Smart Parking System

Author: M. m. nkosi

supervisor: Paul Tarwireyi

University of zululand

version 1

2019

**Declaration**

This is to certify that this final year project report ***‘Smart Parking System’*** is submitted by the authors for the purpose of obtaining a degree in Bachelors of Science in Computer Science. I hereby declare all the work presented in this final project paper are authentic and any inspiration of the work have been accredited with proper referencing.

Signature of Supervisor Signature of Authors

**Contents**

[Section 1 - Project Description](#_Toc24384669)

[1.1 Project](#_Toc24384670)

[1.2 Description](#_Toc24384671)

[1.3 Revision History](#_Toc24384672)

[Section 2 - Overview](#_Toc24384673)

[2.1 Purpose](#_Toc24384674)

[2.2 Scope](#_Toc24384675)

[2.3 Requirements](#_Toc24384676)

[2.3.1 Estimates](#_Toc24384678)

[2.3.2 Traceability Matrix](#_Toc24384679)

[Section 3 - System Architecture](#_Toc24384680)

[Section 4 - Data Dictionary](#_Toc24384681)

[Section 5 - Software Domain Design](#_Toc24384682)

[5.1 Software Application Domain Chart](#_Toc24384683)

[5.2 Software Application Design](#_Toc24384684)

[5.2.1 Activity diagram for the system](#_Toc24384685)

[5.2.2 Sequential Diagrams of the system](#_Toc24384686)

[5.2.3 **Class Diagrams for the system**](#_Toc24384687)

[Section 6 – Database Design](#_Toc24384688)

[6.1 Persistent/Static Data](#_Toc24384689)

[6.1.1 Dataset](#_Toc24384690)

[6.1.2 Static Data](#_Toc24384691)

[6.1.3 Persisted data](#_Toc24384692)

[6.2 Transient/Dynamic Data](#_Toc24384693)

[6.3 External Interface Data](#_Toc24384694)

[6.4 Transformation of Data](#_Toc24384695)

[Section 7 - User Interface Design](#_Toc24384696)

[7.1 User Interface Design Overview](#_Toc24384697)

[7.2 User Interface Navigation Flow](#_Toc24384698)

[7.3 User Function Description](#_Toc24384699)

[Section 8 - Other Interfaces](#_Toc24384700)

[8.1 Interface X](#_Toc24384701)

[Section 9 - Extra Design Features / Outstanding Issues](#_Toc24384702)

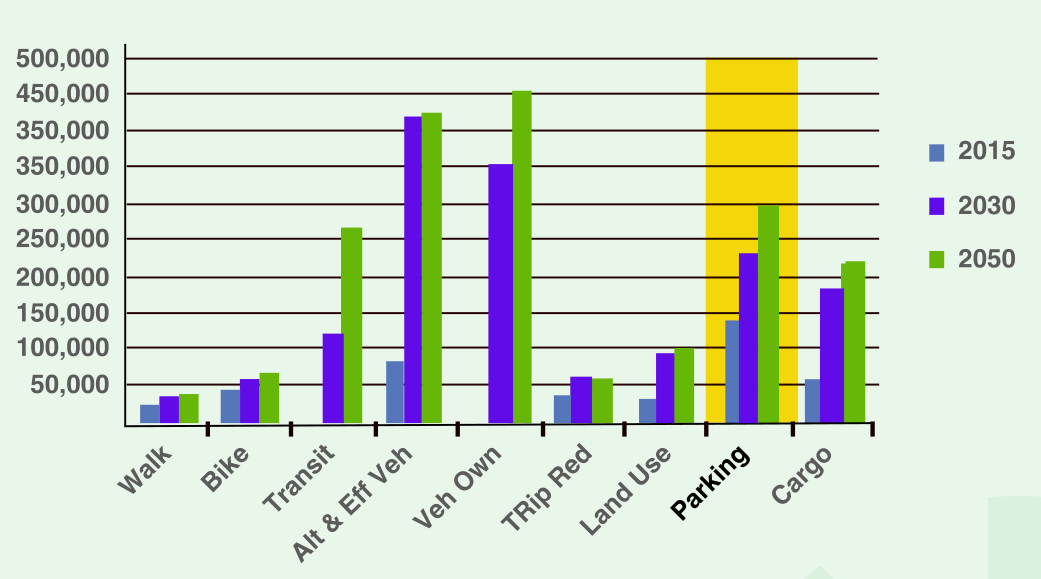
[Section 10 – References](#_Toc24384703)

[Section 11 – Glossary](#_Toc24384704)

**Introduction**

Traffic congestion caused by vehicle is an alarming issue worldwide and it has been extremely growing exponentially. Car parking problem is a major contributor and it has been, still is a major problem with the increasing number of vehicle ownership and vehicle size in the luxurious segments and confined parking spaces in metropolitan areas or cities. Looking for a parking lot/space is an activity to normal vehicle owners and often very frustrating for many people in the cities around the world. Thus, such kind of search burns about one million barrels of the world’s oil every day. As the global population continues to urbanize, without a well-planned, convenience-driven retreat from using our cars these problems will grow bigger.

