SMART WASTE MANAGEMENT SYSTEM FOR METROPOLITIAN CITIES

PROJECT REPORT

Team ID:

PNT2022TMID44143

Team Leader:

AJMEER KAJA (724019104001)

Team Members:

ANSAR M (724019104002)

ARUN VINOD K (724019104003)

NIJAAR AHAMAD (724019104017)

GUIDED BY: MOHAMMED NOORDEEN AP/CSE

ABSTRACT

Timely cleaning of dustbin is a big challenge and if left unaddressed, it may pose several health risks by making the place unhygienic. Current system forthe waste management in local areas of small and highly populated cities is sluggish which leads to a lot of garbage strewn all over the city. The rate of generation of waste is so high that if the garbage collector doesn't visit a place for a couple of days it creates the conditions adverse. In covid-19 pandemic situation, it was very important to monitor and decompose medical waste properly. The handling of normal home garbage was also challenging due to lockdown. In this situation automatic monitoring and controlling of garbage using IOT can play a significance role in garbage management. This paper proposes a smart and fast approach for waste management by creating a network of smart dustbins equipped with sensors and microcontrollers in a city which is monitored by a central control unit to speed up the process in an intelligent and smart way thereby eliminating such hazardous conditions caused by the current sluggish system. The proposed system also takes into account the issue of improper internet connectivity.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

IoT	INTERNET OF THINGS
I-SMAC	IoT IN SOCIAL, MOBILE, ANALYTICS AND CLOUD
GPS	GLOBAL POSITIONING SYSTEM
FR	FUNCTIONAL REQUIREMENTS
NFR	NON-FUNCTIONAL REQUIREMENTS
DFD	DATA FLOW DIAGRAM
MQTT	MQ TELEMETRY TRANSPORT
SQL	STRUCTURED QUERY LANGUAGES
STT	SECURITY TRANSACTIONS TAX
DB	DATABASE
RFID	RADIO FREQUENCY IDENTIFICATION
UAT	USER ACCEPTANCE TESTING
WIFI	WIRELESS FIDELITY
OTP	ONE TIME PASSWORD
MYSQL	MY STRUCTURED QUERY LANGUAGE
NOSQL	NOT ONLY STRUCTURED QUERY LANGUAGE

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The Internet of Things (IoT) is a concept in which surrounding objects are connected through wired and wireless networks without user intervention. In the field of IoT, the objects communicate and exchange information to provide advanced intelligent services for users.

Project deals with the problem of waste management in smart cities, where the garbage collection system is not optimized. Project enables the organizations to meet their needs of smart garbage management systems. This system allows the user to know the fill level of each garbage bin in a localityor city at all times, to give a cost-effective and time-saving route to the truck drivers.

The rate at which solid wastes are produced in most developing countries is becoming alarming. There are two categories of Urban waste namely, organic and inorganic. The organic waste category can be further categorized into three units: nonfermentable, fermentable and putrescible . The Putrescible wastes tend to decay faster, and if not cautiously managed, decomposition can lead to an offensive odour with an unpleasant view.

Fermentable waste which also tends to decompose rapidly do so without the accompanying of offensive odour. Non-fermentable waste most times do not decompose or do so at a very slow rate. Unless organic waste is managed appropriately, the stricken negative effect it has will continue until full decomposition or stabilization will be occurs. Decomposed products which are poor managed or uncontrolled can and often times lead to the problems such as contamination of air, water and soil resource.

1.2 PURPOSE

Waste management is intended to reduce adverse effects of waste on human health, the environment, planetary resources and aesthetics. The aim of waste management is to reduce the dangerous effects of such waste on the environment and human health. Waste management is an important element of environmental protection. Its purpose is to provide hygienic, efficient and economic solid waste storage, collection, transportation and treatment or disposal of waste without polluting the atmosphere, soil or water system.

Waste management is needed to prevent pollution. Sewage treatment is also a kind of waste water treatment where wastes are removed from water.It helps us to separate bio degradable(eg-paper) and non-degradable wastes(Eg-plastics)

Keeping cities clean is essential for keeping their resident healthy depends not just on personal hygiene and nutrition but critically also on how clean we keep our cities and their surroundings the spread of dengue and chicken gunya are intimately linked to the deteriorating state of public health condition in our cities.

The good news is that waste management to keep cities clean is now getting attention through the swatch Bharat mission how where, much of the attention began in and stops with the brooms and the dustbins, extending at most to the collection and transportation of the mixed waste to some distant or not so distant place preferably out of sight.

Waste management involves the regular collection, transportation as well as processing and disposal or recycling and monitoring of different types of waste materials.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

M. Bhuvaneswari et a1. Waste segregation and disposal mechanisms are among the severe problems associated with smart cities, which have a negative impact on our society and health.

K N Fallavi et a1. As the population is growing, the garbage is also increasing. This huge unmanaged accumulation of garbage is polluting the environment, spoiling the beauty of the area and also leading to the health hazard. In this era ofInternet, IOT (Internet of Things) can be used effectively to manage this solid waste.

M.Selvi et a1. The sensed data by the smart objects are transmitted to the sink for further processing using multi hop communication. The smart cities use the analyzed data to improve their infrastructure, public utilities and they enhance their services by using the IoT technology for the betterment of livelihood of the common people. For IoT based smart cities, waste collection is a prominent issue for municipalities that aim to achieve a clean environment.

Mahmoud Ali Ahmed et a1. Three factors make it a big challenge, behind its natural, small area, short period of time and the increasing of the Pilgrimages' member. The process of collected wastes, separated it, and transports the containers daily and quickly to avoid any prospect of a spread of diseases is a complex process.

Proposed smart systems for waste management system with recycling .The proposed system will use the sensors technique insite the container, as a lower level, to separate the waste into 4 categories [food, plastics, papers, and metal] and use actuator at a top level to inform the management system to collect the container.

2.2 REFERENCES

1. M. Bhuvaneswari; K. Tansin; S. Tazmeel Ahamed; N.Tharun Sri Ram; S. Venu Prasath 2022 7th International Conference on Communication and Electronics Systems (ICCES) Internet of Things based Intelligent Waste Segregation and ManagementSystem for Smart Home Application

2. K N Fallavi; V Ravi Kumar; B M Chaithra 2017 International Conference onI-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I- SMAC) Smart waste management using Internet of Things: A survey

3. Jacob John, Mariam Sunil Varkey, Riya Sanjay Podder, Nilavrah Sensarma, M. Selvi, 19 August 2021 Smart Prediction and Monitoring of Waste Disposal System Using IoTand Cloud for IoT Based Smart Cities

4. Rasha Elhassan; Mahmoud Ali Ahmed; Randa AbdAlhalem 2019 4th MEC International Conference on Big Data and Smart City (ICBDSC) Smart Waste Management System for Crowded area : Makkah and HolySites as a Model

5. Inna Sosunova and Jari Porras 18 May 2022, accepted 17 June 2022, date of publication 4 July 2022, date of current version 18 July 2022 IoT-Enabled Smart Waste Management Systems for Smart Cities: A Systematic Review

2.3 PROBLEM STATEMENT DEFINITION

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.

This process in current city setting solves the waste problem partially while itcreates other problems such as

- Some trash bins are overfilled while others are underfilled by the trash collection time,
- Overfilled trash bins create unhygienic conditions,
- Unoptimized truck routes result in excessive fuel usage and environmental pollution and
- All collected trash is combined which complicates sorting at the recycling facility.

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1EMPATHY MAP CANVAS

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

It is a useful tool to helps 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 andchallenges.

