GARBAGE MONITORING SYSTEM

CHANDRASEKAR.P

Register No: 1751002

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
M.C.A (Master of Computer Applications)
OF ANNA UNIVERSITY

April 2019

DEPARTMENT OF COMPUTER APPLICATIONS
COIMBATORE INSTITUTE OF TECHNOLOGY
(Autonomous Institution affiliated to Anna University)
COIMBATORE - 641014

COIMBATORE INSTITUTE OF TECHNOLOGY

(Autonomous Institution affiliated to Anna University)

COIMBATORE 641014

(Bonafide Certificate)

Mini-Project Work

Fourth Semester

GARBAGE MONITORING SYSTEM

Bonafide record of Work done by

CHANDRASEKAR.P

(Register No: 1751002)

Submitted in partial fulfillment of the

requirements for the degree of

M.C.A. (Master of Computer Applications)

of Anna University

April 2019

_____ _____

Faculty Guide Head of the Department

Submitted for the viva-voce held on _____

_____ _____

Internal Examiner External Examiner

CONTENTS

ACKNOWLEDGEMENT

I sincerely thank **Dr. V. SELLADURAI**, Principal, Coimbatore Institute of Technology, for giving me an opportunity to undertake this mini project and for his constant encouragement.

I am grateful to **Dr. P. G. SAPNA**, Head, Department of Computer Applications, Coimbatore Institute Of Technology, Coimbatore, for her continued encouragement and support during the course of this project.

My sincere thanks to my guide **Dr. P. G. SAPNA** Associate Professor, Department of Computer Application, Coimbatore Institute Of Technology, Coimbatore, for providing me valuable guidance in this project.

Thanks to guiding **Dr. M. POONGOTHAI** Associate professor, Department of Electronics and Communication Engineering.

Finally, we wish to thank my parents, friends, and well-wishers who have been a catalyst throughout the development of my project work.

SYNOPSIS

Garbage collection is one of the major challenges in the world. In India, given the population of more than 1 billion people, garbage collection and disposal is a major problem to be tackled. Improper/late collection and disposal if garbage adversely affects the health of all living beings to the planet we live in. Improper/late disposal leads to pollution of soil, air, and water and spread of disease. Hence, timely collection of garbage is an important factor for a clean and healthy environment.

To overcome the above issues the project “Garbage Monitoring System” aims at monitoring garbage fill in garbage bins across locations. Factors that delay collection of garbage on time include the paucity of resources like vehicles and drivers as well as information on garbage fill in the bin. The status of garbage bins is provided automatically to the authorities in the municipality/corporation and the drivers of the vehicle through a mobile application.

The Project aims to provide timely and correct information on the status of each garbage
bin through the help of the mobile application.

PREFACE

Chapter I – Introduction gives a detailed description of the objective and scope of the system. It also describes the environment in which the system runs.

Chapter II – System Analysis is the study phase where the client requirement is studied. This system requirement user-interaction hardware and software requirement and test plans.

Chapter III - System Design is the conversion of requirements specification into actual designs. The chapter gives the detailed design of architecture, circuits, the design of modules, user interfaces.

Chapter IV – System Testing tests the system at the development and deployment of the application process to fix the bug.

Chapter V – System Implementation discusses the steps involved in the implementation of the system.

Chapter VI – Conclusion put forth the successful features of the system and suggestions for future enhancements of this project

CHAPTER I

INTRODUCTION

Garbage consists of the unwanted material left over from Public area, Society, College, home. This system "Garbage Monitoring System" is motivated by Swachh Bharat Abhiyan Government of India. This System will help to minimize the garbage disposal problem. Garbage Monitoring is a very innovative system which will help to keep the cities clean. This system monitors the garbage bins and informs about the level of garbage in the garbage bins via Android Application. For this, the system uses ultrasonic sensors placed over the bins to detect the garbage level and compare it with the garbage bins depth. The system makes use of GPS and Node MCU Esp8266(wi-fi) for sending data to the cloud. An Android Application is used to view the level of waste in the bins. The Application gives the Location of the garbage bins and highlights the marker when the bin is full.

1.1 PROBLEM DEFINITION

OBJECTIVE

The primary objective of "**GARBAGE MONITORING SYSTEM***" is:*

1. Avoid garbage overflow from bins.
2. Timely collection of garbage through knowledge of bin filling states.
3. Provide alerts to garbage collector app based on garbage level in the bin based on set threshold level.
4. Provide a route for collection by providing knowledge of bins that are to be collected.
5. Prevent disease and another health environment because of poor garbage collection.

SCOPE

This project is used to reduce the garbage disposal problem and also avoid garbage overflowing from bins. It can help to keep cities clean. This project is also helpful in the government project of Swachh Bharat Abhiyan.

LIMITATIONS

1. A tracking device should run 24*7.
2. Tracking is difficult when the mobile is switched off.
3. Internet connection is compulsory.

SANITATION STAFF

In this module, deal with login interface and the user who wish to initialize bin status depending on whether the user is authenticated. After authentication user can able to see bin status and location of the filled bin.

Features of sanitation staff are:

- Authenticate using Android application.
- Viewing the Location on Google Maps.
- Sanitation staff finds the shortest path via Google Map.

ADMIN

Admin can able to view the user details and track the bin by using admin login.
They can modify user details.
A feature of admin are:

- Admin can add sanitation staff details.
- Admin can able to view the bin location on Google map.
- Admin can also view the bin status

1.2 SYSTEM ENVIRONMENT
REQUIREMENTS SPECIFICATION

Arduino can sense the environment by receiving input from Ultrasonic sensors and can affect its surroundings by controlling lights. The microcontroller on the board is programmed using the (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash, Processing).

The boards can be built by hand or purchased preassembled; the software can be downloaded for free. The hardware reference designs (CAD files) are available under an open-source license, you are free to adapt them as per your project requirement.

HARDWARE SPECIFICATION

For the Garbage Monitoring system, we have some module that will be attached to each other. this module together will perform as a GPS tracking.

- Processor: Intel core i5
- RAM: 8GB
- Hard Disk: 500GB

OTHER HARDWARE

- Node MCU 1.0 (wi-fi)
- GY-NEO6MV2 (GPS)
- Connecting Wires
- Ultrasonic sensor HC-SR04
- USB Cable
- LED lights

SOFTWARE SPECIFICATION

- Operating system: Windows 7
- Front End: Android Studio 3.3.2
- Back End: Thingspeak, Arduino IDE 1.8.5

CHAPTER II

SYSTEM ANALYSIS

2.1 SYSTEM DESCRIPTION

EXISTING SYSTEM

In Existing system used ultrasonic sensor and GSM module and Arduino Board. Ultrasonic sensor has to monitor the level of garbage in the bin. If waste is reaching the maximum level of the bin Gsm module has to send the particular member and view details in a Web application.

LIMITATIONS

- In the existing system used a web application to monitor the bin.
- The web application is not possible for monitoring frequently.

PROPOSED SYSTEM AND ITS FEATURES

In the proposed system here going to use the Internet of Things i.e. Node MCU and GPS module and the ultrasonic sensor which is helpful for monitor the bin level and find the location with help of the mobile application. This is designed to improve efficiency and reduce manual work. Whenever the bin is full sanitation staff can collect the waste. The mobile device helps to find the bin location on Google Map and view the level of waste in the bin on the Android application.

ADVANTAGES

- Easy to identify with the help of the hand-held device.
- Sensors help in receiving real-time date of each bin.
- Frequent monitoring of the Garbage bin.

DISADVANTAGES

- Internet connection is compulsory.
- When the mobile is switched off, the user cannot able to find a bin.

2.2 SOFTWARE REQUIREMENTS SPECIFICATION

HARDWARE REQUIREMENTS

NODE MCU

Purpose: To Send and receive data

Description:

Node MCU is an IoT platform (wi-fi module) it helps to connect local WLAN and to send/receive internet and data. Before the connection needs to write code on Arduino IDE. Once the code is uploaded it should start sending and receiving. In our project, we need to send data of location coordinates with the help of this module.