According to a report, Smart Parking could result in 2,20,000 gallons of fuel saving till 2030 and approximately 3,000,000 gallons of fuels saved by 2050, if implemented successfully.



Smart Parking System obtains information about available parking lots in a particular place/area and processes it in real-time to guide vehicles to available positions. It involves using low-cost sensors (HC-SR04, FDIR, etc.), real-time collection of data and also mobile phone enabled automated payment systems that allows people to easily reserve a parking in advance, be able to clearly predict where they are most likely to get the next parking lot. When deployed as a system, it will reduce car emissions in urban areas by reducing the need for drivers to needlessly circle the city blocks looking for a parking lot. It also permits cities to carefully manage their parking supply. Smart Parking System aims rightness/safety and convenience.

**Problem Statement.**

****

Fig.1: Old traditional way of parking

**Key Issues of Parking Problems around the world:**

The main reason for parking problems in any place can be concluded as the disparity between the supply of parking facilities and parking demands. The parking demand is high due to the fact that car ownership has increased exponentially, high tension using and high density gathering.

The key issues of parking problems in Beijing are discussed as the following:

* The disparity between the rapid increment of motor vehicle and limited new parking facilities results the parking diﬃculty directly.
* Since 2004, the number of private cars in South Africa grows 0.34 million annually.
* In South Africa, the sharp gap between demand and supply of parking spaces is widening year by year. Therefore, the parking problems seem to be more and more serious. It must be mentioned that the ineﬃciency of facility usage increases the parking problem.
* For the reasons involving with economic, zone bit, and the characteristics of drivers, some of the parking facilities are used frequently while some others are used infrequently

**Section 1 - Project Description**

## 1.1 Project

The project Smart Parking System is a constructive approach for decent parking space and vehicle safety using the Internet of Things (**IoT**) as both of them are of crucial importance in current scenario. Method Analysis**:** Internet of Things play an important role where everything is connected to/with everything else. Ultrasonic sensors, RFID, switch, have been widely used to overcome problems faced with vehicle parking in order to meet up with the 4IR standard and provide safety.

Already there have been studies which have come up with prototypes which shall aid user or user in finding the available parking space, reserve a parking lot, report illegal behaviour within or bad service, and make efficient use of a mobile application.

## 1.2 Description

For the purpose of parking facility, specialized sensor is used to check whether the slot is available, if so then the Boom gate (entrance) opens or else not. Fee is charged on the time basis. The automobile industry and ICT industry amalgamate, the smart vehicle is an issue. To make vehicles safe, we aim at using ultrasonic sensors and a mobile app that has details about a particular driver which are unique (like drivers’ licence) thus the identification of an authorized user becomes more and more reliable. To improve more on safety, we also use Google Map features to make it easier to locate the nearest parking lots. The above prototype is to address traffic congestion, time wastage, vehicle safety, and its current location. Applications/Improvements: The best part about this prototype is that it has been developed using ultrasonic sensors, servo motors, mobile app, Arduino boards, and NodeMCU WIFI modules. It not only enhances communication between the two or more connected devices but also maintains the transparency between them. All these is achieved at cheaper rates.

## 1.3 Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Reason for consultation** | **Results** |
| 20/7/2019 | Collecting resources and tips on where to start with the project. | Positive |
| 1/8/2019 | Needed assistance on connection with Arduino and LCD screen | Positive |
| 10/8/2019 | First task was completed | Positive |
| 27/9/2019 | Stuck on task two | Negative |
| 3/10/2019 | Needed help on connecting Ethernet shield | Negative |
| 7/10/2019 | LCD and circuit wiring issues | Positive |
| 12/10/2019 | Final Prototype | Positive |
|  |  |  |
|  |  |  |