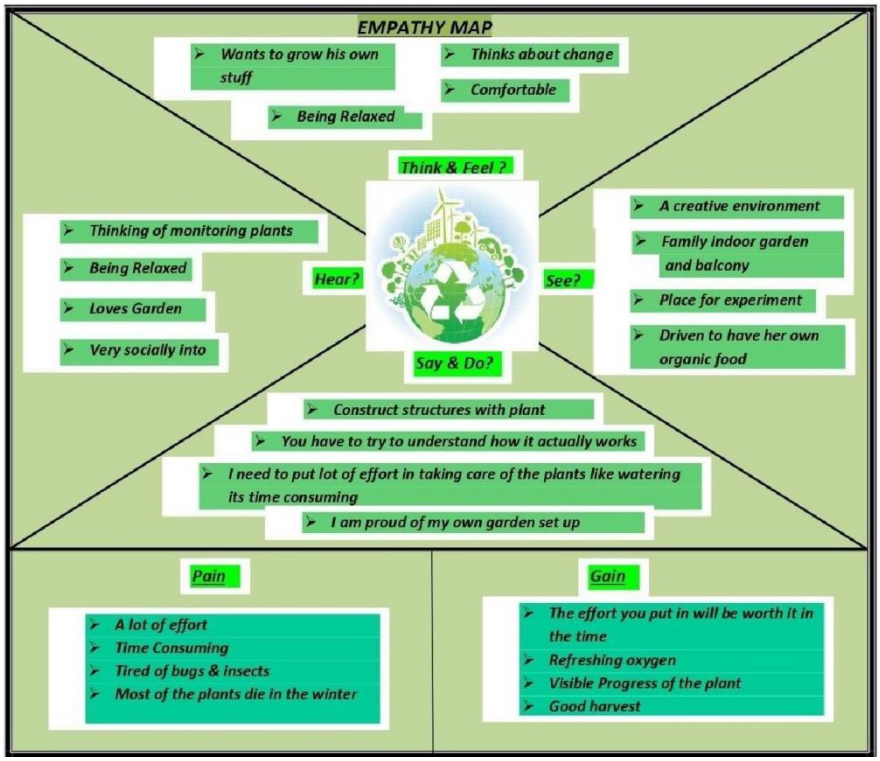


Fig 3.1.1 Empathy Map

3.2 IDEATION & BRAINSTORMING

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.

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

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 andproductive session.

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

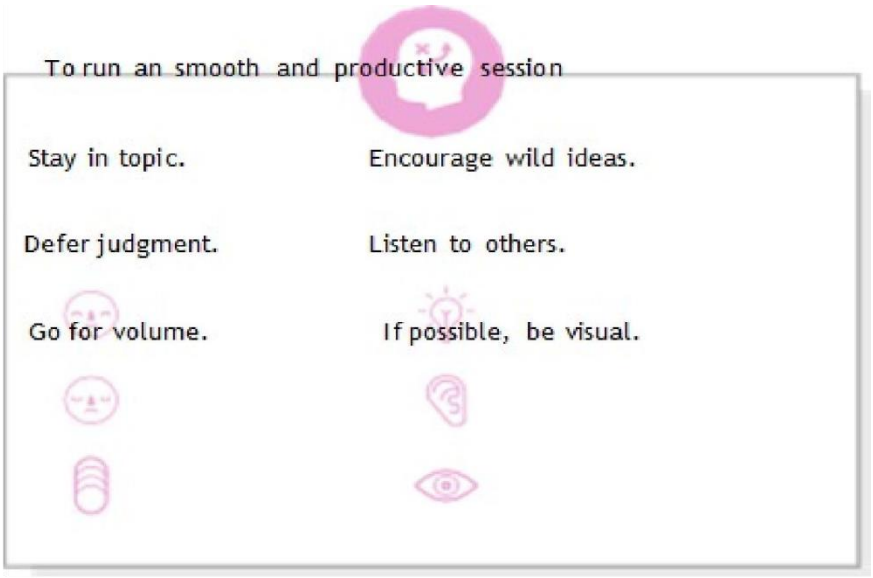


Fig 3.2.1 Problem Statement

Brainstorm

Write down any ideas that come to mind that address your problem statement.