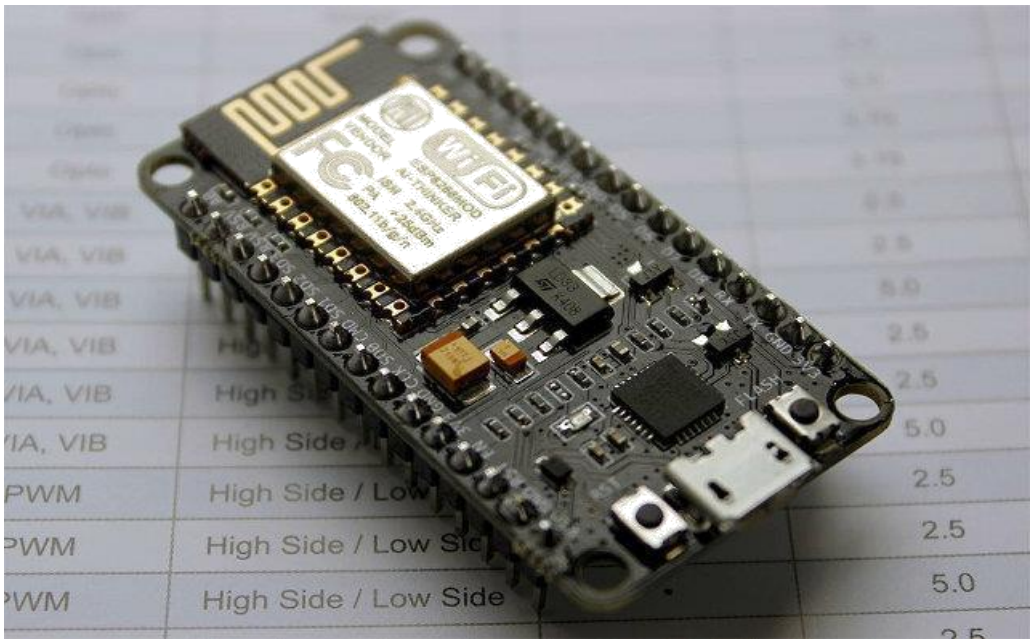


Figure: 2.1 Node MCU

The version of the board (V1.0) has the following specifications and features:

- Wi-Fi Module – ESP-12E module similar to ESP-12 module but with 6 extra GPIOs.
- USB – micro USB port for power, programming and debugging
- Headers – 2x 2.54mm 15-pin header with access to GPIOs, SPI, UART, ADC, and power pins
- Misc – Reset and Flash buttons
- Power – 5V via micro USB port
- Dimensions – 49 x 24.5 x 13mm

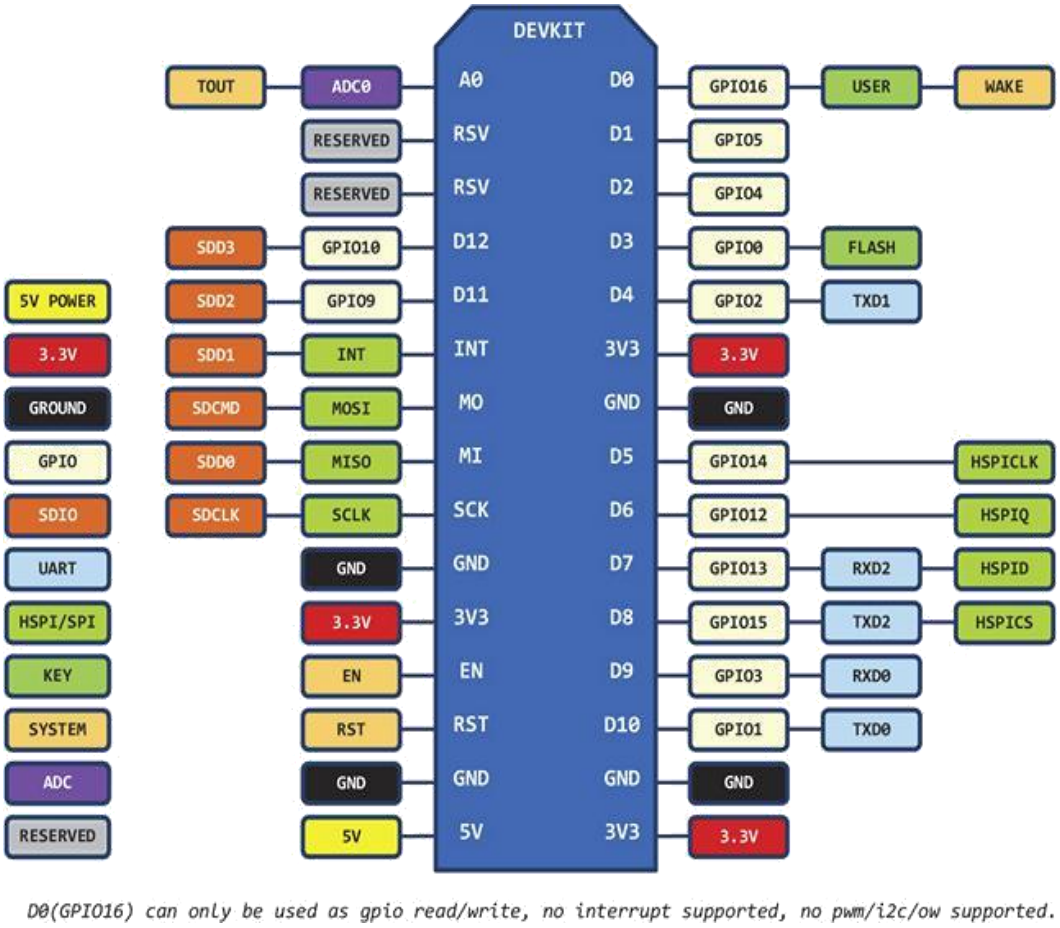


Figure:2.2 Pin of Node MCU

Pin of Node MCU

Node MCU provides access to the GPIO (General Purpose Input/Output) and for developing, purposes below pin mapping table from the API documentation should be referenced.

IO index	ESP8266 pin	IO index	ESP8266 pin
0 [*]	GPIO16	7	GPIO13
1	GPIO5	8	GPIO15
2	GPIO4	9	GPIO3

3	GPIO0	10	GPIO1
4	GPIO2	11	GPIO9
5	GPIO14	12	GPIO10

NEO6M GPS

Purpose: Getting started with GPS module

In this project, GY-NEO6MV2 (GPS module) is used. It is interconnected with the wi-fi module. The process of the module is to get location coordinate detail and the wi-fi module share the GPS data using Node MCU. This features the u-blox NEO-6M GPS module with antenna and built-in EEPROM. This is compatible with various flight controller modules that provide GPS computer test software. This is compatible with various flight controller boards designed to work with a GPS module.

Figure:2.3 NEO6M GPS

Specification

1. Supply Voltage: 2.7 to 3.6V
2. Supply current: 67 mA
3. Antenna gain: 50 dB
4. Operating temperature: -40 to 85°c
5. Antenna Type: Passive and active antenna.
6. Interfaces: UART, USB, SPI, DDC

7. Sensitivity:
    - Tracking & Navigation: -160 dBm
    - Reacquisition: -160 dBm
    - Cold Start (Autonomous): -146 dBm

Pin Description

- **Vcc**-Supply Voltage
- **Gnd**-Ground pin
- **TX and RX**-These 2 pins act as a UART interface for communication

Hardware connections

The connections are made as follows:

- Vcc to 3.3V
- Gnd to Gnd
- TX to Digital Pin 11
- RX to Digital Pin 10

ULTRASONIC SENSOR

Figure:2.4 Ultra Sonic Sensor

The Ultrasonic Sensor is used to measure the distance with high accuracy and stable readings. It can measure the distance from 2cm to 400cm or from 1 inch to 13 feet. It emits an ultrasound wave at the frequency of 40KHz in the air and if the object will come in its way then it will bounce back to the sensor. By using that time which it takes to strike the object and comes back, you can calculate the distance. Distance can be measured by equation 1. Distance = Time * sound speed /2. (1) Where Time = the time between an ultrasonic wave is received and transmitted. It has four pins. Two are VCC and GND which will be connected to the 5V and the GND of the Arduino while the other two pins are Trig and Echo pins which will be connected to any digital pins of the Arduino. The trig pin will send the signal and the Echo pin will be used to receive the signal. To generate an ultrasound signal, you will have to make the Trig pin high for about 10us which will send an 8-cycle sonic burst at the speed of sound and after striking the object, it will be received by the Echo pin.

Specifications of 5V / 5W SMPS Module:

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm
- Practical Measuring Distance: 2cm to 80cm
- Accuracy: 3mm
- Measuring angle covered: <15°
- Operating Current: <15mA
- Operating Frequency: 40Hz

SOFTWARE REQUIREMENTS

SYSTEM FEATURES

The system should provide the current location of the bin being tracked at request. The system position of the bin being tracked at the maximum level of the bin. The system should let admin can view the location on Google Map.

- Android Studio
- Arduino IDE
- Firebase

FUNCTIONAL REQUIREMENTS

Register

Admin can Register sanitation staff details using admin login.

Validation

The admin can log in into the application by providing their mail id and password and if valid they can only use this mobile application and admin can register the sanitation staff details then only staff can log in into the application.

View Bin status

Sanitation staff /admin able to view the level of garbage in the bin. This screen shows the garbage bin status.

Track Bin location

Sanitation staff /admin able to find where the garbage bin is locating on. This screen pointing the garbage location on Google Maps. The marker shows the location it helps to track a bin

Staff Details

Admin can able to view sanitation staff details.

NON-FUNCTIONAL SPECIFICATION

Nonfunctional requirements specify the system's quality characteristics or quality attributes. The system has a number of non-functional requirements. The non-functional requirements are classified in terms of security, performance, availability, reliability, and ease of use.

1. Usability

The system allows the staff to access information from the internet. It is a sanitation staff friendly application.

2. Reliability

The reliability of this system gives the right result within the minimum delay time.

3. Performance

Response time

The response time for each operation is minimal

Throughput

Several staff can access the system at the same time the system can able to respond to all staff.

4. Availability

The application is available for 24 hours because it is accessed through the internet.

5. Accuracy

This application gives an accurate location and level of garbage in the bin.

CHAPTER III

SYSTEM DESIGN

3.1 ARCHITECTURAL DESIGN

A three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage, and user interface are developed and maintained as independent modules on separate platforms. A three-tier architecture is a software design pattern and well-established software architecture. The three-tier architecture allows any one of the three tiers to be upgraded or replaced independently. The user interface is implemented on Android devices and uses a standard graphical user interface with different modules running on the web server. The relational database management system on the database server contains the computer data storage logic. The middle tiers are usually multi-tiered. It involves the client tier, application tier, and database tier.