# 1.4 Project Management Table

|  |  |  |
| --- | --- | --- |
| No. | Project Complexity Question | Answer |
| 1 | What is the estimated time to finish the project? | Less than 6 months. |
| 2 | Have sufficient project resources been budgeted and allocated? | Most resources are budgeted and allocated. |
| 3 | How much confidence is there in the expenditure and information projections? | IoT is very complex and a bit challenging thus the experience of learning something new and useful makes me proud of doing this final year project. |
| 4 | Are the resources available for maintenance of the project deliverable after project closure? | No, after project is done. School will be closed thus there will be no access to the lab. |
| 5 | Is the project supervisor fully resourcing the project? | My supervisor owns most of the resources needed |
| 6 | What minor setbacks did you experience? | * Getting sick. * Family issues that needed my full attention. * Balancing time for the project and studying. * Keeping up with Supervisors deadline. |
| 7 | What major setbacks did you encounter? | * Learning about electronics and circuits. * Limited resources. * Unpredictability of circuit components. * Less people understood less about my project thus besides my supervisor, other helping hands were scarce. |
| 8 | How much time taken each day to do the project? | 4 – 5 hours |
| 9 | Where else did you get additional help? | Online video tutorials and eBooks |
| 10 | Has the project students executed similar projects? | Even though there are other students doing IoT project, there is no one doing a similar project to mine. |
| 11 | What is the level of your commitment? | I have made the project a number one priority. |
| 12 | What experience and training level is needed for doing IoT? | Well trained and very experienced in handling electronic elements and programming. |
| 13 | What is the project's duration? | Less than 6 months |
| 14 | How much variation in the timeframe can be tolerated? | Schedule is not fixed and therefore highly flexible |
| 15 | Are there any dependencies and/or inter-related projects? | There are no major dependencies or inter-related projects |

|  |  |  |
| --- | --- | --- |
| 16 | Does the project address externally generated mandates? | The project has little or no direct impact on accomplishment of external mandates |
| 17 | How will the failure of the project impact your final year results? | Impact of project failure on my results is minimal |
| 18 | How politically sensitive is the project? | Political climate has no impact on the project. |
| 19 | How significant will the project process or activities be impacted? | No project process is impacted |
| 20 | What is the level of change to the project unit? | It will impact a number of other project units |
| 21 | Is the project using proven technology? | The technology is proven and has been available for a number of years |
| 22 | Is the proposed solution applied in a new, proven, or tried way? | Application of the technology is tried and proven |
| 23 | What is the anticipated involvement of the end users with system design and testing? | Highly involved with development stage, provide significant input and have significant ownership of system at large. |
| 24 | What is the anticipated involvement of the end users in the definition of project requirements and scope? | Requirements well-established, baseline defined, user acceptance high with no changes |
| 25 | Does this project require data conversion? | No data conversion is required |

# CHAPTER 2

# Section 2 - Overview

## 2.1 Purpose

Due to the increase in traffic, many drivers roam around in search of decent safe parking space. This not only creates traffic congestion but also time wasting. What if a driver has advance information of available parking spaces in the required destination lot.

Studies have shown illegal encroachments by vehicles parking on the roads have a massive effect of the flow in traffic [1]. The purpose of this paper is to remove the all too familiar sight of cars parked on the roads by introducing smart parking systems.

## 2.2 Scope

In this Smart Parking System, I set the ultrasonic sensor in such a way that when a car in parked in a particular slot, the ultrasonic sensors will sense the presence of a vehicle in that particular slot and update the database.

The user can view how many parking lots are available while still at home. Make reservation. The user gets real time updates on available and occupied parking spaces on the mobile applications. When the user has reached the desired parking lot, the system makes necessary updates and notifies the database. The user vehicle will remain protected there until departure time has been reached thus payment will be made by leaving the premise or by extending the stay. Upon leaving the system will also get notified and make necessary updates on the database.

## 2.3Definitions and Acronyms

|  |  |
| --- | --- |
| **Name** | **Definition** |
| FCM | Firebase Cloud Messaging |
| SPS | Smart Parking System |
| IoT | Internet of Things |
| SRS | Software Requirement Specifications |
| LED | Light Emitting D |
|  |  |

## 2.4 Overview

I am going to build a Smart Parking System, that will enable drivers to be updated earlier anytime anywhere. Help the administrator to easily maintain the system. This type of IoT project has been done before but not in an exact way, thus this project is an improvement.

The reason behind building this app is the fact that as a student I saw a strong need to make during the graduation ceremony. A lot of cars were parked even in unnecessary places and most cars got some damages one way or the other.