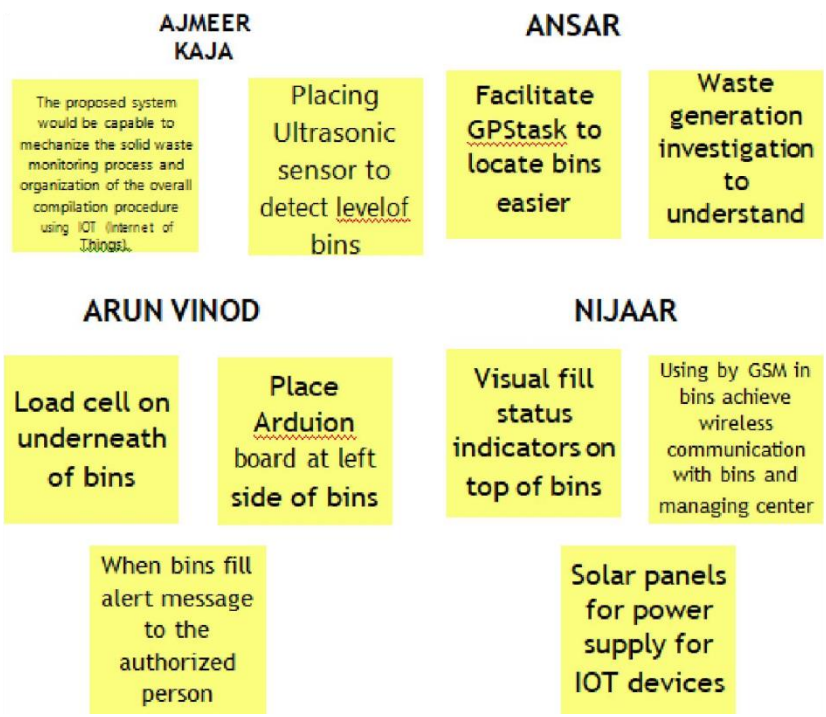


Fig 3.2.2 Brainstorming

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

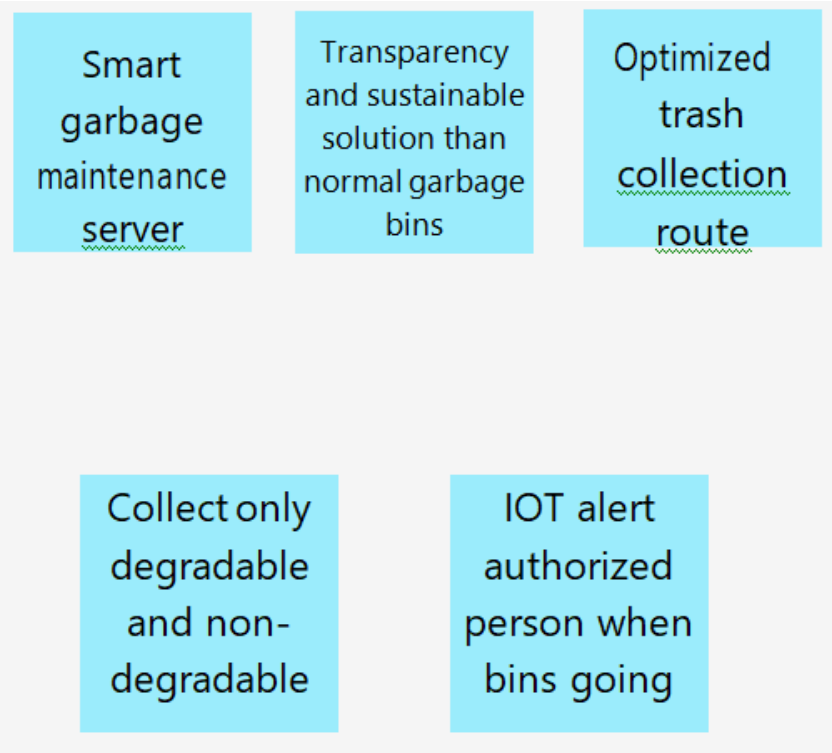


Fig 3.2.3 Group Ideas

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible

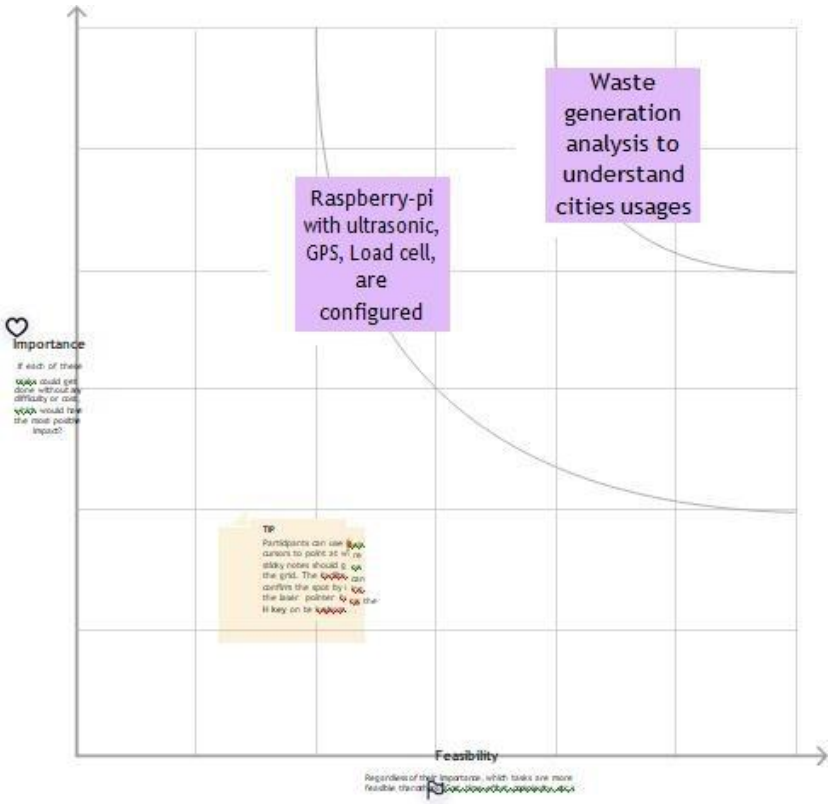


Fig 3.2.4 Prioritize

After you collaborate

A. Share the mural

Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B. Export the mural

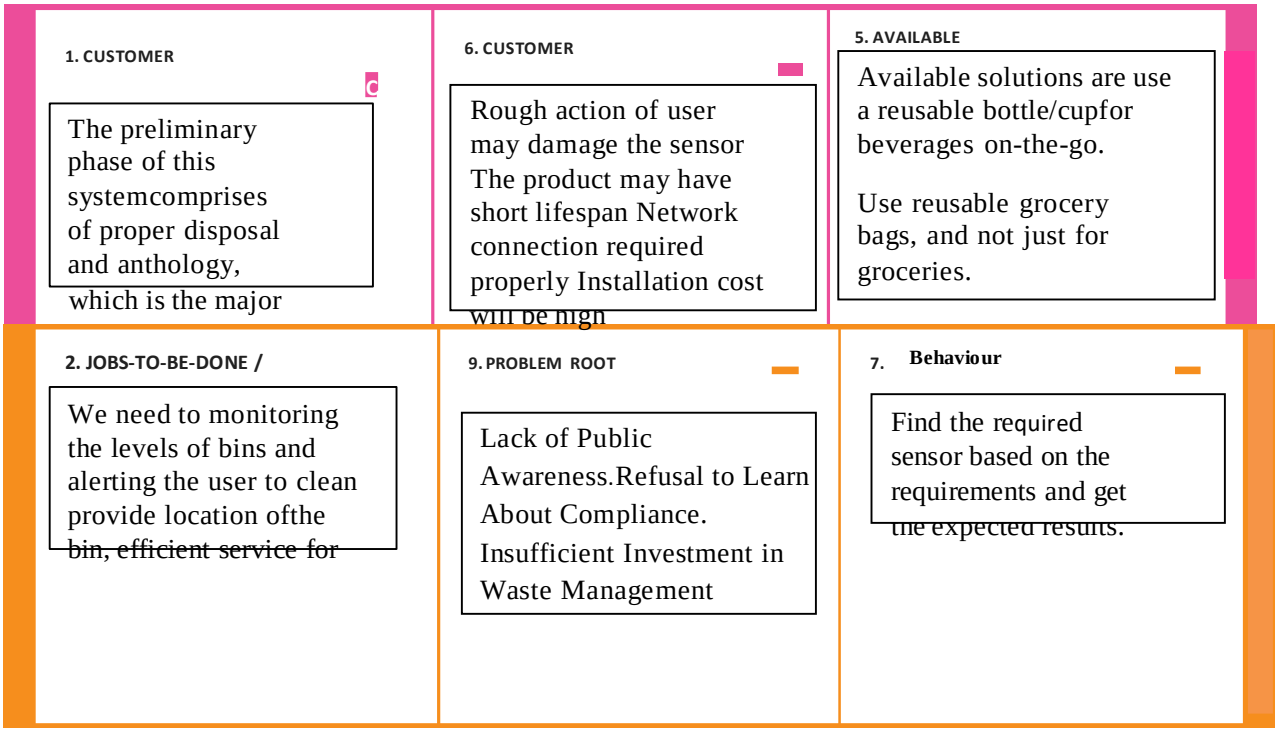
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive

3.3PROPOSED SOLUTION

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	The set and disposal of garbage waste is in unordered, ineffective way which leads to overfilling of bins, rotting garbage smell and more fuel utilization of collecting trucks.
2.	Idea / Solution description	Using sensors, weighing machine; real timemonitoring the level of waste in bins. The information get shared with appropriateauthorities and fellow citizens through web application.
3.	Novelty / Uniqueness	Citizens & industries behavior during precise festival, events at different seasons are monitored and are predicted for garbage overflowing. Also to find the shortest path to reach the fate for trucks in basis of fuel and time consumption.
4.	Social Impact / Customer Satisfaction	Informative, effective management of waste in bigcities reduces waste impacts over environment pollution.
5.	Business Model (Revenue Model)	• Eco-friendly. • Optimized route navigation system. • Reduce fuel consumption. • Alerts authority by real-time monitoring. • Promote 3R's(Reduce, Reuse ,Recycle).
6.	Scalability of the Solution	• The need-driven waste collection eliminates unnecessary traffic blockage. • Generate important statistical data for monitoring for waste collection. • Recycling is promoted between residents, results in clean &sustainable environment.

Table 3.3.1 Proposed Solution

3.4PROBLEM SOLUTION FIT



3. TRIGGERS

- Landfill –growth
- Increases their profit.
- Incineration
- best way too trigger the customers to buy the product

4. EMOTIONS: BEFORE / AFTER

Provide better environment for people live around bins.This technology can lead towards the development and adoption of a cleaner production, circular Economy and effective waste management.

10. YOUR SOLUTION

Our solution is to provide a smart waste management system where sensors are fittedinsidethe dustbins which collect the waste in the locality and alert the respective people to collect and segregate the waste. The system also provides routeplanning for the collection of the waste.

8. CHANNELS of BEHAVIOUR

Online:

- Use emails and articles instead of letters and magazines
- Create voluntary awareness in social
- Reduce recycle reuse
- Buy second handsand reduce goods
- Use biodegradablecovers
- Compost it

Fig 3.4.1 Problem solution fit

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

The functional requirements of smart waste management system are User registration, User conformation, GPS Access, Bin level Analysing, Transport Router.

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Registration**	Registration through Gmail
FR-2	**User Confirmation**	Confirmation via Email Confirmation via OTP
FR-3	**GPS Access**	GPS admission to recognize the location
FR-4	**Bin level Analysing**	Obtain the levels of Waste bins in a regular interval of time.
FR-5	**Transport Router**	To make a efficient route for the collection of garbage in the region of a area.

Table 4.1.1 Functional Requirements

4.2 NON FUNCTIONAL REQUIREMENTS

The non functional requirements are Usability, Security ,Reliability, Performance , Availability and Scalabilility

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	A smart solution has been planned to make the waste sorting more simple and accurate, and look up the user experience, usability, and satisfaction. It aims to optimize ease of use while offering maximum functionality.
NFR-2	**Security**	The information of the users will be highly secured, the accounts are verified with Gmail. If the products are misplaced then the GPSdriven sensor gives an alert.
NFR-3	**Reliability**	Operates in a defined environment without failure resulting in less manpower, emissions, fuel use and traffic congestion.
NFR-4	**Performance**	The system will provide accurate reports, thus increasing the efficiency of the system. The real-time monitoring of the garbage levelwith the help of sensors and wireless communication will reduce the total number of trips required of Garbage collecting truck. This will reduce the total expenditure associated with the garbage collection.
NFR-5	**Availability**	The smart waste bins are available in Convention centers, buildings, stadiums, and transportationfacilities and captures high-quality waste data and informs staff when it gets full.
NFR-6	**Scalability**	A versatile scalable smart waste-bin system based on limited waste management could potentially lead to great improvements.Once these smart bins are implemented on a largescale by replacing the traditional bins, the waste can be quickly managed to its efficient level as it avoids unnecessary lumping of wastes on roadside.

Table 4.2.1 Non-Functional Requirements

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.

