IOT Based SMART DUSTBIN

MINI PROJECT REPORT

Submitted by

SANJAI KUMARAN	2116230701284
MONESH G J	2116230701193

In partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

BONAFIDE CERTIFICATE

Certified that this project "SMART DUSTBIN WITH REAL-TIME STOCK REDUCTION AND ALERTS **" is the bonafide work of "SANJAI KUMARAN (2116230701284) and MONESH G JB(2116230701193)" who carried out the project work under my supervision.**

SIGNATURE
Dr.N.Duraimurugan, M.Tech., Ph.D.

Associate Professor,
Computer Science & Engineering
Rajalakshmi Engineering College
(Autonomous)
Thandalam, Chennai -602105.

Submitted for the **ANNA UNIVERSITY** practical examination Mini-Project work viva voice held on______________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincerethanks to our Chairman **Mr. S.MEGANATHAN, B.E, F.I.E.,** our Vice Chairman**Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respectedChairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.,** for providing us withthe requisite infrastructure and sincere endeavoring in educating us in their premierinstitution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principalfor his kind support and facilities provided to complete our work in time. We express oursincere thanks to **Dr. P. KUMAR, M.E., Ph.D.,** Professor and Head of the Department ofComputer Science and Engineering for his guidance and encouragement throughout theproject work.

We also extend our sincere and hearty thanks to our Internal Guide **Dr.N.Duraimurugan,M.Tech., Ph.D.**Associate Professor, Department of Computer Science and Engineering for his valuable guidance and motivation during the completion of this project. Our sincere thanks to our family members, friends and other staff members of information technology.

SANJAIKUMARAN	**2116230701284**
MONESH G J	**2116230701193**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

ABBREVIATION	ACRONYM
IOT	Internet of Things
HTTP	HyperText Transfer Protocol
TEMP	Temperature
DHT	Digital Humidity and Temperature
SQL	Structured Query Language

ABSTRACT

The IoT-based Smart Dustbin is a modern solution to urban waste management, designed to automatically open its lid using an ultrasonic sensor and notify the status of the bin through IoT technology. This project reduces human effort, promotes hygiene, and helps municipal corporations track bin levels in real time.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Urbanization has led to increased waste generation, posing challenges in waste management. Traditional methods often result in overflowing bins, leading to unsanitary conditions and environmental hazards. Integrating IoT into waste management offers a solution by enabling real-time monitoring and efficient collection strategies.

1.2 SCOPE OF THE WORK

- Develop a smart dustbin system that monitors waste levels in real-time.
- Automate the lid mechanism for hands-free operation.
- Transmit waste level data to a centralized platform for monitoring.
- Notify waste management authorities when bins are full.

1.3 PROBLEM STATEMENT

Manual monitoring of waste bins is inefficient and often leads to delayed waste collection. There's a need for an automated system that provides real-time data on bin status to optimize collection schedules and maintain hygiene.

1.4 AIM AND OBJECTIVES OF THE PROJECT

To implement automatic smart dustbin eduction using RFID and ultrasonic sensors. To notify administrators of low stock levels via buzzer and email alerts.To integrate a responsive web app for real-time Smart dustbin control and billing. To enhance overall efficiency and reduce human error in tracking.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1IOT DEVICES

- Arduino Uno or NodeMCU ESP8266
- Ultrasonic Sensor (HC-SR04)
- Servo Motor (SG90)
- Wi-Fi Module (ESP8266 – if using Arduino Uno)
- Breadboard
- Jumper Wires
- USB Cable or 9V Battery with Connector
- Dustbin (Plastic or Metal)
- Resistors (220Ω, 10kΩ)
- LEDs
- LCD Display (16x2) *(optional)*
- Buzzer *(optional)*
- IoT Platform (Blynk / ThingSpeak)
- IFTTT or Webhooks *(optional)*
- Smartphone with Internet

2.2SOFTWARE SPECIFICATIONS

Operating System	Windows 11
Front – End	React JS
Back – End	Node JS
Browser	Google Chrome
IDE	Arduino IDE