**User Needs**

* **Mobile phone**: so that they can install the smart parking System app.
* **Connection to Wi-Fi:** so that as a user you can connect to the internet and be able to view real-time status of the system.
* **Registered Car:** To avoid stolen cars**.**

**Assumptions and Dependencies**

By doing this project I assume I will be done before the due date and that most features will be working. At the same time, I will assume that if the day comes and I am not done, it would be on the software application part as android studio as it tends to get updated every now and then. Those updates always interfere with the build-up of the app.

## 2.5 Specific Requirements

### The design requirements are about how the system will be built, not how it will work. Below is the required requirements in order to fully build the Smart Parking System.

**Functional Requirements**:

* The application should allow driver/admin to login.
* The system should allow users to register.
* The user provides login details
* If login successful, the user will be taken to home page
* If login is unsuccessful, the user will be taken to forgot password

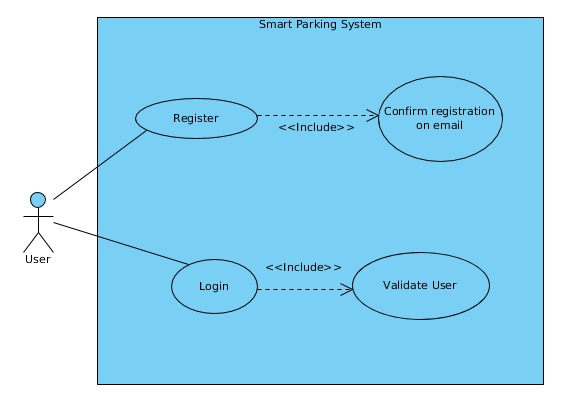


Fig.2: Registration/Login diagram

* The system should allow users to get another password if it’s forgotten.
* The user clicks reset password.
* Then the user is asked for a new password.
* Then new password will get validated.
* A confirm message will display if all went successful.



Fig.3: Forgot Password Diagram

* The system should allow users to make reservations.
* The user makes reservation after checking parking status
* User first must make payments
* After payments, if all goes well user will have reserved parking lot
* User still has option to cancel reservation
* Admin updates the system

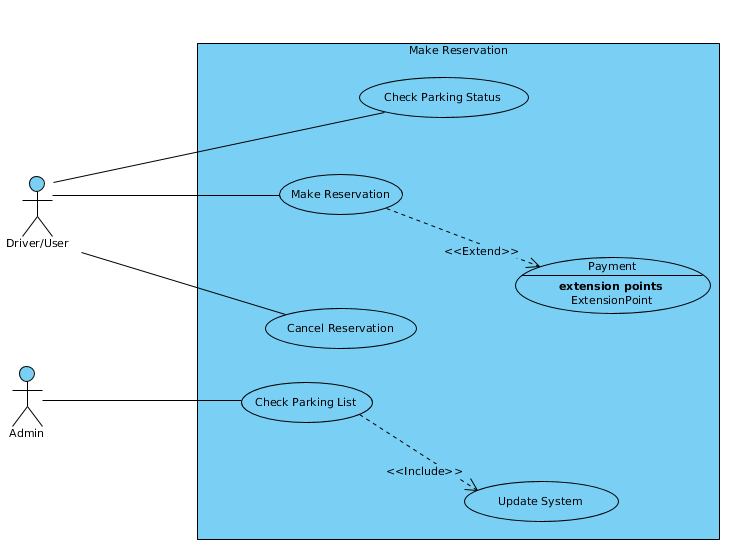


Fig.4: Make Reservations Diagram

* The system should allow users to cancel reservation.
* After booking, user can cancel booking a slot.
* After cancelation, user can go back to home page or logout.

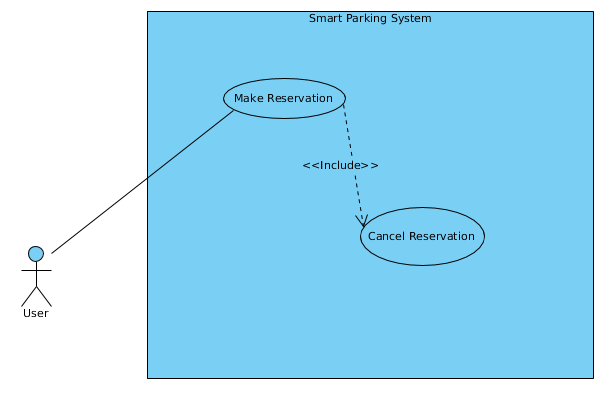


Fig.5: Cancel Reservation Diagram

* The system should allow sensors to detect a car. The system should allow sensors to transmit data to the database.
* When car reaches the entrance gate, sensor detects the car and let it in.
* When car is parked, sensor detects again.
* Every sensor detection data is collected and stored to the database.

