VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

Project Report, JUNE 2020

B.Tech, Winter-2019-2020

TITLE	Smart Waste Management System
COURSE CODE	ITE1007
COURSE NAME	Object Oriented Analysis & Design
SLOT	G1+TG1
FACULTY	Prof. BHAVANI S

TEAM DETAILS:

REG. NO.	NAME
18BIT0231	KUSHAGRA AGARWAL
18BIT0272	PRIYAL BHARDWAJ

OVERVIEW:

Waste management is all the activities and actions required to manage waste from its inception to its final disposal. This includes collection, transportation, treatment and disposal of waste together with monitoring and regulation. Waste collection methods vary widely among different countries and regions. Domestic waste collection services are often provided by local government authorities.

Curb side collection is the most common method of disposal in most countries, in which waste is collected at regular intervals by specialized trucks. Waste collected is then transported to an appropriate disposal area.

Nowadays, cities with developing economies experience exhausted waste collection services, inadequately managed and uncontrolled dumpsites and the problems are worsening. Waste collection method in such countries is an on-going challenge and many struggles due to weak institutions and rapid urbanization.

Efforts are made for having smarter ways of city waste management. Waste management always finds difficulty in knowing when to collect waste. If the vehicle comes to collect too early it would lead to unnecessary trips. And when comes to collect too late will result in an overflow of waste. Even though the level is moderate the odor produced by the waste can cause air pollution. These problems stated can be resolved using IoT technology used by this project. The workers need not want to approach the bin to know whether a waste is ready to be removed. Also, it is used to monitor whether the bin is not full and if not checks the smell of waste and then the bin can be removed.

ALGORITHM/ PROCEDURE:

The code is basically divided into two major parts. First one is the selection of bins which are filled as per the parameters sent in and the second part is deciding shortest path for the filled bins so as to reduce the garbage truck fuel consumption.

1. Selection of bins

5 different parameters are sent in to check whether the bin needs to be emptied or not

- Level of waste:
  if (level of waste > 60% of dustbin height)
  then level of waste is quite high, return true
  else
  return false

- Temperature of waste:
  if (temperature > 20 C)
  then temperature is high, may become home for breeding insects, return true
  else
  return false
- Odour level:
  if (content of gas > 15 ppm)
  then odour is high, return true
  else
  return false
- Type of waste:
  if (liquid level > 40% of dustbin height)
  then liquid level is quite high, return true
  else
  return false
- Weekly collection:
  if (last pickup of waste >= 7 days)
  then return true
  else
  return false
  This also helps to maintain the waste detection system because if the system is broken and is not sending signals, then the weekly collection will prevent garbage rotting and also system can be fixed.

  Now different combinations are fixed for collection of bins:
  1 level of waste is greater than desired level
  2 temperature and odour both are high
  3 level of liquid and odour both are high
  4 weekly collection

2. Selecting the shortest path for Waste bin collection

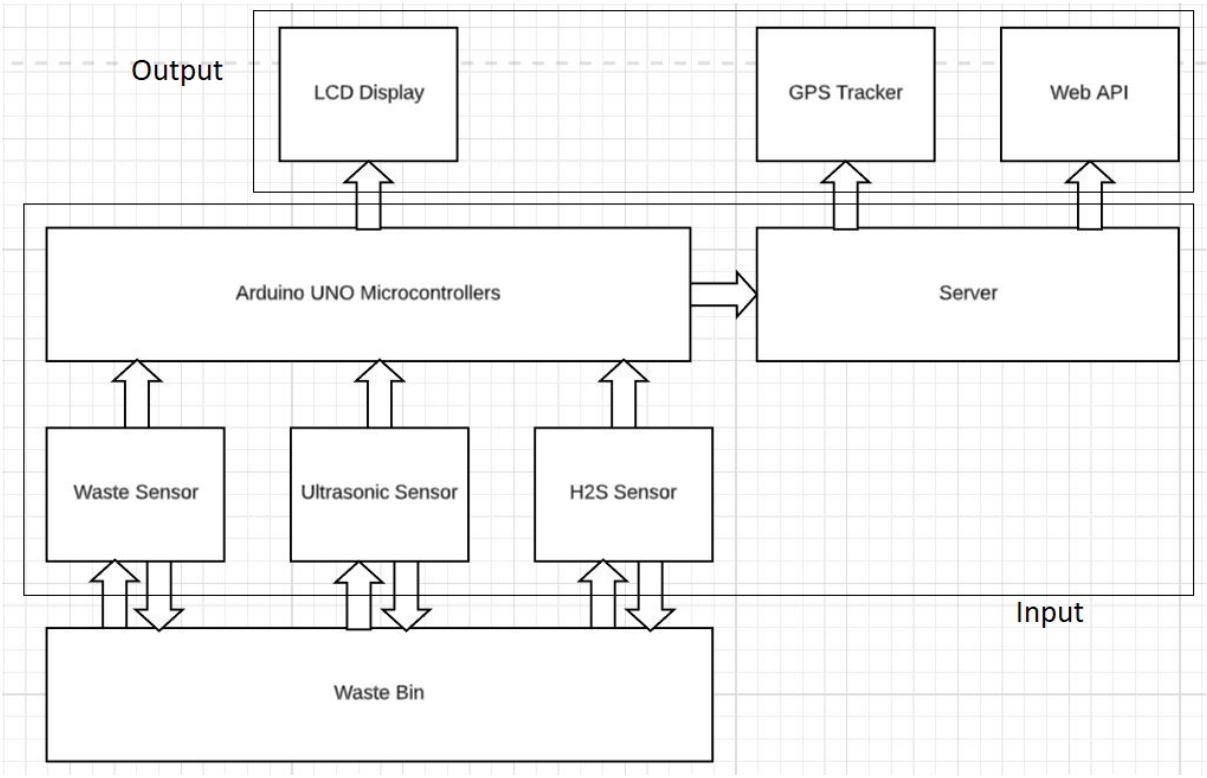
This is done using the Dijkstra Algorithm. It consists of the following steps:

1. Initialization of all nodes with distance "infinite"; initialization of the starting node with 0
2. Marking of the distance of the starting node as permanent, all other distances as temporarily.
3. Setting of starting node as active.

