03-134151-046 ZeeshanMaqsood

03-14151-042 Usama Waseem

03-134151-047 Abdul Rehman

SMART CITY

In partial fulfilment of the requirements for the degree of

Bachelor of Science in Computer Science

Supervisor: Numan Aslam

Department of Computer Sciences

Bahria University, Lahore Campus

October 2018

C e r t i f i c a t e

We accept the work contained in the report titled

"Smart City",

written by

Zeeshan Maqsood

Usama Waseem

Abdul Rehman

as a confirmation to the required standard for the partial fulfilment of the degree of

Bachelor of Science in Computer Science.

Approved by:

Supervisor: Numan Aslam

______________________________________(Signature)

October 25th, 2018

DECLARATION

We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at Bahria University or other institutions.

Enrolment	Name	Signature
03-134151-046	Zeeshan Maqsood	
03-134151-042	Usama Waseem	
03-134151-047	Abdul Rehman	

Date : October 25th 2018

Specially dedicated to

My beloved Father Fiaz Maqsood Ali

Zeeshan Maqsood

my beloved parents and Teachers

(Usama Waseem)

My beloved parents and teachers

(Abdul Rehman)

ACKNOWLEDGEMENTS

We would like to thank everyone who had contributed to the successful completion of this project. We would like to express our gratitude to my research supervisor, Mr NUMAN ASLAM for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, we would also like to express my gratitude to our loving parent and friends who had helped and given me encouragement.

Zeeshan Maqsood
Usama Waseem
Abdul Rehman

SMART CITY

ABSTRACT

Automation plays an important role in today's human life and people's life is changing gradually with smart living due to modern technology development and wireless remote control. Our main goal is to build a smart city. We are going to develop an efficient system which will automate the entire city including car parking, mosque, traffic signals and street lights systems. Smart city project have a wide range of customization in which we are currently covering just five modules in this project. This project will be developed by using incremental methodology as this methodology is used where the project is broken down into parts and developed separately. This will consist of automating the parking system, mosques, street lights, signal and plant watering system. We can also check status of these modules at android application. The mosque module includes player alert timing and temperature maintaining of mosque. The parking module will check the parking space availability. Street light module will check the darkness and automatically on the lights. Density base signal module and plants watering system is also included in this project. This system consist of a hardware model consisting of all five modules implementation using aurdino and android application showing status of these modules. Moreover, by using of this system man power also reduced.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Automation plays an important role in today's human life and people's life is changing gradually with smart living due to modern technology development and wireless remote control. Technology is upgrading day by day. Everyone wants to live smart living therefore we are upgrading city to smart city which is basically the automation of complete city. Our project is basically the automation of complete city. This project have a wide range of scope and customization, therefore we are going to cover just five modules here. This will consist of automating the parking system, mosques, street lights, plant watering system and signal module. We are also developing and android application for checking status of all these modules. The mosque module will alert the prayer timing, then check temperature of mosque if temperature is greater than a limit turn on air conditioner and if less than that limit turn on heaters and the water heat up system just few minutes before the prayer time. The car parking module will check the availability of parking space to only allow the authorized vehicles to pass by the barriers, and update the status in database and not allow to enter the parking if it’s full. Street light module will check the darkness and automatically on the lights. After 12pm streets mostly become empty. Therefore lights will be off and automatically on when an object will be detected there. Density base signal module and plants watering system is also included in this project. The plant watering system is basically controlling watering system of parks. Density base signal module is controlling traffic signal on the basis of density from side where traffic rush is more will open for more time as compared to other sides.

1.2 Problem Statements

Technology is upgrading day by day. Everyone wants a smart living in their life. There it’s necessary to upgrade smart system for city which reduced manpower. Everyone is in hurry and does not have time to wait for checking the parking. Even we are also facing an issue of load shedding and we cannot waste electricity. Therefore its necessary to have a system which can control all city smartly and without manpower.

1.3 Aims and Objectives

The objective of the mission is to promote cities that provide its citizens a decent quality of life, a clean and sustainable environment. Applying Smart Solutions to infrastructure and services in area-based development in order to make them better. For example, making Areas less vulnerable to disasters, using fewer resources, and providing cheaper services it provides the guidelines for smart parking, intelligence traffic management, Street light system and Mosque automation

- To automate the parking system
- To give alerts of prayer time
- To control temperature of mosque
- To check the density of traffic on signals
- To automate the watering system of parks
- To control all this from a single platform
- To implement street lights automation

1.4 Scope of Project

Smart city system is automation of city designed for smart living. This project will only cover the parking system, Mosques, parks, street lights and traffic lights.

1.4.1 Parking Module

In this module parking system is covered. Any user need parking space he will check the app where parking slot is available and park his vehicle. If no parking slot available application will notify him.

1.4.2 Mosque Module

In this module we are covering Mosque automation. It will check the temperate of mosque if it's greater than the limit. Air conditioners will turn on automatically and if it's less than that limit heater and geezer will turn on automatically. In addition to this it will also create alert user of prayer time.

1.4.3 Street Light Module

In this module we will cover the street lights system. Street lights will automatically turned on in darken. And will be turned off after a specific time. After that if they sense motion they will be turned on otherwise will remain turned off. This will save electricity.

1.4.4 Park watering Module

In this module park watering system will be covered. This works on the basis of soil dryness. If soil is dry watering will start automatically. This reduces man power.

1.4.5 Traffic Signal Module

This module will cover the traffic signals. In this we will check the traffic load. The signal from which traffic load is high will open for more time and on other hand signal from which traffic load is less will open for less time.

CHAPTER 2

Software Requirement Specification

2.1 User Classes and Characteristics

Smart city is divided into five modules consisting of parking module, Mosque Module, Park watering system module. Basically all citizens are the user of this project they can check the player time and get alert of prayer and parking space and the status of all modules on an application.

2.2 Operating Environment

Smart City is embedded system so it requires both software and hardware environment. Therefore we are developing a 12 volt hardware system with implementation of all these modules and an android application linked with this hardware system.

2.2.1 Software Environment

For Android application development android studio will be used as a tool and java language for development with support of xml. Different APIs used for interface design.

2.2.2 Hardware Environment

Hardware component such as sensors, Microcontroller, raspberry pi, Arduino IDE is used to program microcontroller.

2.3 Design and Implementation Constraints

Mobile application is android specified. We are developing android application only. It uses a modular design where every feature is wrapped into a separate module and the modules depend on each other through well-written APIs and libraries. For data storage; system is using MYSQL/ 000 webhost where all the record of user and other modules will be stored. For hardware we are using Aurdino microcontroller for programing.

2.4 Assumption and Dependencies

Smart City has Database so one assumption about the product is that admin must have access to database from Mobile Application which is compatible with our application. Another assumption is that users must have internet connection, android mobile and must be connected to server, so that status of availability could update on real time using database. Internet is required for the system. Server down effect on communication between database and mobile application.

2.5 Software Requirement Chart

Here is the software requirement chart for smart city

Table 1: System Requirement Chart

ID	Priority	Type	Source	Used in Use case	Description
1	High	Functional	End User’s	N/A	Install the application
2	High	Functional	End User’s	U1	Registration for new user
3	High	Functional	End User’s	U2	Login in to application to

					use the application
4	Medium	Non-Functional	End User’s	U2	App user forget password
5	Medium	Functional	End User’s	U2	Generate new password
6	Medium	Functional	End User’s	N/A	Delete User
7	Medium	Functional	End User’s	U2	User logout
8	High	Functional	End User’s	N/A	Update USER Profile
9	High	Functional	End User’s	U5	Check space for Car parking
10	Low	Non Functional	End User’s	U5	Car is parked if space is available
11	Low	Non Functional	End User’s	U5	Parking barriers will not open if there is no space
12	High	Non-Functional	End User’s	U5	If a car leaves the parking. Database will be updated and there will show space for new user.
13	High	Non Functional	End User’s	U3	An Alert notification will be generated on prayer time

14	High	Functional	End User's	U3	Check temperature after every azan
15	High	Functional	End User's	U3	Auto start heater and geezer if temperature is LOWER than specific point.
16	High	Functional	End User's	U3	Auto start fan and AC if temperature is higher than specific point.
17	High	Functional	End User's	U6	Auto start lights when it will detect darkness
18	High	Functional	End User's	U6	Activate another sensor after mid midnight for electricity saving
19	High	Functional	End User's	U4	Start water shower if sensor feels the ground dry
20	High	Functional	End User's	U7	Signal will be open for the more time in which side the flow of traffic is more than the other sides.
21	Medium	Non	End User's	N/A	All hardware components will

		Functional			be placed in a model
22	High	Non Functional	End User’s	N/A	An demo of project will be working on all of the above modules.
23	High	Non Functional	End User’s	N/A	Availability of internet is necessary for using the application and communication b/w software and hardware.
24	Medium	Non Functional	End User’s	N/A	Admin is authorized to update the prayers time if required.
25	High	Non Functional	End User’s	N/A	Database should be consistent no same Record should be taken by the database.

2.6 Final Deliverable of project

Final deliverable of this project are

- Android application
- Hardware system

2.7 Beneficiaries of Project

Every Citizen present in the city or this geographical area will be the beneficiaries.

2.8 Resources Requirements

We need the resources that are mentioned below:

- Formic Sheet
- Aurdino Mega
- Aurdino UNO
- Wi-Fi Module
- Android Mobile
- Sensors
- Lithium battery
- Laptop
- Color charts
- Wires
- Motors

CHAPTER 3

DESIGN AND METHODOLOGY

Following artefacts included in this Chapter

1. Use case diagram
2. Use case description
3. Sequence Diagram
4. Collaboration Diagram
5. Domain Model
6. Design Class Diagram
7. Data Model
8. Methodology

3.1 Use case Diagram

Use case diagram of system level at all module at system level is given below.

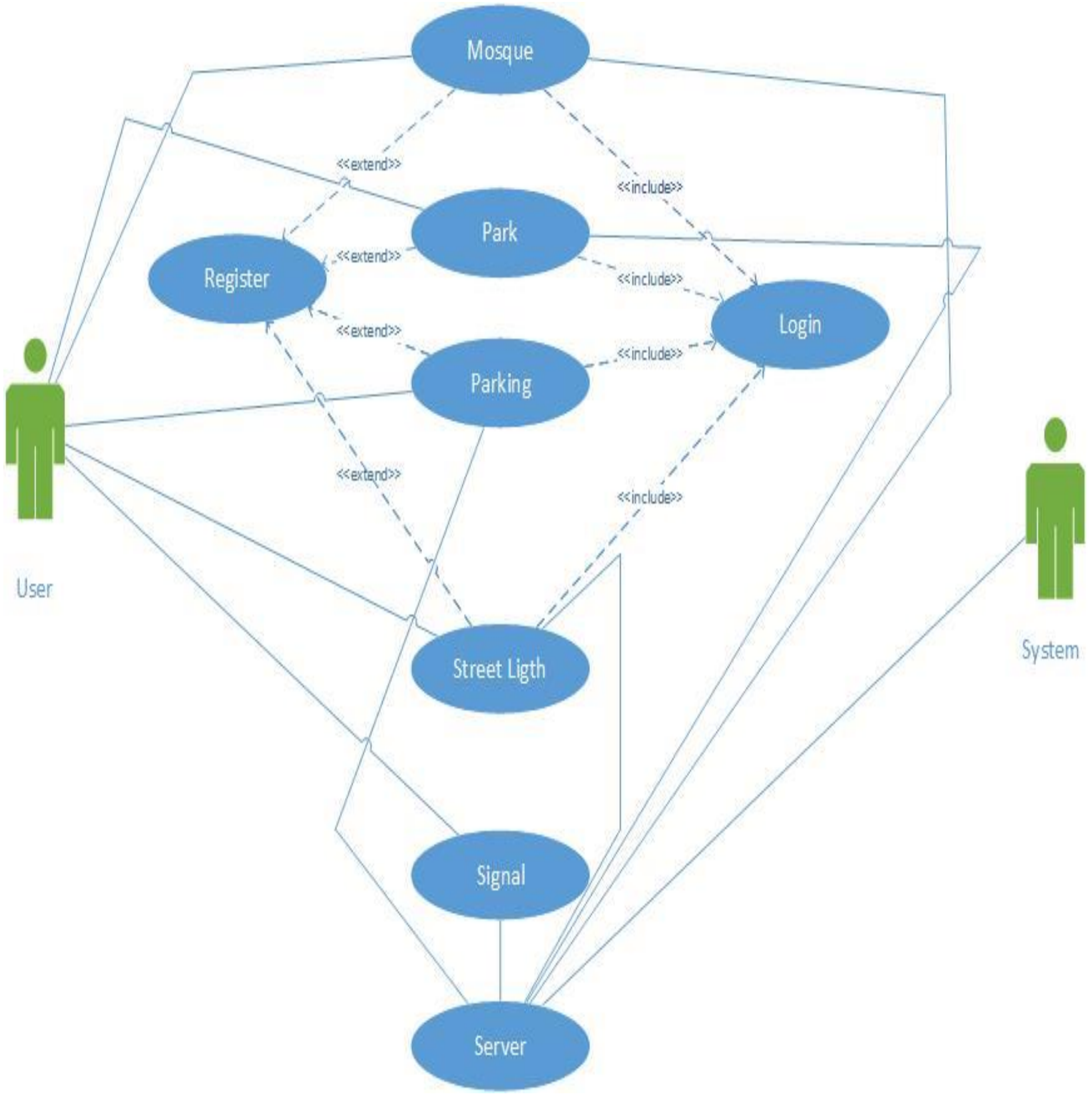


Figure 1: Smart City System Level Use Case Diagram

3.1.1 Mosque Module Use case Diagram

The use case diagram of Mosque module is given below

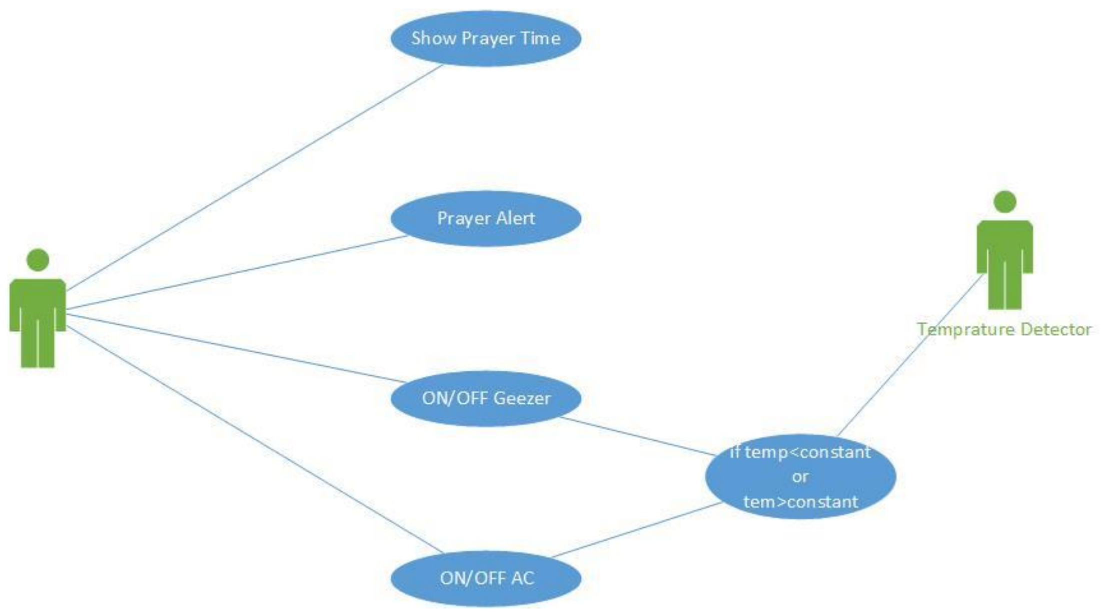


Figure 2: Mosque Module Use Case Diagram

3.1.2 Park Watering Module Use Case Diagram

The use case diagram of park watering system is given below.