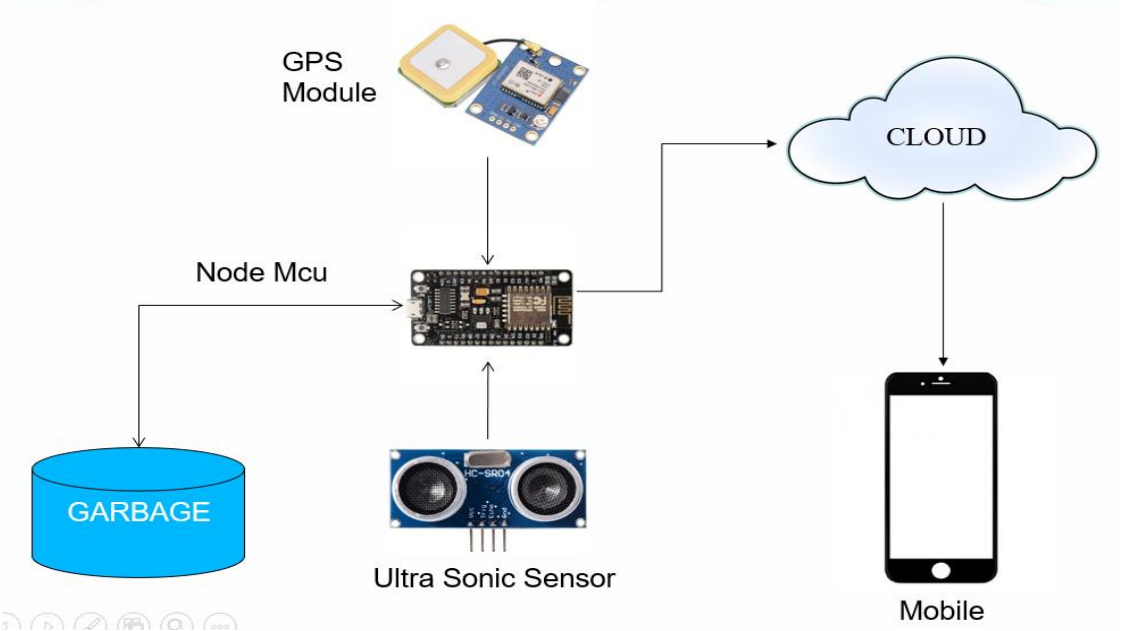


Figure: 3.1 Architecture Design of Garbage monitoring system

3.2 CIRCUIT DESIGN

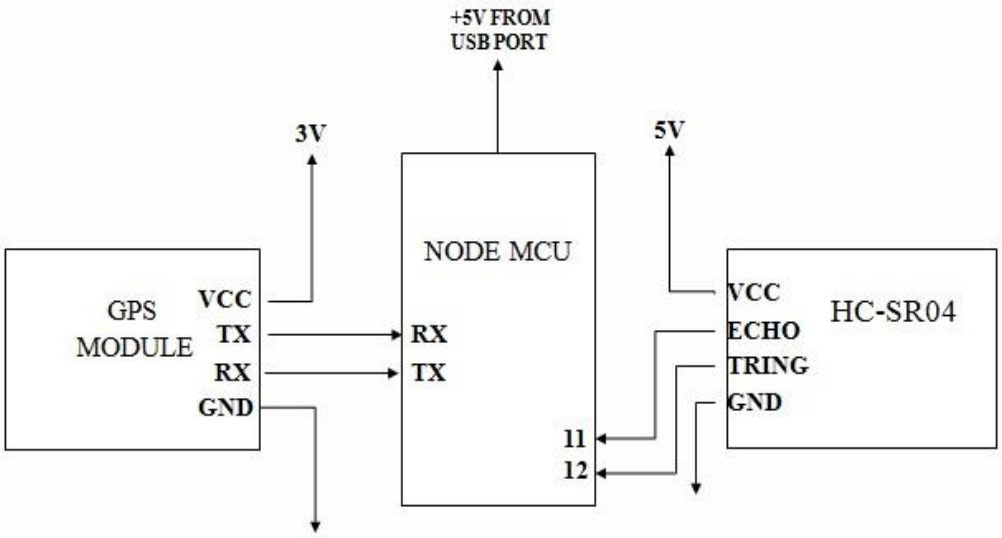


Figure: 3.2 Circuit design of Garbage monitoring system

The circuit diagram shows that soldered connection in between modules using dot board which can be detailed as below. There is two way of input current one from the Node MCU which is connected to PC via USB power get sourced and so we can upload the code using Arduino IDE. Another way of current inflow is from direct current which is enabled through power card after the program execution. The LED light is to check whether the tracking device is working correctly. The fuse holder is to hold fuse and Fuse is an electrical safety device that operates to provide overcurrent protection of an electrical circuit. Node MCU is open source IoT platform it is used to control the rest of the device by using Arduino IDE. The purpose of the Node MCU (wi-fi) module is to receive internet data and send GPS data to the android application. GPS is a receiver is capable of information from GPS satellites and then to calculate geographical position with the help of the wi-fi module.

