TABLE OF CONTENTS

INTRODUCTION

1.1 PROJECT DETAILS

A major problem in day to day life is parking of vehicles especially the car parking at an appropriate place. And this issue indirectly leads to traffic congestion. This project presents the basic concept of using server or cloud based smart parking services in smart cities as an important application of the Internet of Things (IoT) paradigm. This system will be accessible through a mobile app or through the webpage provided and can be used to monitor or find the empty slots in that area.

1.2 PURPOSE

Moving towards smart city application, smart parking is a good example for a common citizen of how the Internet-of-Things (IoT) will be effectively and efficiently used in our daily living environments to provide different services to different users. Any citizen may use his mobile device, a computer having Internet to access the smart city application from anywhere in the world to find a free parking spot in the city and get to know the which parking spot is still available. The main purpose of this application is to reduce the on road traffic and fuel consumption and make travelling eco-friendly and social. This entire process is made easier by the means of an application which people can use from their smartphones.

1.3 SCOPE

- The data can be retrived from the server using WiFi module into another arduino and then onto LCD display or app.
- In future we will continue to add new features and make the system more user-friendly.

1.4 OBJECTIVE

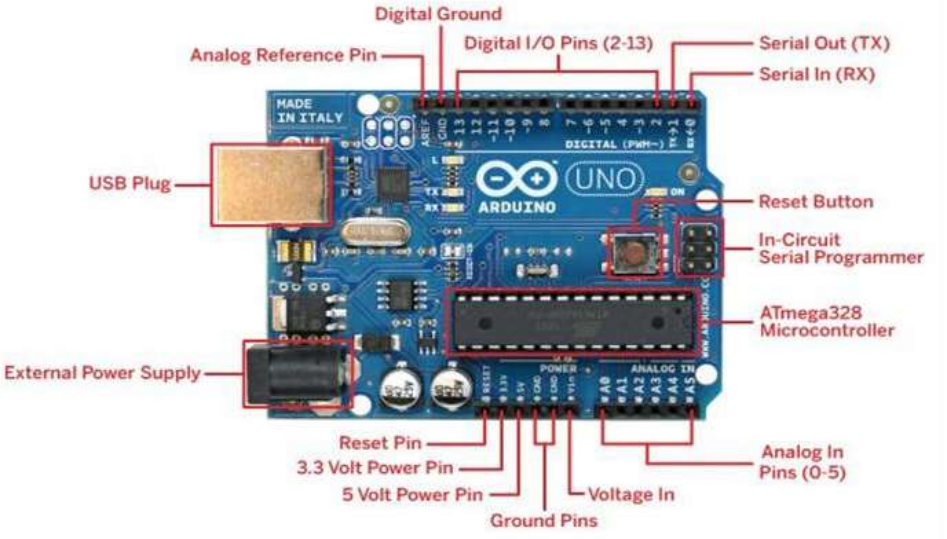
- Saves commuter's time in searching for parking slots.
- More no. of cars can be parked in minimum space.

1.5 TECHNOLOGY AND LITERATURE REVIEW

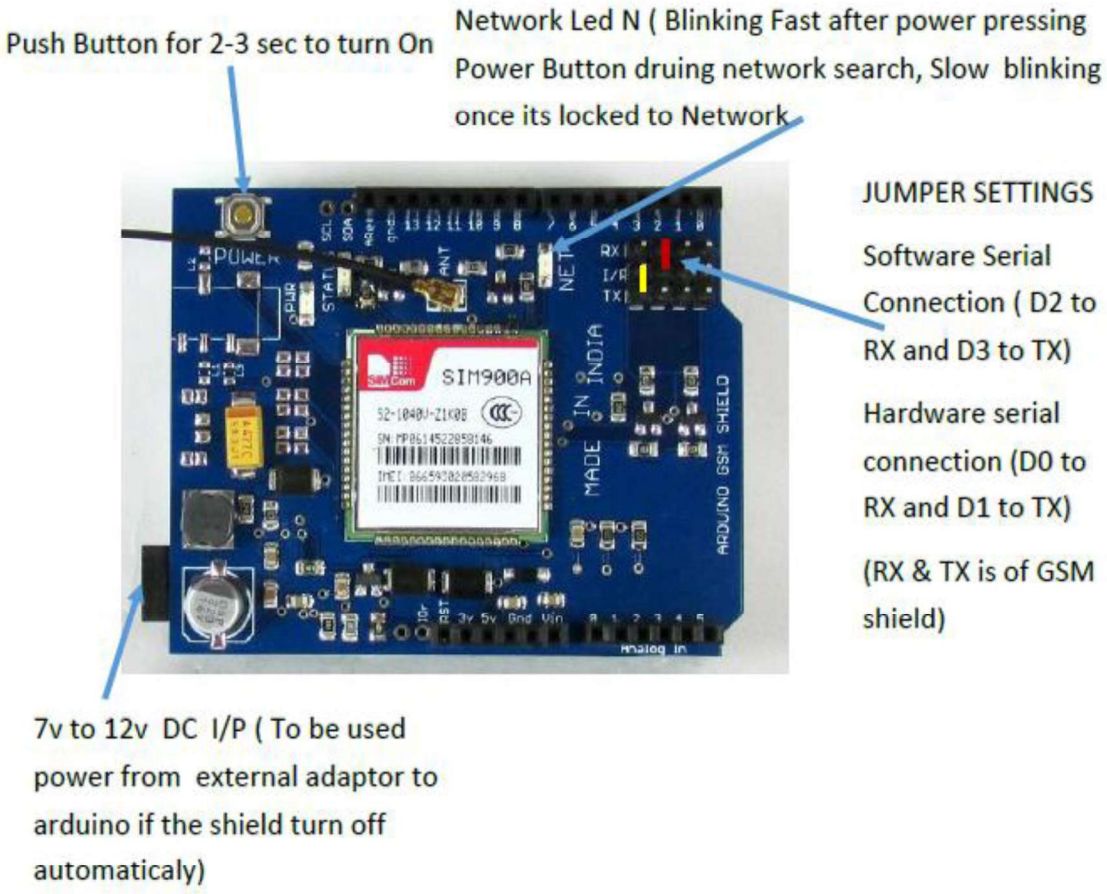
ANDROID: Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics. As of 2015, Android has the largest installed base of all operating systems.