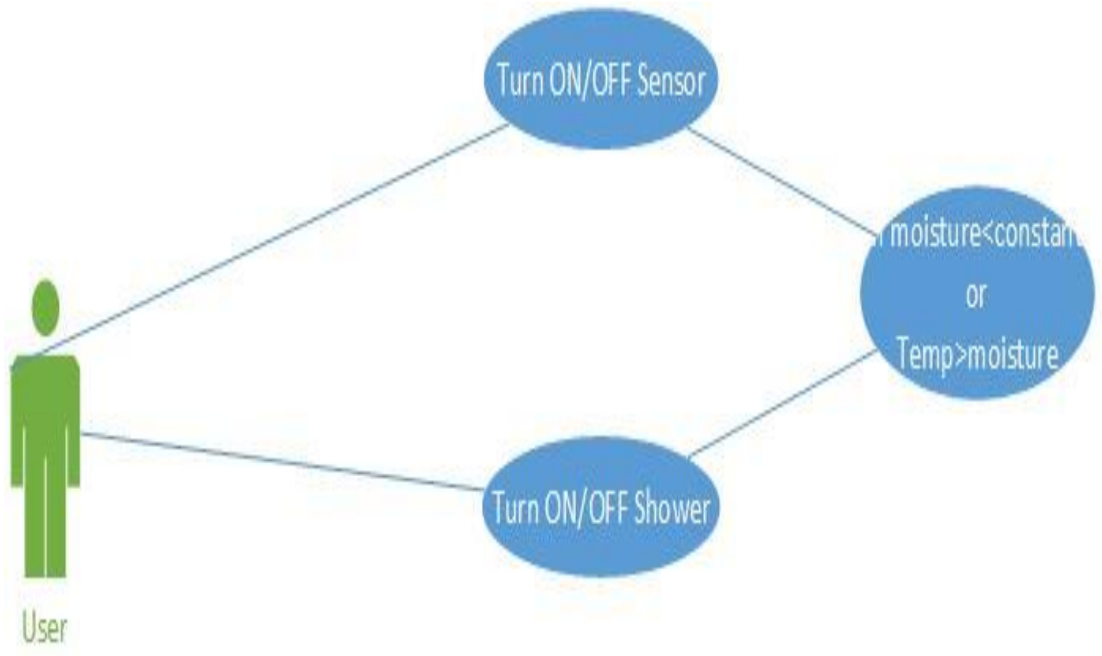


Figure 3: Park watering Module Use Case Diagram

3.1.3 Street light Module Use Case Diagram

Use Case diagram of street light module is given below

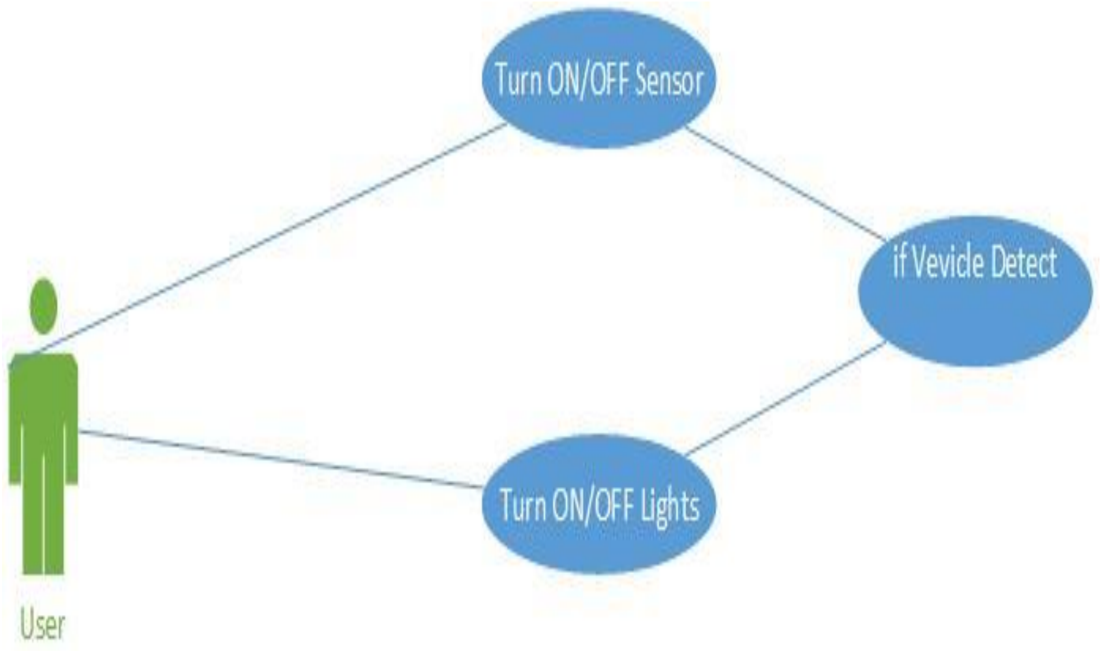


Figure 5: Street light Module Use Case Diagram

3.1.4 Traffic signal Module Use Case Diagram

The use case diagram of Signal module is given below

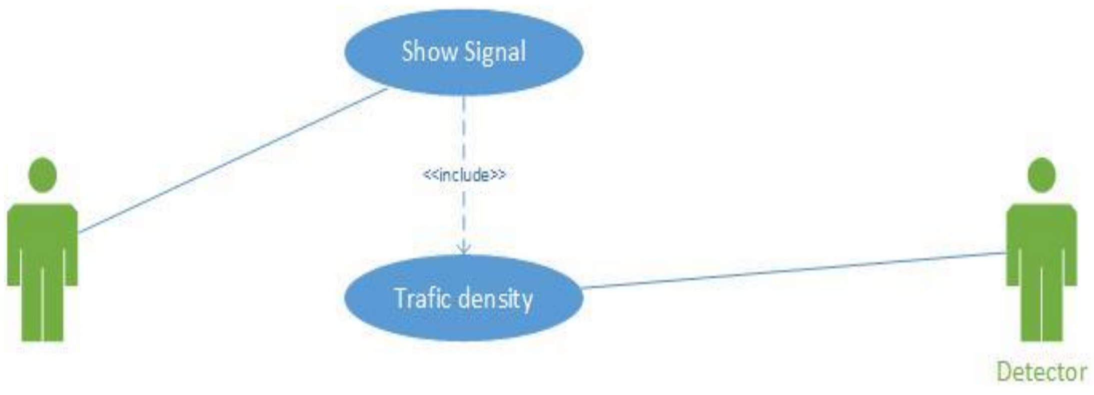


Figure 6: Traffic signal Module Use Case Diagram

3.2 Use Case Description

Following are the narrative parts of every bubble in above use case diagrams.

Table 2: Use Case User Registration

Name	Registration
Use-Case ID	U1
Priority	High
Primary Actor	User
Other participating Actor(s)	System
Description	This use case describes the event of a user registering for android application.
Pre-condition	User should not register yet
Trigger	This use case initiate when a new user going to register i
Typical flow of events	1. User gives new username and a new password 2. User will be registered.
Alternate flow of event	Alt-1 user gives wrong username and a popup error displays Alt-3 user gives wrong password and a popup error displays
Post condition	User is successfully registered
Alternate post condition	User isn’t register

Table 3: Use Case User login

Name	Login
Use-Case ID	U2
Priority	High
Primary Actor	User

Other participating Actor(s)	System
Description	This use case describes the event when user going to login at android application.
Pre-condition	1.User must be registered at Smart city application 2.User should not login
Trigger	This use case initiate when a user going to Login
Typical flow of events	1. User gives username and a password 2. System verifies User Credentials
Alternate flow of event	Alt-1 User gives wrong credentials, system generate a popup error and take user to login page
Post condition	User successfully logged in
Alternate post condition	Login failed

Table 4: Use Case Mosque Module

Name	Mosque Module
Use-Case ID	U3
Priority	High
Primary Actor	User
Other participating Actor(s)	System
Description	This use case describes the event when User enters Mosque Module
Pre-condition	Customer must open Smart application and clicks on Mosque Module
Trigger	This use case initiate when a user clicks Mosque Module

Typical flow of events	1.Enters the smart city application 2. Check prayer time 3.initiate alert for prayers 4.Check the Temperature status
Alternate flow of event	Alt-1.User will not check prayer time. Alt-2 user will not check temperature.
Post condition	Successfully check prayer time Successfully check temperature of Mosque
Alternate post condition	User will not enter mosque module

Table 5: Use Case Park watering Module

Name	Park Watering Module
Use-Case ID	U4
Priority	High
Primary Actor	User
Other participating Actor(s)	---
Description	This use case describes the park watering Module. In automatically watering of plants is checked .
Pre-condition	1.User must enter the park watering module to check status
Trigger	This use case initiate when a user will enter park watering module to check the status
Typical flow of events	1. Open Smart city application. 2. Enter park watering Module. 3. Check the status of watering

Alternate flow of event	Alt-3.User will not enter the park watering module
Post condition	Successfully check the status of park watering system and control this system through application
Alternate post condition	Cannot check the status of park watering system

Table 6: Use Case Parking Module

Name	Parking Module
Use-Case ID	U5
Priority	High
Primary Actor	User
Other participating Actor(s)	---
Description	This use case describes the event when user will enter the parking module to check out parking
Pre-condition	1.user will enter the smart city application 2. User will enter parking module to check parking status. 3-if slot available he will park his vehicle.
Trigger	This use case initiate when a user will enter parking module to check parking status
Typical flow of events	1. Open smart city application. 2.can check parking status of the parking space 3. will park his vehicle
Alternate flow of event	Alt-3.user will not enter parking module
Post condition	Successfully enter parking module
Alternate post condition	Have not check the parking status

Table 7: Use Case Street light Module

Name	Street light module
Use-Case ID	U6
Priority	High
Primary Actor	User
Other participating Actor(s)	---
Description	This use case describes the event when user will enter the street light module .
Pre-condition	1.User must open Smart city application 2. User must enter the street light Module
Trigger	This use case initiate when a user enter the street light Module
Typical flow of events	1. Open smart city application. 2.Must enter street light module 3. Can check status of street lights
Alternate flow of event	Alt-3.User has not enter the street light module Alt-4 .User has not checked the status of lights
Post condition	Successfully enter the street light module
Alternate post condition	Has not enter the street light module and check the status

Table 8: Use Case Signal Module

Name	Signal Module
Use-Case ID	U7
Priority	High

Primary Actor	User
Other participating Actor(s)	---
Description	This use case describes the event when user will enter the signal module to check the status of signal
Pre-condition	1.User must open Smart city application 2. User must enter the signal module
Trigger	This use case initiate when a user enter the signal module
Typical flow of events	1.Open smart city application 2.Enter signal module 3. check the status of parking
Alternate flow of event	Alt-3. User will not enter the signal module
Post condition	Successfully enters the signal module
Alternate post condition	Have not enter the signal module

3.3 Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system. There exists sequence diagram against every use case. Following are the sequence diagrams of Smart City.

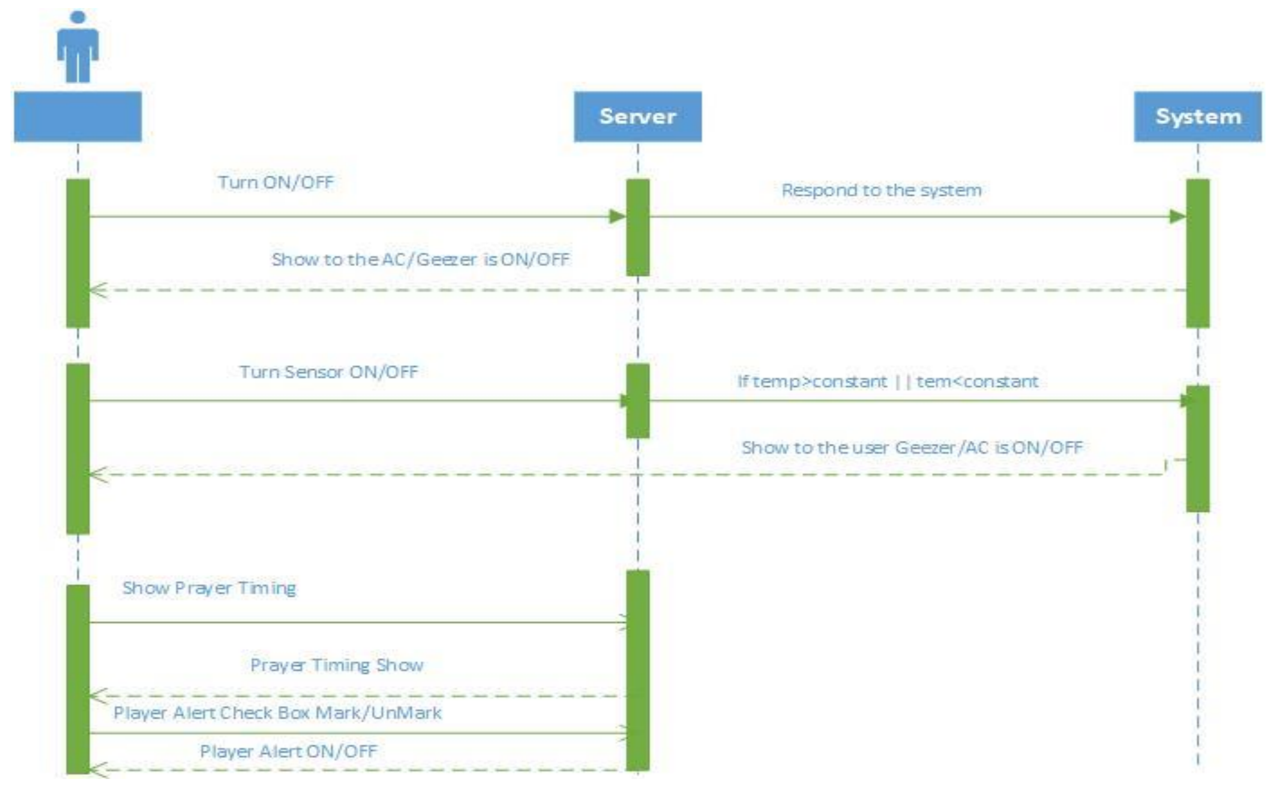

Server
System
Turn ON/OFF
Respond to the system
Show to the AC/Geezer is ON/OFF
Turn Sensor ON/OFF
If temp>constant | | tem<constant
Show to the user Geezer/AC is ON/OFF
Show Prayer Timing
Prayer Timing Show
Player Alert Check Box Mark/UnMark
Player Alert ON/OFF

Figure 7: Mosque Module Sequence Diagram

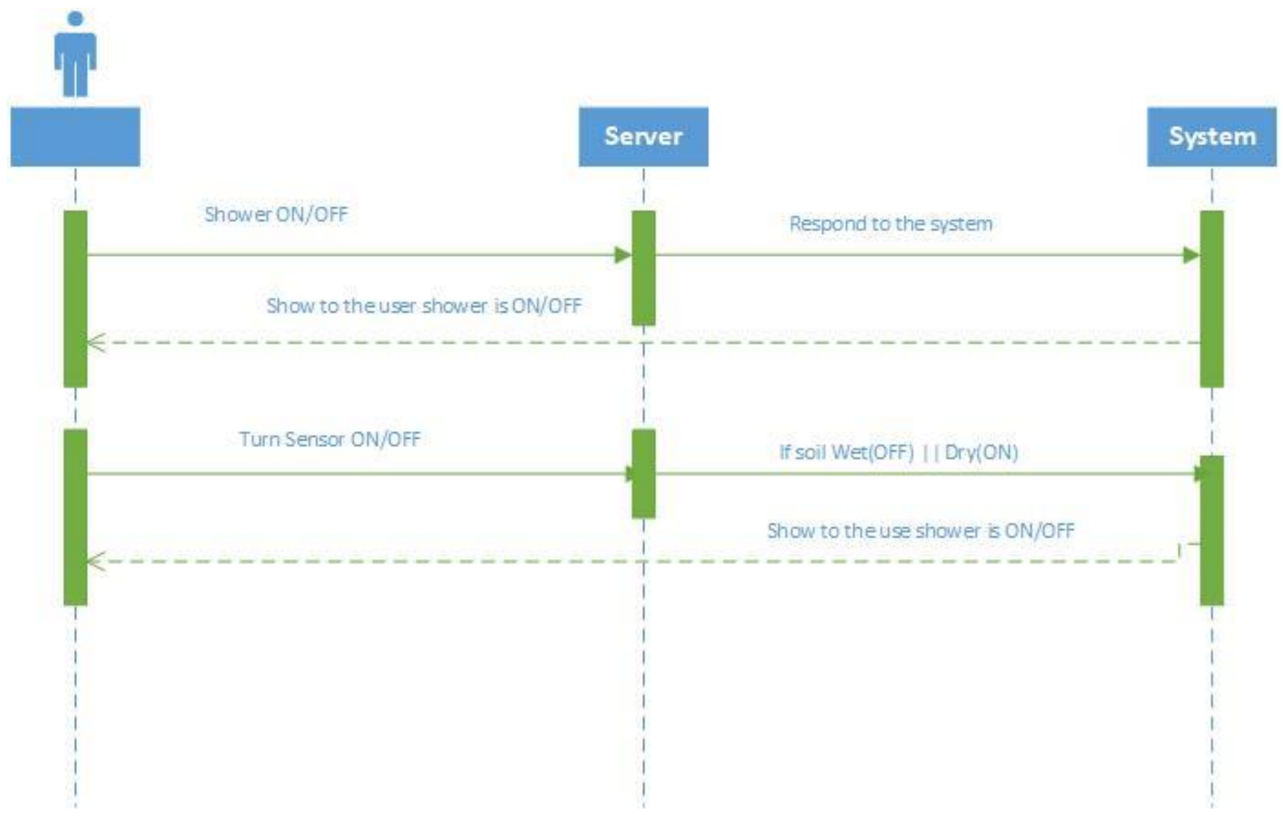

Server
System
Shower ON/OFF
Respond to the system
Show to the user shower is ON/OFF
Turn Sensor ON/OFF
If soil Wet(OFF) | | Dry(ON)
Show to the use shower is ON/OFF