3.3 STRUCTURAL DESIGN

BLOCK DIAGRAM

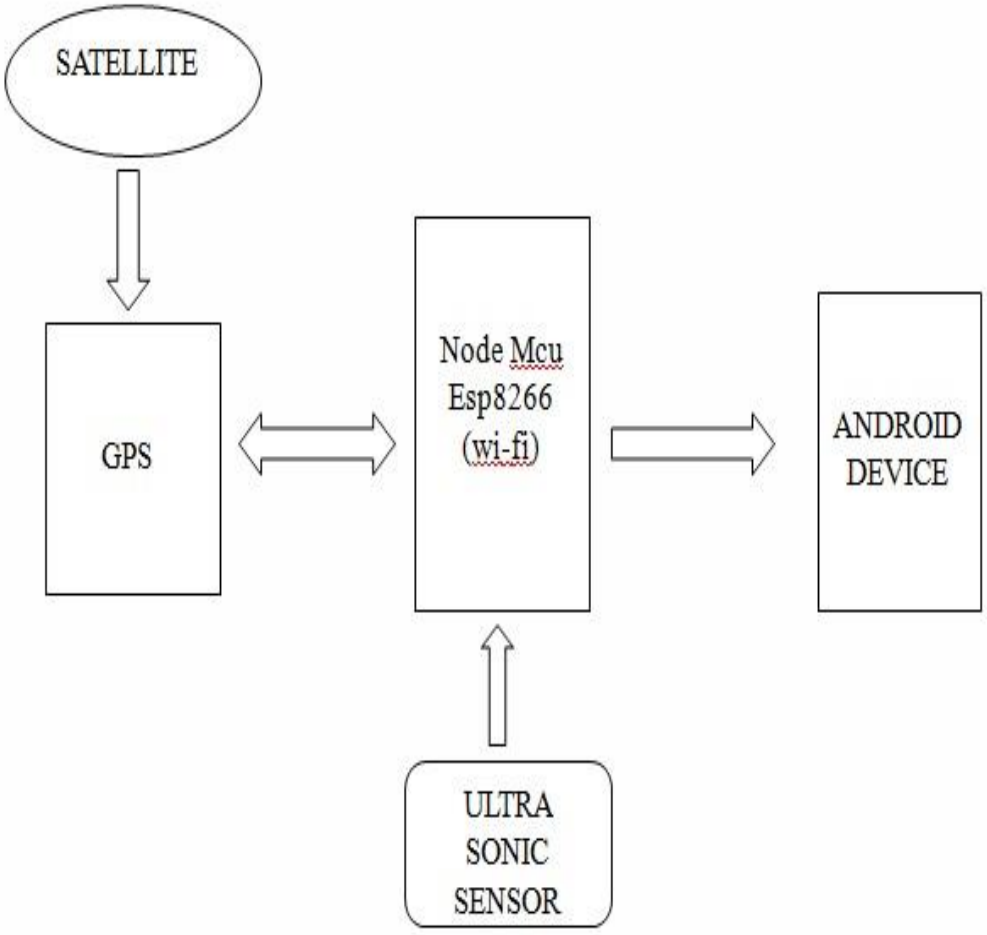


Figure: 3.3 Block diagram of Garbage Monitoring system

3.4 USE CASE MODEL

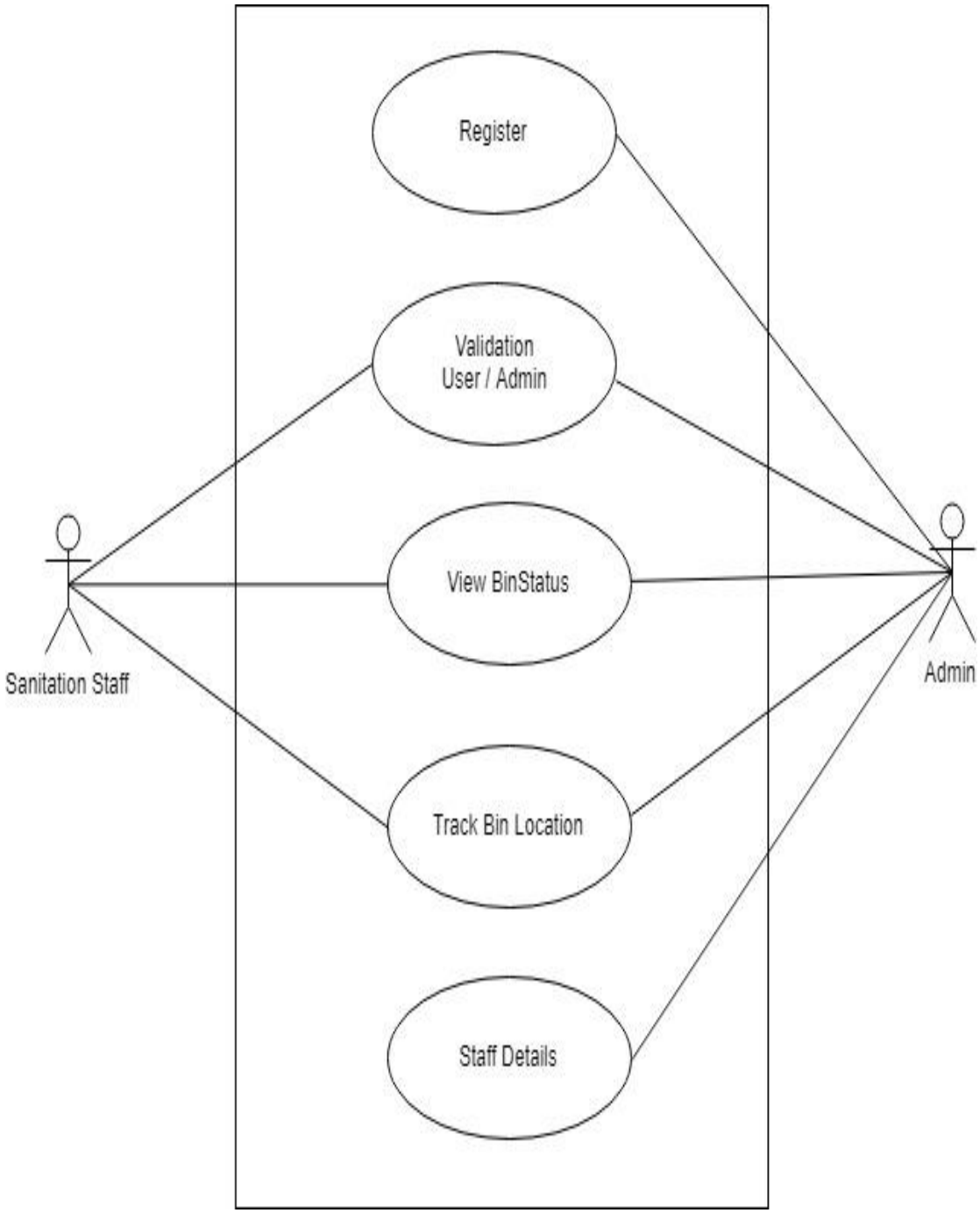


Figure: 3.2 Use Case design of Garbage monitoring system

Use Case 1: Register

Admin can Register sanitation staff details using admin login.

Use Case 2: Validation

The admin can log in into the application by providing their mail id and password and if valid they can only use this mobile application and admin can register the sanitation staff details then only staff can log in into the application.

Use Case 3: View Bin status

Sanitation staff /admin able to view the level of garbage in the bin. This screen shows the garbage bin status.

Use Case 4: Track Bin location

Sanitation staff /admin able to find where the garbage bin is locating on. This screen pointing the garbage location on Google Maps. The marker shows the location it helps to track a bin

Use Case 5: Staff Details

Admin can able to view sanitation staff details.

3.5 TABLE DESIGN

TABLE NAME: LOGIN

FIELD NAME	*DATA TYPE*	*SIZE*	*CONSTRAINT*
Mail-Id	Varchar	15	Primary key
Password	Varchar	12	Not Null

TABLE NAME: LOCATION TABLE

ID	*SENSOR*	*LATITUDE*	*LONGITUDE*	*DATE*
01	25	11.0283	77.0273	17-03-2018
02	50	11.0591	77.0082	02-04-2018

3.6 USER INTERFACE DESIGN

SCREEN 1: LOGIN SCREEN

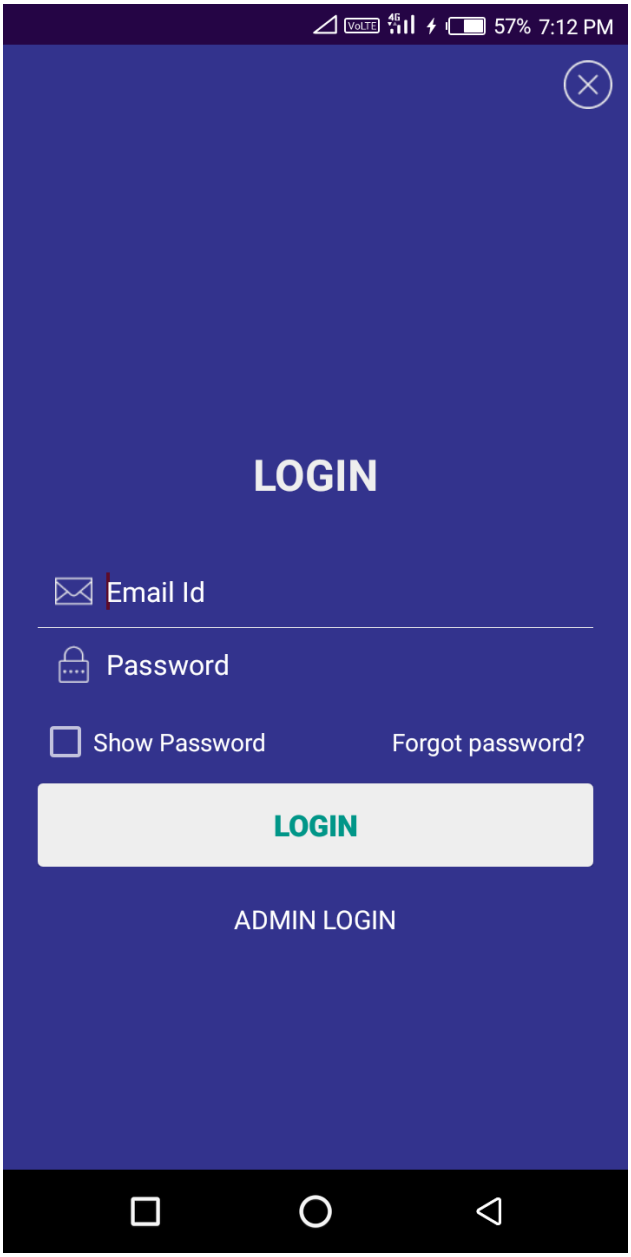


Figure: 3.4 Login Screen for Android application

DESCRIPTION:

The admin can log in into the application by providing their mail id and password and if valid they can only use this mobile application and admin can register the cleaning staff details then only cleaning staff can log in into the application.

SCREEN 2: NAVIGATION SCREEN

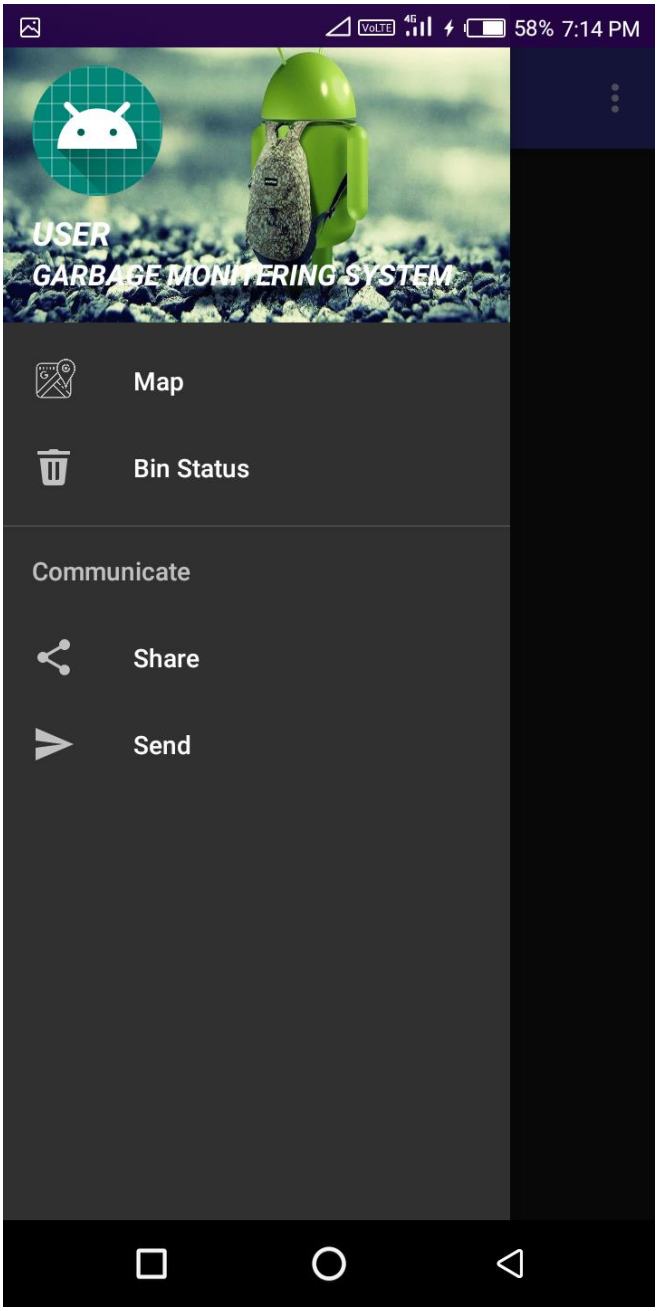


Figure: 3.5 Navigation view of Android application

DESCRIPTION:

Sanitation staff can be able to view the navigation toolbar

SCREEN 3: MAP VIEWSCREEN

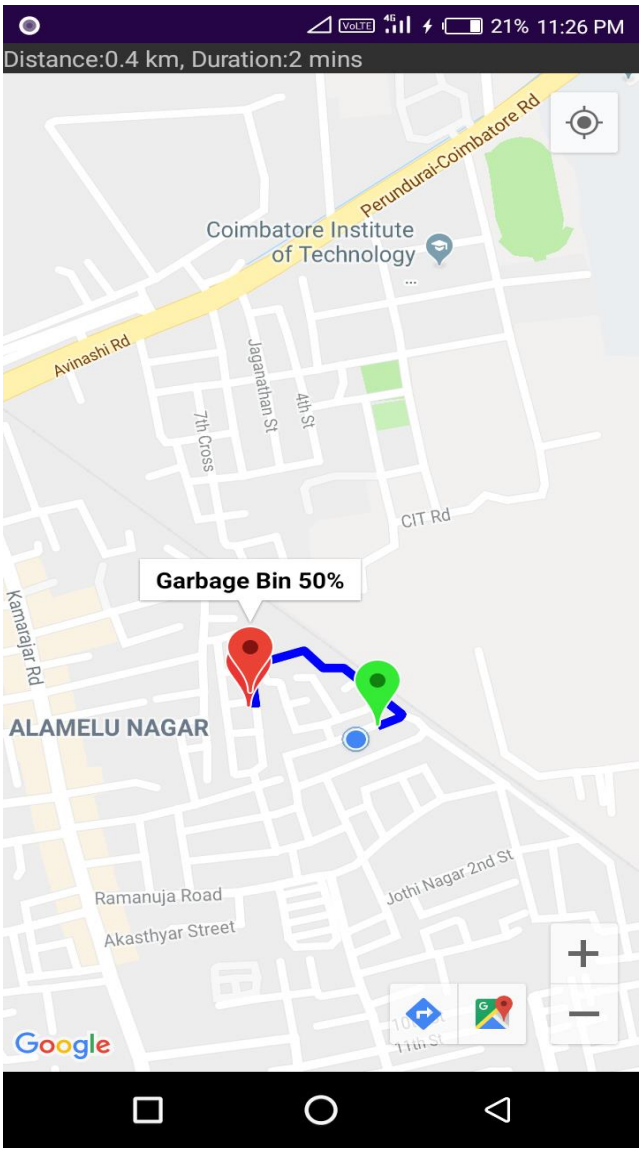


Figure: 3.6 Map view of Android application

DESCRIPTION:

After a successful login user/admin able to find where the garbage bin is locating on. This screen pointing the garbage location on Google Maps. The marker shows the location it helps to track a bin.

SCREEN 4: BIN_STATUS SCREEN

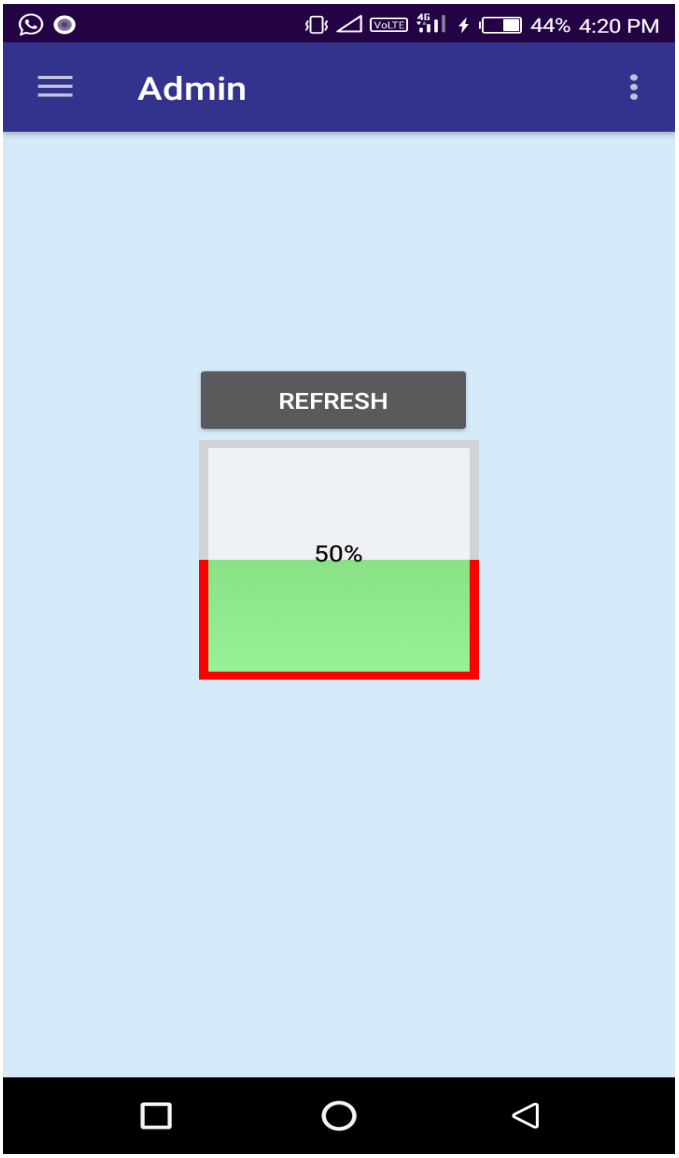


Figure: 3.6 Bin Status view of Android application

DESCRIPTION:

After a successful login sanitation staff/admin able to view the level of garbage in the bin. This screen shows the garbage bin status.

