMULATE_HX711:

import RPi.GPIO as GPIO from hx711 import HX711

else:

import RPi.GPIO as GPIO from emulated_hx711 import HX711

def cleanAndExit():

print("Cleaning...")

if not EMULATE_HX711:
```

```
GPIO.cleanup()
print("Bye!")
sys.exit()
hx = HX711(5, 6)
hx.set_reading_format("MSB", "MSB")
hx.reset()
hx.tare()
print("Tare done! Add weight now...")
def loop():
try:
val = hx.get_weight(5)
print(val)
return val
hx.power_down()
hx.power_up()
time.sleep(0.1)
except (KeyboardInterrupt, SystemExit):
cleanAndExit()
```

Distance.py

```
import RPi.GPIO as GPIO
import time
def distancesensor():
try:
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
PIN_TRIGGER = 23
PIN_ECHO = 33
GPIO.setup(PIN_TRIGGER, GPIO.OUT)
GPIO.setup(PIN_ECHO, GPIO.IN)
GPIO.output(PIN_TRIGGER, GPIO.LOW)
time.sleep(2)
GPIO.output(PIN_TRIGGER, GPIO.HIGH)
time.sleep(0.00001)
GPIO.output(PIN_TRIGGER, GPIO.LOW)
while GPIO.input(PIN_ECHO)==0:
pulse_start_time = time.time()
while GPIO.input(PIN_ECHO)==1:
pulse_end_time = time.time()
pulse_duration = pulse_end_time - pulse_start_time
global distance
distance = round(pulse_duration * 17150, 2)
print(distance)
return distance
finally:
GPIO.cleanup()
```

HX711.py

```
 import RPi.GPIO as GPIO
import time
import threading
class HX711:

def    init (self, dout, pd_sck, gain=128):
self.PD_SCK = pd_sck
self.DOUT = dout
self.readLock = threading.Lock()
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(self.PD_SCK, GPIO.OUT)
GPIO.setup(self.DOUT, GPIO.IN)

self.GAIN = 0
self.REFERENCE_UNIT = 1
self.REFERENCE_UNIT_B = 1

self.OFFSET = 1
self.OFFSET_B = 1
self.lastVal = int(0)

self.DEBUG_PRINTING = False

self.byte_format = 'MSB'
self.bit_format = 'MSB'
# Think about whether this is necessary.
time.sleep(1)

def convertFromTwosComplement24bit(self, inputValue):
return -(inputValue & 0x800000) + (inputValue & 0x7fffff)

def is_ready(self):
return GPIO.input(self.DOUT) == 0

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

GPIO.output(self.PD_SCK, False)

# Read out a set of raw bytes and throw it away.
self.readRawBytes()

def get_gain(self):
if self.GAIN == 1:
return 128
if self.GAIN == 3:
return 64
if self.GAIN == 2:
return 32

# Shouldn't get here.
return 0

def readNextBit(self):
# Clock HX711 Digital Serial Clock (PD_SCK). DOUT will be
GPIO.output(self.PD_SCK, True)
GPIO.output(self.PD_SCK, False)
value = GPIO.input(self.DOUT)

# Convert Boolean to int and return it.
return int(value)



def readNextByte(self):
byteValue = 0

# Read bits and build the byte from top, or bottom, depending
# on whether we are in MSB or LSB bit mode.
for x in range(8):
```

```
if self.bit_format == 'MSB':
byteValue <<= 1
byteValue |= self.readNextBit()
else:
byteValue >>= 1
byteValue |= self.readNextBit() * 0x80

# Return the packed byte.
return byteValue

def readRawBytes(self):
# Wait for and get the Read Lock, incase another thread is already
# driving the HX711 serial interface.
self.readLock.acquire()

# Wait until HX711 is ready for us to read a sample.
while not self.is_ready():
pass

# Read three bytes of data from the HX711.
firstByte = self.readNextByte()
secondByte = self.readNextByte()
# HX711 Channel and gain factor are set by number of bits read
# after 24 data bits.
for i in range(self.GAIN):
# Clock a bit out of the HX711 and throw it away.
self.readNextBit()