ANDROID STUDIO: Android Studio is an integrated development environment (IDE) for developing for the Android platform. It was announced on May 16, 2013 at the Google I/O conference by Google's Product Manager, Katherine Chou. Android Studio is freely available under the Apache License 2.0
Android Studio was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014.[4] The first stable build was released in December 2014, starting from version 1.0
Based on JetBrains' IntelliJ IDEA software, Android Studio is designed specifically for Android development.[6] It is available for download on Windows, Mac OS X and Linux,[7][8] and replaced Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development.

ARDUINO: Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing. Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments.



GSM Module: Global System for Mobile communication (GSM) is used for mobile communication in many countries. It features sending and reading short messages (SMS), the management on phonebook of SIM card, sending SMS to group users, moreover, a flexible solution on real-time reading SMS is also proposed. There are different kinds of GSM modules available in market. We are using most popular module based on Arduino Uno **&** Sim-Com SIM900**A** for this tutorial. GSM module contains SIM card holder, antenna for receiving and sending signals to the SIM, and RS232 based serial port for connection. AT commands are used to use a GSM modem.



2) FEASIBILITY STUDY

2.1 FEASIBILITY STUDY REPORT

The feasibility of software and hardware can be tested in four dimensions: **Technology**- is the project technically feasible? As in our case, smart car parking application we have a few existing examples of the same, so no technical infeasibility are there. **Finance**- Is it financially feasible? Does it have too much cost of development. **Time**- will it take too much time to complete? We have planned each phases and it seems to be in controlled and within time so no extra time cost will be added. **Resources**- Do we have sufficient resource to succeed?

There are four categories of feasibility tests: operational feasibility, technical feasibility, schedule feasibility and economic feasibility.

2.1.1 Operational Feasibility:

A major problem in day to day life is parking of vehicles especially the car parking at an appropriate place. And this issue indirectly leads to traffic congestion. Finding parking slots is a major concern. With this project the problem is eliminated in one go.

2.1.2 Technical feasibility:

With the availability of online and offline resources such as online hosting websites which provides facility of online database and offline resources such as IDEs and Emulators, it is needless to say that the implementation is technically feasible.

2.1.3 Scheduling feasibility:

I used a very simple approach for scheduling feasibility by dividing the work in 3 phases and allotted time to each phase accordingly keeping in mind the deadline. For the first phase, the backend was allotted a time of two weeks, second phase, the front end required 4 weeks for designing, developing and deploying, the last phase however took only a week for testing and debugging.

2.1.4 Implementation feasibility:

With the latest IDE's and Emulators, the implementation was not an issue at all. The application was tested extensively on the various emulators and handheld smartphones. The emulators used were: Android Virtual Device, GenyMotion and Windroye. The handheld devices ranged from Sony Xperia smartphone to Google Nexus 5 smartphone. Hence the application is compatible with vast range of devices..

2.2 PROJECT PLANNING

2.2.1 Project development approach and justification

The project idea was clicked by the requirement of a system to curb the parking problems created by commuters which increased traffic and fuel consumption during journeys. The solution was to create an application which gives each commuter an option to travel together with other commuters to reduce the number on vehicles wrongly parked on road and parking area, save cost and fuel consequently. Android platform is most widely used and hence it was the best option to create an application for the Android platform to maximize number of users.