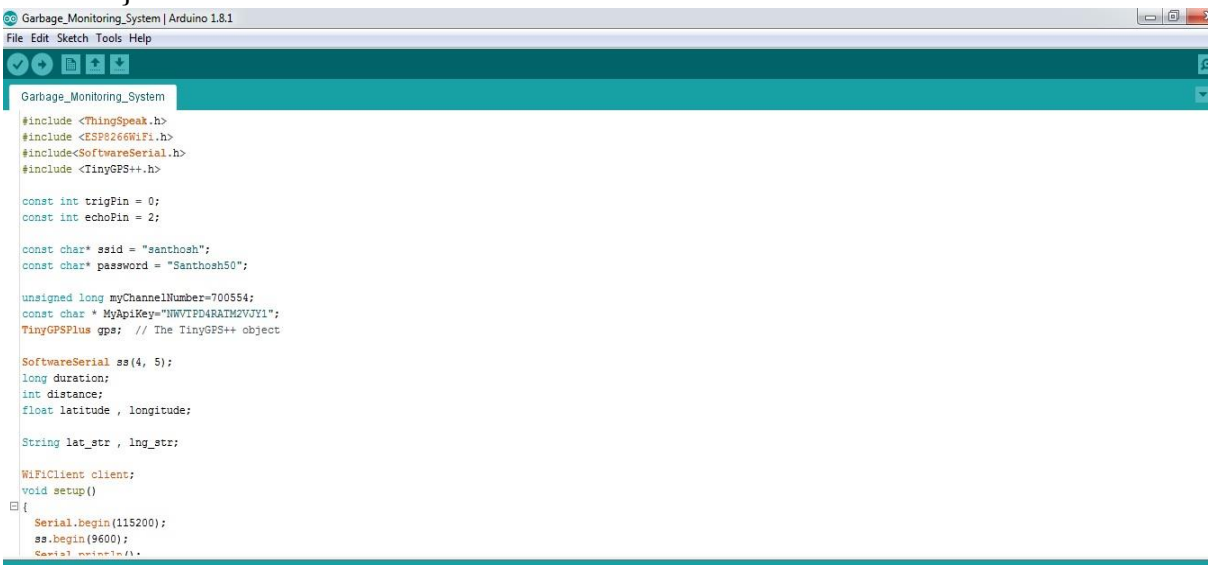
3.7 CODE DESIGN

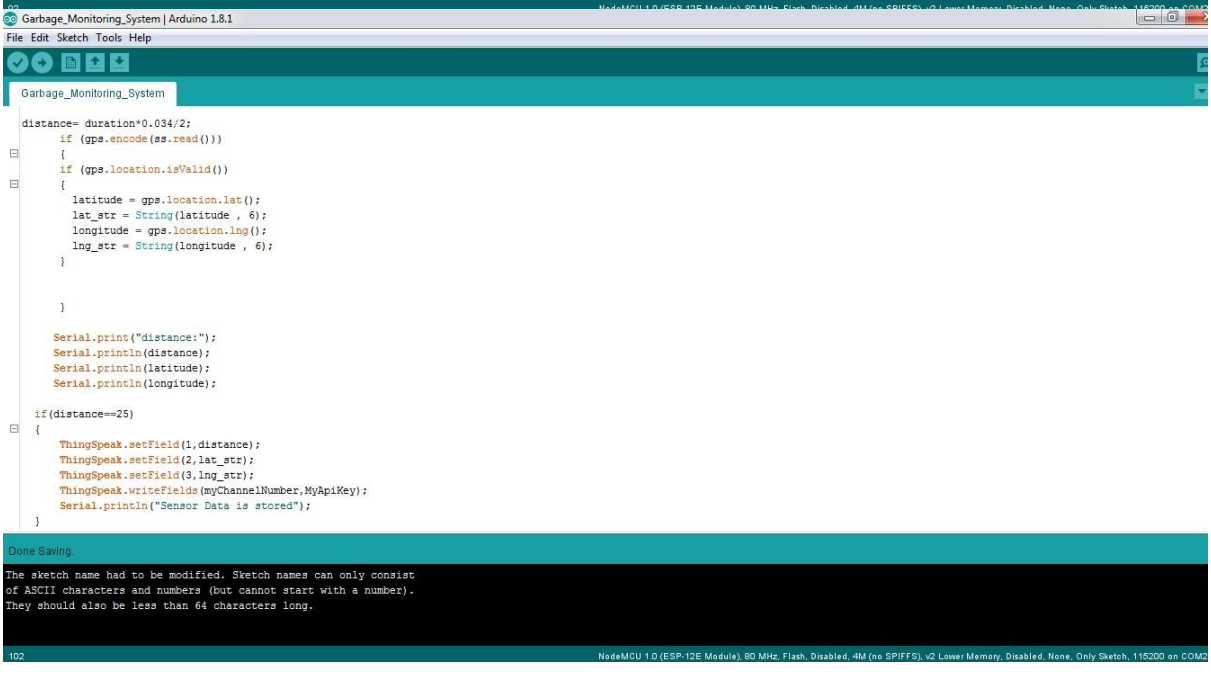
ARDUINO CODE:

```
#include <ThingSpeak.h>
#include <ESP8266WiFi.h>
#include<SoftwareSerial.h>
#include <TinyGPS++.h>

const int trigPin = 0;
const int echoPin = 2;

const char* ssid = "santhosh";
const char* password = "Santhosh50";

unsigned long myChannelNumber=700554;
const char * MyApiKey="NWVTPD4RATM2VJY1";
TinyGPSPlusgps;  // The TinyGPS++ object

SoftwareSerialss(4, 5);
long duration;
int distance;
float latitude , longitude;

String lat_str ,lng_str;

WiFiClient client;
void setup()
{
Serial.begin(115200);
ss.begin(9600);
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);

WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)
  {
delay(500);
Serial.print(".");
  }
Serial.println("");
Serial.println("WiFi connected");



pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
```

```
  // Print the IP address
Serial.println(WiFi.localIP());
ThingSpeak.begin(client);

}

void loop()
{
digitalWrite(trigPin, LOW);
delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);

distance= duration*0.034/2;
     if (gps.encode(ss.read()))
     {
     if (gps.location.isValid())
     {
       latitude = gps.location.lat();
lat_str = String(latitude , 6);
       longitude = gps.location.lng();
lng_str = String(longitude , 6);
     }


     }

Serial.print("distance:");
Serial.println(distance);
Serial.println(latitude);
Serial.println(longitude);

  if(distance==25)
  {
ThingSpeak.setField(1,distance);
ThingSpeak.setField(2,lat_str);
ThingSpeak.setField(3,lng_str);
ThingSpeak.writeFields(myChannelNumber,MyApiKey);
Serial.println("Sensor Data is stored");
  }
  else if(distance==50)
  {
ThingSpeak.setField(1,distance);
```

```
ThingSpeak.setField(2,lat_str);
ThingSpeak.setField(3,lng_str);
ThingSpeak.writeFields(myChannelNumber,MyApiKey);
Serial.println("Sensor Data is stored");
  }
  else if(75<=distance && distance <= 100)
  {
ThingSpeak.setField(1,distance);
ThingSpeak.setField(2,lat_str);
ThingSpeak.setField(3,lng_str);
ThingSpeak.writeFields(myChannelNumber,MyApiKey);
Serial.println("Sensor Data is stored");
  }
```





ANDROID CODE:

ManiActivity.java

```
package com.Garbage;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.view.View.OnClickListener;
import com.google.firebase.auth.FirebaseAuth;
public class MainActivity extends AppCompatActivity {
   private static FragmentManagerfragmentManager;
   private FirebaseAuthmAuth;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
fragmentManager = getSupportFragmentManager();
mAuth = FirebaseAuth.getInstance();
      // If savedinstnacestate is null then replace login fragment
       if (savedInstanceState == null) {
fragmentManager
.beginTransaction()
.replace(R.id.frameContainer, new Login_Fragment(),
Utils.Login_Fragment).commit();
       }
       // On close icon click finish activity
findViewById(R.id.close_activity).setOnClickListener(
             new OnClickListener() {
                @Override
                public void onClick(View arg0) {
finish();
                }
             });
   }
   // Replace Login Fragment with animation
   protected void replaceLoginFragment() {
fragmentManager
.beginTransaction()
.setCustomAnimations(R.anim.left_enter, R.anim.right_out)
.replace(R.id.frameContainer, new Login_Fragment(),
Utils.Login_Fragment).commit();
   }
   @Override
   public void onBackPressed() {
      // Find the tag of signup and forgot password fragment
```

```
        Fragment SignUp_Fragment = fragmentManager
.findFragmentByTag(Utils.SignUp_Fragment);
        Fragment ForgotPassword_Fragment = fragmentManager
.findFragmentByTag(Utils.ForgotPassword_Fragment);
        Fragment Admin=fragmentManager
.findFragmentByTag(Utils.Admin);
        // Check if both are null or not
        // If both are not null then replace login fragment else do backpressed
        // task
        if (SignUp_Fragment != null)
replaceLoginFragment();
        else if (ForgotPassword_Fragment != null)
replaceLoginFragment();
        else if(Admin!=null)
replaceLoginFragment();
        else
super.onBackPressed();
    }
}
```

MapsActivity.java

```
package com.Garbage;
import android.Manifest;
import android.content.Intent;
import org.json.JSONException;
import org.json.JSONObject;
import org.json.JSONTokener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback{
        private static final String TAG = "UsingThingspeakAPI";
    private static final String THINGSPEAK_CHANNEL_ID = "700554";
    private static final String THINGSPEAK_API_KEY = "2F0SWOE6NCBDQNOI"; //GARBAGE
KEY
    private static final String THINGSPEAK_API_KEY_STRING =
"2F0SWOE6NCBDQNOI";//replace your read key
        /* Be sure to use the correct fields for your own app*/
        private static final String THINGSPEAK_FIELD1 = "field1";
        private static final String THINGSPEAK_FIELD2 = "field2";
        private static final String THINGSPEAK_FIELD3 = "field3";
        private static final String THINGSPEAK_UPDATE_URL =
```

```
"https://api.thingspeak.com/update?";
      private static final String THINGSPEAK_CHANNEL_URL =
"https://api.thingspeak.com/channels/";
      private static final String THINGSPEAK_FEEDS_LAST = "/feeds/last?";
   private static final int LOCATION_PERMISSION_REQUEST_CODE = 1;
ArrayListmarkerPoints = new ArrayList();
ArrayList<LatLng>MarkerPoints;
   private GoogleMapmMap;
   private TextView text;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_maps);
      // Obtain the SupportMapFragment and get notified when the map is ready to be used.
SupportMapFragmentmapFragment = (SupportMapFragment) getSupportFragmentManager()
.findFragmentById(R.id.map);
mapFragment.getMapAsync(this);
   }
GoogleMap.OnMyLocationButtonClickListeneronMyLocationButtonClickListener =
        new GoogleMap.OnMyLocationButtonClickListener() {
          @Override
          public booleanonMyLocationButtonClick() {
mMap.setMinZoomPreference(15);
            try {
              new FetchThingspeakTask().execute();
            } catch (Exception e) {
Log.e("ERROR", e.getMessage(), e);
            }
            return false;
          }
        };
GoogleMap.OnMyLocationClickListeneronMyLocationClickListener =
        new GoogleMap.OnMyLocationClickListener() {
          @Override
          public void onMyLocationClick(@NonNull Location location) {
mMap.setMinZoomPreference(12);
LatLng me = new LatLng(location.getLatitude(), location.getLongitude());
mMap.addMarker(new MarkerOptions().position(me).title("Me"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(me));
          }
        };
   public void onMapReady(GoogleMapgoogleMap) {
mMap = googleMap;
mMap.setOnMyLocationButtonClickListener(onMyLocationButtonClickListener);
mMap.setOnMyLocationClickListener(onMyLocationClickListener);
enableMyLocationIfPermitted();
mMap.getUiSettings().setZoomControlsEnabled(true);
mMap.setMinZoomPreference(11);
mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {
```