Figure 8: Park Watering Module Sequence Diagram

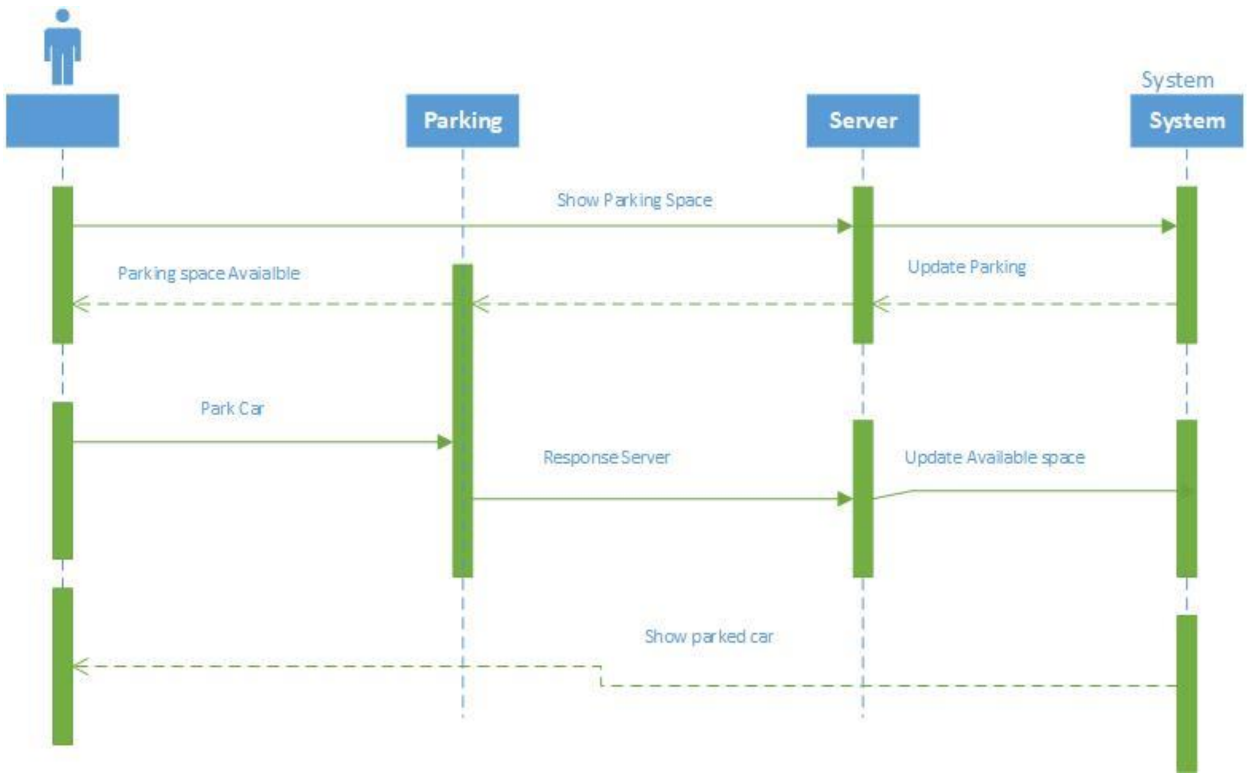


Figure 9: Parking Module Sequence Diagram

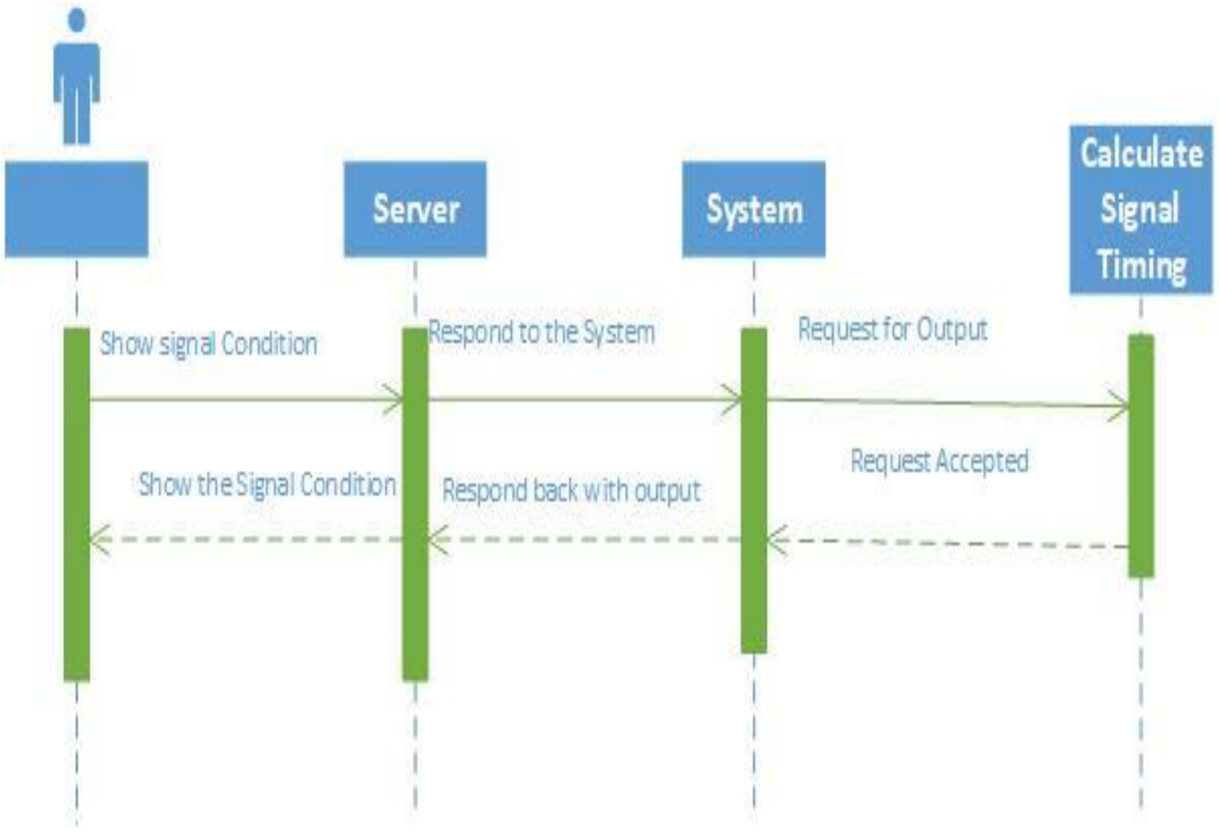


Figure10: Signal Module Sequence Diagram

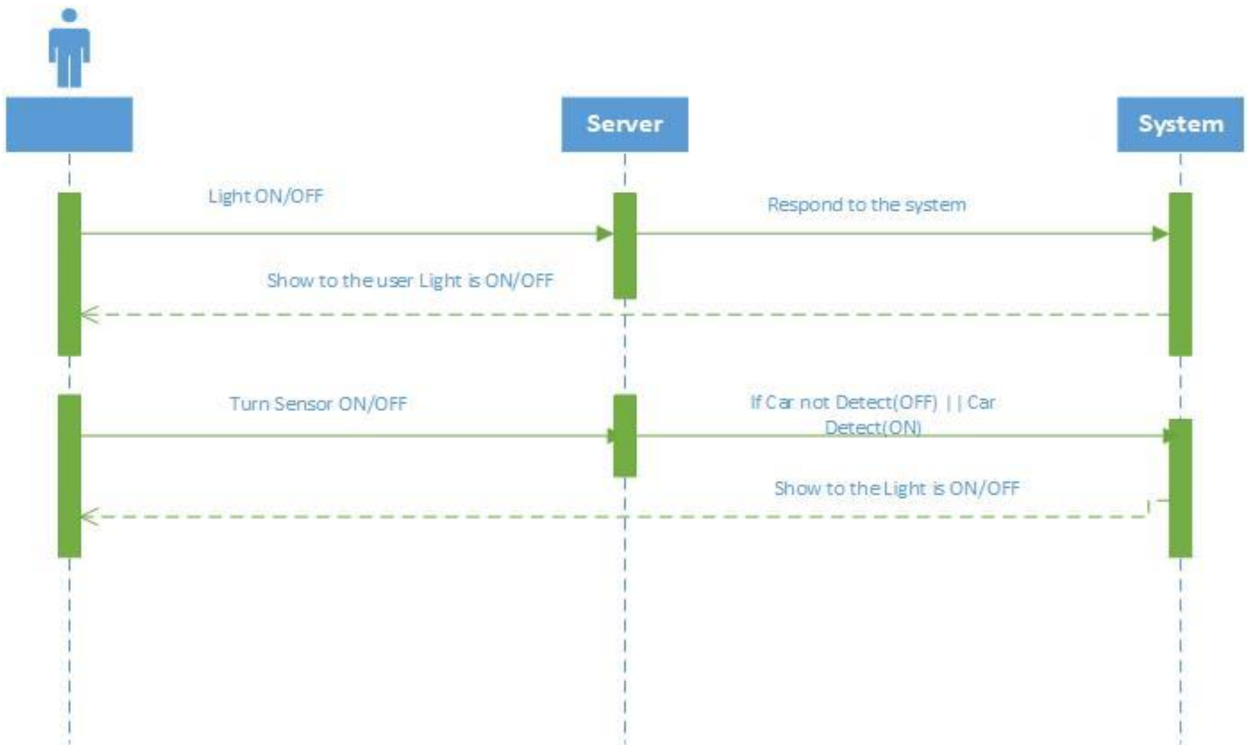

Server
System
Light ON/OFF
Respond to the system
Show to the user Light is ON/OFF
Turn Sensor ON/OFF
If Car not Detect(OFF) | | Car Detect(ON)
Show to the Light is ON/OFF

Figure 11: Street Light Sequence Diagram

3.4 Collaboration Diagram

A collaboration diagram describes a pattern of interaction among objects; it shows the objects participating in the interaction by their links to each other and the messages that they send to each other.

Collaboration diagrams are used to show how objects interact to perform the behaviour of a particular use-case, or a part of a use-case. Along with sequence diagrams, collaborations are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use-case. They are the primary source of information used to determining class responsibilities and interfaces. Unlike a sequence diagram, a collaboration diagram shows the relationships among the objects. Sequence diagrams and collaboration diagrams express similar information, but show it in different ways. Collaboration diagrams show the relationships among objects and are better for understanding all the effects on a given object and for procedural design. Because of the format of the collaboration diagram, they tend to better suited for analysis activities. Specifically, they tend to be better suited to depicting simpler interactions of smaller numbers of objects. As the number of objects and messages grows, the diagram becomes increasingly hard to read. In addition, it is difficult to show additional descriptive information such as timing, decision points, or other unstructured information that can be easily added to the notes in a sequence diagram. Following are the Collaboration Diagrams of Smart city.