2.2.2 Project Plan

The plan composed to make application for commuters to get details of other commuters so as to initiate car-pooling.

2.2.3 Milestones and Deliverables

Management needs information. As software is intangible, this information can only be provided as documents that describe that state of the software being developed. Without this information, it is impossible to judge progress and cost estimates and schedules cannot be updated. When planning a project series of milestones are established.

Milestone:

- Milestone is an end-point of the software process activity.
- At each milestone there should be formal output, such as report, that can be represented to the management. The weekly report is submitted to project guide, which include day to day work report.
- Milestone represents the end of the distinct, logical stage in the project.

Deliverables:

- Deliverables is a project report that is delivered to the administrator of the project.

- Deliverables are delivered to the administrators of our organization at the end of the some major project phase such as specification, design, etc.
- Deliverables are usually milestones

- Milestones may be internal project results that are used by the project manager to check progress but which are not delivered to the administrator.

2.2.4 Roles & Responsibilities

Name	Role				
	Analysis	Designing	Coding	Testing	Documentation
Bharvee Acharya	✓	✓	✓	✓	✓
Aditi Desai	✓	✓	✓	✓	✓

3) SYSTEM REQUIRMENT STUDY

3.1 STUDY OF CURRENT SYSTEM

The current system provides output on app as per input.
The current system can automatically change output as per environmental conditional changes.
The current system requires phpMyadmin database.
The current system uses GSM module for data exchange.

3.2 PROBLEMS AND WEAKNESSES OF CURRENT SYSTEM

The current system displays data bit late.
The current system is only for a single board unit. That means the object sensor and Arduino board should be on a single breadboard connected in single circuit.

3.3 USER CHARACTERISTICS

User can check parking slots from anywhere.
The application is easy to use.

3.4 HARDWARE AND SOFTWARE REQUIREMENTS

3.4.1 Hardware Requirements

- Arduino-Uno
- IR object sensor
- GSM module
- Android Smartphone with OS 6.0+ (can use emulator)

5V Voltage Source (can be connected using USB)

3.4.2 Functional Requirements

Description:-We will use an android app for checking the presence of car on the parking slot.

State-The Android Application Main Page showing empty parking slots.

Input-Positioning a car on parking slot.

Output- Occupied parking slot and empty slot with different colors is marked on app

Processing-When cars are sensed on the slots using IR object sensors,the data will be sent over from GSM module using Arduino and will be received by the cloud.The data will therefore be fetched on android application.

3.4.3 Non-Functional Requirements

Interface Issue
The user-interface is designed in a way that any user can use the application comfortably and with ease, after all, aesthetic appeal is what keeps the users glued to the application.

Accuracy
It was taken great care of when creating the application, that the requirements were met precisely and no loophole in the system had gone unnoticed.

3.5 Constraints

3.5.1 Regulatory Policy

As per the Company's policy any developer has to maintain the Coding Standards. Also each and every user should maintain the subversion and commit the modification with appropriate comment so to have track of work and also of the code modification.

From the client's perspective:

Developer should use well known technology.
Developer should use well coding standard.

3.5.2 Hardware Limitations

The hardware limitation is very low. The system has been designed according to current android versions availability which client is using

From the developer's perspective:

It is assumed that client is having smart phone.
Client is having required system configuration to run the app.

3.6 ASSUMPTIONS AND DEPENDENCIES

Android Marshmallow 6.0+ be present on user's mobile phones.
5V Voltage Source is present.

4) SYSTEM ANALYSIS

4.1 Requirements of New System

4.1.1 User Requirements

Availability across all android devices.
Ability to control fan speed manually.