4. Calculation of the temporary distances of all neighbour nodes of the active node by summing up its distance with the weights of the edges.
5. If such a calculated distance of a node is smaller as the current one, update the distance and set the current node as antecessor. This step is also called update and is Dijkstra's central idea.
6. Setting of the node with the minimal temporary distance as active. Mark its distance as permanent.
7. Repeating of steps 4 to 7 until there aren't any nodes left with a permanent distance, which neighbours still have temporary distances.

```
1:  function Dijkstra (Graph, source):
2:      for each vertex v in Graph:                 // Initialization
3:          dist[v]: = infinity                     // initial distance from source to vertex v is set to infinite
4:          previous[v]: = undefined                // Previous node in optimal path from source
5:      dist[source]: = 0                           // Distance from source to source
6:      Q: = the set of all nodes in Graph          // all nodes in the graph are unoptimized - thus are in Q
7:      while Q is not empty:                       // main loop
8:          u: = node in Q with smallest dist[ ]
9:          remove u from Q
10:         for each neighbor v of u:               // where v has not yet been removed from Q.
11:             alt: = dist[u] + dist_between(u, v)
12:             if alt < dist[v]                    // Relax (u,v)
13:                 dist[v]: = alt
14:                 previous[v]: = u
15:     return previous [ ]
```

ARCHITECTURE/ BLOCK DIAGRAM:



CODE:

```
#include <iostream>
using namespace std;
int count;
#define INFINITY 9999


int waste_check(float ht_bin, float lvl)   //Level of Waste Check (Fn1)
{
    float chk = 0.6 * ht_bin;
    if(lvl > chk)
        return 1;
    else
        return 0;
}
int waste_check(float temp)     //Temperature Check of Waste (Fn2)
{
    if(temp>20)
        return 1;
    else
        return 0;
}
int waste_check(int ppm)        //Odour Check of Waste (Fn3)
```

```
{
    if(ppm>15)
        return 1;
    else
        return 0;
}
int waste_check(float ht_bin, int depth)    //Type Of Waste Check (Fn4)
{
    float chk = 0.4 * ht_bin;
    if(depth>ht_bin)
        return 1;
    else
        return 0;
}


int periodic_chk(int day)       //Weekly Waste pickup (Fn5)
{
    if(day>=7)
        return 1;
    else
        return 0;
}


int aerosol(float temp, int ppm)        //Aerosol Spray Check (Fn6)
{
    int t_chk,ppm_chk;
    t_chk = waste_check(temp);
    ppm_chk = waste_check(ppm);
    if (t_chk && ppm_chk)
        cout<<"Aerosol hit";
    else
        cout<<"Aerosol not hit";
}

int waste_pickup(int fn1,int fn2,int fn3,int fn4,int fn5)
//Condition to pickup waste
{
    if(fn1)
        return 1;
    else if(fn2 && fn3)
        return 1;
    else if(fn4 && fn3)
        return 1;
    else if(fn5)
        return 1;
    else
        return 0;
```

```
}



int main()
{
    float ht_bin,lvl,temp;
    int ppm,depth,day;
    int Fn1,Fn2,Fn3,Fn4,Fn5,bin[5];
    int i,j;

    for(i=0;i<5;i++)
    {
    cout<<"\n\tBin No.: "<<i<<"\n";
    cout<<"Enter Height of bin: ";
    cin>>ht_bin;
    cout<<"Enter Level of waste: ";
    cin>>lvl;
    cout<<"Enter Temperature of waste: ";
    cin>>temp;
    cout<<"Enter Level of Odour(in ppm): ";
    cin>>ppm;
    cout<<"Enter depth of liquid waste: ";
    cin>>depth;
    cout<<"Enter number of days since last waste pickup: ";
    cin>>day;
    Fn1 = waste_check(ht_bin,lvl);
    Fn2 = waste_check(temp);
    Fn3 = waste_check(ppm);
    Fn4 = waste_check(depth);
    Fn5 = periodic_chk(day);
    if(Fn2 && Fn3)
        aerosol(temp,ppm);
    bin[i] = waste_pickup(Fn1,Fn2,Fn3,Fn4,Fn5);
    }

    int flag=1,count=0;
    cout<<"\nBins which are filled and need to be picked up are: \n";
    /*if the different parameters to pick up the waste are satisfied then they
     return a Boolean value which is stored in an array
    */
    for(i=0;i<5;i++)
    {
        if(bin[i]==1)
        {
        cout<<"Bin "<<i<<"\t";
        flag = 0;
        count++;
        }
```

```
    }


    if (flag == 1)          //if none of the bins are filled then exit the code
    {
        cout<<"No bin";
        exit(0);
    }
    else                    //else making the shortest path to collect the bins
    {
        int loc[5][5] = {{0,1,0,3,10},{1,0,5,0,0},{0,5,0,2,1},{3,0,2,0,6},{10,
0,1,6,0}};
        int bin_loc[count][count];
        int x=0,y=0;
        for(i=0;i<5;i++)        //Choosing those bins only which are filled
        {
            if(bin[i]==1)
            {
                for(j=0;j<5;j++)
                {
                    if(bin[j]==1)
                    {
                    bin_loc[x][y]=loc[i][j];
                    y++;
                    }
                }
                y=0;
                x++;
            }
        }
        // Using dijkstra algorithm to calculate the shortest path for bin
        collection
        int n=count;
        int startnode=0;
        int cost[count][count],distance[count],pred[count];
        int visited[count],ct,mindistance,nextnode,i,j;
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
            if(bin_loc[i][j]==0)
                cost[i][j]=INFINITY;
            else
                cost[i][j]=bin_loc[i][j];
        for(i=0;i<n;i++)
        {
            distance[i]=cost[startnode][i];
            pred[i]=startnode;
            visited[i]=0;
        }
```

```
        distance[startnode]=0;
        visited[startnode]=1;
        ct=1;
        while(ct<n-1)
        {
            mindistance=INFINITY;
            for(i=0;i<n;i++)
                if(distance[i]<mindistance&&!visited[i]) {
                mindistance=distance[i];
                nextnode=i;
        }
        visited[nextnode]=1;
        for(i=0;i<n;i++)
            if(!visited[i])
        if(mindistance+cost[nextnode][i]<distance[i]) {
            distance[i]=mindistance+cost[nextnode][i];
            pred[i]=nextnode;
        }
        ct++;
        }
        for(i=0;i<n;i++)
        if(i!=startnode) {
         cout<<"\nDistance of bin "<<i<<"="<<distance[i];
            cout<<"\nPath="<<i;
            j=i;
        do {
         j=pred[j];
         cout<<"<-"<<j;
        }while(j!=startnode);
        }
    }
    return 0;
}
```