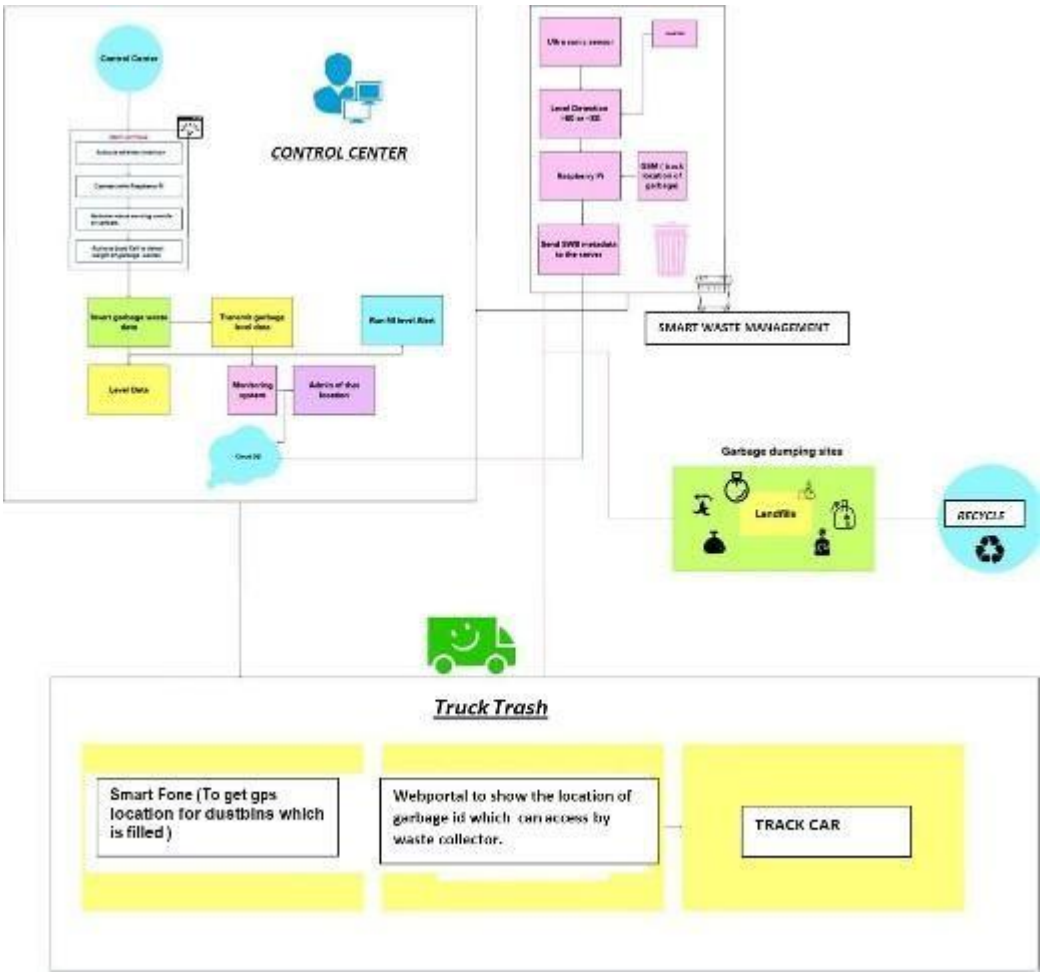


Figure 5.1.1 Data Flow Diagram

5.2 SOLUTION AND TECHNICAL ARTCHITECTURE

The Deliverable shall include the architectural diagram as below and the information as per the Table1 & Table 2

Example: Order processing during pandemics for offline mode

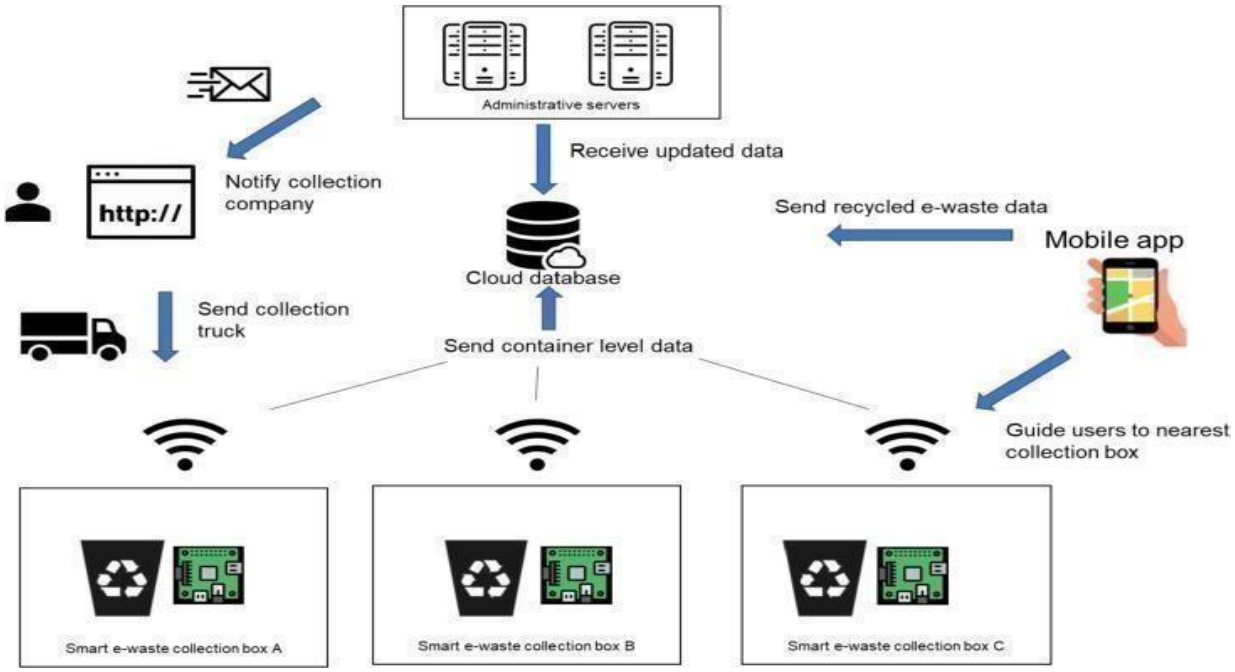


Figure 5.2.1 Solution & Technical Architecture

S. No	Component	Description	Technology
1.	User Interface	IBM Watson IOT cloud platform	MQTT Protocol
2.	Application Logic-1	The bin waste data's are collectedusing sensors	Python
3.	Application Logic-2	The collected data's are monitoredusing IOT	IBM Watson STTservice

4.	Application Logic-3	Based on data's the alerting message will send to the workersfor disposing the wastes.	IBM Watson Assistant
5.	Database	• MySQL is a relational databasethat is based on a tabular design. • NoSQL is non-relational andhas adocument -based design.	MySQL, NoSQL
6.	Cloud Database	This module will receive real time status updates from all the garbage bins and continuously display it on web application and also push the notifications on client sides (Municipal Corporation,Garbage collector truck drivers etc.) mobile application.	IBM DB2, IBM Cloud
7.	File Storage	Data storage makes it easy to back up files for safekeeping and quick recovery in the event of an unexpected computing crash or cyber attack.	IBM Block storage orother storage device

Table 5.2.1 Solution Architecture

S. No	Characteristics	Description	Technologies
1.	Open - Source Frameworks	Transport, treatment, and disposal of waste together with monitoring and regulation.	Python
2.	Security Implementations	Fundamental component of data security that dictates who allowedto access and it use company information and resources. Firewalls use rule based access to control model with rules expressed in an accesscontrol list.	Firewall

3.	Scalable Architecture	Using smart waste bins, reduce the number of bins inside town and cities because that we capable to monitor the garbage 24/7.It will be more cost efficient and scalable when we moves to smarter.	Technologyused
4.	Availability	By developing & deploying resilient hardware and beautiful software we empower cities, businesses, and countries to manage waste smarter.	IOT, RFID

Table 5.2.2 Technical Architecture

5.3 USER STORIES

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Admin (who manage server)	Login	USN-1	As an Admin, I gave userid and password for ever workers and manage them.	I can access my account / dashboard	Medium	Sprint-2
Co Admin	Login	USN-2	As a Co Admin, I'll manage garbage level monitor .if garbage get filling alert I will post location and garbage e idto trash truck	I can receive confirmation email & click confirm	High	Sprint-1
Truck Driver	Login	USN-3	As Truck Driver, I'll follow the route send byCo Admin to reach the filled garbage	I can register& access the dashboard with Facebook Login	Medium	Sprint-2