CHAPTER 3
SYSTEM DESIGN

3.1 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components

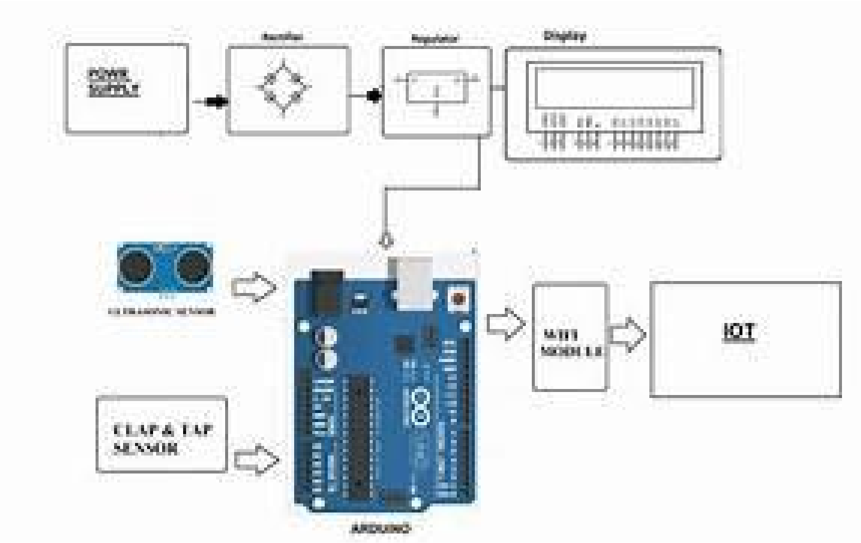

Figure 3.1 Architecture Diagram

From the above Figure 3.1, the architecture of the system is well understood.

3.2 USE CASE DIAGRAM

A use case is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modelling Language as an actor) and a system to achieve a goal. The actor can be a human or other external system.

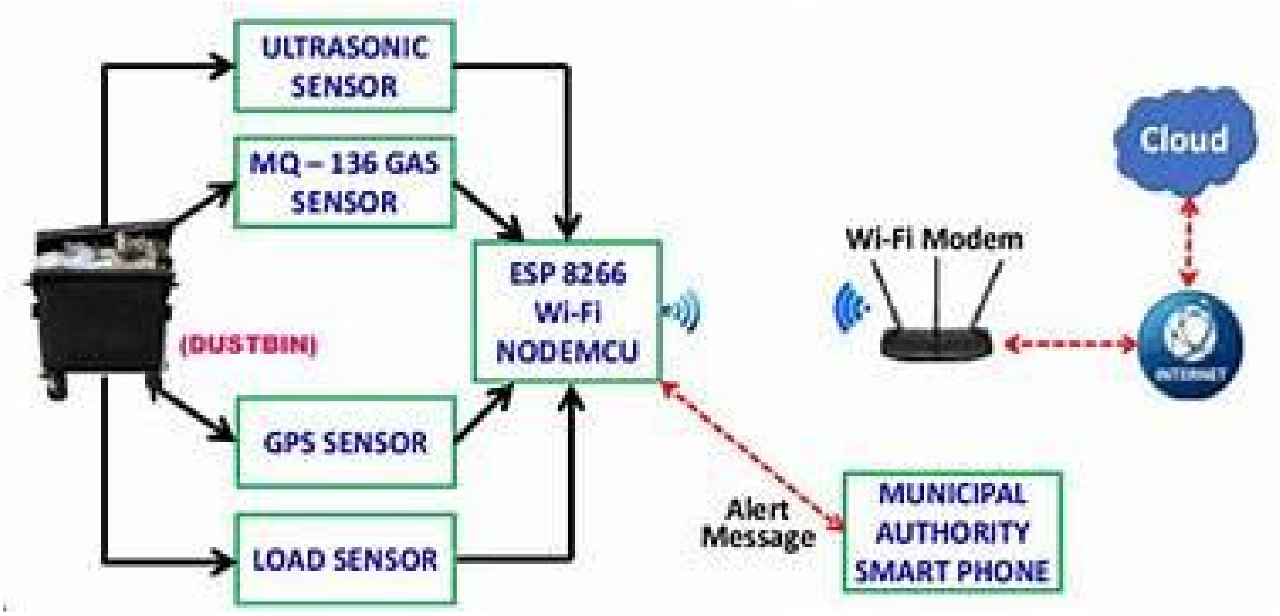

Figure 3.2 Use case diagram

From the above figure 3.2, the interactions between a role in the system is shown

3.3 ACTIVITY DIAGRAM

An activity in Unified Modelling Language (UML) is a major task that must take place in order to fulfill an operation contract. Activities can be represented inactivity diagrams. An activity can represent: The invocation of an operation. A step in a business process.