OUTPUT:

```
        Bin No.: 0
Enter Height of bin: 50
Enter Level of waste: 40
Enter Temperature of waste: 15
Enter Level of Odour(in ppm): 12
Enter depth of liquid waste: 10
Enter number of days since last waste pickup: 3

        Bin No.: 1
Enter Height of bin: 80
Enter Level of waste: 20
Enter Temperature of waste: 12
Enter Level of Odour(in ppm): 11
Enter depth of liquid waste: 12
Enter number of days since last waste pickup: 5

        Bin No.: 2
Enter Height of bin: 60
Enter Level of waste: 15
Enter Temperature of waste: 18
Enter Level of Odour(in ppm): 7
Enter depth of liquid waste: 8
Enter number of days since last waste pickup: 2

        Bin No.: 3
Enter Height of bin: 80
```

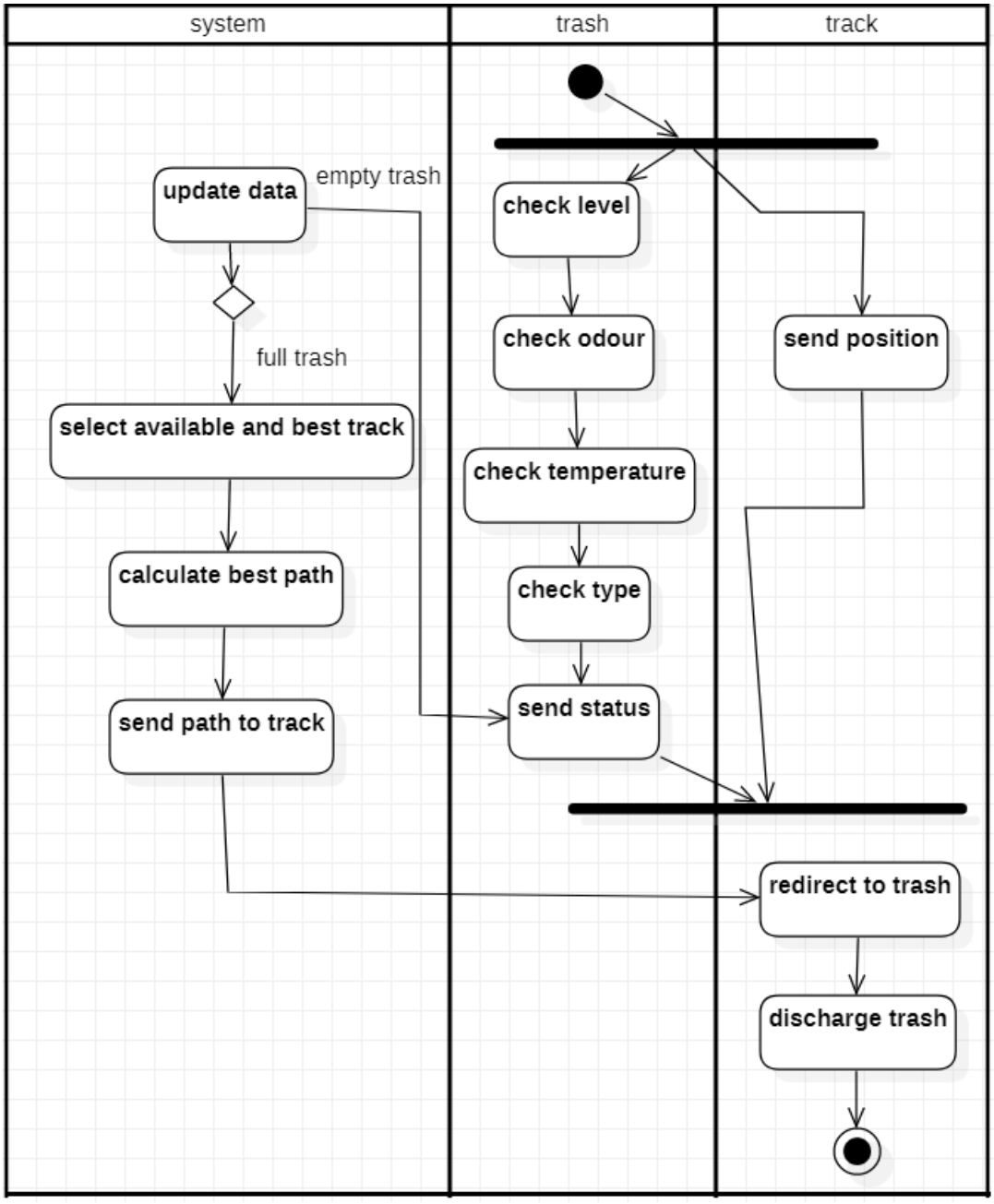
```
Enter depth of liquid waste: 8
Enter number of days since last waste pickup: 2

       Bin No.: 3
Enter Height of bin: 80
Enter Level of waste: 35
Enter Temperature of waste: 25
Enter Level of Odour(in ppm): 20
Enter depth of liquid waste: 11
Enter number of days since last waste pickup: 4
Aerosol hit
       Bin No.: 4
Enter Height of bin: 40
Enter Level of waste: 10
Enter Temperature of waste: 15
Enter Level of Odour(in ppm): 11
Enter depth of liquid waste: 20
Enter number of days since last waste pickup: 9
Bins which are filled and needs to be picked up are:
Bin 0Bin 3Bin 4
Distance of node1=3
Path=1<-0
Distance of node2=9
Path=2<-1<-0

...Program finished with exit code 0
```

```
       Bin No.: 0
Enter Height of bin: 80
Enter Level of waste: 46
Enter Temperature of waste: 16
Enter Level of Odour(in ppm): 24
Enter depth of liquid waste: 36
Enter number of days since last waste pickup: 4

       Bin No.: 1
Enter Height of bin: 74
Enter Level of waste: 67
Enter Temperature of waste: 15
Enter Level of Odour(in ppm): 35
Enter depth of liquid waste: 6
Enter number of days since last waste pickup: 2

       Bin No.: 2
Enter Height of bin: 65
Enter Level of waste: 31
Enter Temperature of waste: 25
Enter Level of Odour(in ppm): 12
Enter depth of liquid waste: 12
Enter number of days since last waste pickup: 5

       Bin No.: 3
Enter Height of bin: 80
```