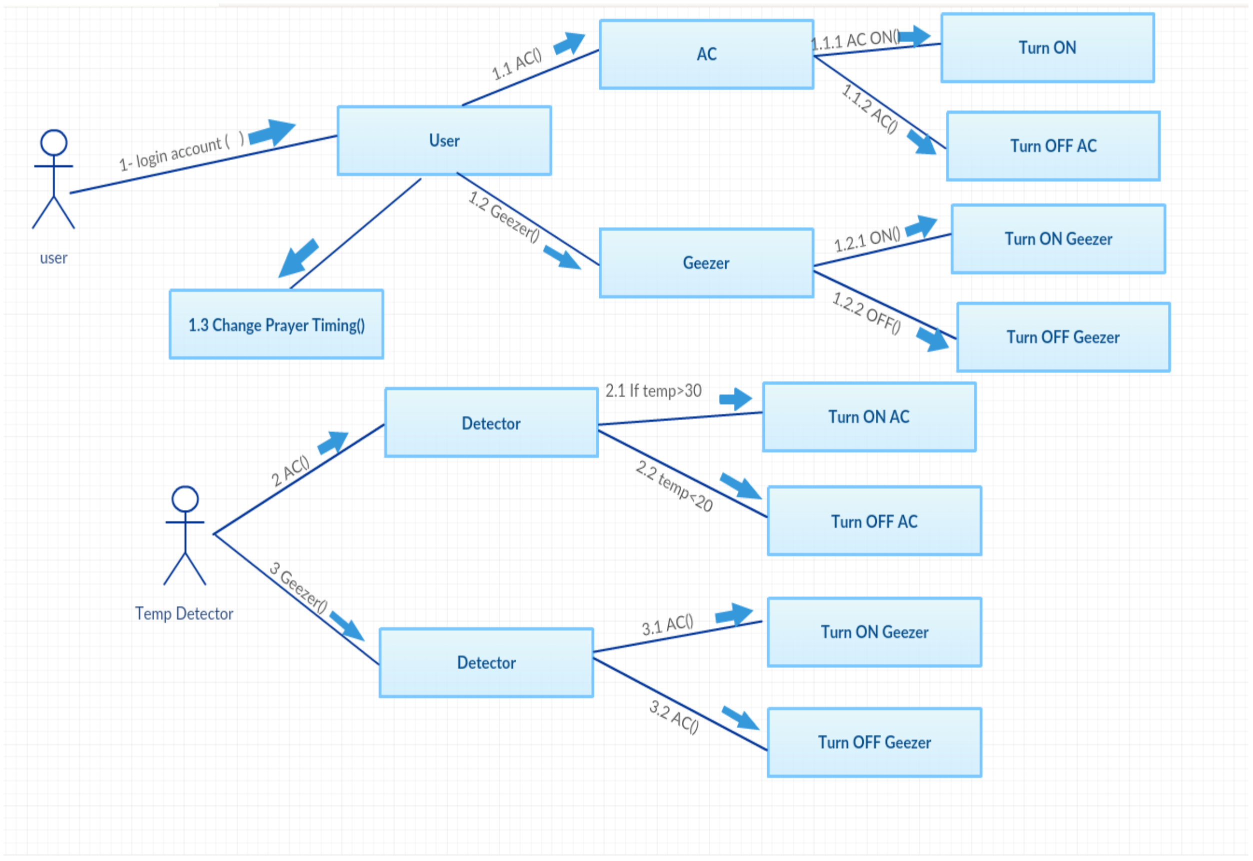


Figure 12: Mosque Module Collaboration Diagram

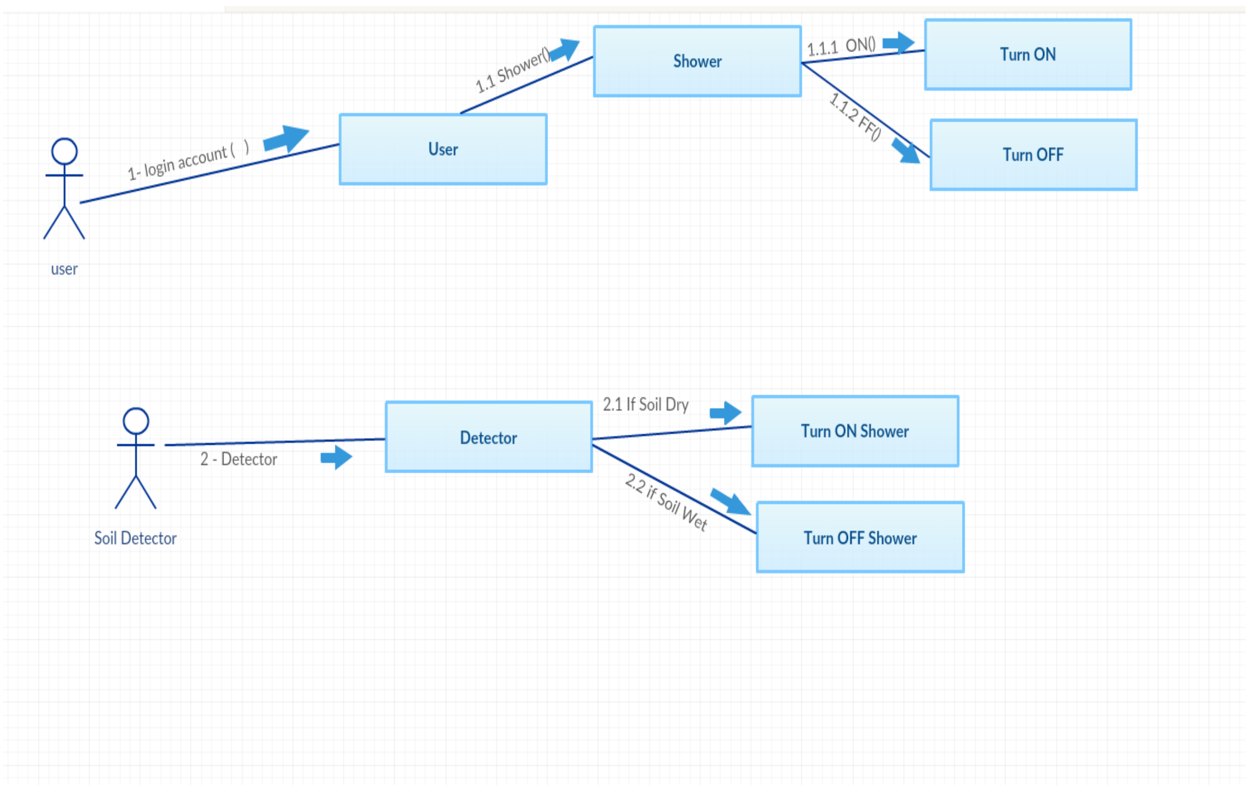


Figure 13: Park watering Collaboration Diagram

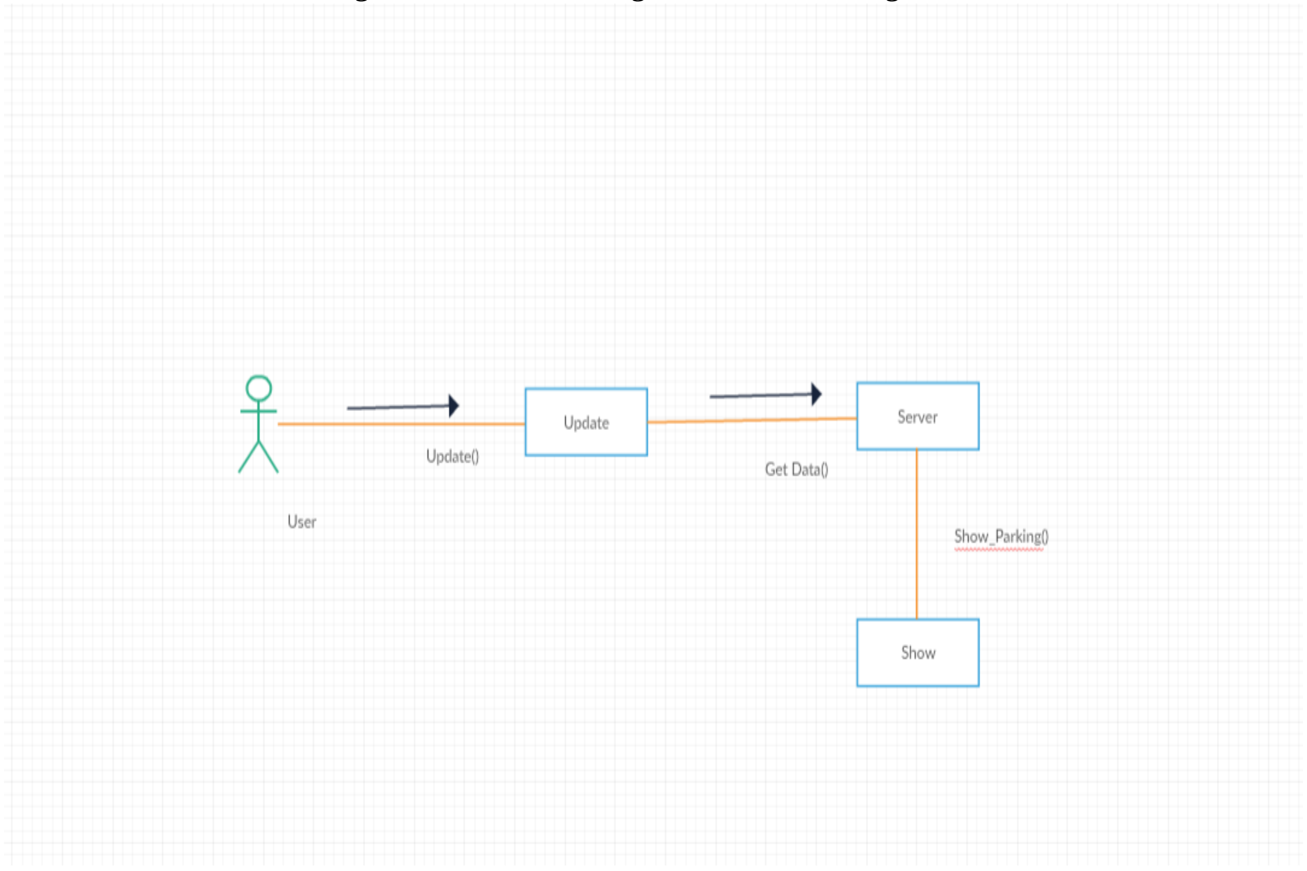


Figure 14: Parking module Collaboration Diagram

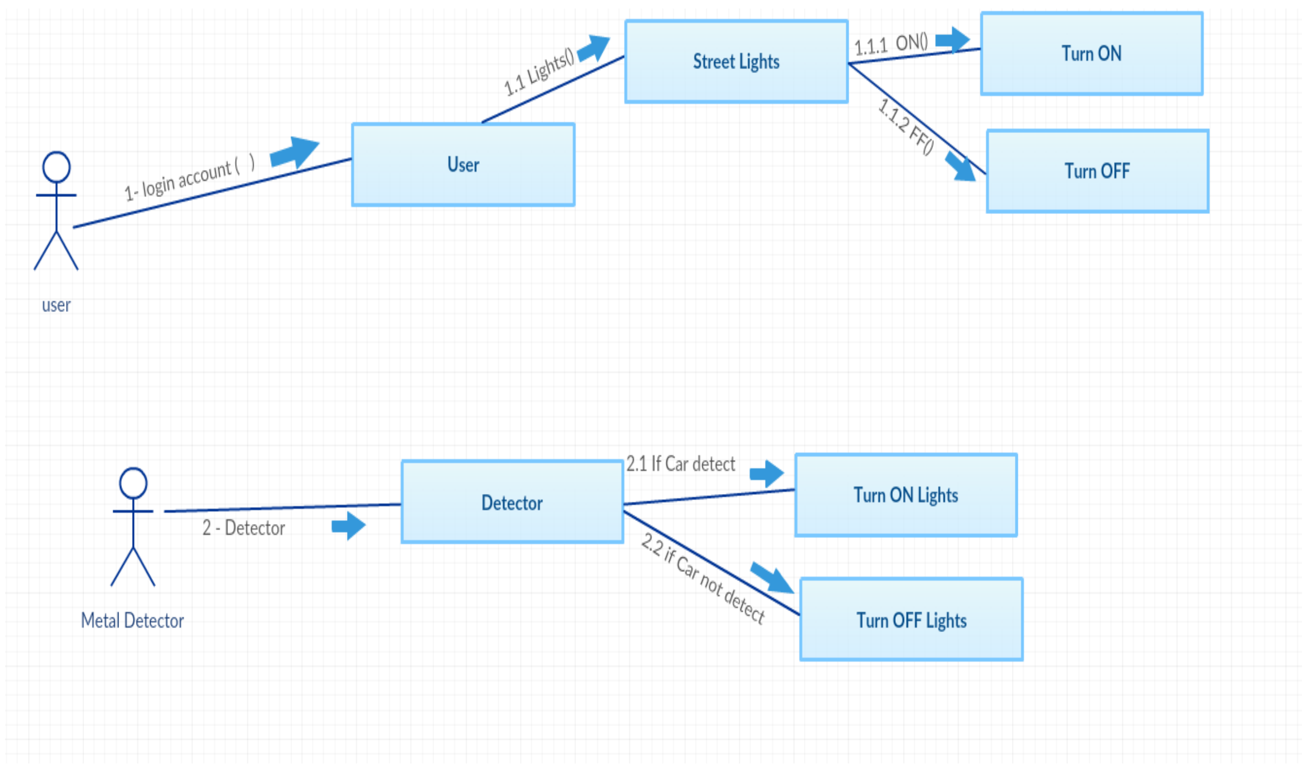


Figure 15: Street Light module Collaboration Diagram

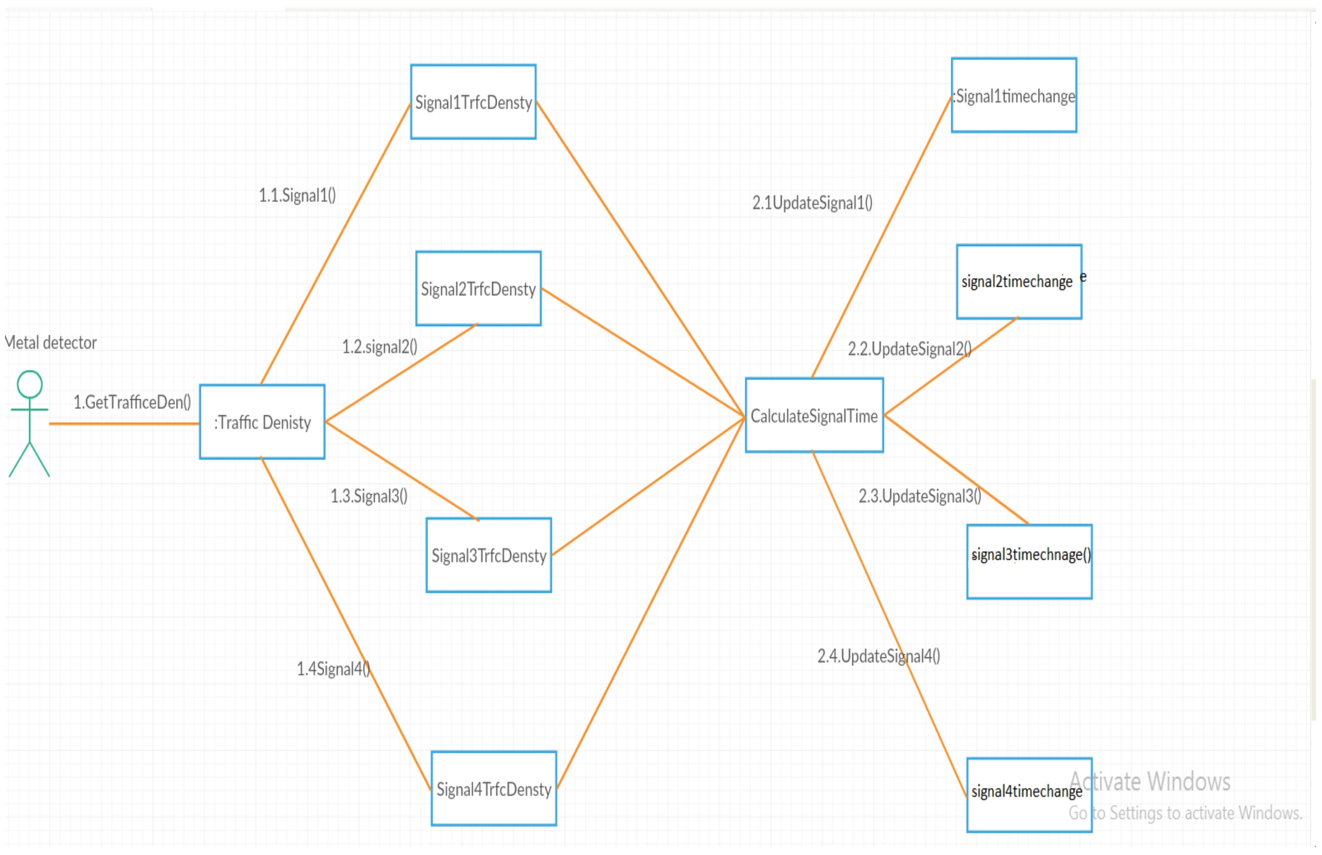


Figure16: Signal Module Collaboration Diagram

3.5 Domain Diagram

A domain model is a conceptual model of the domain that incorporates both behaviour and data. Domain Modelling is a way to describe and model real world entities and the relationships between them, which collectively describe the problem domain space. Derived from an understanding of system-level requirements, identifying domain entities and their relationships provides an effective basis for understanding and helps practitioners design systems for maintainability, testability, and incremental development. Following is the Domain Model of Smart City.

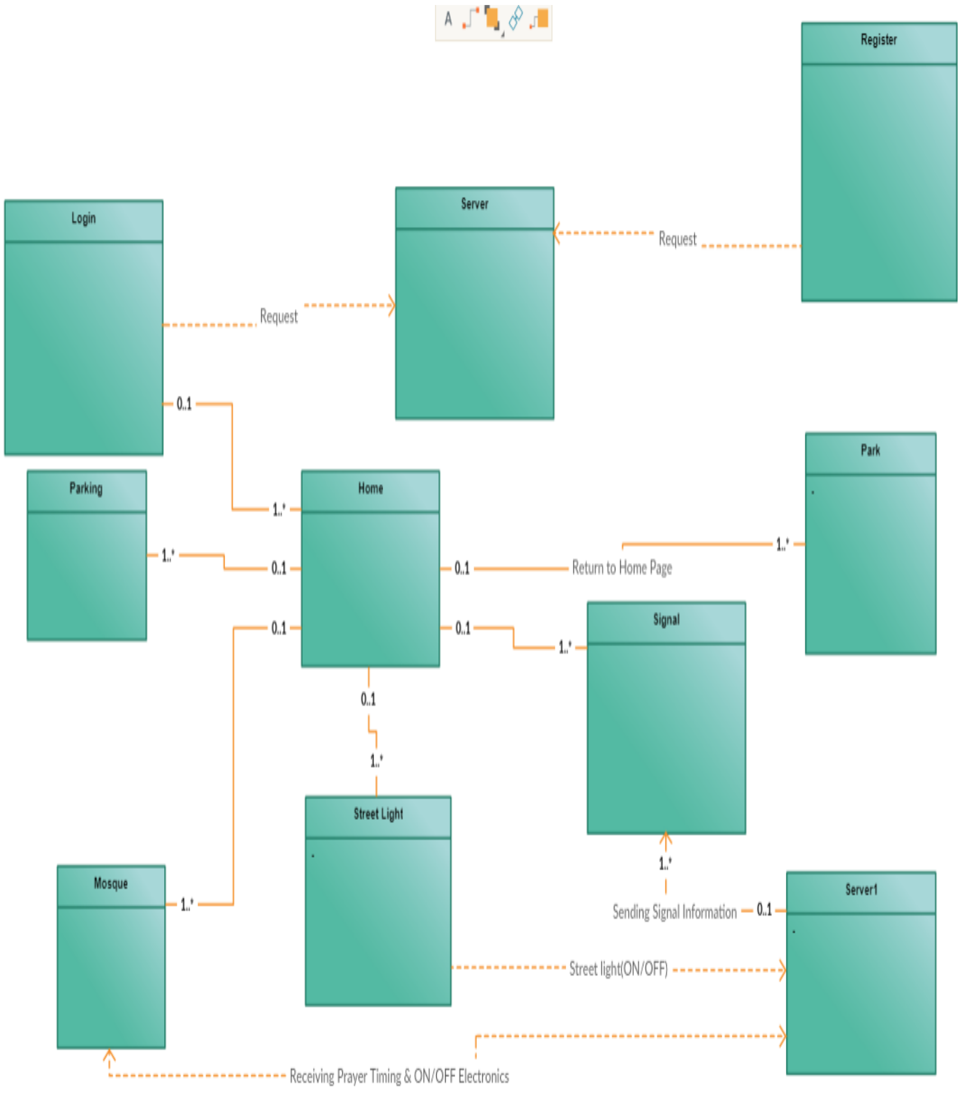


Figure 17: Domain Model Diagram

3.6 Design Class Diagram

Classes are the work-horses of the design effort—they actually perform the real work of the system. The other design elements—subsystems, packages and collaborations simply describe how classes are grouped or how they interoperate.

Capsules are also stereotyped classes, used to represent concurrent threads of execution in real-time systems. In such cases, other design classes are 'passive' classes, used within the execution context provided by the 'active' capsules. When the software architect and designer choose not to use a design approach based on capsules, it is still possible to model concurrent behaviour using 'active' classes. Active classes are design classes, which coordinate and drive the behaviour of the passive classes - an active class is a class whose instances are active objects, owning their own thread of control. Following are class diagram of Smart city.

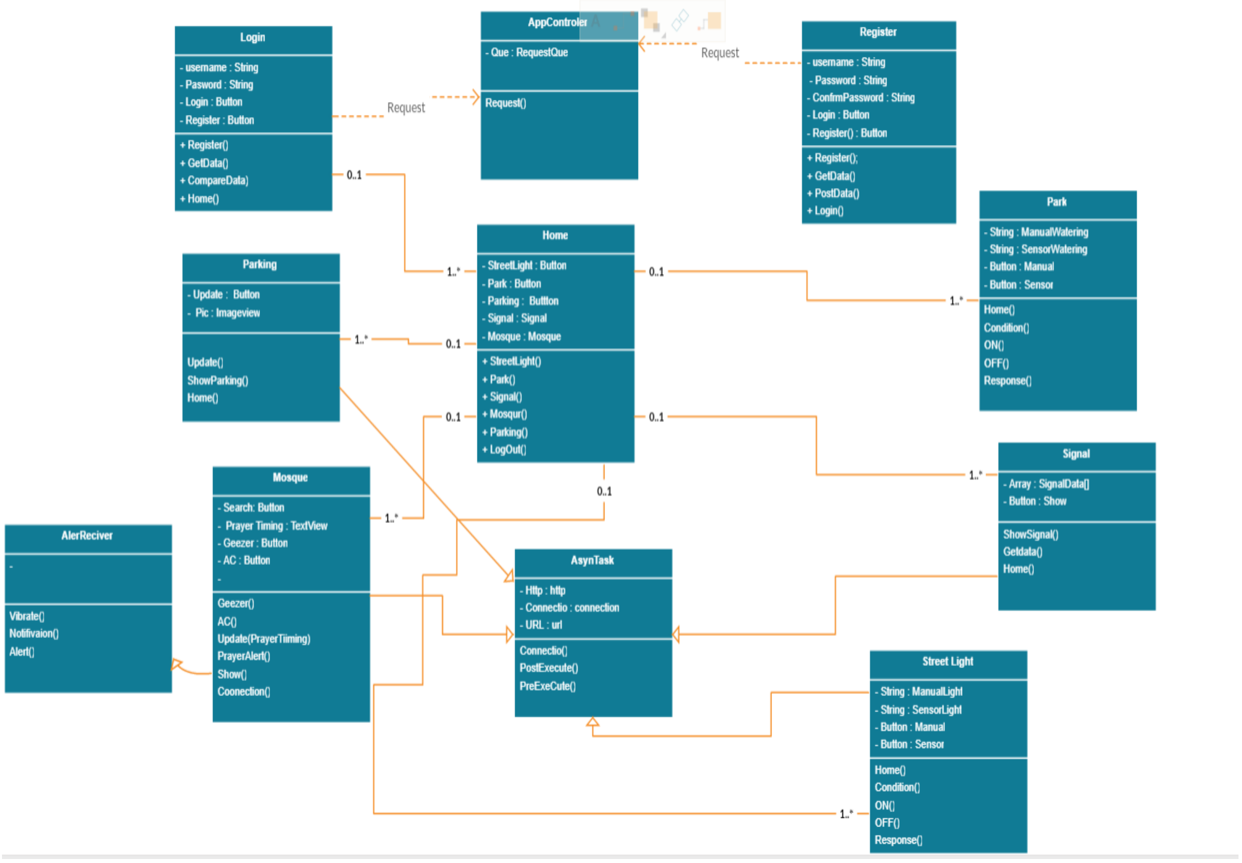


Figure 18: Design Class Diagram

3.7 Data Model

The data model is a subset of the implementation model, which describes the logical and physical representation of persistent data in the system.