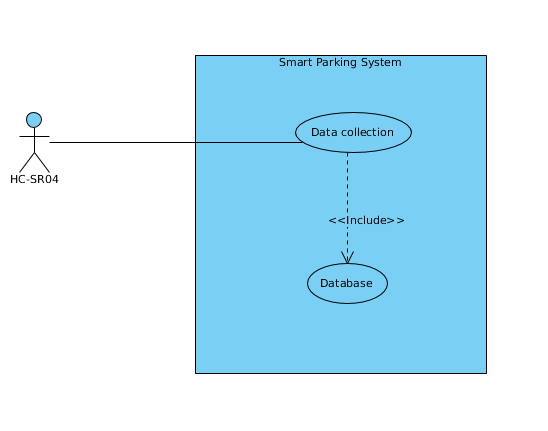


Fig.6: Sensor Diagram

* The system allows the logged in User/admin to access their home page for more feature to use.
* When user logs in, they are taken to the home page.
* When Admin logs in they are taken to their home page.

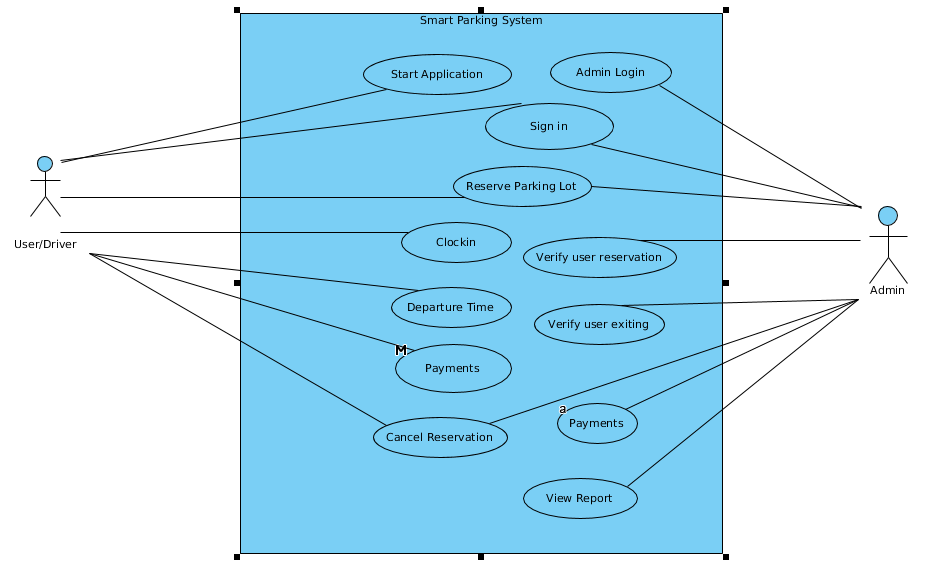


Fig.7: Home Page for User and Admin diagram

**Non-Functional Requirements**:

* **Usability**: The system shall provide uniform look and feel between all the application

pages. The system shall provide icons and instant search buttons.

* **Accessibility**: The system shall provide handicap access.

The system shall provide easy to understand interface.

* **Reliability and Availability**: The system shall provide stable database to store all the

Data. The system shall provide for replication of database

to off-site storage location.

* **Performance**: The application will run on android mobile phone as they are more

versatile. The system shall take initial load time depending on the internet

connection strength which also depends on the media from which the

system runs.

* **Data Storage**: Users password will be kept private. It shall always be echoed with

Special characters representing typed characters.

The systems backend servers shall never display the user’s password or

car details.

The systems backend servers shall only be accessible to authenticated

Administrators.

The systems backend database shall be encrypted.

* **Supportability**: The source code developed for this system shall be maintained in

configuration management tool.

* **Availability**: Sensors have a status update signal if a certain distance has been reached

The parking sensor has an easily understandable LED colour display set.

* **Software requirements**:

1. **Arduino IDE** [2], which will be used to create commands/instructions for the microcontroller boards.
2. **Android Studio** [3], will be used to create the mobile application which is a useful tool when it comes to designing the layouts for the user interface.
3. **Firebase Database** [4], it’s a stable real-time database that will be important for making sure that the systems information get stored for better management.
4. **Operating System** [5], Linux Mint Sonya 19.2, any Windows OS
5. **Browser** [6], Chrome or Firefox

* **Hardware requirements**:

1. **Servo motor** [7], will be used as boom gate at the entrance of SPS and the exit.