4.1.2 System Requirements

4.1.2.1 Hardware Requirements

1. Arduino Uno
2. GSM module
3. Android (version : 06+)

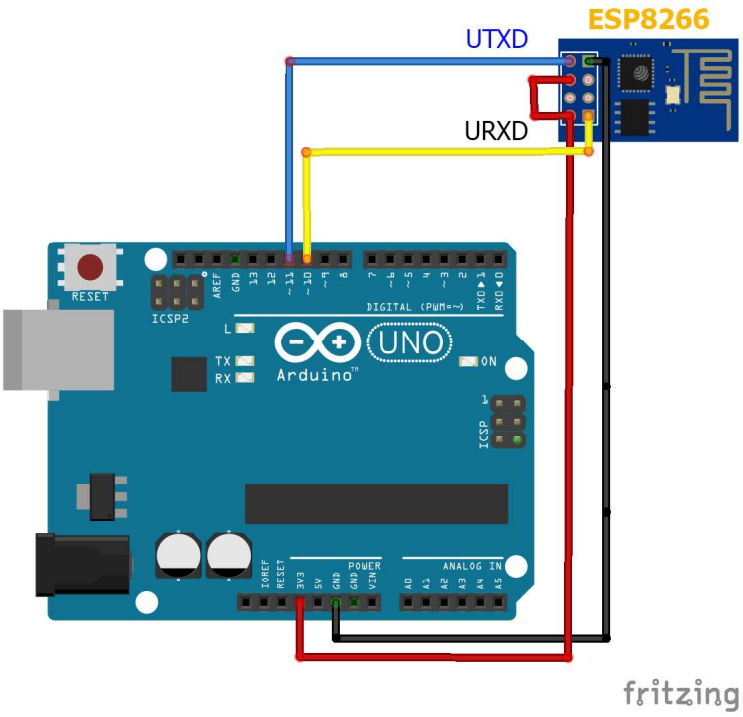
4.1.2.2 Functional Requirements

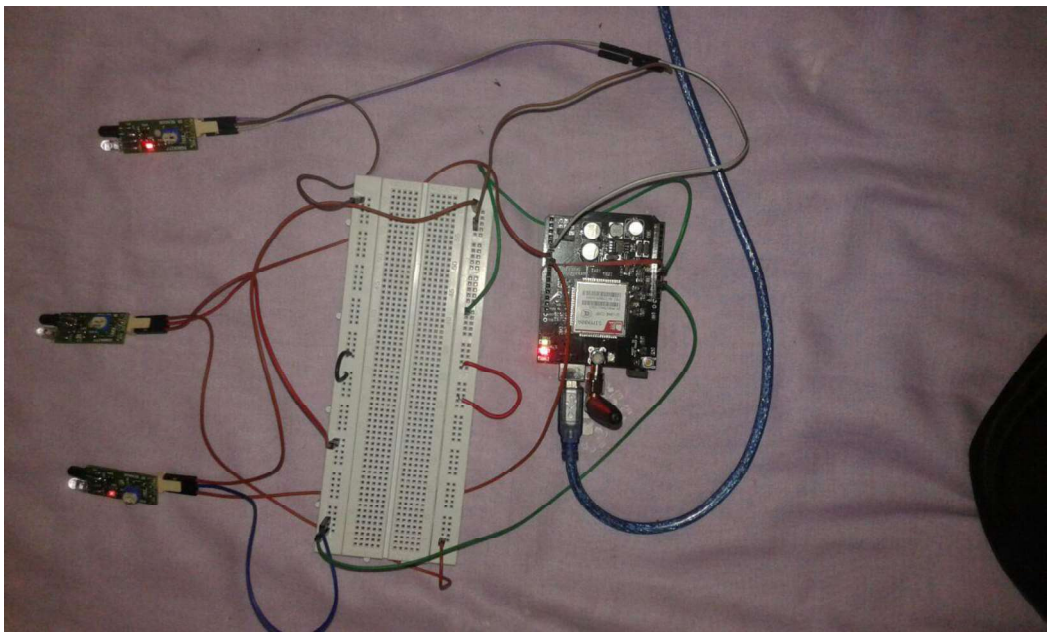
Voltage Source	:	5V Battery or USB.
Hard Di	:	5MB
Screen	:	Touch Screen