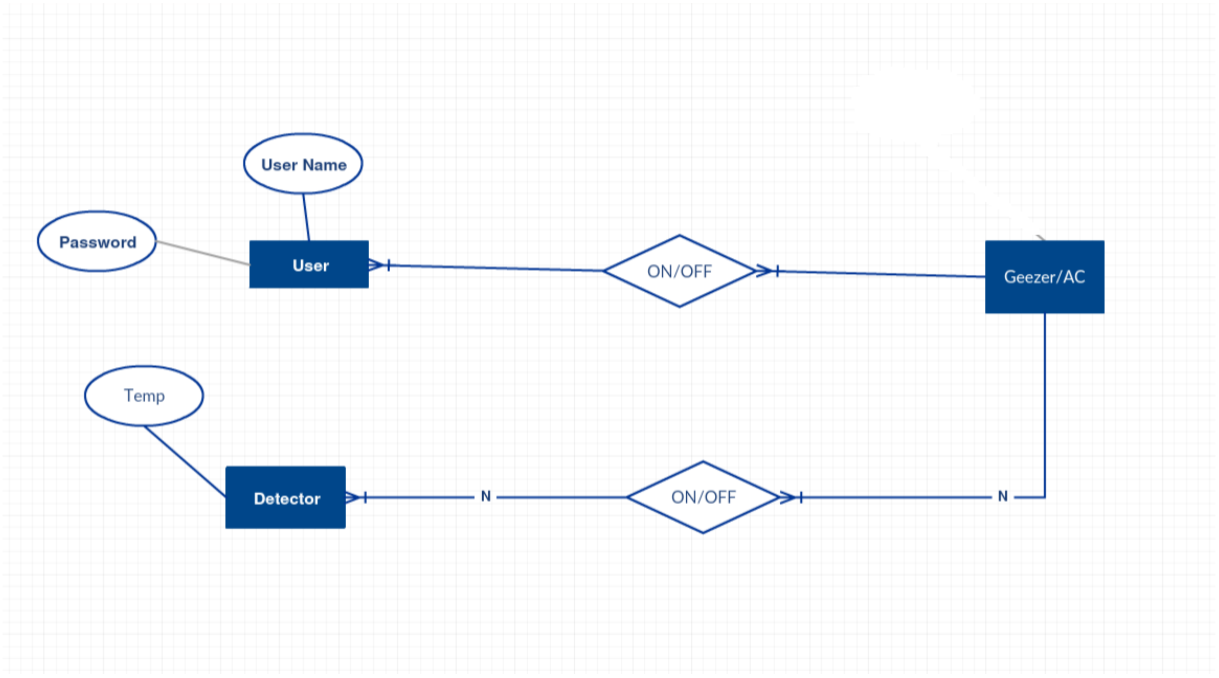


Figure 19: Mosque ERD Diagram

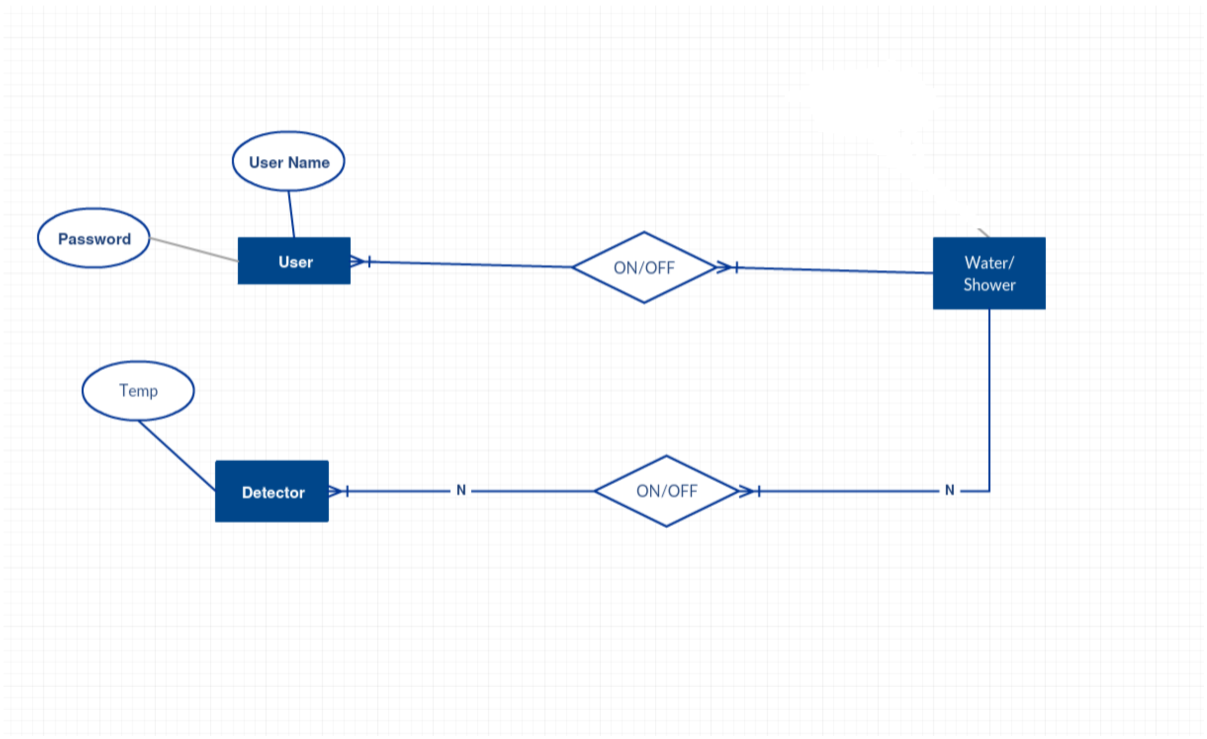


Figure 20: Park watering ERD Diagram

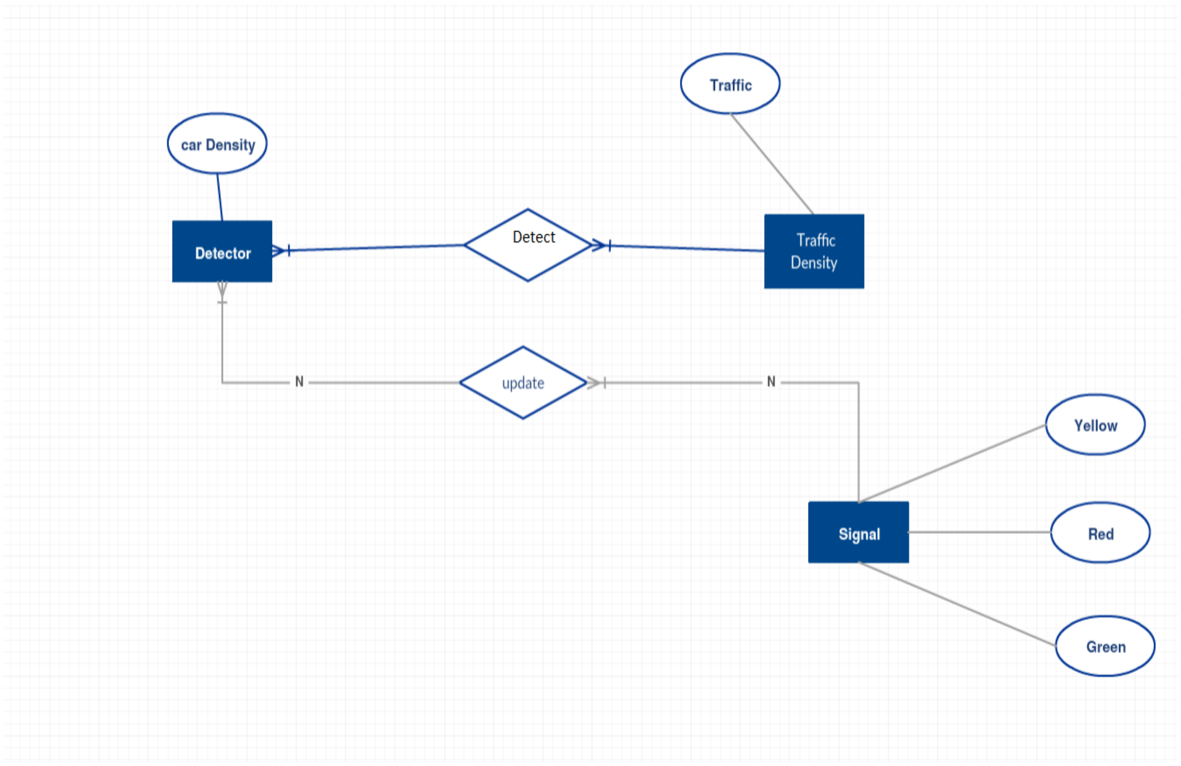


Figure 21: Signal Module ERD Diagram

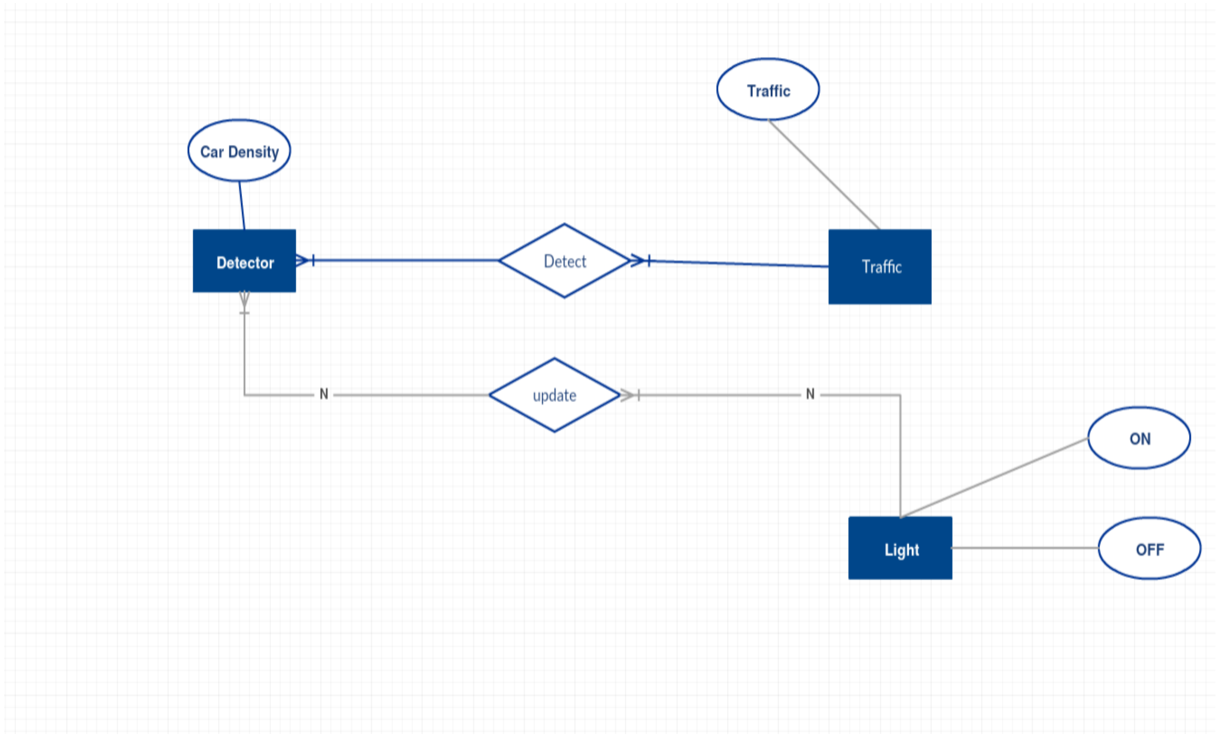


Figure 22: Street Light Module ERD Diagram

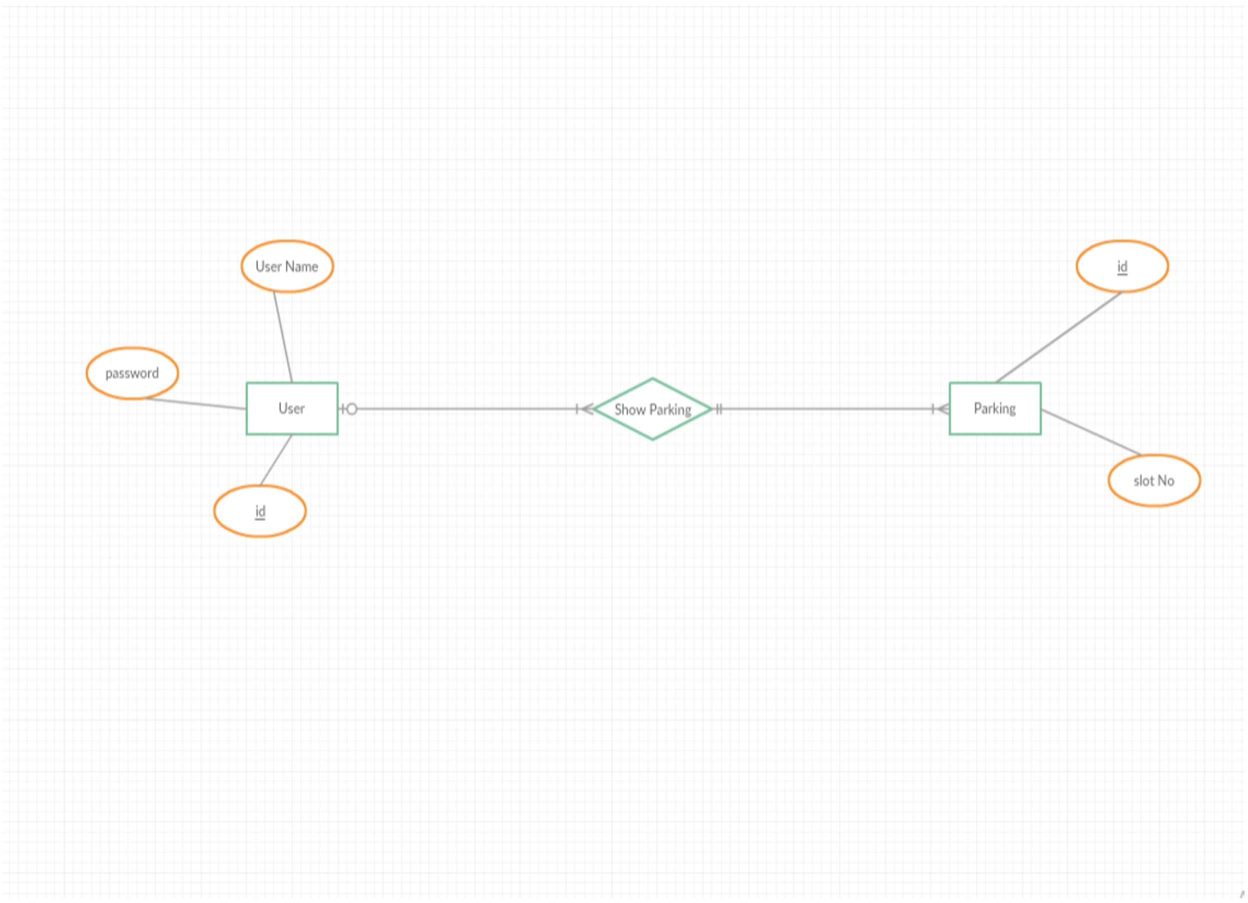


Figure 23: Parking Module ERD Diagram

3.8 Methodology

Many methodologies now exist for developing projects, so to choose an appropriate style for this project, research into the various options is necessary. We are using incremental methodology for this project. In this methodology each phases after requirement gathering passes through the design, implementation and deployment phases. Here working module is produced early. Our project is combination of different modules. Each module has separate implementation. Therefore we are using incremental methodology for our project. The benefits of this methodology are

- This model is less costly to changes requirement and scope.
- Easier to manage risks because risks are identified earlier.
- It is easier to test and identify the errors.

CHAPTER 4

IMPLMENTATION

4.1 Implementation

Smart city is cluster of hardware and software. With the help of this city is automated and become modern and controlled with the help of android application. Smart city is basically combination of different modules. Therefore we will develop the modules separately and then combines together and implemented on a formic board. Along this an android application will be developed and connected together through a database.

4.1.1 Implementation of first Stage

The first Stage of development process includes separate module wise development. In this stage we developed the modules separately.

4.1.1.1 Mosque Module

In Mosque module we have done mosque automation. Which includes the automation of geezer lights, fan and air conditioner. If temperature is less a normal temperature it will turn on geezer and if temperature is greater than normal temperature it will turn on Air conditioner.

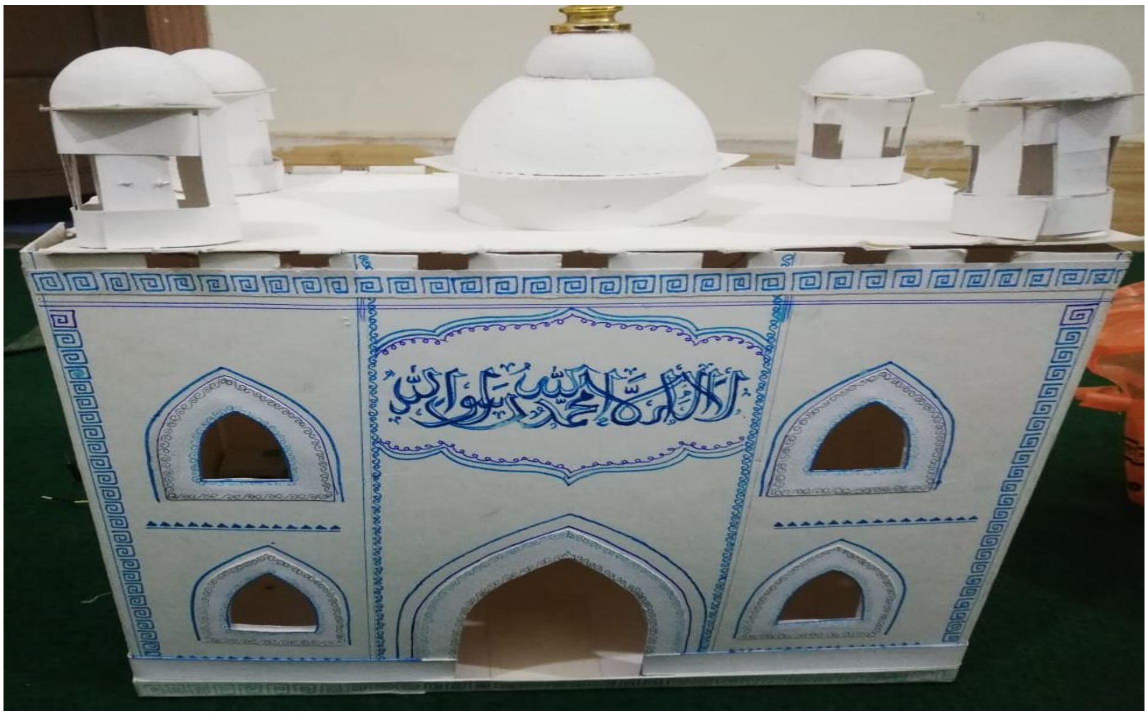


Figure 24: Mosque Module Diagram

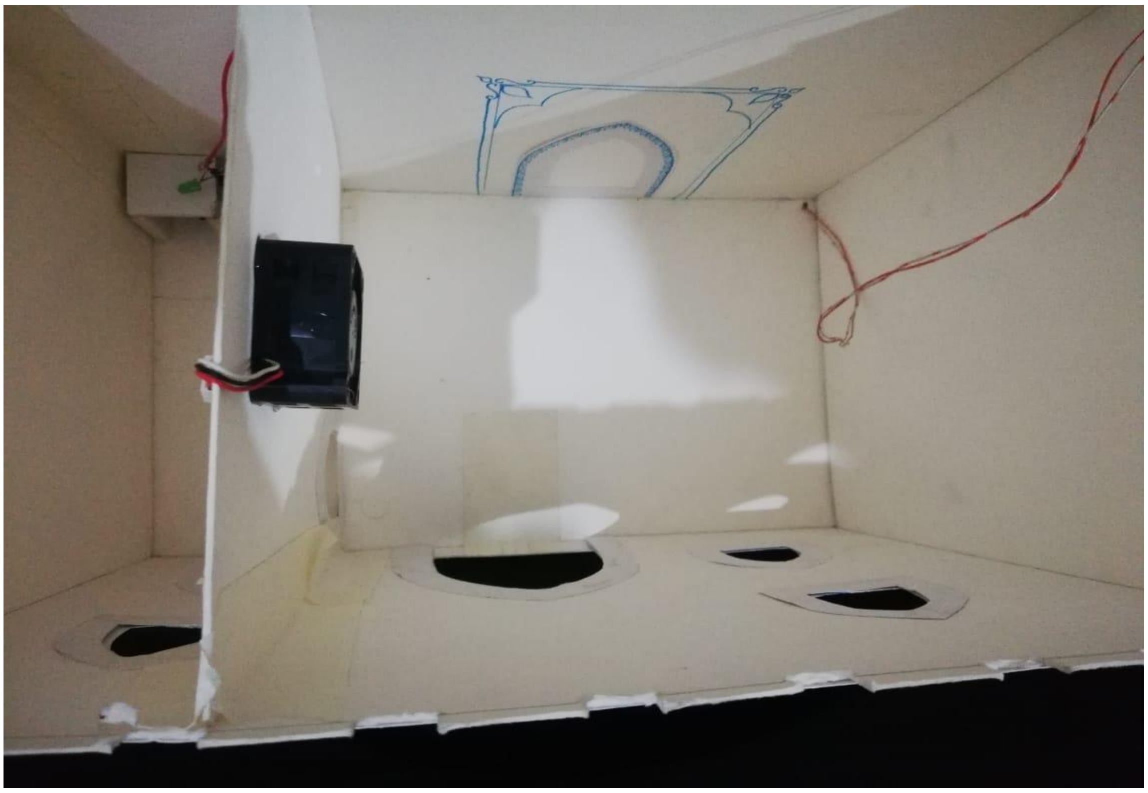


Figure 25: Mosque Module implementation Diagram

4.1.1.2 Park Module

In park module the implementation of park module is finalized. In the park watering system is automated. Which is controlled with the help of a sensor which detect the moisture and if the soil is moist it stops watering and of soil is dry it start watering plants.