They have high power and efficiency.

1. **Arduino Boards** [8], it is cheap. It comes under open source hardware, so

experienced circuit designers can make their own version of the

module, extending Linux operating systems. Most micro-

controller systems are limited to

Windows only.

1. **NodeMCU** [9], It is also open source. Operates on low power which is good for

Demonstration of the project. It’s able to connect to the database with.

1. **HC-SR04** [10], the ultrasonic sensor operates on signals which are usually in the 40- to

70-kHz range. These signals are used like radar—they’re radiated

Toward a target and reflected back to the source. They are low cost,

Flexible, and. especially useful in shorter-range applications, especially

Automotive design.

1. **Ethernet cable:** Since the application must run over the internet, all hardware shall

### 2.4.1 Estimates

|  |  |  |
| --- | --- | --- |
| **#** | **Description** | **Wks. Est.** |
| 1 | Do a research on each electronic component and sensors | 1-2 weeks |
| 2 | Oder/purchase required components | 1-2 weeks |
| 3 | Designing the project | 3-4 weeks |
| 4 | Testing | 2 weeks |
|  | **TOTAL**: | 8-9 weeks |

# Section 3 - System Architecture

Below is the proposed system architecture for my project: Fig. 8:

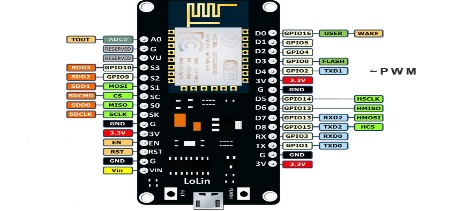


User

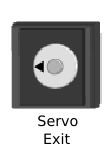


LAN



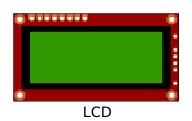
Parking Entrance

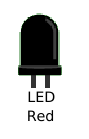


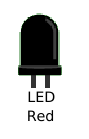


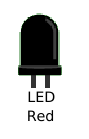
Parking Exit

nodeMCU









Parking 3

Parking 2

Parking 1

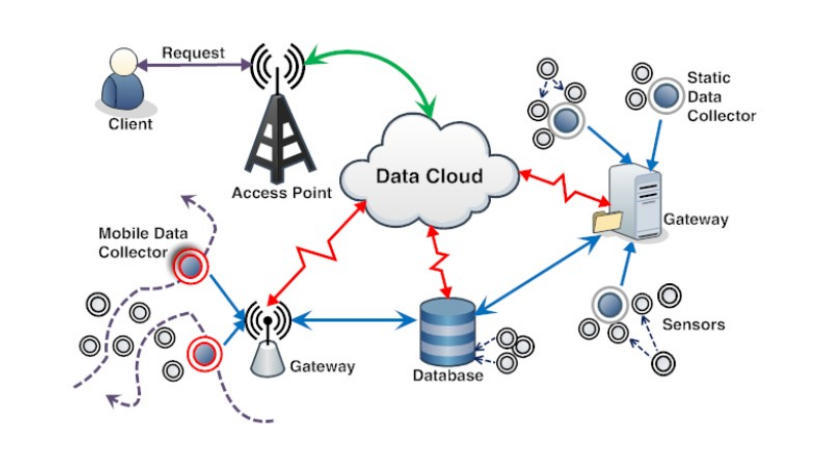


Fig. 9: An actual proposed architecture

Use of each component to the project.

* **Ultrasonic Sensor** (HC-SR04):

They are used to detect the vehicle at the entrance, parking spaces, and also at the exit point. They are good at detecting objects since they have high frequency, high sensitivity and high penetrating power to provide high accuracy.

* **LED** (Green and Red):

The LED’s are used to specify the parking space condition, if a green LED is on then that means the parking lot is available (vacant) but if a red LED in on that means that particular parking space is booked or occupied.

* **LCD Screen 16\*2:**

Is used to display parking slots statuses.

* **Servo Motor**:

Is used to act as a boom gate to increase security within the system.

* **NodeMCU:**

Used to collect data on the Arduino board and store it on Firebase Database.

* **Arduino microcontroller:**

It was used as a microcontroller to hold instructions and commands which will be used by the sensors and other electronic components.

* **Router** (Gateway):

A router is used to connect to firebase and be able to store data in order to learn from it as time passes.

* **User:**

The user interacts with the system in order to get a better parking space.