# Release the Read Lock, now that we've finished driving the HX711
# serial interface.
self.readLock.release()

# Depending on how we're configured, return an orderd list of raw byte
# values.
if self.byte_format == 'LSB':
```

```
return [thirdByte, secondByte, firstByte]
else:
return [firstByte, secondByte, thirdByte]
def read_long(self):
# Get a sample from the HX711 in the form of raw bytes.
dataBytes = self.readRawBytes()
if self.DEBUG_PRINTING:
print(dataBytes,)
# Join the raw bytes into a single 24bit 2s complement value.
twosComplementValue = ((dataBytes[0] << 16) |
(dataBytes[1] << 8) |
dataBytes[2])

if self.DEBUG_PRINTING:
print("Twos: 0x%06x" % twosComplementValue)
# Convert from 24bit twos-complement to a signed value.
signedIntValue        =
self.convertFromTwosComplement24bit(twosComplementValue)

# Record the latest sample value we've read.
self.lastVal = signedIntValue

# Return the sample value we've read from the HX711.
return int(signedIntValue)
# Make sure we've been asked to take a rational amount of samples.
if times <= 0:
raise ValueError("HX711()::read_average(): times must >= 1!!")

# If we're only average across one value, just read it and return it.
if times == 1:
return self.read_long()

# If we're averaging across a low amount of values, just take the
# median.
if times < 5:
```

```
return self.read_median(times)

# If we're taking a lot of samples, we'll collect them in a list, remove
# the outliers, then take the mean of the remaining set.
valueList = []

for x in range(times):
valueList += [self.read_long()]

valueList.sort()

# We'll be trimming 20% of outlier samples from top and bottom of collected set.
trimAmount = int(len(valueList) * 0.2)

# Trim the edge case values.
valueList = valueList[trimAmount:-trimAmount]

# Return the mean of remaining samples.
return sum(valueList) / len(valueList)
# A median-based read method, might help when getting random
value spikes
# for unknown or CPU-related reasons
def read_median(self, times=3):
raise ValueError("HX711::read_median(): times must be greater
than zero!")

# If times == 1, just return a single reading.
if times == 1:
return self.read_long()
valueList = []
for x in range(times):
valueList += [self.read_long()]
valueList.sort()

# If times is odd we can just take the centre value.
```

```
if (times & 0x1) == 0x1:
return valueList[len(valueList) // 2]
else:
# If times is even we have to take the arithmetic mean of
# the two middle values.
midpoint = len(valueList) / 2
return sum(valueList[midpoint:midpoint+2]) / 2.0
# Compatibility function, uses channel A version
def get_value(self, times=3):
return self.get_value_A(times)
def get_value_A(self, times=3):
return self.read_median(times) - self.get_offset_A()
def get_value_B(self, times=3):
# for channel B, we need to set_gain(32)
g = self.get_gain()
self.set_gain(32)
value = self.read_median(times) - self.get_offset_B()
self.set_gain(g)
return value
# Compatibility function, uses channel A version
def get_weight(self, times=3):
return self.get_weight_A(times)
def get_weight_A(self, times=3):
value = self.get_value_A(times)
value = value / self.REFERENCE_UNIT
return value
def get_weight_B(self, times=3):
value = self.get_value_B(times)
value = value / self.REFERENCE_UNIT_B
return value
# Sets tare for channel A for compatibility purposes
def tare(self, times=15):
return self.tare_A(times)
def tare_A(self, times=15):
backupReferenceUnit = self.get_reference_unit_A()
```

```
self.set_reference_unit_A(1)
value = self.read_average(times)

if self.DEBUG_PRINTING:
print("Tare A value:", value)
self.set_offset_A(value)

# Restore the reference unit, now that we've got our offset.
self.set_reference_unit_A(backupReferenceUnit)
return value
def tare_B(self, times=15):
# Backup REFERENCE_UNIT value
backupReferenceUnit = self.get_reference_unit_B()
self.set_reference_unit_B(1)

# for channel B, we need to set_gain(32)
backupGain = self.get_gain()
self.set_gain(32)

value = self.read_average(times)
if self.DEBUG_PRINTING:
print("Tare B value:", value)
self.set_offset_B(value)
# Restore gain/channel/reference unit settings.
self.set_gain(backupGain)
self.set_reference_unit_B(backupReferenceUnit)
return value
def   set_reading_format(self,        byte_format="LSB",
bit_format="MSB"):
if byte_format == "LSB":
self.byte_format = byte_format
elif byte_format == "MSB":
self.byte_format = byte_format
else:
raise ValueError("Unrecognised    byte_format:   \"%s\""         %
byte_format)
if bit_format == "LSB":
self.bit_format = bit_format
elif bit_format == "MSB":
self.bit_format = bit_format
raise ValueError("Unrecognised bitformat: \"%s\"" % bit_format)
# sets offset for channel A for compatibility reasons
def set_offset(self, offset):
self.set_offset_A(offset)