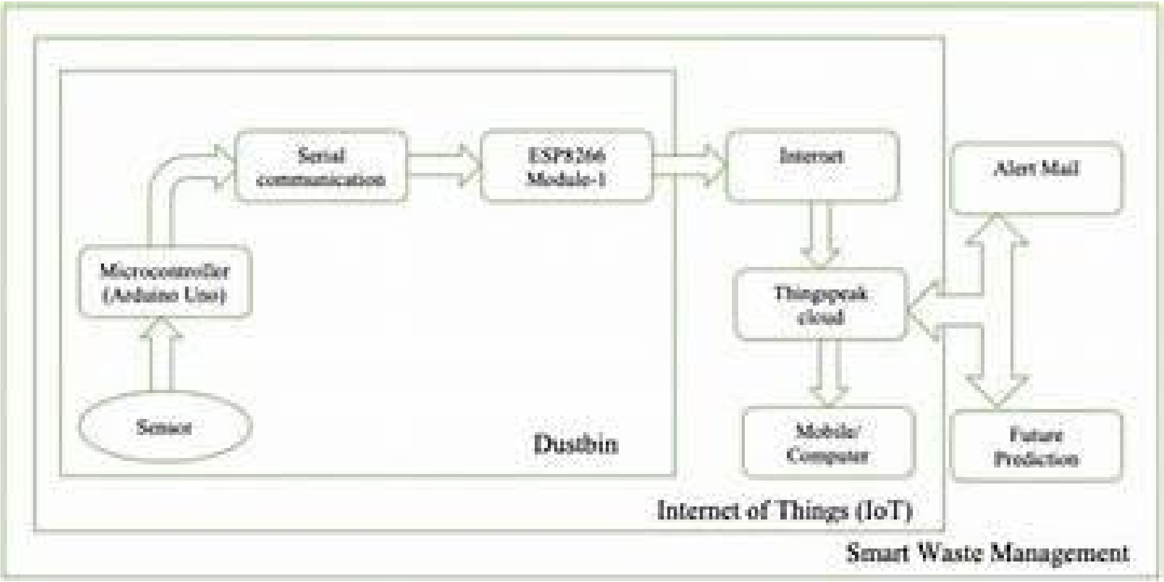

Figure 3.3 Activity Diagram

From the above figure 3.3, the activities of the system are shown

3.4 CLASS DIAGRAM

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modelling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

```
      +---------------------+
|     SmartDustbin    |
+---------------------+
| - binID: String     |
| - location: String  |
| - fillLevel: float  |
| - isLidOpen: bool   |
+---------------------+
| +checkFillLevel()   |
| +openLid()          |
| +closeLid()         |
| +sendStatus()       |
+---------------------+
      |
      | 1
      |----------------+
                  | uses
                  v
        +-------------------------+
        |    UltrasonicSensor    |
        +-------------------------+
        | - distance: float       |
        +-------------------------+
        | +measureDistance():float|
        +-------------------------+

        +------------------------+
        |      ServoMotor       |
        +------------------------+
        | - angle: int          |
        +------------------------+
        | +rotate(angle:int)     |
        +------------------------+

        +------------------------+
        |       WiFiModule       |
        +------------------------+
        | - ssid: String         |
        | - password: String     |
        +------------------------+
        | +connect()             |
| +sendData(data: String)|
+------------------------+

+-------------------------+
|    CloudServer/API    |
+-------------------------+
| - database: String      |
+-------------------------+
| +storeData()            |
| +displayStatus()        |
+-------------------------+
```

Figure 3.4 Class Diagram

The above Figure 3.4 is the class diagram for the system.

CHAPTER 4

MODULE DESCRIPTION

4.1 Sensor Module

Utilizes the HC-SR04 ultrasonic sensor to measure the distance from the lid to the waste surface, determining the bin's fill level.

4.2 Control Module

The microcontroller (Arduino Uno or NodeMCU) processes sensor data and controls the servo motor for lid operation. It also manages data transmission to the IoT platform.

4.3 Communication Module

Employs the ESP8266 Wi-Fi module to connect the system to the internet, enabling real-time data updates and remote monitoring.