Local	Login	USN-4	As a Waste Collector, I'll collect all the trash from	I can collect trash and	Medium	Sprint-2
Garbage collector			garbage and load into garbage truck and send them to landfill	pulled to truck and send off		
Municipality	Login	USN-5	As a Municipality, I'll checkthe process are happening in disciplinemanner without any issues.	I can manageall these process goinggood	High	Sprint-1

Table 5.3.1 User Stories

CHAPTER 6

PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email	2	High	Ajmeer Kaja
Sprint-2		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Ansar
Sprint-3		USN-3	As a user, I can register for the application through Face book	2	Low	Arun Vinod
Sprint-4	Dashboard	USN-4	As a user, I can register for the application through Gmail	2	Medium	Nijaar Ahamad

Table 6.1.1 Sprint Planning & Estimation

6.2 Sprint Delivery scheduling

Sprint	Total Story Points	Duration	SprintStart 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	30	30 OCT 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	49	6 NOV 2022
Sprint-4	20	6 Days	14 Nov2022	19 Nov 2022	50	7 NOV 2022

Table 6.2.1 Sprint Delivery Scheduling

6.3 REPORT FROM JIRA

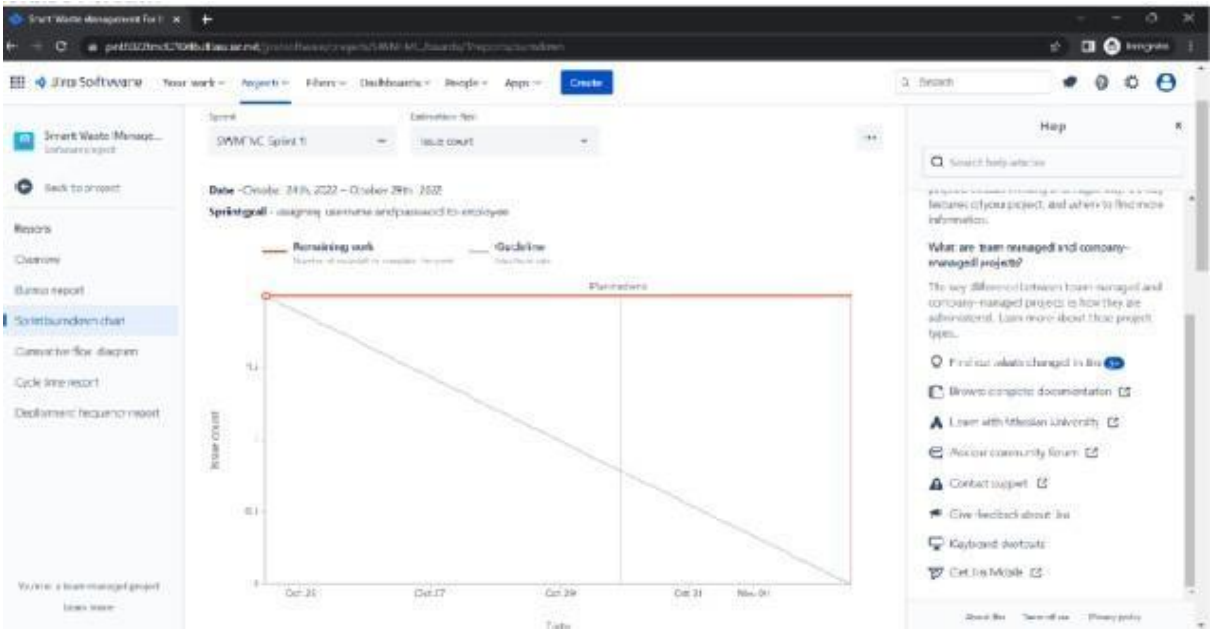


Figure 6.3.1 Reports From JIRA

CHAPTER 7

CODING & SOLUTIONING

7.1 FEATURE 1

7.1.1 User Module

As members of an IoT ecosystem, users notify about their needs and desires, and provide feedback within a networked intelligence to mutually progress their individual ability to rule the actuators of the system at their service. User Device means a mobile or other handheld device.

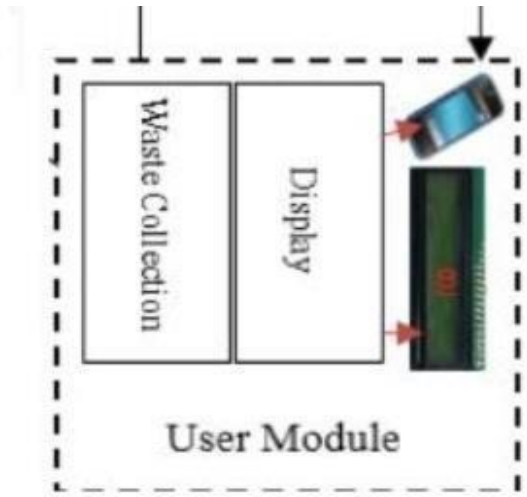


Fig 7.1.1.1 User Module

Program:
For Site Routes

```
from flask import Blueprint
site = Blueprint('site',_____name___)
```

```
@site.rout
e('/')def
index():
return "<h1>Welcome to Our Waste Management
System</h1>"@site.route('/health')
def
health_check():
return  "ok"
```

For Validate

```
def validate(roles,
data):if roles is
None:
    return True
for role in
roles:
print(role)
print(data)
print(role in data)
if role not in data:
        return False
return True
```

Explanation

In User Module, User can using a Blueprint to Access the Waste Management to define the routes of the area. User will also check the health by defining this Module. For Validating the Route user can define the roles and data to check the given routes have garbage or not.

7.2FEATURE 2

7.2.1 Sensing Module

Sensors and modules (having extra electronic circuitry along with sensor) are Electronic devices that detect and respond to some type of input from the physical environment..Sensors play a pivotal role in the internet of things (IoT). They make it possible to create an ecosystem for collecting and processing data about a specific environment so it can be monitored, managed and controlled more easily and efficiently.

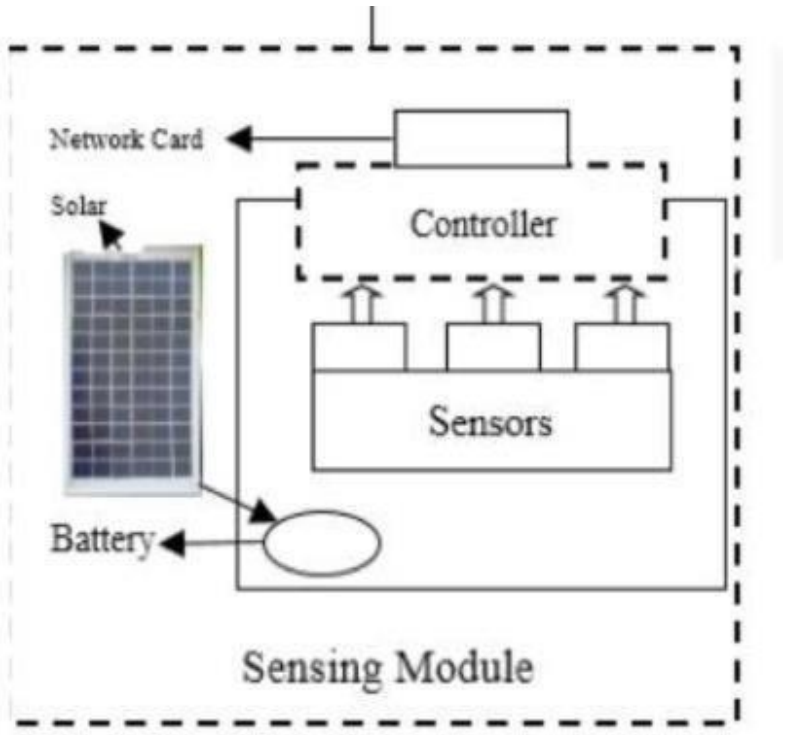


Fig 7.2.1.1 Sensing Module

Program: <u>For Scripts</u>

"""${message}

Revision ID: ${up_revision}
Revises: ${down_revision |
comma,n}Create Date:
${create_date}
"""
from alembic
import opimport
sqlalchemy as sa
${imports if imports else ""}
revision identifiers, used by
Alembic. revision =
${repr(up_revision)} down_revision
= ${repr(down_revision)}
branch_labels =
${repr(branch_labels)} depends_on
= ${repr(depends_on)}
def upgrade():
    ${upgrades if upgrades else
"pass"}def downgrade():
${downgrades if downgrades else "pass"}