def set_offset_A(self, offset):
self.OFFSET = offset
def set_offset_B(self, offset):
self.OFFSET_B = offset

def get_offset(self):
return self.get_offset_A()

def get_offset_A(self):
return self.OFFSET

def get_offset_B(self):
return self.OFFSET_B
def set_reference_unit(self, reference_unit):
self.set_reference_unit_A(reference_unit)
def set_reference_unit_A(self, reference_unit):
# Make sure we aren't asked to use an invalid reference unit.
if reference_unit == 0:
raise ValueError("HX711::set_reference_unit_A() can't accept 0
as a reference unit!")
return

self.REFERENCE_UNIT = reference_unit
def set_reference_unit_B(self, reference_unit):
# Make sure we aren't asked to use an invalid reference unit.
```

```
if reference_unit == 0:
raise ValueError("HX711::set_reference_unit_A() can't accept 0
as a reference unit!")
return
self.REFERENCE_UNIT_B = reference_unit
def get_reference_unit(self):
return get_reference_unit_A()
def get_reference_unit_A(self):
return self.REFERENCE_UNIT
def get_reference_unit_B(self):
def power_down(self):
# Wait for and get the Read Lock, incase another thread is already
# driving the HX711 serial interface.
self.readLock.acquire()

# Cause a rising edge on HX711 Digital Serial Clock (PD_SCK). We then
# leave it held up and wait 100 us. After 60us the HX711 should
be
powered down.
GPIO.output(self.PD_SCK, False)
GPIO.output(self.PD_SCK, True)

time.sleep(0.0001)

# Release the Read Lock, now that we've finished driving the HX711
# serial interface.
self.readLock.release()
def power_up(self):
# Wait for and get the Read Lock, incase another thread is already
# driving the HX711 serial interface.
self.readLock.acquire()
# Lower the HX711 Digital Serial Clock (PD_SCK) line.
GPIO.output(self.PD_SCK, False)
# Wait 100 us for the HX711 to power back up.
time.sleep(0.0001)
```

```
# Release the Read Lock, now that we've finished driving the HX711
# serial interface.
self.readLock.release()
# HX711 will now be defaulted to Channel A with gain of 128. If
this
# isn't what client software has requested from us, take a sample
and
# throw it away, so that next sample from the HX711 will be from
the
# correct channel/gain.
if self.get_gain() != 128:
self.readRawBytes()
def reset(self):

self.power_down()
self.power_up()
```

WEBSITE CODING

Index.html

```
<html>
 <head>
<linkrel="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"
};

</head>

firebase.initializeApp(firebaseConfig)
</script>
<script defer src="database.js"></script>
<body style="background-color:#1F1B24;">
<script src="map.js"></script>
</div>
</div>
<div id="map_container">
<h1 id="live_location_heading" >LIVE LOCATION</h1>
<div id="map"></div>
<div id="alert_msg">ALERT MESSAGE!</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>
```

Database.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);
});
```

Map.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);




});
}, function (error) {

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

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;
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;
 }