4.4 Power Module

Provides necessary power to the system components, ensuring uninterrupted operation.

4.5 User Interface Module

A mobile application or web dashboard displays real-time bin status, historical data, and sends notifications when the bin is full.

CHAPTER 5

TABLE

5.1 MEDICINE TABLE

S.NO	ATTRIBUTE	TYPE
1	MED_ID	NUMBER(5)
2	MED_NAME	VARCHAR(45)
3	MANUFACTURER	VARCHAR(45)
4	PREF_MIN_TEMP	NUMBER(5,2)
5	PREF_MIN_HUM	NUMBER(5,2)
6	PREF_MIN_LIGHT	NUMBER(7,2)
7	PREF_MAX_TEMP	NUMBER(5,2)
8	PREF_MAX_HUM	NUMBER(5,2)
9	PREF_MAX_LIGHT	NUMBER(7,2)

5.2 STORAGE TABLE

S.NO	ATTRIBUTE	TYPE
1.	STORAGE_ID	NUMBER(5)
2.	MED_ID	NUMBER(5)
3.	MFD	DATE
4.	EXPIRY_DATE	DATE

5.3 HISTORY TABLE

S.NO	ATTRIBUTE	TYPE
1.	HISTORY_ID	NUMBER(5)
2.	STORAGE_ID	INTEGER
3.	TIME	TIMESTAMP WITHOUT TIMEZONE
4.	TEMP	NUMBER(5,2)
5.	LUX	NUMBER(7,2)
6.	HUMIDITY	NUMBER(5,2)

5.4 CURRENT DATA TABLE

S.NO	**ATTRIBUTE**	**TYPE**
1.	TEMP_IN_C	NUMBER(5,2)
2.	TEMP_IN_F	NUMBER(5,2)
3.	LIGHT	NUMBER(7,2)
4.	HUMIDITY	NUMBER(5,2)

CHAPTER 6

SAMPLE CODING

ESP8266-12E NODEMCU Program

```
#include <Servo.h>

#define TRIGGER_PIN  9 // Arduino pin connected to the trigger pin of
ultrasonic sensor
#define ECHO_PIN     10 // Arduino pin connected to the echo pin of
ultrasonic sensor
#define SERVO_PIN    11 // Arduino pin connected to the signal pin of
servo motor
#define MAX_DISTANCE 20 // Maximum distance threshold for
triggering servo (in centimeters)

Servo servo;

void setup() {
  Serial.begin(9600);
  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  servo.attach(SERVO_PIN);
}

void loop() {
  long duration, distance;
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);
  duration = pulseIn(ECHO_PIN, HIGH);
  distance = (duration / 2) / 29.1; // Calculate distance in centimeters

  if (distance <= MAX_DISTANCE) {
    // If object is within range, open the lid
    servo.write(90); // 90 degrees position (adjust as needed)
    delay(1000); // Wait for 1 second
  } else {
    // If no object is detected, close the lid
    servo.write(0); // 0 degrees position (adjust as needed)
  }

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(1000); // Wait for 1 second before taking the next reading
}
```