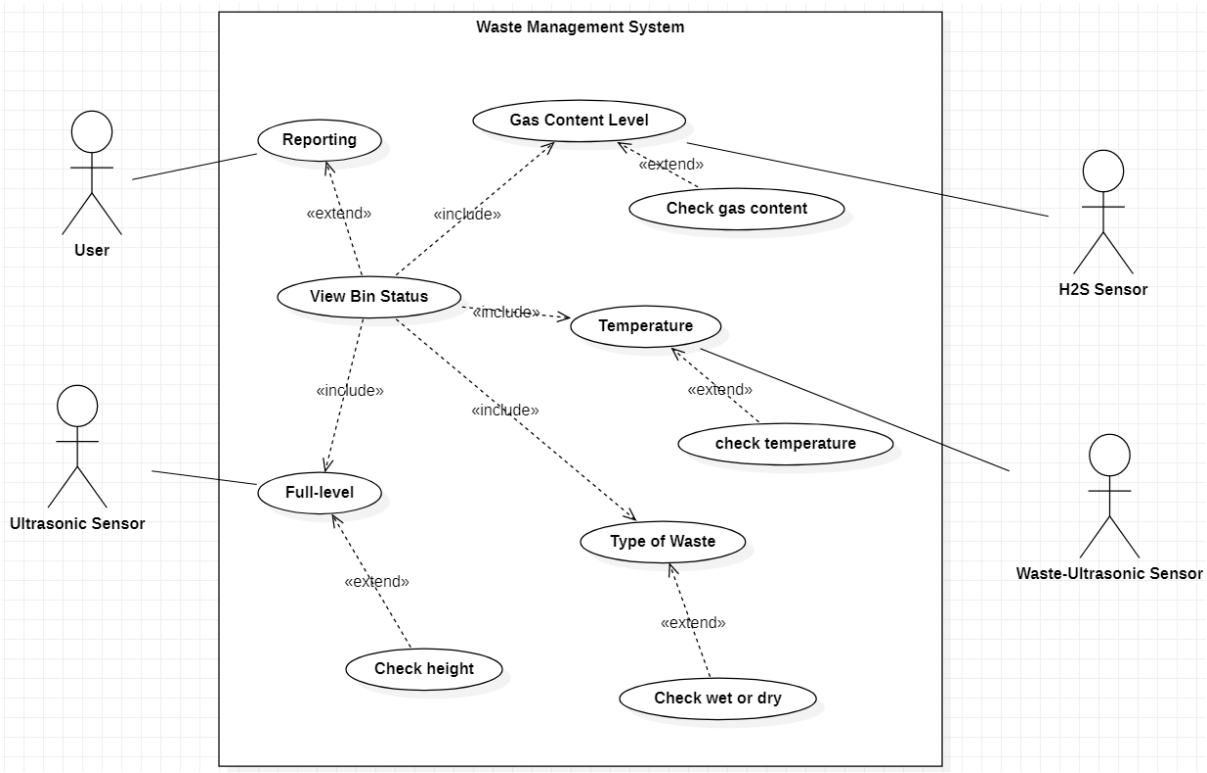
```
Enter number of days since last waste pickup: 5

       Bin No.: 3
Enter Height of bin: 80
Enter Level of waste: 45
Enter Temperature of waste: 32
Enter Level of Odour(in ppm): 18
Enter depth of liquid waste: 22
Enter number of days since last waste pickup: 1
Aerosol hit
       Bin No.: 4
Enter Height of bin: 68
Enter Level of waste: 6
Enter Temperature of waste: 10
Enter Level of Odour(in ppm): 5
Enter depth of liquid waste: 2
Enter number of days since last waste pickup: 1

Bins which are filled and need to be picked up are:
Bin 0   Bin 1   Bin 3
Distance of bin 1=1
Path=1<-0
Distance of bin 2=3
Path=2<-0

...Program finished with exit code 0
Press ENTER to exit console.
```

UML DIAGRAMS:

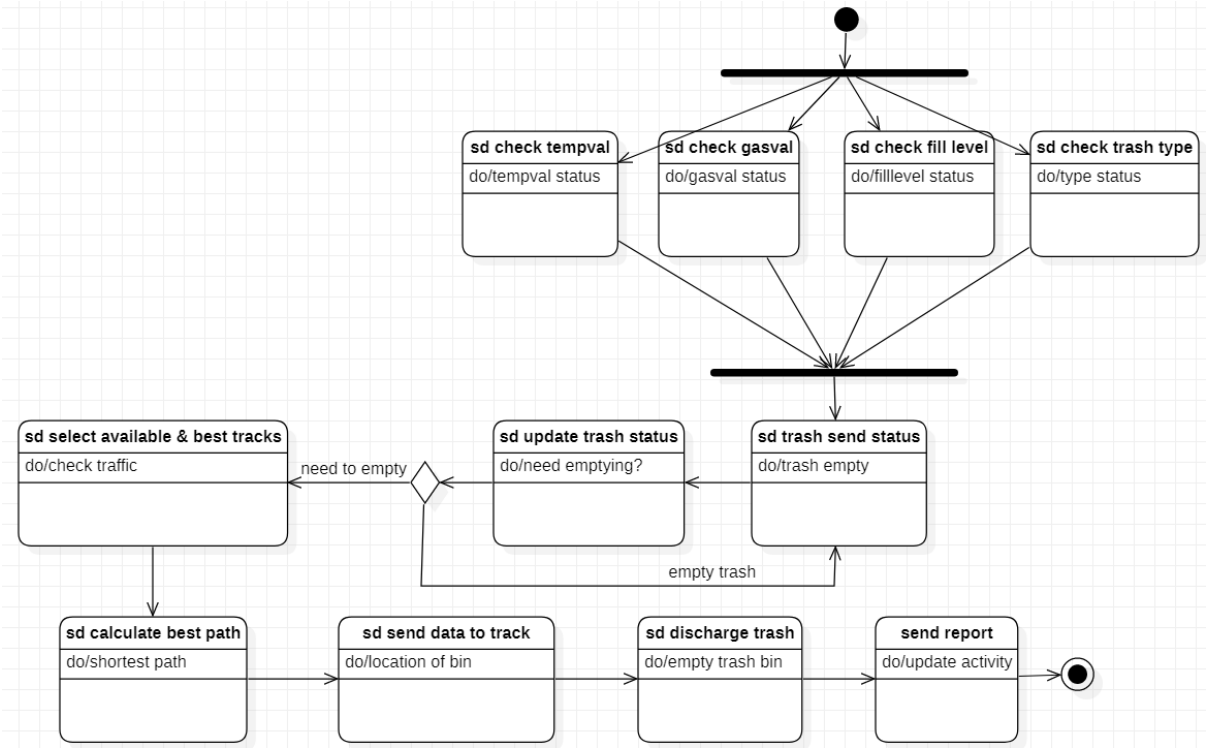
1. ACTIVITY DIAGRAM



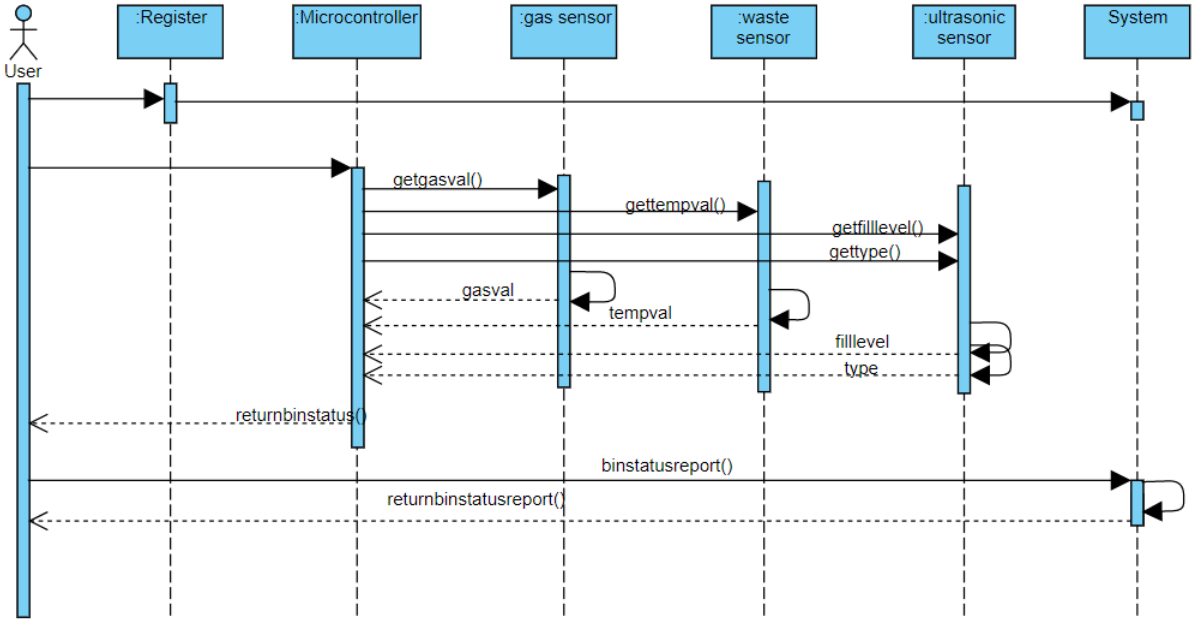
2. USE CASE DIAGRAM