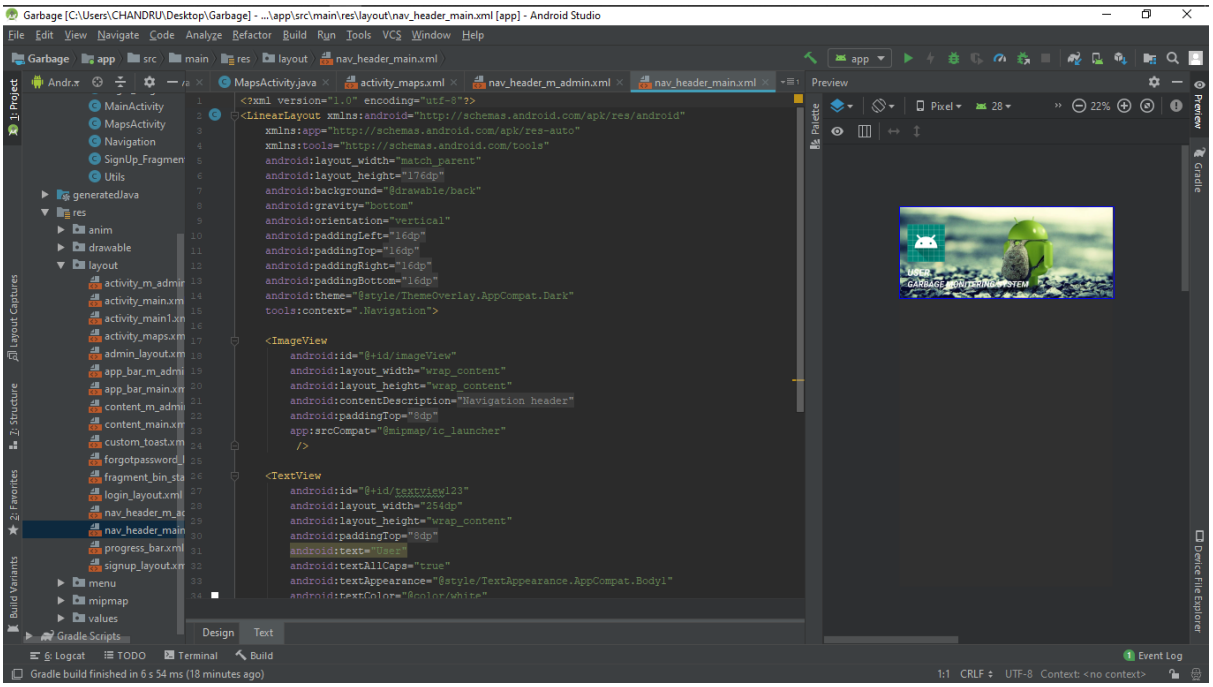
```
        @Override
        public void onMapClick(LatLnglatLng) {
            if (markerPoints.size() > 1) {
markerPoints.clear();
mMap.clear();
            }
            // Adding new item to the ArrayList
markerPoints.add(latLng);
            // Creating MarkerOptions
MarkerOptions options = new MarkerOptions();
            // Setting the position of the marker
options.position(latLng);
            if (markerPoints.size() == 1)
options.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));
            } else if (markerPoints.size() == 2)
            // Add new marker to the Google Map Android API V2
mMap.addMarker(options);
            // Checks, whether start and end locations are captured
            if (markerPoints.size() >= 2) {
LatLng origin = (LatLng) markerPoints.get(0);
LatLngdest = (LatLng) markerPoints.get(1);
                // Getting URL to the Google Directions API
                String url = getDirectionsUrl(origin, dest);
DownloadTaskdownloadTask = new DownloadTask();
                // Start downloading json data from Google Directions API
downloadTask.execute(url);
            }          }
      });    }
   private void enableMyLocationIfPermitted() {
      if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
!=PackageManager.PERMISSION_GRANTED) {
ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_FINE_LOCATION},
                LOCATION_PERMISSION_REQUEST_CODE);
      } else if (mMap != null) {
mMap.setMyLocationEnabled(true);
      }
   }
   @Override
   public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                          @NonNullint[] grantResults) {
      switch (requestCode) {
         case LOCATION_PERMISSION_REQUEST_CODE: {
            if (grantResults.length> 0
&&grantResults[0] == PackageManager.PERMISSION_GRANTED) {
enableMyLocationIfPermitted();
            } else {
```

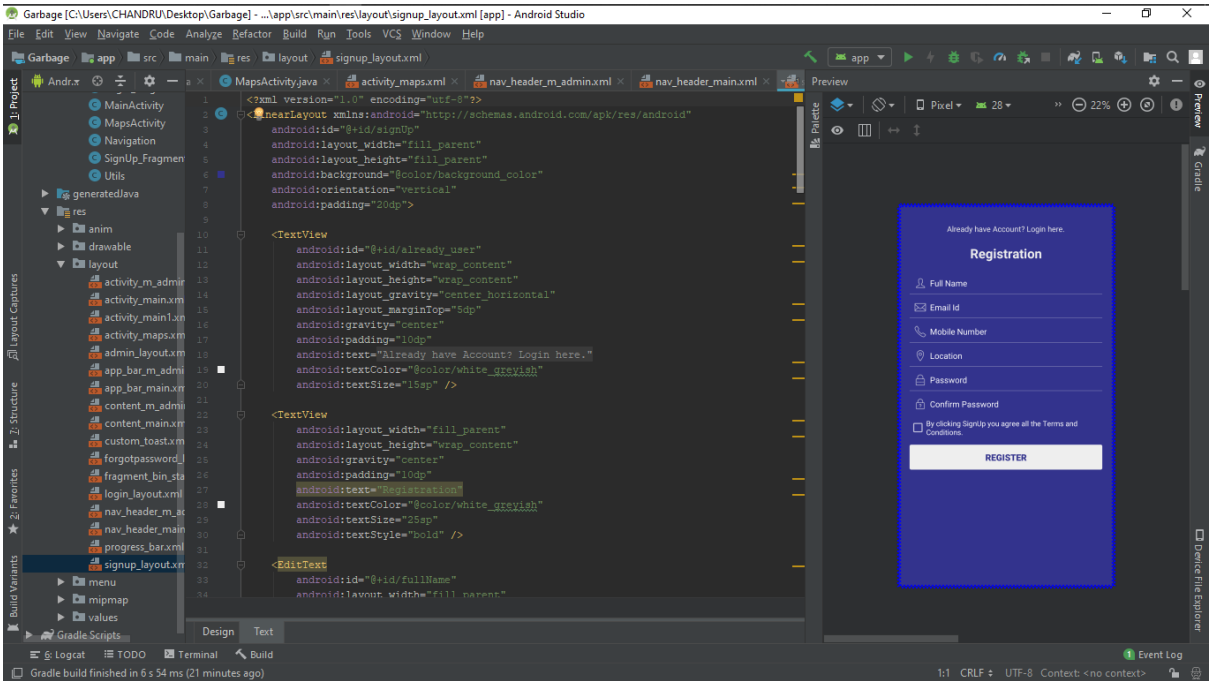
```
            }
            return;
          }
        }
    }
    public void onDraw(GoogleMapgoogleMap) {
        Location location = googleMap.getMyLocation();
        final Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://maps.google.com/maps?" + "saddr=" + location.getLatitude() + "," +
location.getLongitude() + "&daddr=" + 11.0283 + "," + 77.023));
intent.setClassName("com.google.android.apps.maps", "com.google.android.maps.MapsActivity");
startActivity(intent);
    }
    private String getDirectionsUrl(LatLng origin, LatLngdest) {
        // Origin of route
        String str_origin = "origin=" + origin.latitude + "," + origin.longitude;
        // Destination of route
        String str_dest = "destination=" + dest.latitude + "," + dest.longitude;
        // Sensor enabled
        String sensor = "sensor=false";
        String mode = "mode=driving";
        // Building the parameters to the web service
        String parameters = str_origin + "&" + str_dest + "&" + sensor + "&" + mode;
        // Output format
        String output = "json";
        // Building the url to the web service'
        String url = "https://maps.googleapis.com/maps/api/directions/" + output + "?" + parameters +
"&key=AIzaSyD4qDkhyYL2n-gT1i7p-n_3yV7QugyJAYI";
System.out.println("GoogleUrl " + url);
        return url;
    }
    private class DownloadTask extends AsyncTask<String, Void, String> {
        @Override
        protected String doInBackground(String... url) {
            String data = "";
            try {
                data = downloadUrl(url[0]);
            } catch (Exception e) {
Log.d("Background Task", e.toString());
            }
            return data;
        }
        @Override
        protected void onPostExecute(String result) {
super.onPostExecute(result);
ParserTaskparserTask = new ParserTask();
parserTask.execute(result);
        }
    }
```

```
   private class ParserTask extends AsyncTask<String, Integer, List<List<HashMap<String,
String>>>> {
       // Parsing the data in non-ui thread
       @Override
  protected List<List<HashMap<String, String>>>doInBackground(String... jsonData) {
JSONObjectjObject;
          List<List<HashMap<String, String>>> routes = null;
          try {
jObject = new JSONObject(jsonData[0]);
DirectionsJSONParser parser = new DirectionsJSONParser();
             routes = parser.parse(jObject);
          } catch (Exception e) {
e.printStackTrace();
          }
          return routes;
       }
       @Override
       protected void onPostExecute(List<List<HashMap<String, String>>> result) {
ArrayList points = null;
PolylineOptionslineOptions = null;
MarkerOptionsmarkerOptions = new MarkerOptions();
          String distance = "";
          String duration = "";
          for (int i = 0; i<result.size(); i++) {
             points = new ArrayList();
lineOptions = new PolylineOptions();
             List<HashMap<String, String>> path = result.get(i);
             for (int j = 0; j <path.size(); j++) {
                HashMap<String, String> point = path.get(j);
                if(j==0){    // Get distance from the list
                   distance = (String)point.get("distance");
                   continue;
}else if(j==1){ // Get duration from the list
                   duration = (String)point.get("duration");
                   continue;
                }
                double lat = Double.parseDouble(point.get("lat"));
                double lng = Double.parseDouble(point.get("lng"));
LatLng position = new LatLng(lat, lng);
points.add(position);
             }
lineOptions.addAll(points);
lineOptions.width(12);
lineOptions.color(Color.BLUE);
lineOptions.geodesic(true);
          }
          if (lineOptions != null)
          {
mMap.addPolyline(lineOptions);
```

```
            text=findViewById(R.id.text);
text.setText("Distance:" + distance + ", Duration:" + duration);
          }
          else {
Toast.makeText(MapsActivity.this, "PolylineOptions cannot be null",
Toast.LENGTH_SHORT).show();
          }
       }
    }
    class FetchThingspeakTask extends AsyncTask<Void, Void, String>
    {
       protected void onPreExecute() {
Toast.makeText(MapsActivity.this,"Fetching Data from Server.Please
Wait...",Toast.LENGTH_LONG).show();
       }
       protected String doInBackground(Void... urls) {
          try {
             URL url = new URL(THINGSPEAK_CHANNEL_URL +
THINGSPEAK_CHANNEL_ID +
                   THINGSPEAK_FEEDS_LAST + THINGSPEAK_API_KEY_STRING + "=" +
                   THINGSPEAK_API_KEY + "");
HttpURLConnectionurlConnection = (HttpURLConnection) url.openConnection();
             try {
BufferedReaderbufferedReader = new BufferedReader(new
InputStreamReader(urlConnection.getInputStream()));
                StringBuilder stringBuilder = new StringBuilder();
                String line;
                while ((line = bufferedReader.readLine()) != null) {
stringBuilder.append(line).append("\n");
                }
bufferedReader.close();
                return stringBuilder.toString();
             }
finally{
urlConnection.disconnect();
             }
          }
catch(Exception e) {
Log.e("ERROR", e.getMessage(), e);
             return null;
          }
       }
          try {
JSONObject channel = (JSONObject) new JSONTokener(response).nextValue();
             int status=channel.getInt(THINGSPEAK_FIELD1);
             String Latitude = channel.getString(THINGSPEAK_FIELD2);
             String Longitude =channel.getString(THINGSPEAK_FIELD3);
             double lat=Double.parseDouble(Latitude);
             double lon=Double.parseDouble(Longitude);
```

```
LatLng bin = new LatLng(lat,lon);
mMap.addMarker(new MarkerOptions().position(bin).title("Garbage Bin "+status+"%"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(bin));
        } catch (JSONException e)
        {
e.printStackTrace();
        }
    }
}
}
```