Web Application

1. index.js

```
          const express = require("express")
const app = express()
const PORT = 3001
const cors = require('cors');
const morgan = require('morgan');
const bodyParser = require("body-parser");

const corsOptions = require('./config/corsOptions')
const db = require('./config/DBConnect')
const medStorageRoute = require('./routes/med_storage.route');
const historyRoute = require('./routes/history.route');

app.use(bodyParser.urlencoded({ extended: false }));
app.use(cors(corsOptions));
app.use(express.json());
app.use(morgan('tiny'));

db.connect().then(()=>{
console.log("Postgres is connected");
})

app.get("/getSensorData", async(req, res) => {
res.send(response)
})
```

```
let StorageID, TemperatureInC, TemperatureInF, Humidity, Light;
let currentData;

app.post('/',async(req,res)=>{
   console.log(req.body);
currentData = req.body;
StorageID = req.body.StorageID;
TemperatureInC = req.body.TemperatureInC;
TemperatureInF = req.body.TemperatureInF;
   Humidity = req.body.Humidity;
   Light = req.body.Light;

if(TemperatureInC !='nan' && Light>0){
      await db.query("INSERT INTO HISTORY(STORAGE_ID, TEMP, LIGHT,
HUMIDITY) VALUES($1,$2,$3,$4)",[StorageID,TemperatureInC,Light,Humidity]);
      await db.query("UPDATE CURRENT_DATA SET TEMP_IN_C = $1, TEMP_IN_F =
$2, LIGHT = $3, HUMIDITY = $4",[TemperatureInC,TemperatureInF,Light,Humidity]);
console.log("Inserted with Light");
}elseif(TemperatureInC !='nan'){
      await db.query("INSERT INTO HISTORY(STORAGE_ID, TEMP, HUMIDITY)
VALUES($1,$2,$3)",[StorageID,TemperatureInC,Humidity]);
      await db.query("UPDATE CURRENT_DATA SET TEMP_IN_C = $1, TEMP_IN_F =
$2, HUMIDITY = $3",[TemperatureInC,TemperatureInF,Humidity]);
console.log("Inserted without Light");
   }
res.status(200).send("POST Request Received");
})

app.get('/currentData',async(req,res)=>{
   const result =await db.query("Select * from current_data");
res.status(200).send(result.rows[0]);
});


app.use('/meds-storage',medStorageRoute);
app.use('/history',historyRoute);
app.listen(PORT, () => {
console.log("Server is online")
})
```

1. Dashboard.jsx

```
import * as React from 'react';
import { styled, createTheme, ThemeProvider } from '@mui/material/styles';
import CssBaseline from '@mui/material/CssBaseline';
import MuiDrawer from '@mui/material/Drawer';
import Box from '@mui/material/Box';
import MuiAppBar from '@mui/material/AppBar';
import Toolbar from '@mui/material/Toolbar';
import List from '@mui/material/List';
import Typography from '@mui/material/Typography';
import Divider from '@mui/material/Divider';
import IconButton from '@mui/material/IconButton';
import Badge from '@mui/material/Badge';
import Container from '@mui/material/Container';
import Grid from '@mui/material/Grid';
import Paper from '@mui/material/Paper';
import Link from '@mui/material/Link';
import MenuIcon from '@mui/icons-material/Menu';
import ChevronLeftIcon from '@mui/icons-material/ChevronLeft';
import NotificationsIcon from '@mui/icons-material/Notifications';
import { mainListItems, secondaryListItems } from './listItems';
import Chart from './Chart';
import Deposits from './Deposits';
import Orders from './Orders';


function Copyright(props) {
  return (
<Typography variant="body2" color="text.secondary" align="center" {...props}>
      {'Copyright © '}
<Link color="inherit" href="https://mui.com/">
        Your Website
</Link>{' '}
      {new Date().getFullYear()}
      {'.'}
</Typography>
  );
}

const drawerWidth = 240;

const AppBar = styled(MuiAppBar, {
shouldForwardProp: (prop) =>prop !== 'open',
})(({ theme, open }) => ({
zIndex: theme.zIndex.drawer + 1,
  transition: theme.transitions.create(['width', 'margin'], {
    easing: theme.transitions.easing.sharp,
    duration: theme.transitions.duration.leavingScreen,
  }),
...(open && {
marginLeft: drawerWidth,
    width: `calc(100% - ${drawerWidth}px)`,
    transition: theme.transitions.create(['width', 'margin'], {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.enteringScreen,
    }),
  }),
}));

const Drawer = styled(MuiDrawer, { shouldForwardProp: (prop) =>prop !== 'open' })(
({ theme, open }) => ({
    '& .MuiDrawer-paper': {
      position: 'relative',
whiteSpace: 'nowrap',
      width: drawerWidth,
      transition: theme.transitions.create('width', {
        easing: theme.transitions.easing.sharp,
        duration: theme.transitions.duration.enteringScreen,
      }),
boxSizing: 'border-box',
...(!open && {
overflowX: 'hidden',
        transition: theme.transitions.create('width', {
          easing: theme.transitions.easing.sharp,
          duration: theme.transitions.duration.leavingScreen,
        }),
        width: theme.spacing(7),
        [theme.breakpoints.up('sm')]: {
          width: theme.spacing(9),
        },
      }),
    },
  }),
);
         const defaultTheme = createTheme();
export default function Dashboard() {
  const [open, setOpen] = React.useState(true);
  const toggleDrawer = () => {
setOpen(!open);
  };
            return (
<ThemeProvider theme={defaultTheme}>
<Box sx={{ display: 'flex' }}>
<CssBaseline />
<AppBar position="absolute" open={open}>
<Toolbar
sx={{
              pr: '24px', // keep right padding when drawer closed
            }}
>
<IconButton
```

```
            edge="start"
            color="inherit"
            aria-label="open drawer"
onClick={toggleDrawer}
sx={{
marginRight: '36px',
...(open &&{ display: 'none' }),
            }}
>
<MenuIcon />
</IconButton>
<Typography
            component="h1"
            variant="h6"
            color="inherit"
noWrap
sx={{ flexGrow: 1 }}
>
            Dashboard
</Typography>
<IconButton color="inherit">
<Badge badgeContent={4} color="secondary">
<NotificationsIcon />
</Badge>
</IconButton>
</Toolbar>
</AppBar>
```

```
<Drawer variant="permanent" open={open}>
<Toolbar
sx={{
            display: 'flex',
alignItems: 'center',
justifyContent: 'flex-end',
px: [1],
          }}
>
<IconButtononClick={toggleDrawer}>
<ChevronLeftIcon />
</IconButton>
</Toolbar>
<Divider />
<List component="nav">
          {mainListItems}
<Divider sx={{ my: 1 }} />
          {secondaryListItems}
</List>
</Drawer>
<Box
        component="main"
sx={{
backgroundColor: (theme) =>
theme.palette.mode === 'light'
              ? theme.palette.grey[100]
              : theme.palette.grey[900],
flexGrow: 1,
          height: '100vh',
          overflow: 'auto',
        }}
>
<Toolbar />
<Container maxWidth="lg" sx={{ mt: 4, mb: 4 }}>
<Grid container spacing={3}>
            {/* Chart */}
<Grid item xs={12} md={8} lg={9}>
<Paper
sx={{
                 p: 2,
                 display: 'flex',
flexDirection: 'column',
                 height: 240,
               }}
>
<Chart />
</Paper>
</Grid>
            {/* Recent Deposits */}
<Grid item xs={12} md={4} lg={3}>
<Paper
sx={{
                 p: 2,
                 display: 'flex',
flexDirection: 'column',
                  height: 240,
                }}
>
<Deposits />
</Paper>
</Grid>
             {/* Recent Orders */}
<Grid item xs={12}>
<Paper sx={{ p: 2, display: 'flex', flexDirection: 'column' }}>
<Orders />
</Paper>
</Grid>
</Grid>
<Copyright sx={{ pt: 4 }} />
</Container>
</Box>
</Box>
</ThemeProvider>
  );
}
```