For Environment Sensing

from_____future____import
with_statementimport logging
from logging.config import fileConfig

```
from sqlalchemy import
engine_from_configfrom sqlalchemy
import pool
from flask import
current_appfrom alembic
import context
# this is the Alembic Config object, which provides

# access to the values within the .ini file in use.
config = context.config
# Interpret the config file for Python
logging.# This line sets up loggers
basically.
fileConfig(config.config_file_name)
logger =
logging.getLogger('alembic.env')#
add your model's MetaData object
here # for 'autogenerate' support
# from myapp import mymodel
# target_metadata =
mymodel.Base.metadata
config.set_main_option(
'sqlalchemy.url',
str(current_app.extensions['migrate'].db.engine.url).replace('%', '%%'))
target_metadata =
current_app.extensions['migrate'].db.metadata# other values
from the config, defined by the needs of env.py,
```

```
# can be acquired:
# my_important_option =
config.get_main_option("my_important_option")# ... etc.
def run_migrations_offline():
"""Run migrations in 'offline' mode.
This configures the context with just a URL
and not an Engine, though an Engine is
acceptablehere as well. By skipping the Engine
creation
we don't even need a DBAPI to be available.

Calls to context.execute() here emit the given string to the
script output.
"""
url =
config.get_main_option("sqlalchemy.url")
context.configure(
url=url, target_metadata=target_metadata, literal_binds=True
)
with context.begin_transaction():context.run_migrations()
def run_migrations_online():
"""Run migrations in 'online' mode.
In this scenario we need to create an
Engineand associate a connection with
the context."""
# this callback is used to prevent an auto-migration from being
```

```
    generated# when there are no changes to the schema
    # reference: http://alembic.zzzcomputing.com/en/latest/cookbook.html
    def process_revision_directives(context, revision, directives):
if getattr(config.cmd_opts, 'autogenerate', False):
    script = directives[0]
if script.upgrade_ops.is_empty():
    directives[:] = []
            logger.info('No changes in schema
    detected.')connectable = engine_from_config(
    config.get_section(config.config_ini_section),
    prefix='sqlalchemy.',
    poolclass=pool.NullPool,
    )
with connectable.connect() as connection:context.configure(
    connection=connection, target_metadata=target_metadata,
    process_revision_directives=process_revision_directives,
    **current_app.extensions['migrate'].configure_args
    )
with context.begin_transaction():
    context.run_migrations()
if
    context.is_offline_mode
    ():
    run_migrations_offline(
    )
    else:
    run_migrations_online()
```

7.3 DATABASE SCHEMA

Step 1: Create A Cloundant DB With Your Credentials

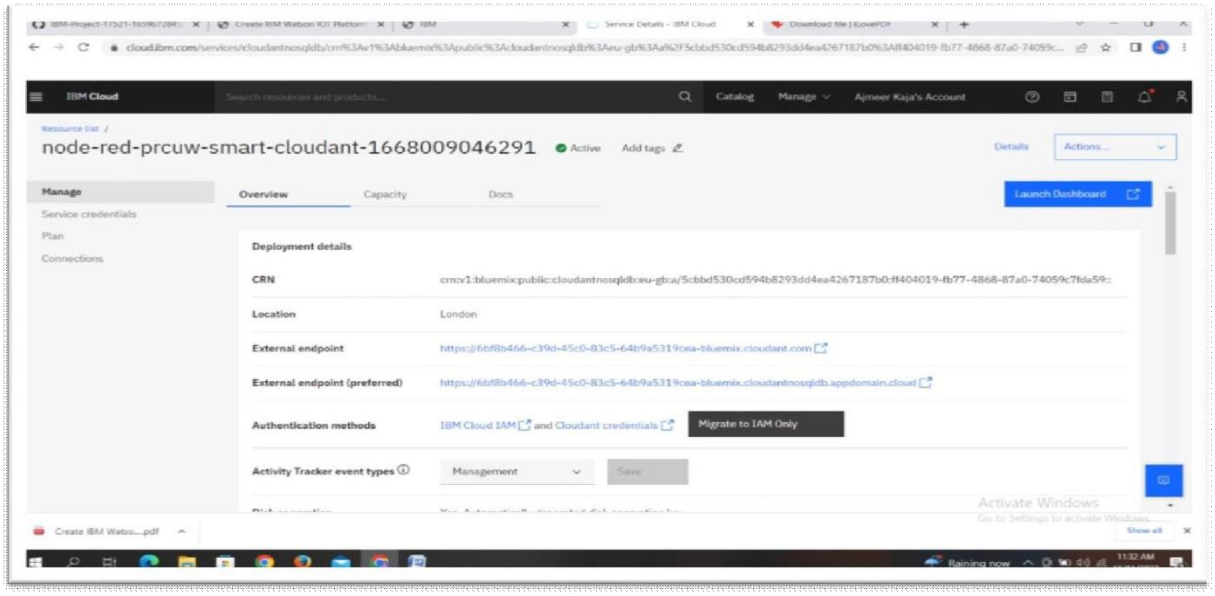


Fig 7.3.1 Cloudant DB

Step 2: Then Launch Dashboard To see your Database Page

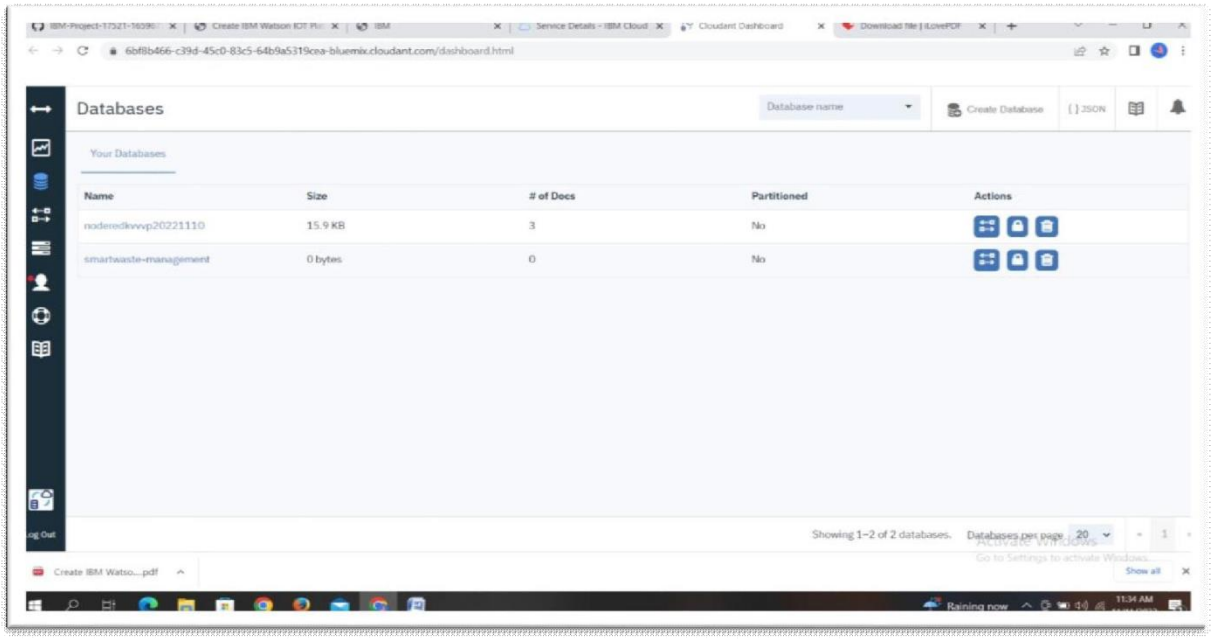


Fig 7.3.2 DB Creation

Step 3: Then Create Your Database for your Project.