CHAPTER IV

SYSTEM TESTING

UNIT TESTING

Unit testing is a level of software testing where individual units/ components of the software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

1. DISPLAY MAP WITH USER'S CURRENT LOCATION

Test Case Id	*Test Case*	*Expected Result*	*Obtained result*
1	Display map with the user's current location	User's current location should be accurate and displayed on the map	Passed

2. DISPLAY BIN STATUS WITH LEVEL OF GARBAGE BIN

Test Case Id	*Test Case*	*Expected Result*	*Obtained result*
2	Display garbage bin level	Bin level should be displayed on Bin status	Passed

3. DISPLAY MAP WITH BIN LOCATION

Test Case Id	*Test case*	*Expected Result*	*Obtained Result*
3	Display bin location on Google Map	Filled Bin location should be accurate and displayed on the map	passed

4. SHOW ROUTE

Test Case Id	*Test case*	*Expected Result*	*Obtained Result*
4	Show routes	Staff can select the place from the current location to reach bin location	passed

INTEGRATION TESTING

Test Case Id	*Description*	*Success/Failure*
1	Display staff current location	success
2	Display the level of garbage in the bin on bin status	success
3	Display the bin location on Google Map	success
4	Show routes	success

SYSTEM TESTING

Test Case Id	*Description*	*Result*
1	Sanitation staff should be able to find bin location and level of garbage in the bin	Success

4.1 TEST CASES AND TEST REPORTS

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. Once the source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer. In order to find the highest possible number of errors, tests must be conducted systematically and test cases must be designed using disciplined techniques.

Test Case No.	Description	Expected Result	Observed result	Status
1	GY-NEO6MV2 (GPS)	It gives the location coordinates continuously.	Data has been given by the GPS.	PASS
2	NODE MCU 1.0 (wi-fi)	It should connect to local WLAN and transfer data	It connects to WLAN and gets data	PASS
3	LED light	To check the device is in power. So LED light is enabled.	The Led will be turned on thus the device working properly.	PASS
4	ULTRASONIC HC-SR04 (sensor)	It is expected to give distance	It gives the distance	PASS

CHAPTER V

SYSTEM IMPLEMENTATION

The purpose of system implementation is to take all possible steps to shows that the upcoming deployment and transition occur suddenly, efficiently and flawlessly.

INSTALLATION PROCEDURE FOR ARDUINO

Arduino is an open-source electronic prototyping platform enabling users to create interactive electronic objects. Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OSX and Linux. The environment is written in Java and based on Processing and another open-source software.

INSTALLING SOFTWARE

The Uno is programmed using the Arduino Software (IDE), our Integrated Development Environment common to all our boards. Before you can move on, you must have installed the Arduino Software (IDE) on your PC.

INSTALL THE BOARD DRIVERS

If you used the Installer, Windows-from XP upto10-will install drivers automatically as soon as you connect your board. If you downloaded and expanded the Zip package or, for some reason, the board wasn't properly recognized, please follow the procedure below. Click on the Start Menu, and open up the Control Panel. While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager. Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COM xx)". If there is no COM & LPT section, look under "Other Devices" for "Unknown Device". Right click on the" Arduino UNO(Comxx)"port and choose the" Update Driver Software" option. Next, choose the "Browse my computer for Driver software" option. Finally, navigate to and select the driver file named "Arduino.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3orolder), choose the Uno driver file named "Arduino UNO.inf" Windows will finish up the driver installation from there.

INSTALLATION PROCEDURE FOR ANDROID STUDIO

Android Studio provides the fastest tools for building apps on every type of Android device. World-class code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system all allow you to focus on building unique and high-quality apps.

5.2.2 INSTALLING SOFTWARE

Launch Android Studio.exe, Make sure before launch Android Studio, Our Machine should require installed Java JDK. To install Java JDK as detailed below. Once you launched Android Studio, it's time to mention JDK7 path or later version in android studio installer. Need to check the components, which are required to create applications, below the image has selected **Android Studio**, **Android SDK**, **Android Virtual Machine** and **performance.** Need to specify the location of local machine path for Android studio and Android SDK. Need to specify the ram space for Android emulator by default it would take 1GB of local machine RAM. At the final stage, it would extract SDK packages into our local machine, it would take a while time to finish the task and would take 2626MB of Hard disk space. After doing all the above steps perfectly, get the finish button and it was gone be an open android studio project.

INSTALLATION PROCEDURE FOR JDK

Step 1: Install JDK. To use Java programming, you need to first install the Java Development Kit (JDK).

Step 2: Download "JDK" from oracle.com

Step 3: Run the downloaded installer

CHAPTER VI

CONCLUSION

The garbage monitoring system is a very innovative system based on Internet of things which will help to keep the cities clean. The garbage monitoring system is used to monitor the level of garbage in the bin. This data can be further used to plan garbage collection trips more efficiently, ultimately reducing overflowing bins and helping have better public sanitation. Using the system, drivers can give information on the status of bins. The garbage Monitoring system can be used anywhere in the world.

DISADVANTAGES FOR GARBAGE MONITORING SYSTEM

Sometimes the GPS may fail due to certain reasons and in that case, need to carry a backup map and directions. If you are using GPS on a battery-operated device, there may be a battery failure and need an external power supply which is not always possible. Sometimes, GPS signals are not accurate due to some obstacles to the signals such as buildings, trees and sometimes by extreme atmospheric conditions such as geomagnetic storms.

BENEFITS FOR GARBAGE MONITORING SYSTEM

ADVANTAGES

1) Real-time data updates.
2) Helps to monitor garbage levels.
3) Ultimately helps in better planning of garbage pickups.
4) Can help to reduce overflowing bins.
5) Reduces trips to areas where the bins still have a lot of capacity.

BIBLIOGRAPHY

REFERENCES BOOKS

1. Massimo Banzi, “Getting started with Arduino”, O'Reilly Media, First edition.
2. Simon monk “programming Arduino getting started with sketches”, McGraw Hill international edition, 2012.
3. Michael Margolis, “Arduino Cookbook”, O'Reilly Media - 2nd Edition.
4. Zigurd Mednieks, “Programming Android: Java Programming for the New Generation of Mobile Devices”, O'Reilly Media - 2nd Edition.

REFERENCES WEBSITES

1. https://www.arduino.cc
2. https://developer.android.com/
3. http://www.instructables.com
4. https://thingspeak.com