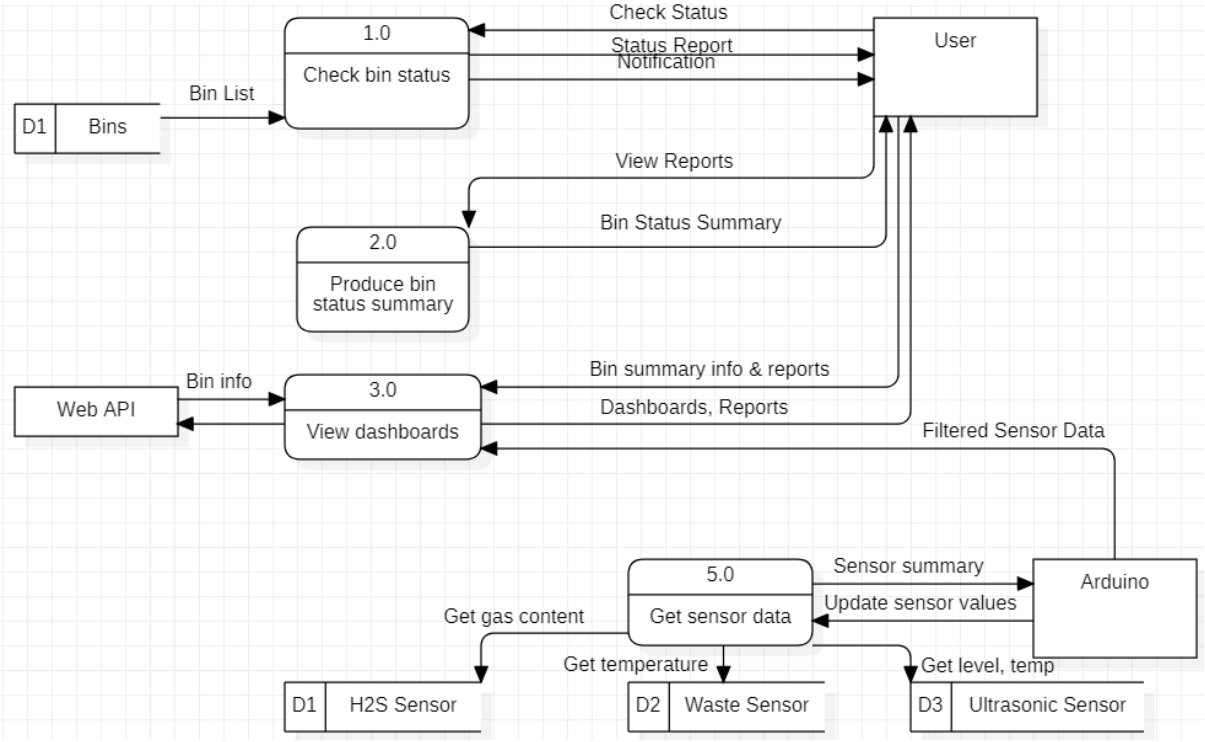
3. STATE DIAGRAM



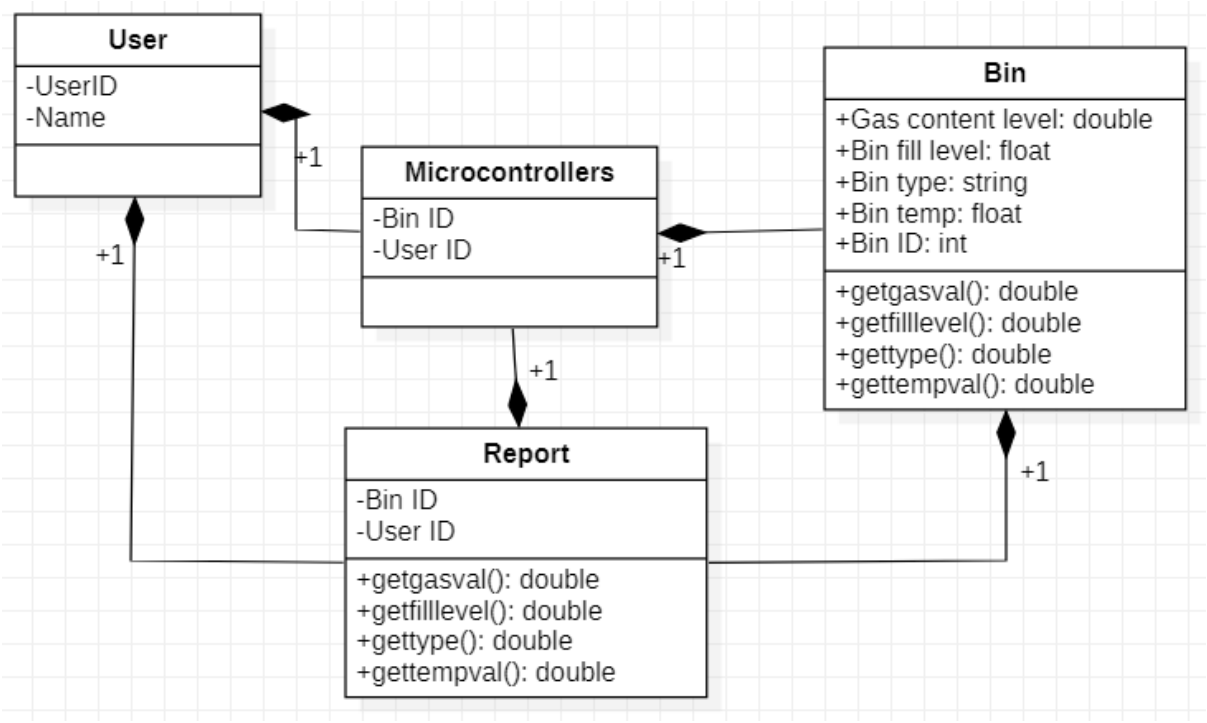
4. SEQUENCE DIAGRAM



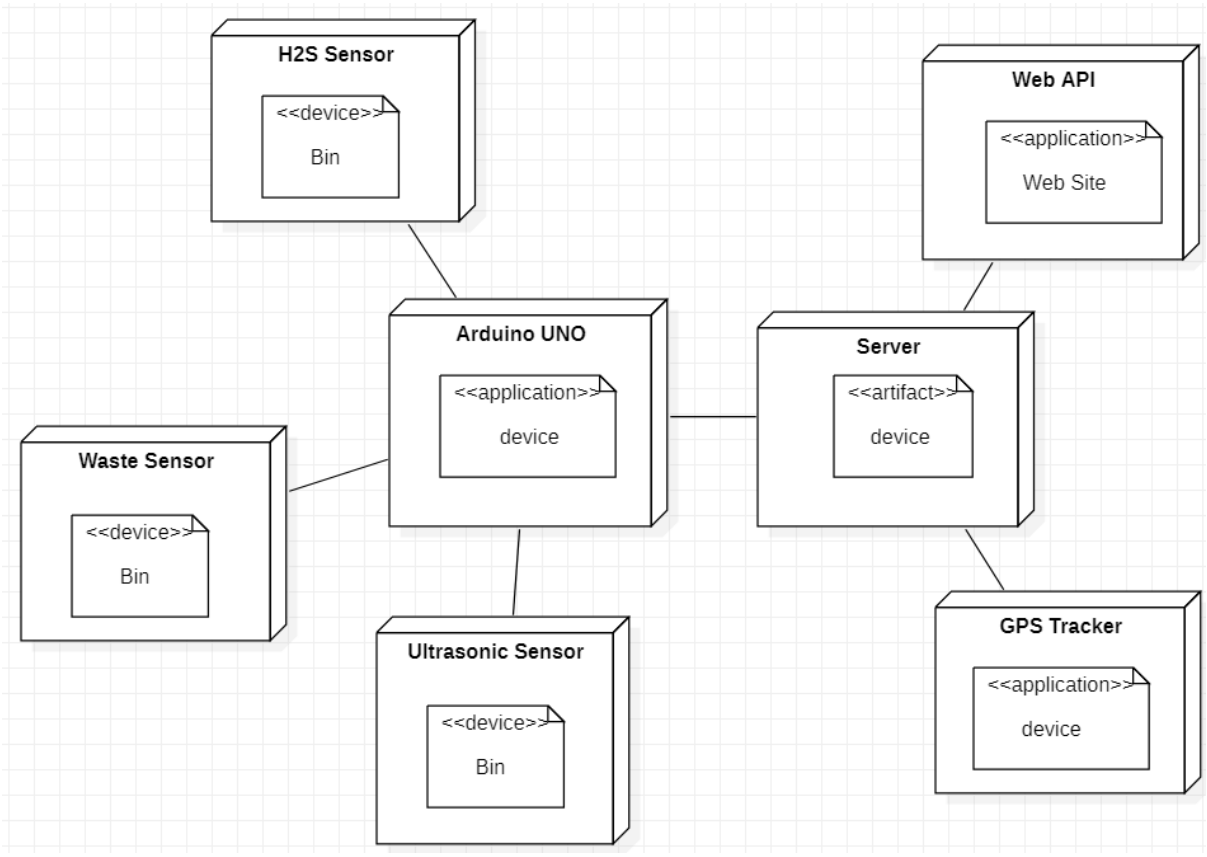