4.2 Features of New System

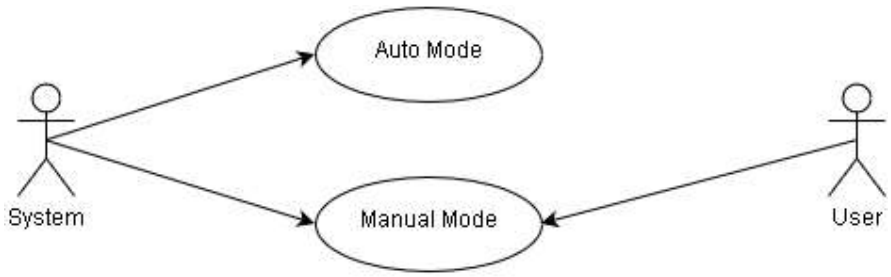
Portability
Improved User Interface

4.3 Circuit Diagram



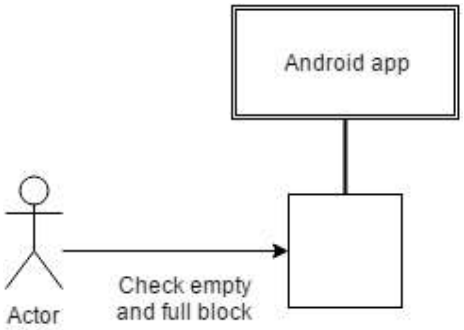


4.4 Usecase Diagram

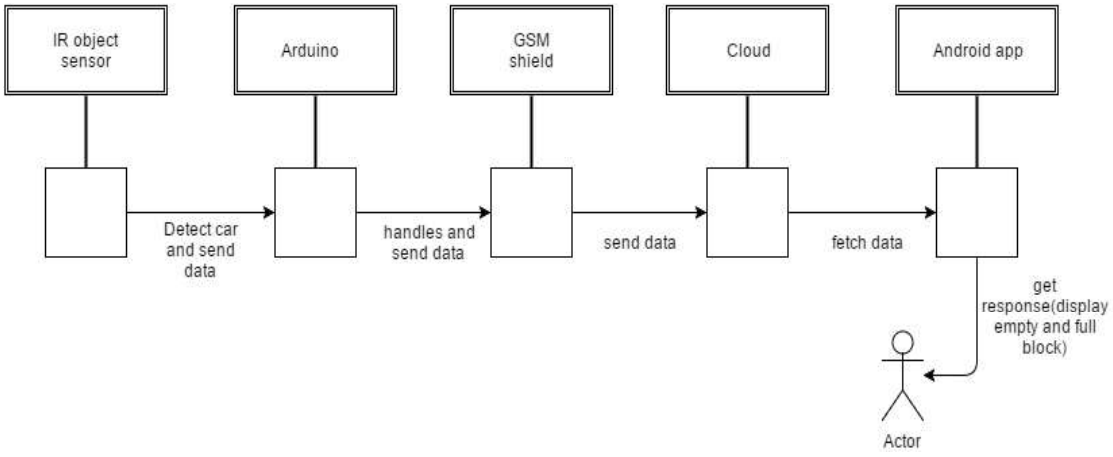


4.5 Sequence Diagram

- **Manual Mode**



- **Auto Mode**



5) IMPLEMENTATION AND PLANNING

5.1 IMPLEMENTATION ENVIRONMENT

- **With GUI**

They are relatively easy to learn and use. User with no computing experience can learn to use the interface brief training session.

The user uses multiple screens for system interaction. Switching from one task to another is possible without losing sight of information generated during the first tasks.

Fast access of data from the device
Good and well structured User Interface

5.2 CODING STANDARDS

Coding Standards contribute to an improved comprehension of source code. Perhaps one of the most influential aids to understanding the logical flow of an application is how the various elements of the application are named. A name should tell "what" rather than "how." By avoiding names that expose the underlying implementation, which can change, you preserve a layer of abstraction that simplifies the complexity.

Naming Conventions make programs more understandable by making them easier to read. They can also give information about the function of the identifier – for example, whether it's a constant, class, etc. which can be helpful in understanding the code.

Reasons for using the coding standards are

- Uniform distribution
- Sound understanding
- Encourages Good programming skills.

- ✓ All code -should be well commented. All procedures and functions should begin with a comment to explain what the function/procedure performs.
- Good and meaningful comments make code more maintainable.
- Do not write comments for every line of code and every variable defined.

Write comments wherever required. But good readable code will require very less comments. If all the variables and methods names are meaningful, that would make the code very readable and will not need more comments.

6) TESTING

6.1 TESTING STRATEGY

Once source code has been generated, software must be tested to uncover as many errors as possible before delivery to customer. Your goal is to design a series of test cases that have a high likelihood of finding errors. Software testing techniques provide systematic guidance for designing tests that (1) exercise the internal logic of software components, and
(2) exercise the inputs and outputs domains of the program to uncover errors in program function, behavior and performance.

During early stages of testing, a software engineer performs all tests. However, as the testing process progresses, testing specialists may become involved. Reviews and other activities can and do uncover errors, but they are not sufficient. Every time the program is executed, the customer tests it! Therefore, you have to execute the program before it gets to the customer with the specific intent of finding and removing all errors. In order to find the highest possible number of errors, tests must be conducted systematically and test cases must be designed using disciplined techniques.

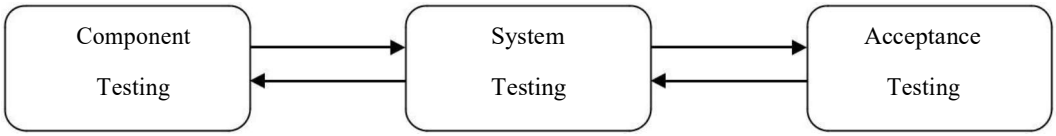


Fig 6.1 The Testing Process

Testing Objective

- Testing is a process of executing a program with intent of finding an error.
- A good test case is one that has a high probability of finding an as-yet undiscovered error.
- A successful test is one that uncover as as-yet undiscovered error.

6.1.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually scrutinized for proper operation. Unit testing is often automated but it can also be done manually. This testing mode is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

Unit testing involves only those characteristics that are vital to the performance of the unit under test. This encourages developer to modify the source code without immediate concerns about how such changes might affect the functioning of the units or the program as a whole. Once of whole of the units in a program have been found to be working in the most efficient and error free manner

possible, larger components of the program can be evaluated by means of integration testing.