* **Server:**

Is for storing data to learn from.

* **Admin:**

Manages the system and make sure everything is running smooth.

# Section 4 - Database Design

Brief description of each element in this module or a link to an actual data dictionary

(template of a database table description)

|  |
| --- |
| **Table** |

|  |  |  |
| --- | --- | --- |
| **Field** | **Notes** | **Type** |
| ID | Unique Identifier from TABLE\_SEQ | DECIMAL |
| NAME | The Name in Object.Name() | VARCHAR |
| VALUE | The Value output from somewhere | VARCHAR |
|  |  |  |
|  |  |  |

# Section 5 - Software Domain Design

## 5.1 Software Application Domain Chart

A flowchart is a diagram that represents a workflow which is the process of the whole System.

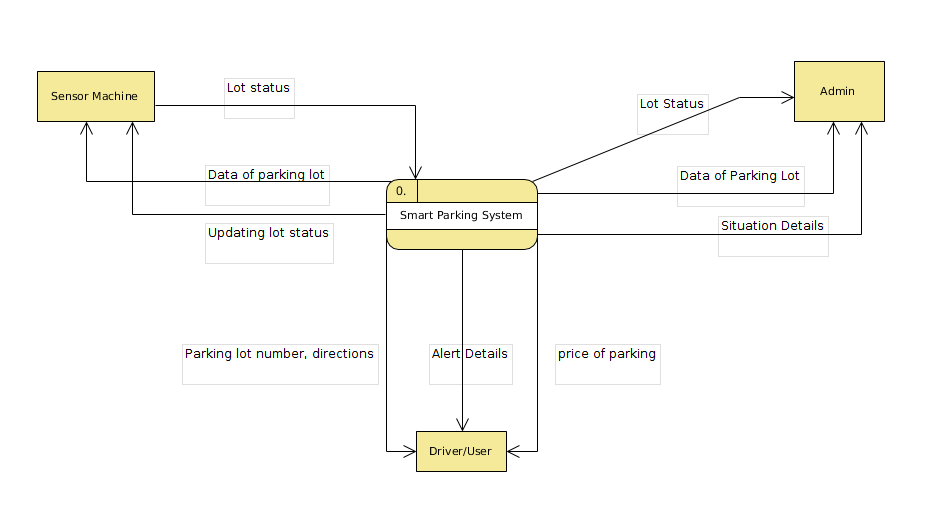


Fig.10: Data Flow Diagram.

## 5.2 Software Application Design

Here I represent the segment of reality for which a software system is developed.

### 5.2.1 Activity diagram for the system

A high-level description of a series of actions or flow of control in a system similar to a flowchart.