5. DATA FLOW DIAGRAM



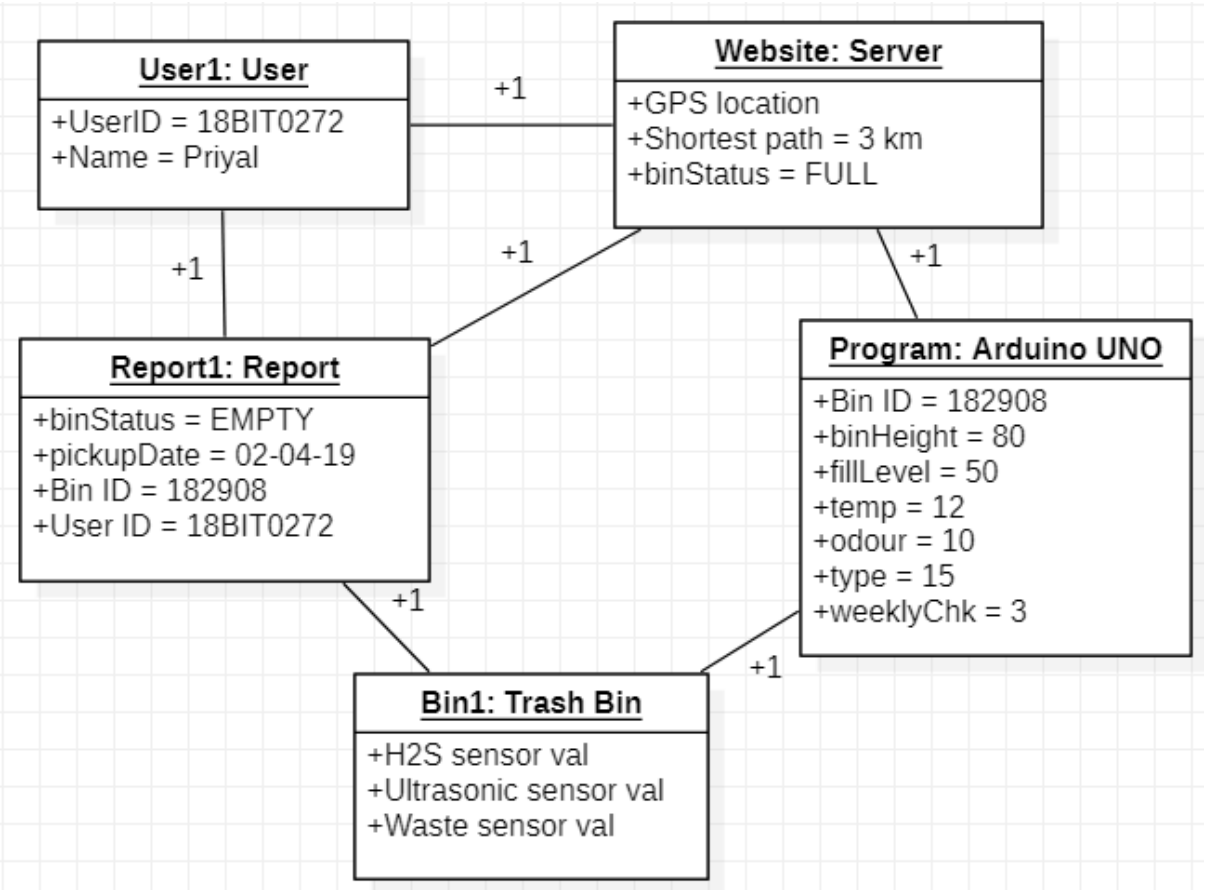
6. CLASS DIAGRAM



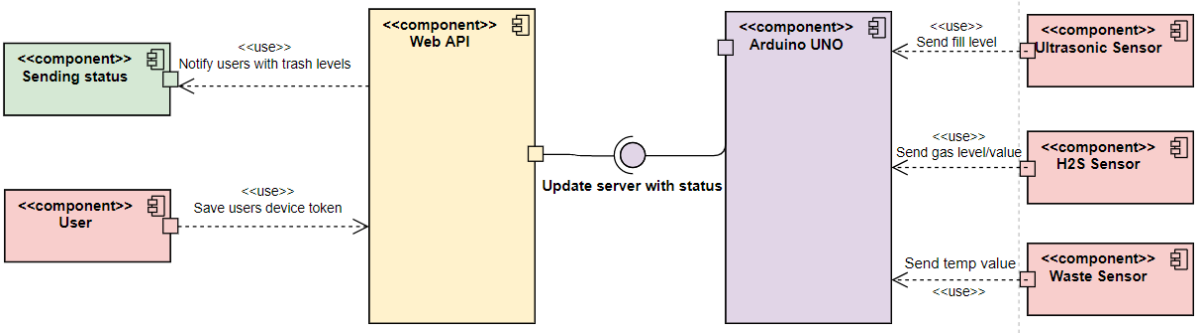