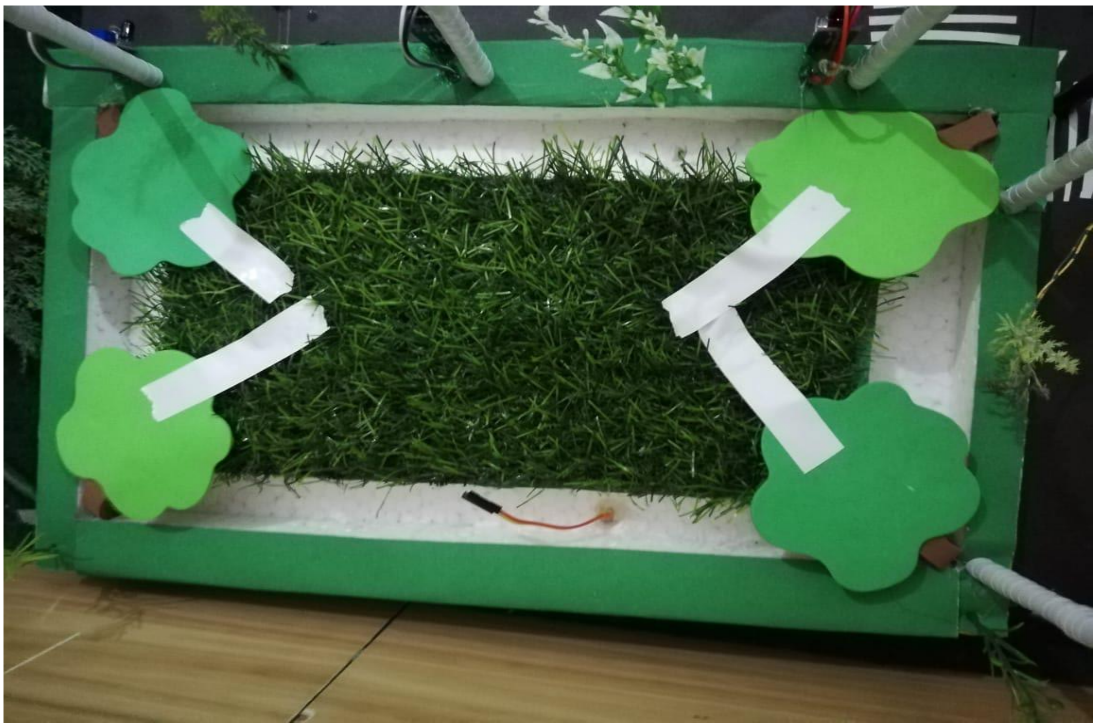


Figure 26: Park watering Module Diagram

4.1.1.3 Parking Module

In this Module the parking of vehicle is automated which tell user space is available or not. If parking is full no vehicle is allowed to enter the parking.

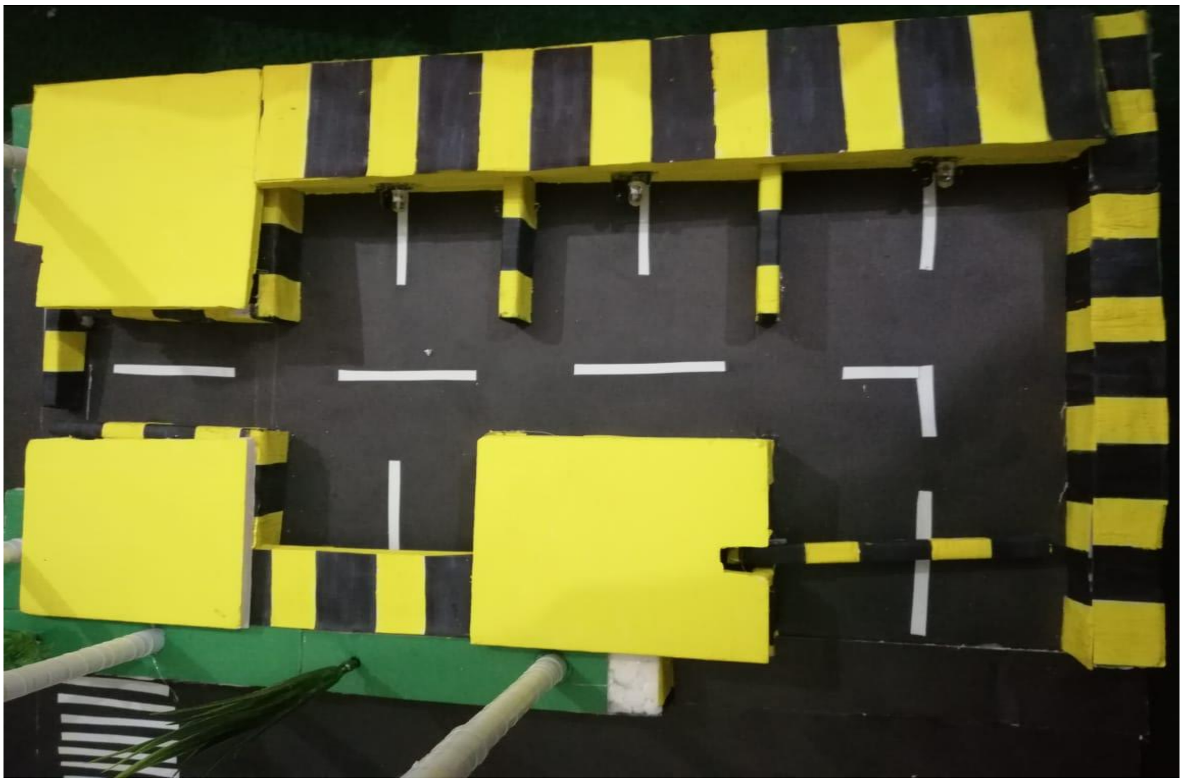


Figure 27: Parking Module Diagram

4.1.1.4 Street light Module

In street light module street lights are automated. All street light will automatically turn on in evening and for saving electricity after 12pm it will turn off because at that time traffic is low. Lights will only turn on when there is motion around it.

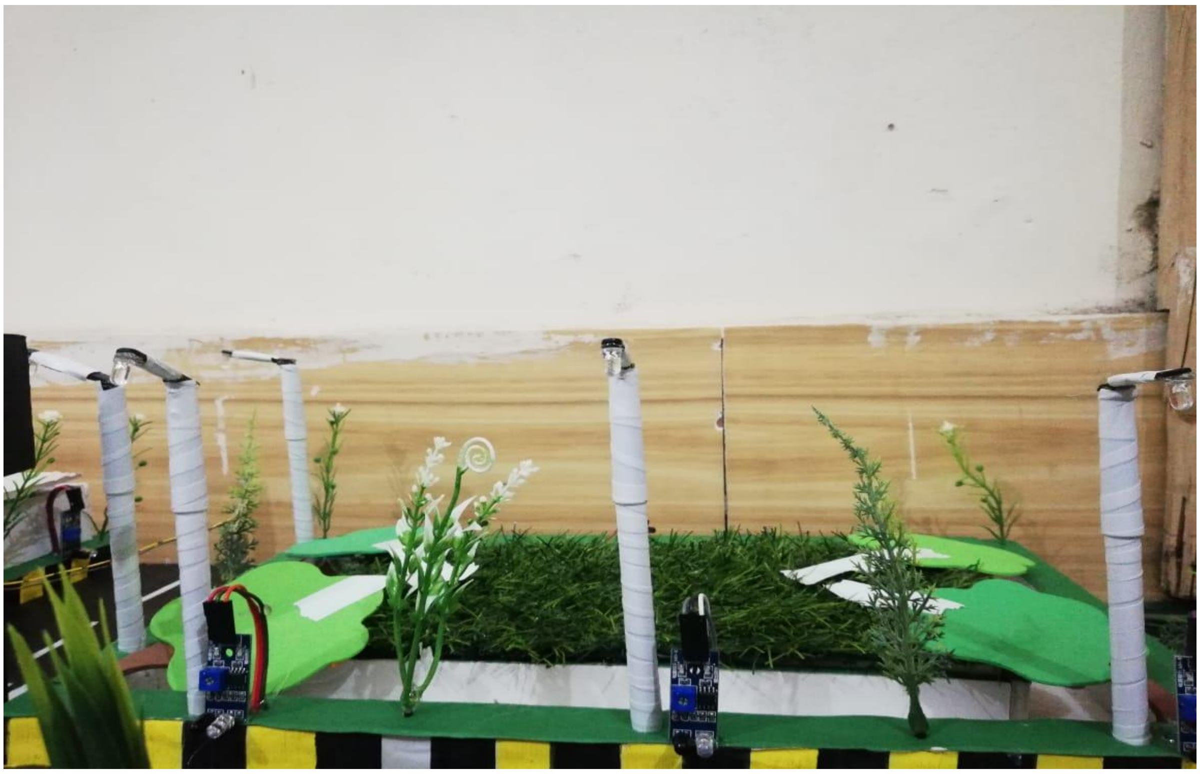


Figure 28: Street Light Module Diagram

4.1.1.5 Traffic Signal Module

In this module density base traffic signal is implemented. It works like the signal of that side where traffic flow is greater will one for more time than other and the signal where there is no traffic will not open.

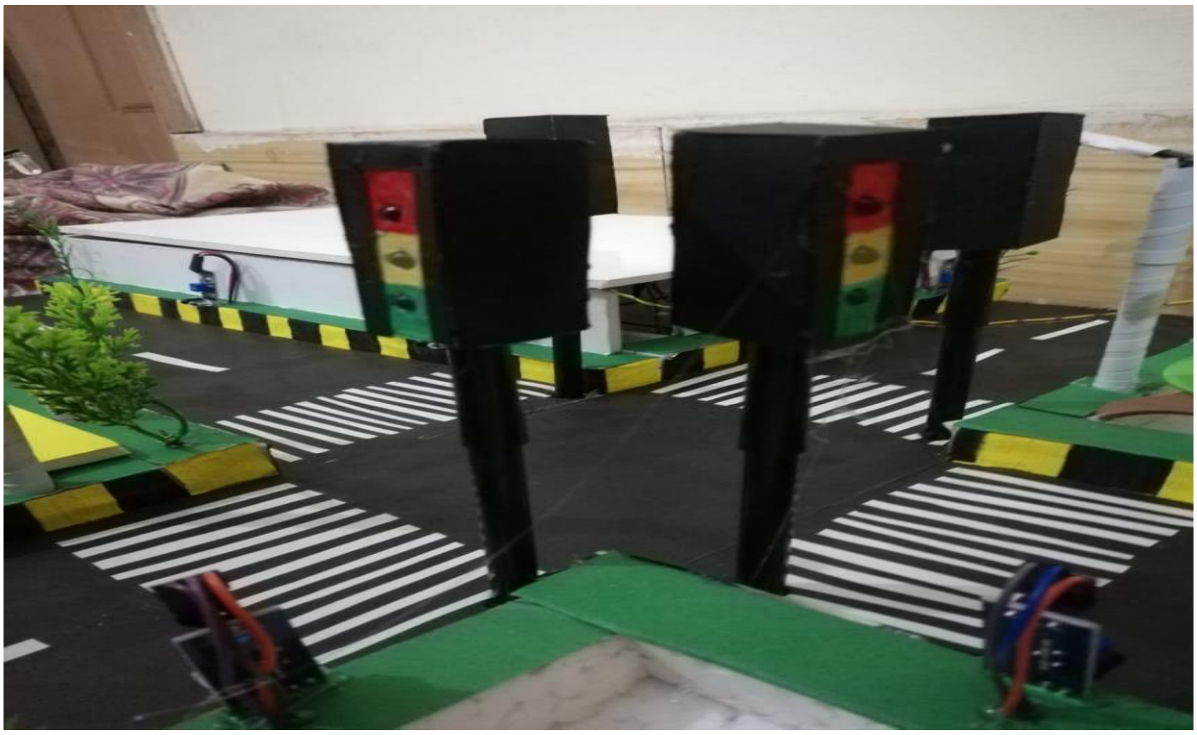


Figure 29: Traffic Module Diagram

4.1.2 Second Stage

In second stage the integration of all modules on a microcontroller is done and they all are implemented on a single formic board.

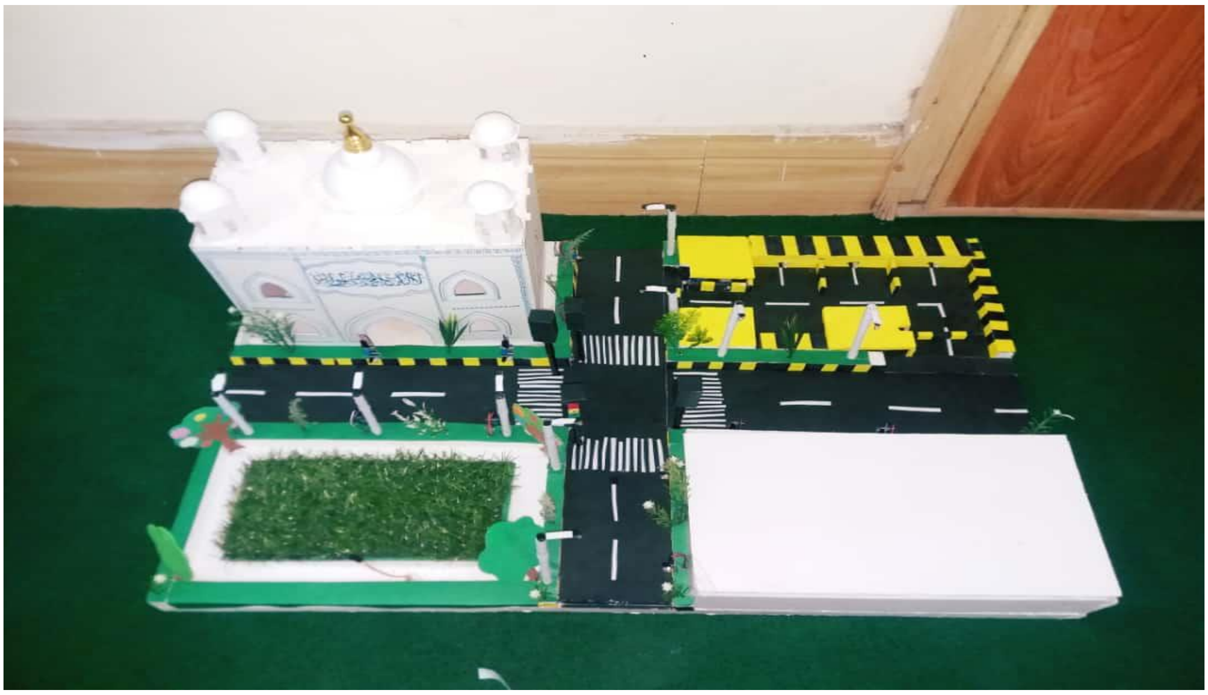


Figure 30: Integration of Module Diagram

4.1.3 Third Stage

In this stage android application is designed and developed. The android application is software side implementation. After development of android application it is also integrated in with hardware and the status of the hardware is checked here.

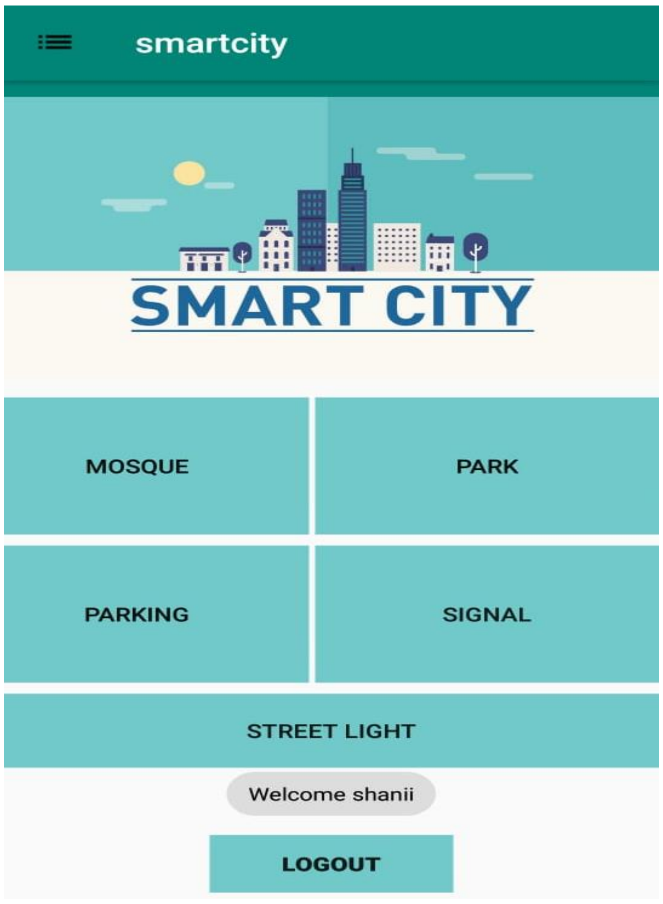


Figure 31: Android Application Diagram

4.2 Source Codes of Hardware Modules

The source code of all modules is given below

4.2.1 Mosque Module

```
int val;
int tempPin = A0;
int ledPin=12;
int ledPin1=13;
void setup()
{
Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
    pinMode(ledPin1, OUTPUT);
}
void loop()
{
val = analogRead(tempPin);
float mv = ( val/1024.0)*5000;
float cel = mv/10;
float farh = (cel*9)/5 + 32;
Serial.print("TEMPRATURE = ");
Serial.print(cel);
Serial.print("*C");
Serial.println();
delay(1000);
/* uncomment this to get temperature in farenhite
Serial.print("TEMPRATURE = ");
Serial.print(farh);
Serial.print("*F");
Serial.println();
*/
if(cel<10.0)
{
 digitalWrite(ledPin, HIGH);
   delay(1000);
}
else if(cel>20.0)
{
 digitalWrite(ledPin1, HIGH);
   delay(1000);
}
else
{
  digitalWrite(ledPin, LOW);
  digitalWrite(ledPin1, LOW);

delay(1000);}
}
```