I tested each single part of the all website, both on the admin side, client side and guest side; I tested each and every module individually. On admin side tested modules like Add new mobile, view mobile, manage key feature, Add retailer, View retailer and others. In Forum Admin can add, edit and delete mobile, mobile Category and mobile Name and for Products the admin can add, update and delete mobile. On the front side I tested modules like Admin panel, Client panel, Guest panel. Similarly for every module I have done Unit testing while coding and before submitting a demo. So, most of the errors have been removed from the website.

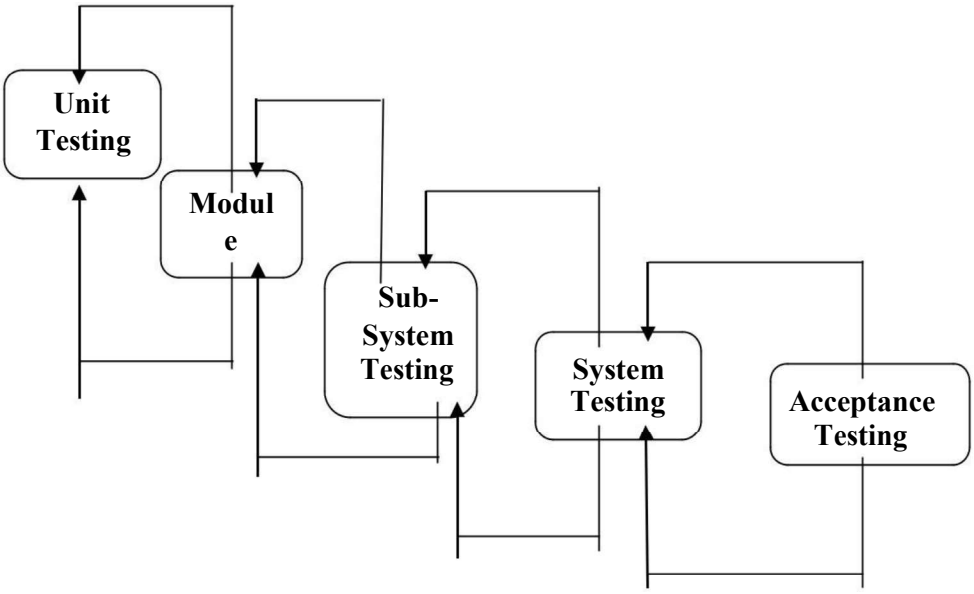


Fig 6.2 Types of Testing

6.1.2 Sub System Testing

After testing each Unit we move on to larger units called sub system. In Mobile studios website, there was admin side, client side and guest side so after unit testing of each module on both sides. In subsystem testing I tested the whole admin side as one system and them the front side as one whole system. On the admin side all the modules like Add mobile, view mobile and Add retailer etc were tested together to see that there was any error or bug found. On the front side all the modules like Admin panel, Client panel, and Guest panel were tested together.

I developed each sub-system such as admin side and front side individually, so tested also at the development time. These sub-systems works fine alone, but after integrating within the websites I found some errors, that is why we done integrating testing after integrating these Sub-system.

6.1.3 System Testing

After testing all the sub-system it is time to test the whole system. System testing of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. While testing the whole system I found many errors like mobile automatically added in a cart when we press F5 & refresh page or this problem was happened because we have wrote the code on page lode event that's why this problem was created to solve this problem we have changed the code for the cart. I worked on each error and exception that I got while testing and most of them are removed or made such correction that it will not happen again. Exception also arises when there was no access available to the database and the website was not able to read the data from database. Such types of modification are done by me to take system reliable and error free.

- Recovery Testing: It is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed.

- Security Testing: It attempts to verify that protection mechanisms build into a system will, in fact, protect it from improper penetration.

- Performance Testing: It is designed to test the run-time performance of software within the context of an integrated system performance testing occurs throughout all step in the testing process.

6.1.4 Acceptance Testing

Acceptance testing can be connected by the end user, customer, or client to validate whether or not to accept the product. Acceptance testing may be performed as part of the hand-off process between any two phases of development. The acceptance test suite is run again the supplied input data or using an acceptance test script to direct the tester. Then the results obtained are compared with the expected results. If there is a correct match for every case, the test suite is said to pass.

I had provided demo to our client at the regular interval of time with my project Manager. So they have complete site of the whole project from the initial stages whatever changes we made in between demo interval, were also been informed to client regularly, so they don't get surprised by seeing new functionality.

6.2 TESTING METHODS

The verification activities fall into the category of static testing. During static testing, you have a checklist to check whether the work you are doing is going as per the set standards of the organization. These standards can be for coding, integrating and deployment. Reviews,

Inspection's and Walkthroughs are static testing methodologist. Dynamic testing involves working with the software giving input values and checking if the output Is as expected. These are the validation activities. Unit test, integration test. System and acceptance tests are few of the dynamic testing methodologies.