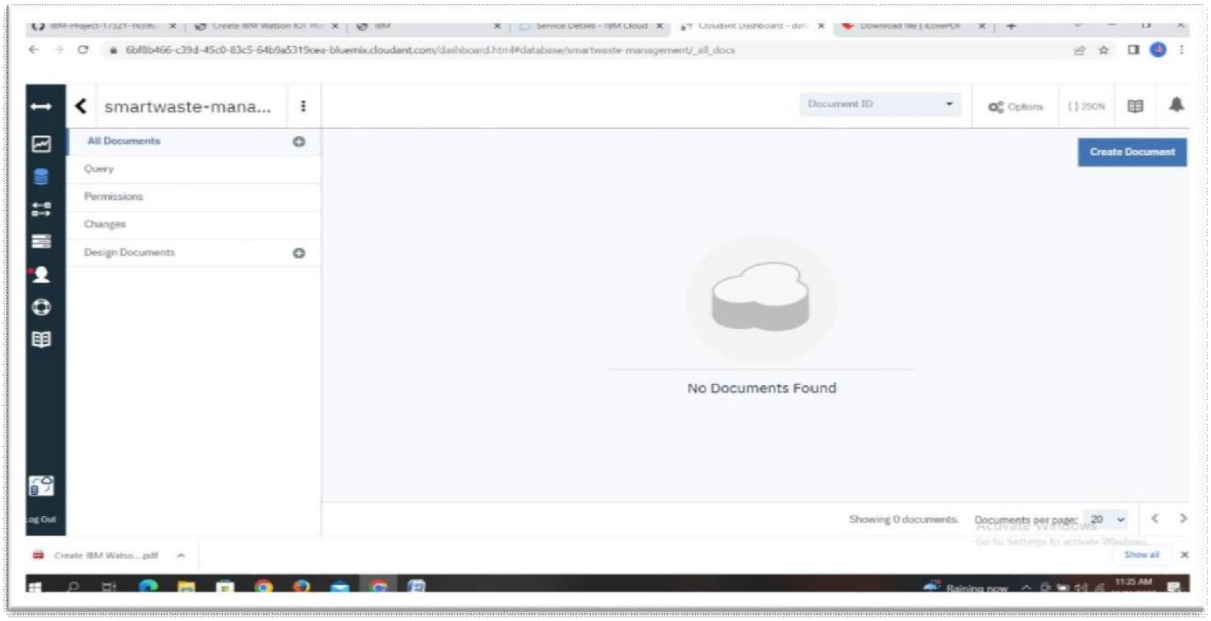


Fig 7.3.3 Smart waste Management DB

CHAPTER 8

TESTING

8.1 TEST CASES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user exceptions and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.2 USER ACCEPTANCE TESTING

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/understand the software system before moving the software application to the invention environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements

Test Results: All the test cases mentioned above are passed approximately with somedefects encountered.

CHAPTER 9

RESULTS

9.1 PERFORMANCE METRICES

Performance metrics are defined as figures and data representative of an organization's actions, abilities, and overall quality. There are many different forms of performance metrics, including sales, profit, return on investment, customer happiness, customer reviews, personal reviews, overall quality, and reputation in a marketplace. Performance metrics can vary considerably when viewed through different industries.

Performance metrics are integral to an organization's success. It's important thatorganizations select their chief performance metrics and focus on these areas because these metrics help guide and gauge an organization's success. Key success factors are only useful if they are acknowledged and tracked. Business measurements must also be carefully managed to make sure that they give right answers, and that the right questions are being asked.

S.No.	Parameter	Values
1.	Model Summary	These may be determined by organizational policy, adherence to a published standard or an analysis of the requirements based on use, ability to measure, or more.
2.	Accuracy	Training Accuracy somedefects Encountered Validation Accuracy Suc–cessfully Passed

Table 9.1.1 Performance metrics

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Improve Productivity and Performance.
- Increase Profitability.
- Boost Sustainability.
- Superior Customer Engagement.
- Become a Smart City.
- Enhance Safety.

DISADVANTAGES

- These results into high initial cost due to expensive smart dustbins compare to other methods.
- Sensor nodes used in the dustbins have limited memory size.
- Wireless technologies used in the system such as zigbee and wifi have shorter range and lower data speed.
- It requires a well structured hardware.
- The onetime cost of installation will be higher than the present technique.

CHAPTER 11

CONCLUSION

Project work is the implementation of smart waste management system using ultrasonic sensor, arduino and Wi-Fi modules such as sensing modules, user modules. This system assures the cleaning of dustbins soon when the garbage level reaches its maximum. This reduces the total number of trips of garbage collection vehicle and hence reduces the overall expenditure associated with the garbage collection. It ultimately helps to keep cleanliness in the society. Therefore, the smart garbage management system makes the garbage collection more efficient. Smartdustbin helps us to reduce pollution. This project ensures waste collection on time which in turn ensures less contamination of environment, no spread of disease and a cleaner surrounding.

CHAPTER 12
FUTURE SCOPE

The future of Waste management starts and proceeds with technological adjustments. Like every other industry, to proceed, the waste management industry needs to become digitized and data-driven to advance its work field. The future is smart and competitive! Especially for businesses, they are required to be one step ahead of their competitors. When smart waste management solutions are applied over time, the data is collected. These data in hand sensors can be used to identify fill patterns, optimize driver routes and schedules, and reduce operational costs. These sensors' cost is steadily decreasing, making smart bins more feasible to implement and more attractive to companies or city leaders. When we say that the future is smart, it also means that it is practical. The selection of the containers minimizes the need fortrash collection crews. The amount of labor and time spent on collection processes is minimized, and ultimately it's profitable. In addition to hardware, the time spent is reduced for management and reduced by using easy to use compact and comprehensive platforms and mobile apps for both ends of the waste management processes.