4.2.2 Parking Module

```
#include <SoftwareSerial.h>
#include <Servo.h>
SoftwareSerial ArduinoMega(25, 24);
int ir = A0;
int ir1 = A1;
int ir2 = A2;
int ir3 = A3;
int ir4 = A4;
String slot1;
String slot2;
String slot3;
int sensorValue = 0;
int sensorrValue = 0;
int ssensorValue = 0;
int seensorValue = 0;
int sennsorValue = 0;
int servoPin = 9;
int servooPin = 8;
int i = 0;
int j = 0;
Servo servo;
Servo servoo;
```

```
String Data = "";
int dat = 0;
void setup() {
  delay(1000);
  Serial.begin(9600); // Starts the serial communication

 ArduinoMega.begin(115200);
   pinMode(24, OUTPUT);
  pinMode(25, INPUT);
   pinMode (ir,INPUT);
    pinMode (ir1,INPUT);
     pinMode (ir2,INPUT);
     servo.attach(servoPin = 9);
servoo.attach(servooPin = 8);
}
void loop(){
ssensorValue = analogRead(ir1);
  seensorValue = analogRead(ir2);
  sennsorValue = analogRead(ir3);

  if (ssensorValue<500 && seensorValue<500 && sennsorValue<500)
  {
        servo.write(0);
    delay(100);
  }
  else
{    sensorValue = analogRead(ir);
      if (sensorValue <600) // entrance Gate
  {
        for (int i=0;i<90;i++)
        servo.write(i);
    delay(50);
        delay(5000);
  }     }
```

```
     if(sensorValue >600)


    {
     for (int i=90;i>0;i--)
    servo.write(i);
          delay(500);
 }
  sensorrValue = analogRead(ir4);
    if (sensorrValue <600) // exit Gate
      {
        for (int j=0;j<=90;j++)
            servoo.write(j);
            delay(30);
        delay(3000);
      }
 if(sensorrValue >600)
  {
  for (int j=90;j>0;j--)
    servoo.write(j);
    delay(60);
  }
int seensorValue=analogRead(A1);
  Serial.println(seensorValue);
       delay(1);
if (analogRead(A1)<500)
  {
    Serial.print("Slot 1 Occupied ");
    Serial.println();
    slot1 = "Occupied";
  }
  else
  {
    Serial.print("Slot 1 empty");
    Serial.println();
    slot1 = "empty";

delay(250);}
  if (analogRead(A2)<500)
  {
    Serial.print("Slot 2 Occupied ");
    Serial.println();
    slot2 = "Occupied";
  }
  else
  {
    Serial.print("Slot 2 empty");
    Serial.println();
    slot2 = "empty";

  delay(250);

  }
  if (analogRead(A3)<500)
  {
    Serial.print("Slot 3 Occupied ");
    Serial.println();
    slot3 = "Occupied";
  }
  else
  {
    Serial.print("Slot 3 empty");
    Serial.println();
    slot3 = "empty";

   delay(250);  }
  // Data += "slot1=" + slot1+ "slot="+ slot2+ "slot="+ slot3;

  Data = slot1 + "-" + slot2 + "-" + slot3 + "-"  "END";
  //Data = "slot1 = empty -";
  Serial.println(Data);
  ArduinoMega.println(Data);
  ArduinoMega.println("\n");
  Data = "";
  delay(2000);  //Post Data at every 2 seconds
}
```

4.2.3 Street Light Module

```
int led = 2;
int led1 = 4;
int led2 = 7;
int led3 = 13;
int ldr = A0;
int ir = A1;
int ir1 = A2;
int ir2= A3;
void setup()
{
  Serial.begin (9600);
  pinMode (led,OUTPUT);
  pinMode (led1,OUTPUT);
  pinMode (led2,OUTPUT);
  pinMode (led3,OUTPUT);
  pinMode (ldr,INPUT);

  pinMode (ir,INPUT);

}
void loop()
{
```

```
Serial.println(analogRead(A0));
 int ldrStatus = analogRead (ldr);
   if (ldrStatus >=800)
    {
      digitalWrite(led, HIGH);
      analogWrite(led,255/4);

      digitalWrite(led1, HIGH);
      analogWrite(led1,255/4);

      digitalWrite(led2, HIGH);
      analogWrite(led2,255/4);

      digitalWrite(led3, HIGH);
          if (analogRead(A1)>500)       // IR 1 CODE
              {
               digitalWrite(led,HIGH);
               analogWrite(led,255/4);
              }
           else
               {
                 digitalWrite(led,HIGH);
                 delay(3000);// micro second
               }

         if (analogRead(A2)>500)       // IR 1 CODE
              {
               digitalWrite(led1,HIGH);
               analogWrite(led1,255/4);
              }
           else
               {
                 digitalWrite(led1,HIGH);
                 delay(3000);// micro second
```

```
            }
            if (analogRead(A3)>500)       // IR 1 CODE
           {
            digitalWrite(led2,HIGH);
            analogWrite(led2,255/4);
           }
         else
            {
              digitalWrite(led2,HIGH);
              delay(3000);// micro second
            }
    }
    else
      {
        digitalWrite(led, LOW);
        digitalWrite(led1, LOW);
        digitalWrite(led2, LOW);
        digitalWrite(led3, LOW);
           }
}
```

4.2.4 Park Watering Module

```
int led =13;
void setup()
{
  pinMode(led,OUTPUT);
  Serial.begin(9600);
}
void loop()
{
 int sensorValue= analogRead(A0);
 Serial.println(sensorValue);
 delay(1000);
 if(sensorValue >=600)
  digitalWrite(led,HIGH);
 else
  digitalWrite(led,LOW);
}
```

4.2.5 Traffic Signal Module

```
#define signal1led1 2
#define signal1led2 3
#define signal1led3 4

#define signal2led1 5
#define signal2led2 6
#define signal2led3 7

#define signal3led1 8
#define signal3led2 9
#define signal3led3 10


#define signal4led1 11
#define signal4led2 12
#define signal4led3 13


#define signel1sensor1 A0

#define signel1sensor2 A1

#define signel2sensor1 A2

#define signel2sensor2 A3

#define signel3sensor1 A4

#define signel3sensor2 A5

#define signel4sensor1 A6
```

```
#define signel4sensor2 A7
void setup() {
  Serial.begin(9600);
  pinMode(signal1led1,OUTPUT);
  pinMode(signal1led2,OUTPUT);
  pinMode(signal1led3,OUTPUT);

  pinMode(signal2led1,OUTPUT);
  pinMode(signal2led2,OUTPUT);
  pinMode(signal2led3,OUTPUT);

  pinMode(signal3led1,OUTPUT);
 pinMode(signal3led2,OUTPUT);
  pinMode(signal3led3,OUTPUT);

  pinMode(signal4led1,OUTPUT);
  pinMode(signal4led2,OUTPUT);
  pinMode(signal4led3,OUTPUT);

 pinMode(signel1sensor1,INPUT);
 pinMode(signel1sensor2,INPUT);
 pinMode(signel2sensor1,INPUT);
 pinMode(signel2sensor2,INPUT);

 pinMode(signel3sensor1,INPUT);
 pinMode(signel3sensor2,INPUT);
 pinMode(signel4sensor1,INPUT);
 pinMode(signel4sensor2,INPUT);

}

void loop() {
  int sensor1=analogRead(A0);
int sensor2=analogRead(A1);
Serial.println(sensor1);
Serial.println(sensor2);
  if(analogRead(A0)<400 && analogRead(A1)<400){
    signal1(20000);
  }
  if(analogRead(A0)<400){
    signal1(10000);
    }
   else
    {
      signal1(5000);
    }




    int sensor3=analogRead(A2);
    Serial.println(sensor3);
 int sensor4=analogRead(A3);
    Serial.println(sensor4);

  if(analogRead(A2)<400 && analogRead(A3<400)){
    signal2(20000);
  }
  if(analogRead(A2)<400){
    signal2(10000);
  }
    else
    {
      signal2(5000);
    }
```

```
      int sensor5=analogRead(A4);
      Serial.println(sensor5);
       int sensor6=analogRead(A5);
      Serial.println(sensor6);

if(analogRead(A4)<400 && analogRead(A5)<400){
      signal3(20000);
}
if(analogRead(A6)<400){
  signal3(10000);
  }
      else
 {
        signal3(5000);
      }

      int sensor7=analogRead(A6);
      Serial.println(sensor7);
      int sensor8=analogRead(A7);
      Serial.println(sensor8);



      if(analogRead(A6)<400 && analogRead(A7)<400){
      signal4(20000);
      }
      if(analogRead(A4)<400){
      signal4(10000);
      }
      else
      {
      signal4(5000);     }


     }
```

```
void signal1(int a){

  digitalWrite (signal1led3,HIGH);
  digitalWrite(signal1led1,LOW);
  digitalWrite(signal1led2,LOW);
  digitalWrite(signal2led1,HIGH);
  digitalWrite(signal2led2,LOW);
  digitalWrite(signal2led3,LOW);
  digitalWrite(signal3led1,HIGH);
  digitalWrite(signal3led2,LOW);
  digitalWrite(signal2led3,LOW);
   digitalWrite(signal4led1,HIGH);
  digitalWrite(signal4led2,LOW);
  digitalWrite(signal4led3,LOW);
  delay(a);
  digitalWrite(signal1led2,HIGH);
  digitalWrite(signal1led3, HIGH);
  digitalWrite(signal2led1,HIGH);
  digitalWrite(signal2led2,HIGH);
  digitalWrite(signal2led3,LOW);
  digitalWrite(signal1led1,LOW);
    digitalWrite(signal3led2,LOW);
  digitalWrite(signal3led3,LOW);
  digitalWrite(signal3led1,HIGH);
   digitalWrite(signal4led1,HIGH);
  digitalWrite(signal4led2,LOW);
  digitalWrite(signal4led3,LOW);
  delay(3000);
}
void signal2(int a){
   digitalWrite (signal2led3,HIGH);
   digitalWrite(signal2led1,LOW);
   digitalWrite(signal1led1,HIGH);
   digitalWrite(signal1led2,LOW);
```

```
     digitalWrite(signal2led2,LOW);
     digitalWrite(signal1led3,LOW);
       digitalWrite(signal3led1,HIGH);
    digitalWrite(signal3led3,LOW);
    digitalWrite(signal3led2,LOW);
     digitalWrite(signal4led1,HIGH);
    digitalWrite(signal4led2,LOW);
    digitalWrite(signal4led3,LOW);
     delay(a);
      digitalWrite (signal2led3,HIGH);
     digitalWrite(signal2led1,LOW);
     digitalWrite(signal1led1,HIGH);
     digitalWrite(signal1led2,LOW);
     digitalWrite(signal2led2,HIGH);
     digitalWrite(signal1led3,LOW);
     digitalWrite(signal3led1,HIGH);
    digitalWrite(signal3led3,LOW);
    digitalWrite(signal3led2,HIGH);
     digitalWrite(signal4led1,HIGH);
    digitalWrite(signal4led2,LOW);
    digitalWrite(signal4led3,LOW);
    delay(3000);
  }
void signal3(int a)
{
  digitalWrite (signal2led3,LOW);
     digitalWrite(signal2led1,HIGH);
     digitalWrite(signal1led1,HIGH);
     digitalWrite(signal1led2,LOW);
     digitalWrite(signal2led2,LOW);
     digitalWrite(signal1led3,LOW);
     digitalWrite(signal3led1,LOW);
    digitalWrite(signal3led3,HIGH);
    digitalWrite(signal3led2,LOW);
```

```
     digitalWrite(signal4led1,HIGH);
    digitalWrite(signal4led2,LOW);
    digitalWrite(signal4led3,LOW);
    delay(a);
      digitalWrite (signal2led3,LOW);
     digitalWrite(signal2led1,HIGH);
     digitalWrite(signal1led1,HIGH);
     digitalWrite(signal1led2,LOW);
     digitalWrite(signal2led2,LOW);
     digitalWrite(signal1led3,LOW);
     digitalWrite(signal3led1,LOW);
    digitalWrite(signal3led3,HIGH);
    digitalWrite(signal3led2,HIGH);
     digitalWrite(signal4led1,HIGH);
    digitalWrite(signal4led2,HIGH);
    digitalWrite(signal4led3,LOW);
    delay(3000);
}
void signal4(int a){
        digitalWrite (signal2led3,LOW);
     digitalWrite(signal2led1,HIGH);
     digitalWrite(signal1led1,HIGH);
     digitalWrite(signal1led2,LOW);
     digitalWrite(signal2led2,LOW);
     digitalWrite(signal1led3,LOW);
     digitalWrite(signal3led1,HIGH);
    digitalWrite(signal3led3,LOW);
    digitalWrite(signal3led2,LOW);
     digitalWrite(signal4led1,LOW);
    digitalWrite(signal4led2,LOW);
    digitalWrite(signal4led3,HIGH);
    delay(a);

      digitalWrite (signal2led3,LOW);
     digitalWrite(signal2led1,HIGH);
     digitalWrite(signal1led1,HIGH);
     digitalWrite(signal1led2,HIGH);
     digitalWrite(signal2led2,LOW);
     digitalWrite(signal1led3,LOW);
     digitalWrite(signal3led1,HIGH);
    digitalWrite(signal3led3,LOW);
    digitalWrite(signal3led2,LOW);
     digitalWrite(signal4led1,LOW);
    digitalWrite(signal4led2,HIGH);
    digitalWrite(signal4led3,HIGH);
    delay(3000);
}
```

CHAPTER 5

USER MANUAL

5.1 How to access application

To access smarty city application simply install application in your mobile and open it.

5.1.1 Login and Registration

After accessing Smart City Application, Login screen will display.

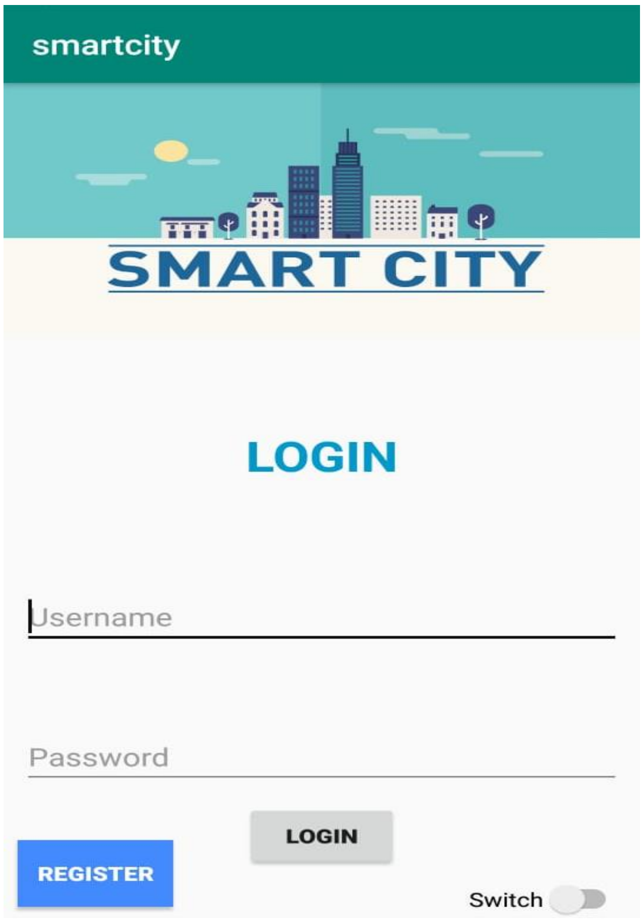


Figure 32: Android Login Diagram

- Enter login credentials if you are already registered at Smart city click on Login button

- Click on ‘Register to register at smart city. After click on ‘Register’ registration form will show as in below figure

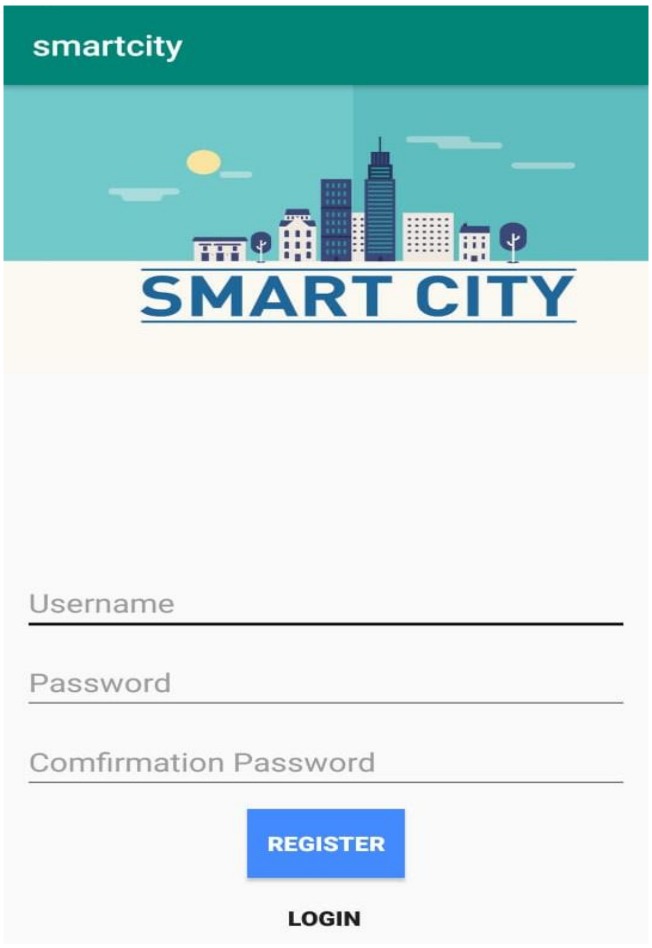


Figure 33: Android Registration Diagram

- Fill the registration form by providing user name and a new password then click on ‘Register button it will redirect you to Login.
- By click on ‘Login’ button, you will go back to Login screen

5.2 Home Screen

- After login in application a home screen will all modules will be shown.

- From this you can access any module which you want to access.