Alpha & beta testing: the alpha test is conducted at the developer's site by a customer. The software is used in a natural setting with the developer "looking over shoulder" of the user and recording errors and usage problems. Alpha test are conducted in a controlled environment. The beta testing is conducted at one or more customer site by the end-user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a "live" application of the software in an environment that cannot be controlled by the developer.

6.2.1 Black box testing

Also known as functional testing. A software testing techniques where by the internal working of the item being tested are not known by the tester. For example, in a black box test on software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs. The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specification.

- The advantages of this type of testing include:

1. The test is unbiased as the designer and the tester are independent of each other

2. The tester does not need knowledge of any specific programming languages

3. The test is done from the point of view of the user, not the designer

4. Test cases can be designed as soon as the specifications are complete

- The disadvantages of this type of testing include:

1. The test can be redundant if the software designer has already run a test case

2. The test cases are difficult to design

3. Testing every possible input stream is unrealistic because it would take an inordinate amount of time: there for, many program paths will go untested

6.2.2 White box testing

Also known as glass box, structural, clear box and open box testing. A software testing technique where by explicit knowledge of the internal workings of the item being tested are used to select the test data. Unlike black box testing, white box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do. He or she can than see if the program diverges from its intended goal.

6.2.3 Design of test Cases

To minimize the number of errors in software, a reach variety of test design methods have evolved for software. These methods provide the developer with a systematic approach to testing. More important, methods provide a mechanism that can help to ensure the completeness of test and provide the highest likelihood for uncovering errors in software.

An engineering product can be tested in one of the two ways: (1) knowing the specified function that product has been designed to perform, tests can be conducted has demonstrate each function is fully operational while at the same time searching for errors in each function: (2) knowing the internal workings of a product, tests can be conducted to ensure that "all gear mesh", that is, internal oppression are performed according to specifications and all internal components have been adequately exercised. Here are the test cases that we had made for our application.

6.3 TEST CASES

Test Case No: 01

Purpose:	Search for free parking slots in parking area
Input:	Park a car on parking slot
Expected Output:	Occupied slot will not be displayed with different colour than that of empty slot on relevant application

Test Case No: 02

Purpose:	Search for free parking slots in parking area
Input:	Remove a car from parking slot
Expected Output:	Free slot will be displayed with different colour than that of empty slot on relevant application

6)SOURCE CODE

- **Arduino & GSM Module Setup**

```
while (!Serial);
 Serial.begin(9600);
 Serial.println(F("FONA basic test"));
 Serial.println(F("Initializing....(May take 3 seconds)"))
 fonaSerial->begin(4800);
 if (! fona.begin(*fonaSerial)) {
  Serial.println(F("Couldn't find FONA"));
  while (1);
 }
 type = fona.type();
 Serial.println(F("FONA is OK"));

 Serial.print(F("Found "));
 switch (type) {
  case FONA800L:      Serial.println(F("FONA 800L"));
                      break;
  case FONA800H:     Serial.println(F("FONA 800H"));
                      break;
  case FONA808_V1:  Serial.println(F("FONA 808 (v1)"));
                      break;
  case FONA808_V2:  Serial.println(F("FONA 808 (v2)"));
                        break;
  case FONA3G_A:       Serial.println(F("FONA 3G (American)"));
                       break;
  case FONA3G_E:       Serial.println(F("FONA 3G (European)"));
                       break;
  default:             Serial.println(F("???"));
                       break;
 // Print module IMEI number.
 char imei[15] = {0}; // MUST use a 16 character buffer for IMEI!
 uint8_t imeiLen = fona.getIMEI(imei);
 if (imeiLen > 0) {
  Serial.print("Module IMEI: "); Serial.println(imei);
 }
 fona.setGPRSNetworkSettings(F("uninor"), F(""), F(""));
 fona.setHTTPSRedirect(true);
```

- **Turn GPRS On/Off**

```
        Int gprs(int s) {
        if( s == 0) {
       // turn GPRS off

    if (!fona.enableGPRS(false))
      { Serial.println(F("Failed to turn off"));
       return 0; }
    else { return 1; }
```

```
    }
    else if( s==1 ) {
  // turn GPRS on
  if (!fona.enableGPRS(true))
  {  Serial.println(F("Failed to turn on"));
    return 0;}
  else { Serial.println(F("**********turned on")); return 1; }
  }
    }
```

- **Read data using sensors**

```
    int p1;
    int p2;
    int p3;
    p1=digitalRead(s1);
   p2=digitalRead(s2);
    p3=digitalRead(s3);
```