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

The IoT-based Smart Dustbin system presents an innovative and efficient approach to addressing the growing challenges of urban waste management. By integrating sensors and wireless communication technologies, the system enables real-time monitoring of waste levels and automates notifications when the bin reaches its capacity. This not only ensures timely waste collection but also contributes to maintaining a cleaner and healthier environment. The system reduces human intervention, minimizes overflow issues, and promotes hygienic waste disposal practices. Overall, it supports the vision of smart cities by enhancing the efficiency and responsiveness of municipal services, ultimately leading to better urban living standards.

Looking ahead, several improvements can be made to enhance the functionality and sustainability of the Smart Dustbin system. One major upgrade could be the integration of solar panels to make the unit energy-independent, allowing it to function efficiently in outdoor or remote locations without relying on conventional power sources. Another important enhancement is the inclusion of waste segregation capabilities using additional sensors to differentiate between biodegradable, non-biodegradable, and recyclable materials. This would support more effective recycling and reduce environmental impact.

REFERENCES

1. Patel, R., & Doshi, D. (2016). *IoT based Smart Garbage and Waste Collection Bin*. International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE), 5(5), 1576–1578.

2. Kumar, V., & Rajkumar, D. (2018). *Smart Waste Bin Monitoring Using IOT*. International Journal of Engineering Research & Technology (IJERT), 7(04), 360–363

3. Singh, P., & Nayak, S. (2020). *A Review on Smart Dustbin System Using IoT*. International Journal of Scientific & Technology Research, 9(1), 1460–1463.

4. Components101. (n.d.). *HC-SR04 Ultrasonic Sensor – Working, Pinout, Datasheet*. Retrieved from https://components101.com/ultrasonic-sensor-working-pinout-datasheet