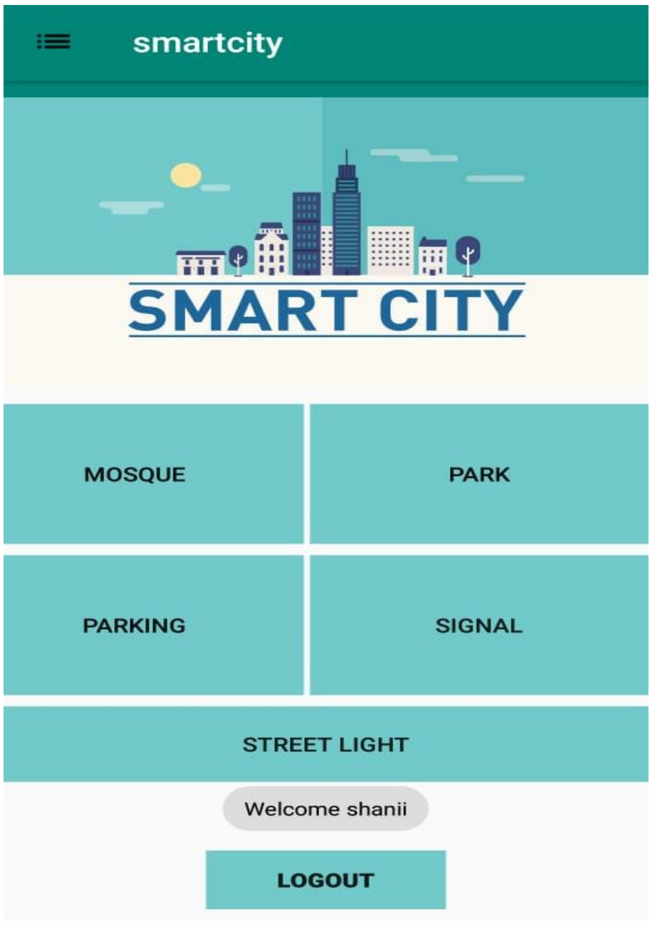
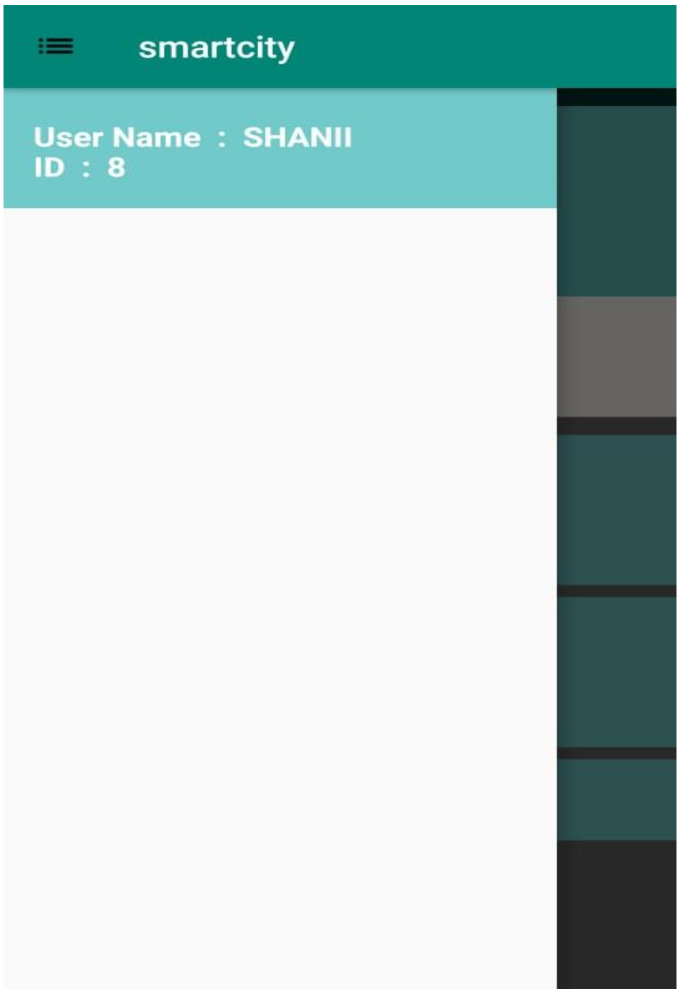


Figure 34: Android Home Screen Diagram

- You can also check your login details from top left column.



User Name : SHANII
ID : 8

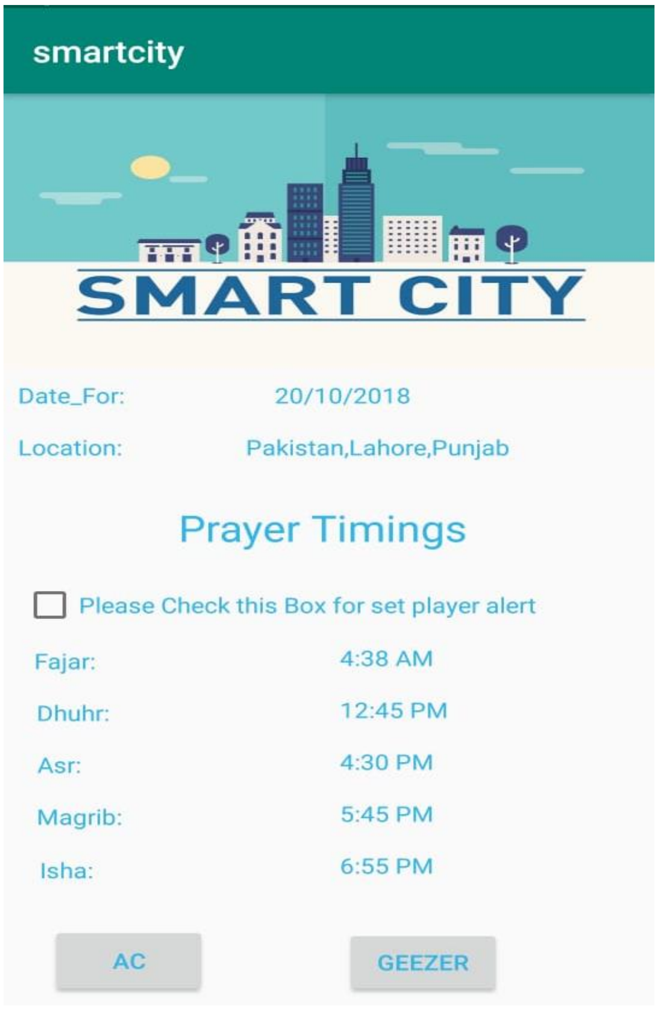
Figure 35: Android Login details Diagram

5.3 Module Description Manual

Here is the manual of every module separately

5.3.1 Mosque Module

- On clicking Mosque Module button you will enter mosque module.
- From there you can check prayer time, prayer alerts.
- You can also check temperature of Mosque.
- You can also check weather ac is on or geezer.


smartcity

SMART CITY

Date_For: 20/10/2018

Location: Pakistan,Lahore,Punjab

Prayer Timings

☐ Please Check this Box for set player alert

Fajar:	4:38 AM
Dhuhr:	12:45 PM
Asr:	4:30 PM
Magrib:	5:45 PM
Isha:	6:55 PM

Figure 36: Android Mosque Module Diagram

5.3.2 Park Watering Module

1. In this module you can check park watering module status of the smart city system.

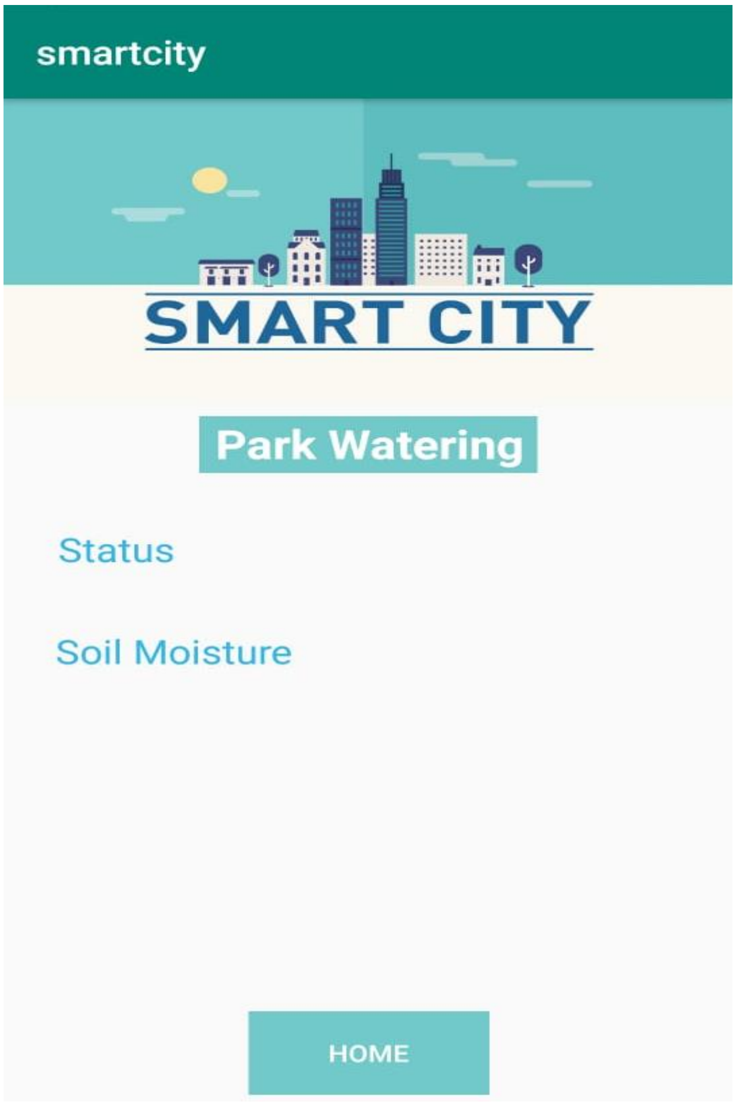


Figure 37: Android park Module Diagram

5.3.3 Parking Module

- In this Module you can check the status of the parking.

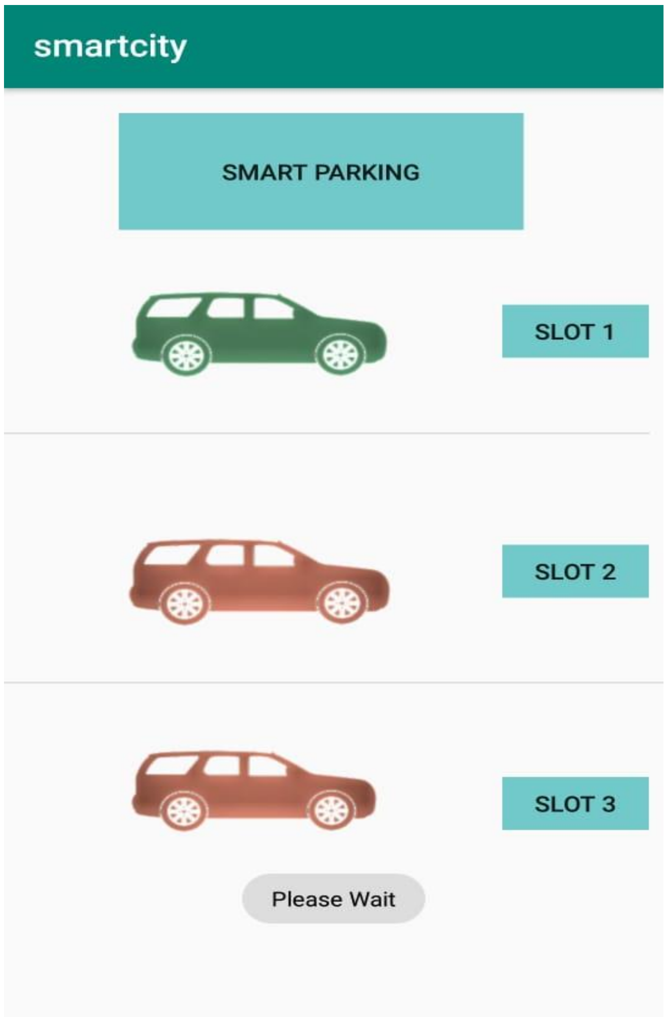


Figure 38: Android parking Module Diagram

5.3.4 Traffic Signal Module

- In this module you can check the status of all signals as shown in the pic below.
- The status of all traffic lights is given in this module.

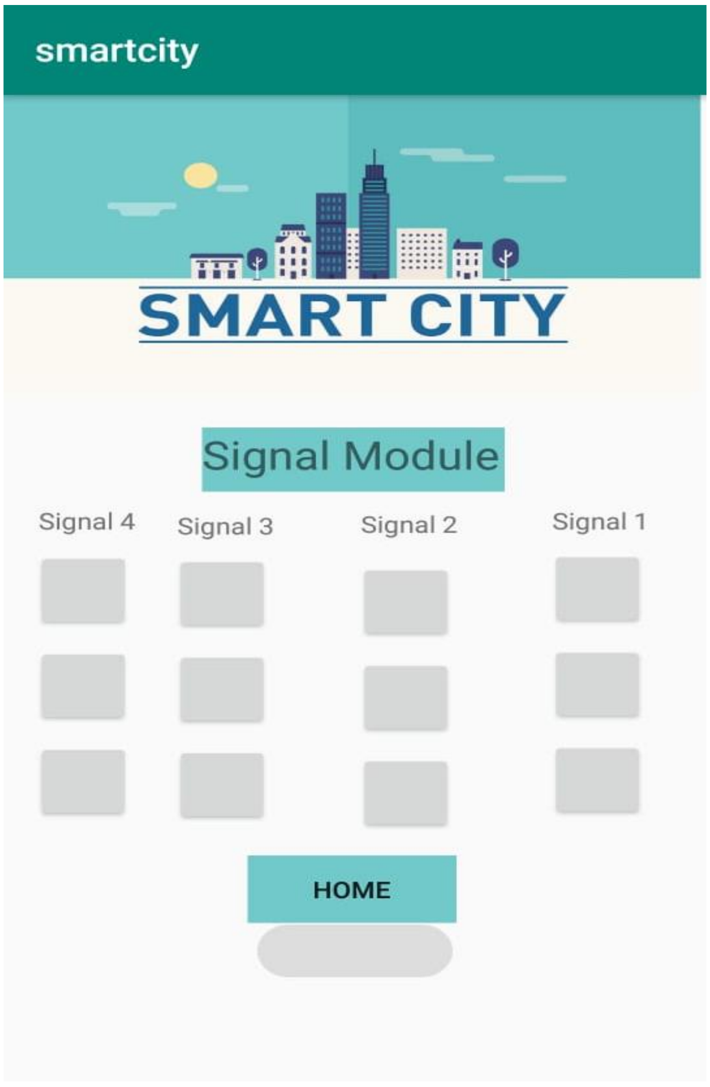


Figure 39: Android Traffic Module Diagram

5.3.5 Street light Module

- In this Module you can check the status of street lights.
- Weather streetlight is on or off.

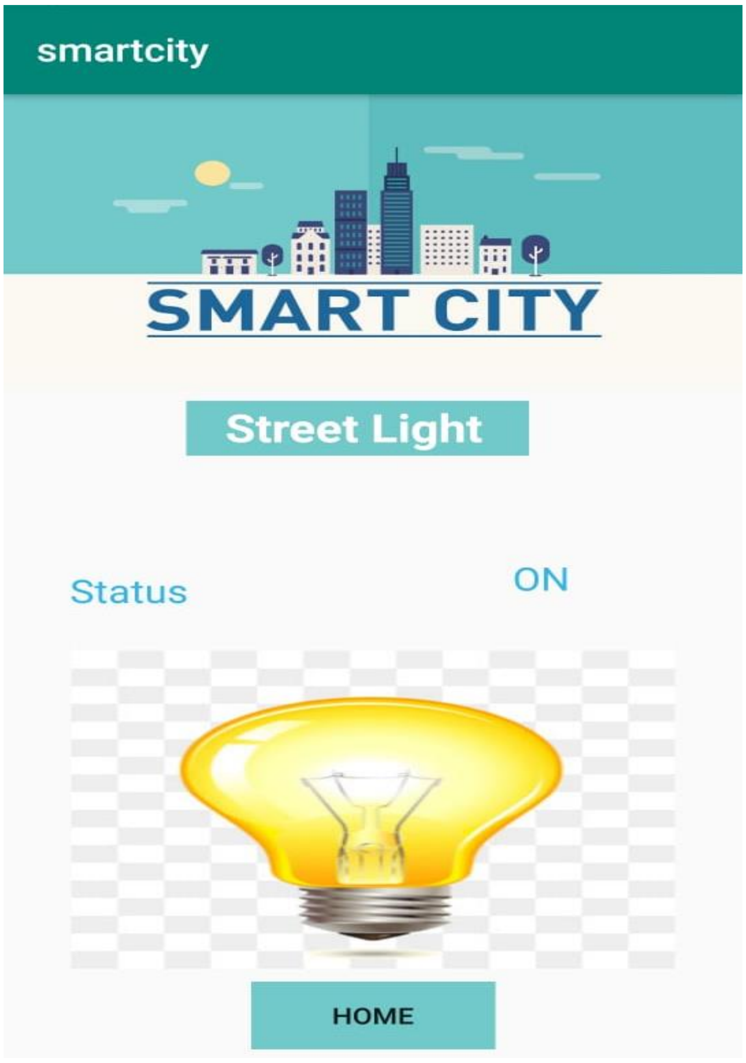


Figure 40: Android Street light Module Diagram

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

In past everything thing is controlled manually it very difficult to control this. The aim of this project is to makes a city smart by automating it. And providing a single platform to control all this and helps the user to check the status of all modules from a single application. This will reduce man power. It giving azan alert, showing status of different modules. It also shows the parking availability to the user. In short it automation of city. And controlling it from single platform. This project is vast therefore we just covered five modules in this project currently.

6.2 Recommendation

It's necessary for user to have strong internet connection otherwise application may be lack or works slowly. User should have little bit know how about the hardware.

REFERENCES

[1]https://www.researchgate.net/publication/236685572_Smart_Cities_Literature_Review_and_Analysis

[2] http://www.mavensystems.com/smart-city.html

[3] http://iot-smartcities.lero.ie/wp-content/uploads/2016/12/A-Systems-Approach-to-Smart-City-Infrastructure-A-Small-City-Perspective.pdf

[4] https://pdfs.semanticscholar.org/f995/13aa551690d639f1480702ed953bd3756685.pdf

[5] https://www.smartparking.com/

[6] http://smartparkingsystems.com/en/

[7] https://cityos.io/Best-Smart-Parking-Systems

[8]https://www.siemens.com/global/en/home/products/mobility/road-solutions/parking-solutions/intelligent-parking-solutions.html

[9] http://smarthomebus.com/dealers/Presentations/Smart%20Mosque.pdf