CHAPTER 13

APPENDIX

Source code

```
from flask import Blueprint, jsonify, request, abort, Response, send_file

from app.models import Basket, User, Waste, Vehicle, Employee, commit, Area,

SoftwareVersion,
BasketType from
app.validate import validate
import json
from datetime import
datetimefrom io import
BytesIO
api = Blueprint('api',_____name___, url_prefix='/api')


@api.rout
e('/')def
index():
return jsonify({"message": "the api is working"})
@api.route('baskets'
) def get_baskets():
baskets = Basket.query.all()

baskets_list = [basket.format() for basket in baskets]
return jsonify({
  "baskets": baskets_list,
  "total_baskets": len(baskets_list)
})


@api.route('baskets/<int:basket_id>'
) def get_basket(basket_id):
basket =
Basket.query.get(basket_id)return
jsonify({
  "basket": basket.format()
})
```

```
@api.route('baskets/<int:basket_id>/wastes
') def get_wastes_of_basket(basket_id):
basket = Basket.query.get(basket_id)

wastes = [waste.format() for waste in basket.wastes]
total_size = 0.0
for waste in wastes:

    total_size = total_size + waste['size']
    print(type(waste['size']), waste['size'])
    print(total_size)
    print(total_size)return jsonify({
    "basket_id": basket.id,
    "total_size": total_size,
    "wastes": wastes
})


@api.route('baskets', methods=['POST'])def add_new_basket():
data = request.json

roles = ['longitude', 'latitude', 'area_code']
  abort(400) if not validate(roles, data) else
  Nonelongitude = data['longitude']
latitude = data['latitude']
area_code = data['area_code']type_id = data['type']
area = Area.query.get(area_code)if not area:
    abort(422)

basket_type = BasketType.query.get(type_id)if not
  basket_type:
  abort(422)

    basket = Basket(longitude=longitude, latitude=latitude, area=area,
basketType=basket_type).save()


return jsonify({
    "success": True,

    "basket": basket.format()

})
```

```
@api.route('baskets/<int:basket_id>',
methods=['DELETE'])def delete_baskets(basket_id):

basket = Basket.query.get(basket_id)

basket = basket.delete() if basket else basket
return jsonify({
    'success': bool(basket)

})
@api.route('baskets',
methods=['PATCH'])def
update_all_baskets():
data = request.json

software_version = data['software_version']

baskets = Basket.query.update({Basket.software_version: software_version})
commit()
return jsonify({
    "baskets_update": baskets
})
@api.route('baskets/<int:basket_id>', methods=['PATCH'])
def update_the_basket(basket_id):
data = request.json
basket_level = data['level']
if basket_level is None:
    abort(400)
try:
    basket = Basket.query.get(basket_id)
    basket.wastes_height = basket_level

basket.save()except:
    abort(422)
return jsonify({
    "success": True,
})
@api.route('areas')def
get_areas():
areas = Area.query.all()
areas_list = [area.format() for area in areas]
return jsonify({
```

```
"total_areas": len(areas_list),
"areas": areas_list
})
@api.route('areas/<int:area_code>')
def get_area(area_code):
area = Area.query.get(area_code)
return jsonify({
    "area": area.format()
})
@api.route('areas/<int:area_code>/baskets') def
get_basket_belong_to_area(area_code):
baskets = Area.query.get(area_code).baskets
baskets_list = [basket.format() for basket in baskets]
return jsonify({
    "total_baskets": len(baskets_list),
    "baskets": baskets_list
})
@api.route('areas/<int:area_code>/users')def
get_user_belong_to_area(area_code):
users = Area.query.get(area_code).users
users_list = [user.format() for user in users]
return jsonify({
    "total_users": len(users_list),
    "users": users_list
})
@api.route('areas',
methods=['POST'])def
insert_new_area():
data = request.json
roles = ['area_code', 'area_name', 'area_size', 'longitude', 'latitude', 'city']
if not validate(roles, data):
    abort(422)
code = data['area_code']
name = data['area_name']
size = data['area_size']
longitude = data['longitude']
latitude = data['latitude']
```

```
city = data['city']
area = Area(code=code, name=name, size=size, longitude=longitude,
latitude=latitude, city=city)



area.save(True)
return jsonify({
    "success": True,
    "area": area.format()
})
@api.route('vehicles')
def get_vehicles():
vehicles = Vehicle.query.all()

vehicles_list = [vehicle.format() for vehicle in vehicles]
return jsonify({
    "vehicles": vehicles_list

})
@api.route('vehicles/<int:vehicle_plate_no>'
) def get_vehicle(vehicle_plate_no):
vehicle = Vehicle.query.get(vehicle_plate_no)
return jsonify({

    "vehicle": vehicle.format()

})
@api.route('vehicles',
methods=['POST'])def
create_vehicle():
data = request.json

roles = ['plate_number', 'container_size', 'tank_size', 'employee_ssn']
if not validate(roles, data):
    abort(400)

plate_number = data['plate_number']
container_size = data['container_size']
tank_size = data['tank_size']
employee_ssn = data['employee_ssn']
driver = Employee.query.get(employee_ssn)
```

```
if not driver:
abort(404)
# try:
vehicle = Vehicle(plate_number=plate_number, container_size=container_size,
tank_size=tank_size, driver=driver)



vehicle.save(True)
return jsonify({
"success": True,
"vehicle": [vehicle.format()]
})
# except:
#abort(422)

@api.route('employees') def
get_employees():
employees = Employee.query.all()
employees_list = [employee.format() for employee in employees]
return jsonify({
"total_employees": len(employees_list),
"employees": employees_list
})
@api.route('employees/<int:employee_ssn>'
) def get_employee(employee_ssn):
employee = Employee.query.get(employee_ssn)
return jsonify({
"employee": employee.format()
})
@api.route("employees",
methods=['POST'])def
create_new_employee():
data = request.json
ssn = data['ssn']
full_name = data['full_name']
```

```
user_name = data['user_name']
password = data['password']
date_of_birth = data['data_of_birth']
phone = data['phone']
print(ssn)

if not ssn or not full_name or not user_name or not password or not date_of_birth
or not phone:
abort(400)



employee = Employee(SSN=ssn, full_name=full_name, user_name=user_name,
password=password, DOB=date_of_birth,
phone=phone).save(True)return
jsonify({
"success": True,

# "employee": [employee.format()]

})
@api.route("employees",
methods=['PATCH'])def
update_supervisor():
# TODO update the supervisor for all employee
return ''

@api.route('employees/<int:employee_ssn>', methods=['DELETE'])
def delete_employee(employee_ssn):
employee = Employee.query.get(employee_ssn)
if employee is None:
abort(404)
employee.update()return jsonify({
"success": True

})
@api.route('users')def get_all_users():
users = User.query.all()

users_list = [user.format() for user in users]return
jsonify({"user": users_list})
```

```
@api.route('users', methods=['POST'])def
create_new_user():
data = request.json

user_name = data['user_name']first_name =
data['first_name']last_name = data['last_name'] email =
data['email']
password = data['password']gender = data['gender']
area = data['area_code'] area = Area.query.get(area)if not
area:
abort(404)

roles = ['user_name', 'first_name', 'last_name', 'email', 'password', 'gender']


 abort(400)

 if not validate(roles, data) else Nonetry:
user = User(user_name=user_name, first_name=first_name,
last_name=last_name, email=email, password=password,
gender=gender, area=area).save(True)return
jsonify({
"success": True, 'user':
user.format()
})

except:

abort(422)


@api.route('wastes') def get_waste():
data = request.args

basket_id = data.get('basket_id', 0, int)

wastes = Waste.query.all() if not basket_id else
Waste.query.filter_by(basket_id=basket_id).all()
wastes_list = [waste.format() for waste in wastes]total_size = 0
for waste in wastes_list: total_size += +waste['size']


return jsonify({ "total_wastes_size": total_size,"wastes": wastes_list,
})
```

```
@api.route('wastes', methods=['POST'])def insert_new_waste():
data = request.json

basket = Basket.query.get(data['basket_id'])

is_full = basket.set_wastes_height(data['waste_height'])if is_full:
abort(422)

waste_size = basket.get_waste_volume(data['waste_height'])

waste = Waste(size=waste_size, type='bio', DOC=datetime.utcnow(),
basket=basket).save()
return jsonify({

"basket_level": basket.get_basket_level(),"waste": waste.format()
})
@api.route('wastes', methods=['DELETE'])def delete_waste():
# waste = Waste.query.filter_by(type='bio').delete()#
db.session.commit()
waste = Waste.query.all()print(waste)


return "

@api.route('test', methods=['POST', "GET"])def test():
data = request.jsonreturn jsonify({
"value": data['value']

})
@api.route("/baskets_types") def get_basket_type():
types_of_baskets = BasketType.query.all()

type_list = [type_of_basket.format() for type_of_basket in types_of_baskets]
return jsonify({
"types": type_list

})
@api.route("/baskets_types", methods=["POST"])def create_basket_type():
data = request.json length = data["length"]height = data["height"]width =
data["width"]
micro_controller = data["micro_controller"]roles = ['length', 'height', 'width']

abort(400) if not validate(roles, data) else Nonetry:
    basket_type = BasketType(length=length, height=height, width=width,
```

```
        micro_controller=micro_controller).save()
    return jsonify({
    "success": True,
        'Type': basket_type.format()
})
except:
    abort(422)
@api.route('/baskets/<int:basket_id>/versio
ns') def
get_basket_software_version(basket_id):
basket = Basket.query.get(basket_id)
software_versions =
SoftwareVersion.query.filter_by(basket_id=basket_id).order_by(SoftwareVersion.dat
e.desc()).all()
list_software_version = []
status = 'update'
for software_version in software_versions:
if software_version.version == basket.software_version:status
= 'rollback'
list_software_version.append(software_version.format('current'))
continue
list_software_version.append(software_version.format(status))return
jsonify({

"software_versions": list_software_version,
"current_version": basket.software_version
})
@api.route("/software_versions/<string:version>"
) def get_file(version):
software = SoftwareVersion.query.get(version)
file_name = "{}.bin".format(software.version)
return send_file(BytesIO(software.file), attachment_filename=file_name,
as_attachment=True)
@api.route("/software_versions",
methods=["POST"])def post_file():
file = request.files['file']
update_type = request.form.get("update_type", None)
```

```
basket_id = request.form.get("basket_id", None)
basket = Basket.query.get(basket_id)
    last_version =
SoftwareVersion.query.filter_by(basket_id=basket_id).order_by(SoftwareVersion.dat
e.desc()).first()
if last_version:major, minor, patch = last_version.version.split(".")if update_type == "patch":
patch = int(patch) + 1
elif update_type == "minor":
        minor = int(minor) +
1patch = 0
elif update_type == "major":
major = int(major) + 1 patch =
0
    minor = 0
else:
    abort(422)
print(last_version.version.split())
    version = "{}.{}.{}".format(major, minor, patch)
else:
    version = "0.1.0"
print(version)
print(last_version)
    software_version = SoftwareVersion(version=version, file=file.read(),
basket=basket)
software_version.save(True)
return jsonify({
"success": True,
"version": software_version.version
}), 201
```

Github Link

https://github.com/IBM-EPBL/IBM-Project-47741-1660801872

Demonstration Link

https://drive.google.com/file/d/1g6p7eg6HIOERET9dG5-nUAwKeOuY97G3/view?usp=sharing