- **Send data to Database**

```
      // read website URL
  uint16_t statuscode;
  int16_t length;
  flushSerial();
  char url[80];
  char dp[10];

strcpy(url,"http://api.pushingbox.com/pushingbox?devid=v3CA760F45ECBC
4D&temp=");
  itoa(T, dp, 10);
  strcat(url,dp);
  strcat(url,"&hu=");
  itoa(H, dp, 10);
  strcat(url,dp);
  strcat(url,"&mo=");
  itoa(M, dp, 10);
  strcat(url,dp);
  Serial.println(url);
  Serial.println(F("****"));
  if (!fona.HTTP_GET_start(url, &statuscode, (uint16_t *)&length)) {
              Serial.println("Failed!");
    return;
   // break;
   }
   while (length > 0) {
    while (fona.available()) {
     char c = fona.read();
     // Serial.write is too slow, we'll write directly to Serial register!
      #if defined(__AVR_ATmega328P__) ||
defined(__AVR_ATmega168__)
     loop_until_bit_is_set(UCSR0A, UDRE0); /* Wait until data register
empty. */
     UDR0 = c;
      #else
     Serial.write(c);
```

```
            #endif
            length--;
            if (! length) break;
          }
        }
        Serial.println(F("\n****"));
        fona.HTTP_GET_end();
```

- **Retrive data on app**

```
package com.example.bhargav.car_parcking;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

import com.android.volley.Cache;
import com.android.volley.Network;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.BasicNetwork;
import com.android.volley.toolbox.DiskBasedCache;
import com.android.volley.toolbox.HurlStack;
import com.android.volley.toolbox.StringRequest;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.concurrent.RunnableFuture;
import java.util.logging.Handler;

public class MainActivity extends AppCompatActivity {
   TextView parking1,parking2,parking3;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      parking1=(TextView)findViewById(R.id.p1);
      parking2=(TextView)findViewById(R.id.p2);
      parking3=(TextView)findViewById(R.id.p3);
      final String url="http://soilmoisture.hol.es/bhargav/carget.php";
      Cache cache = new DiskBasedCache(getCacheDir(), 1024 * 1024);
      Network network = new BasicNetwork(new HurlStack());
      final RequestQueue mRequestQueue=new RequestQueue(cache,network);
      mRequestQueue.start();
      final android.os.Handler handler=new android.os.Handler();
      Runnable updateParking=new Runnable() {
         @Override
         public void run() {
```

```
            StringRequest serviceRequest=new
StringRequest(Request.Method.GET, url, new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    JSONArray ja;
                    try {
                        ja=new JSONArray(response);
                        JSONObject parkingObject1=ja.getJSONObject(0);
                        JSONObject parkingObject2=ja.getJSONObject(1);
                        JSONObject parkingObject3=ja.getJSONObject(2);
                        int parkingValue1=parkingObject1.getInt("senser_data");
                        int parkingValue2=parkingObject2.getInt("senser_data");
                        int parkingValue3=parkingObject3.getInt("senser_data");
                        System.out.println("PARKING1:"+parkingValue1);
                        if(parkingValue1==0){
                            parking1.setBackgroundColor(Color.parseColor("#04e461"));
                        }else if(parkingValue1==1){
                            parking1.setBackgroundColor(Color.parseColor("#ff0505"));
                        }
                        if(parkingValue2==0){
                            parking2.setBackgroundColor(Color.parseColor("#04e461"));
                        }else if(parkingValue2==1){
                            parking2.setBackgroundColor(Color.parseColor("#ff0505"));
                        }
                        if(parkingValue3==0){
                            parking3.setBackgroundColor(Color.parseColor("#04e461"));
                        }else if(parkingValue3==1){
                            parking3.setBackgroundColor(Color.parseColor("#ff0505"));
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            },
                    new Response.ErrorListener() {
                        @Override
                        public void onErrorResponse(VolleyError error) {

                        }
                    }
            );
            mRequestQueue.add(serviceRequest);

            //Change Here If You Want Currently It Refresh Every 2 Sec
            handler.postDelayed(this,2000);
        }
    };
    //Change Here If You Want Currently It Refresh Every 2 Sec
    handler.postDelayed(updateParking,2000);
  }
}
```

8.)Output

9) LIMITATION AND ENHANCEMENT

9.1 Limitation

Need good network connection on both sides on arduino side and on app side to get the result updated.

9.2 Future Enhancement

This project can be easily converted to large scale by adding similar hardware parts and programming for the same.

10) Conclusion

The smart parking system based on IoT concept has been implemented using various sensor circuitry and cloud (server). It is an efficient system for car parking which prevails traffic congestion.
Using resources like Arduino IDE ,Arduino UNO, PhpMyadmin database and Wifi Module which provides an online cloud one can easily make an android app which monitors the parking area .
We chose Smart Car Parking System using Arduino and android because concepts of IoT are helping people a lot. Here we can easily check free parking slots with our object sensors and other modules using internet and android app with concepts of IoT, which helps people and other professional orgaization.

11) Bibliography

11.1 Websites Reffered

www.google.com
www.developer.android.com
www.javapoint.com
www.tutorialpoint.com
www.esp8266.com
www.youtube.com
www.draw.io