}
```

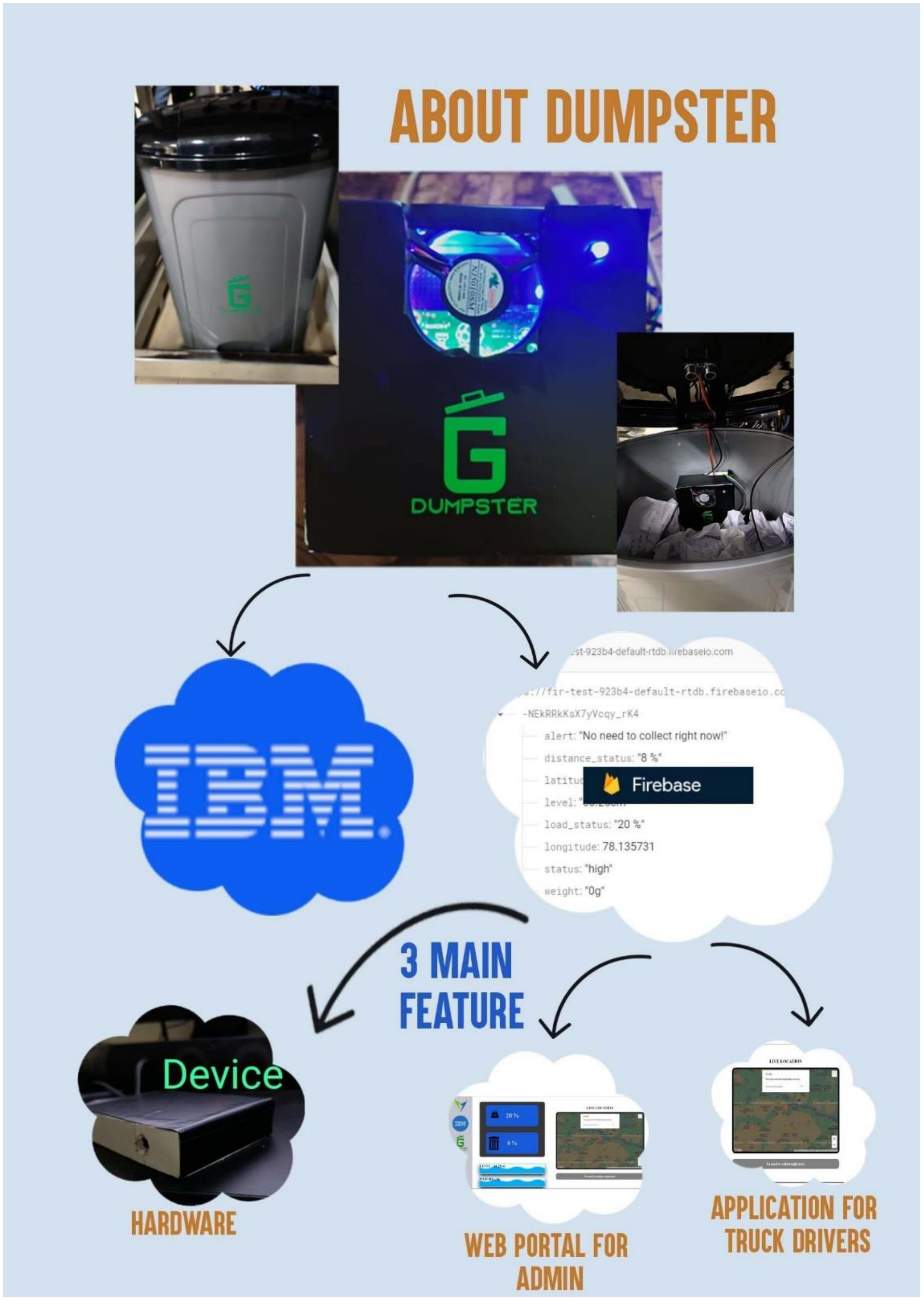
For simulator python code

BIN1.PY

```
 import requests import json
  import ibmiotf.application import ibmiotf.device import time
  import random import sys

  # watson device details

  organization = "4yi0vc" devicType = "BIN1" deviceId = "BIN1ID" authMethod=
      "token" authToken= "123456789"

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




  def myCommandCallback(cmd):
  global a
  print("command recieved:%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("caught exception 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 = 'alert :' ' Dumpster poundage getting high, Time to collect :) ' '90 %'



elif distance < 40 and distance >16:
dist = 'alert :' 'dumpster is above " 60%'

elif distance < 60 and distance > 41:
dist = 'alert :' 'dumpster is above "40 %' else:
dist ='alert :' 'No need to collect right now "17 %'




if load == "90 %" or distance == "90 %": warn = 'alert pushed to ibm sucessfully :'

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

warn = 'alert pushed to ibm sucessfully :' else :
warn = 'alert pushed to ibm sucessfully :'
def myOnPublishCallback(lat=10.678991,long=78.177731):
print("Gandigramam, Karur")
print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon = %s "
    %long,"lat = %s" %lat) print(load)
print(dist) print(warn)

time.sleep(4)
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(4)
deviceCli.commandCallback=myCommandCallback #disconnect the device
deviceCli.disconnect()
```

10.2 OUTPUT PICTURE



```
st-923b4-default-rtdb.firebaseio.com
://fir-test-923b4-default-rtdb.firebaseio.co
-NEkRRkKsX7yVcqy_rK4
    alert: "No need to collect right now!"
    distance_status: "8 %"
    latitu
    level:
    load_status: "20 %"
    longitude: 78.135731
    status: "high"
    weight: "0g"
```

10.3 LINKS GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-46573-1660750606