7. DEPLOYMENT DIAGRAM



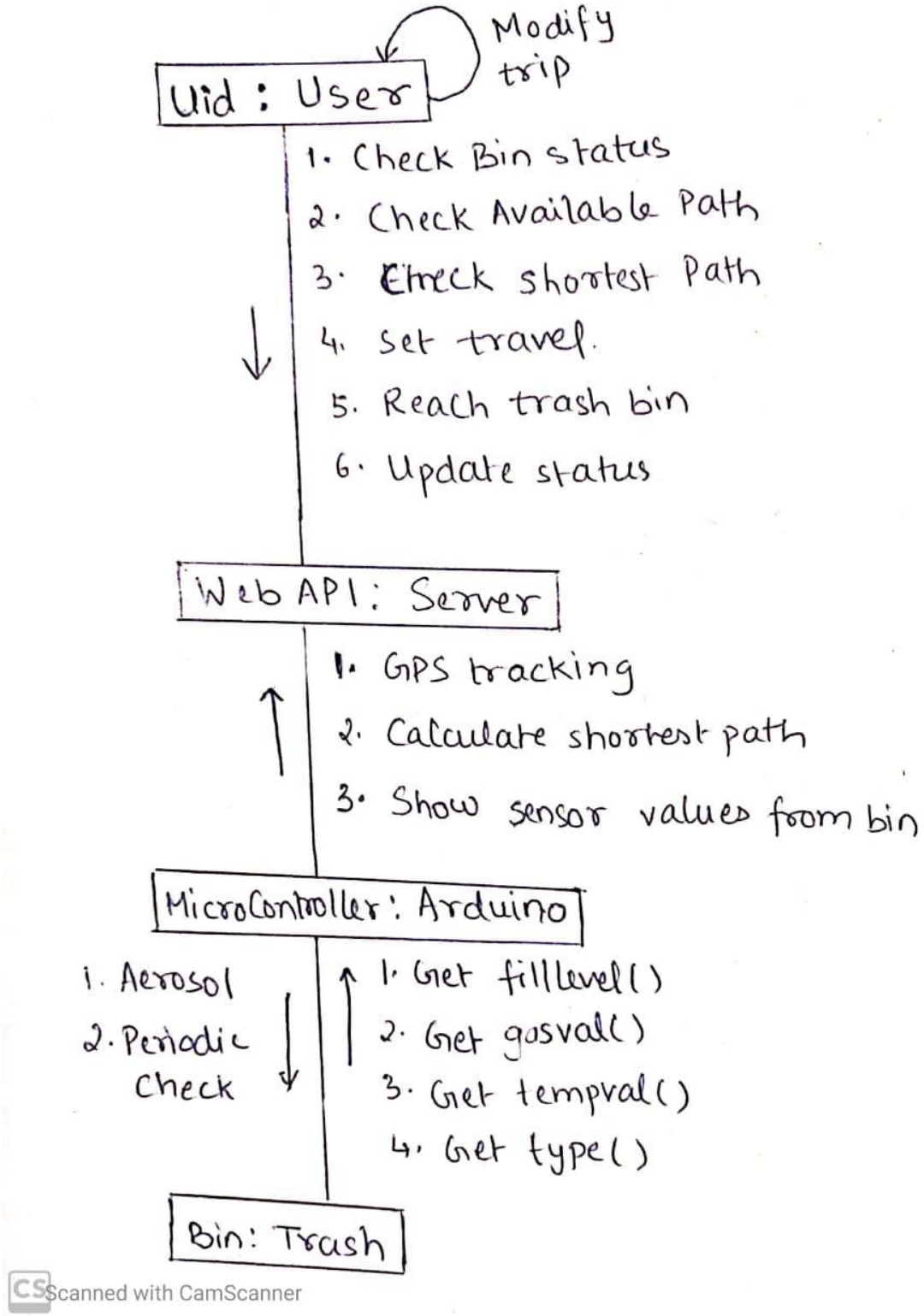
8. OBJECT DIAGRAM



9. COMPONENT DIAGRAM



10. COLLABORATION DIAGRAM



****