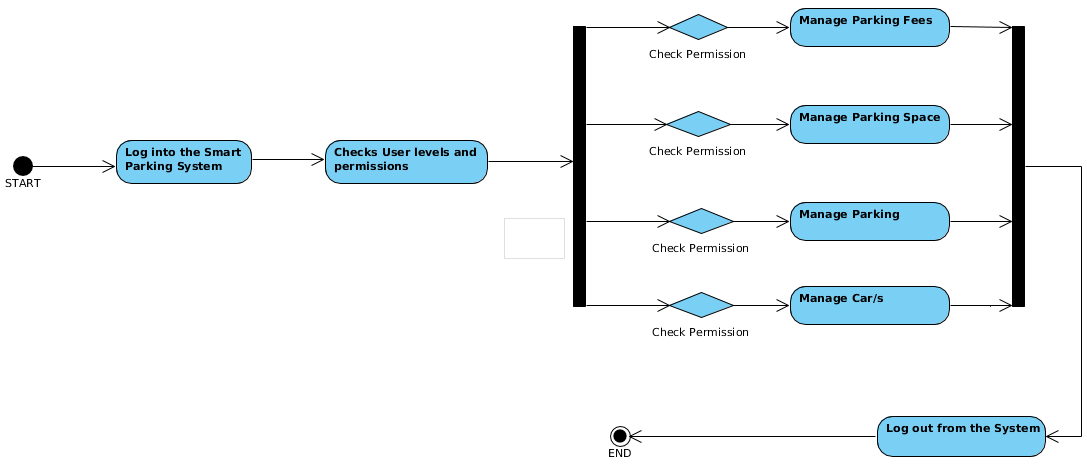


Fig.11: Admin Activity diagram

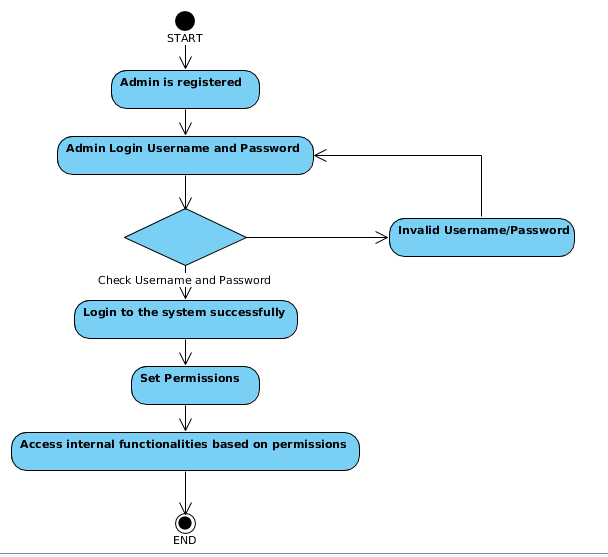


Fig.12: Admin login activity

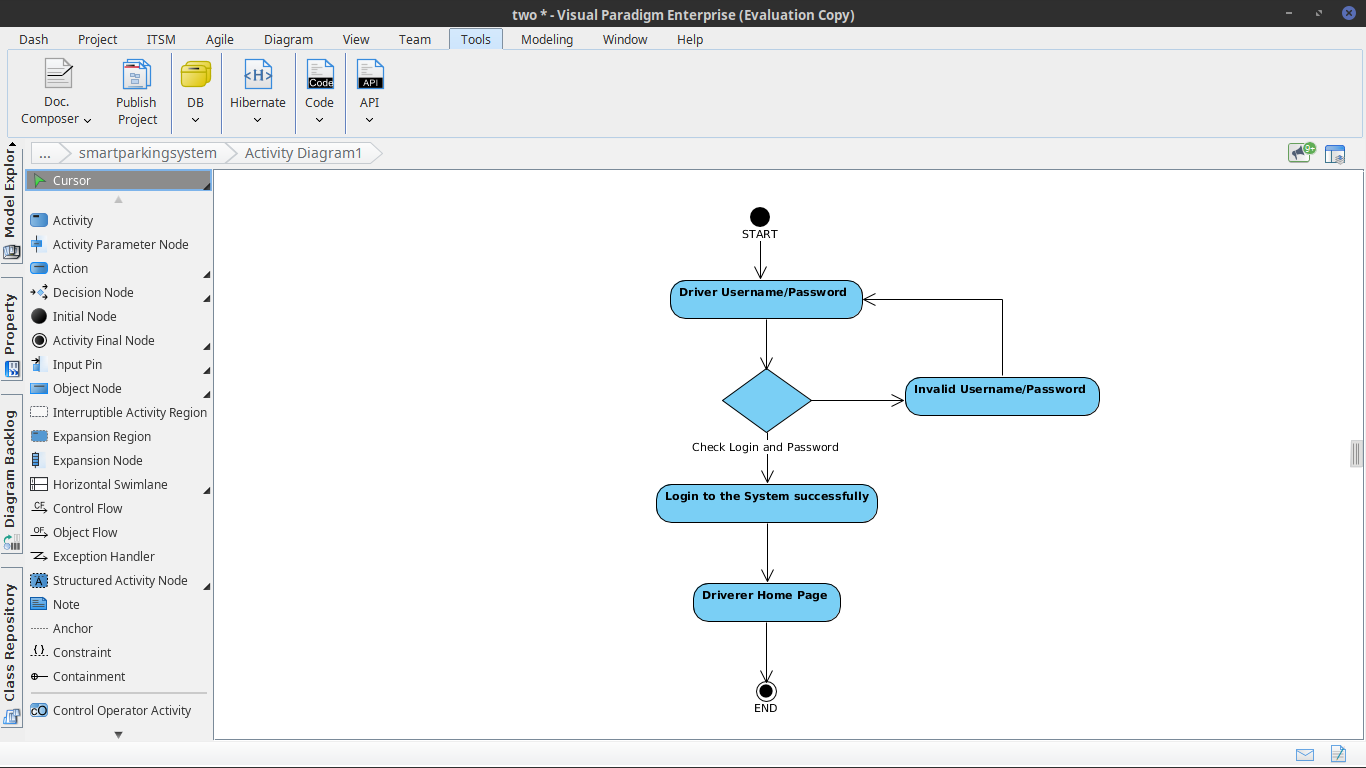


Fig.13: Driver login activity

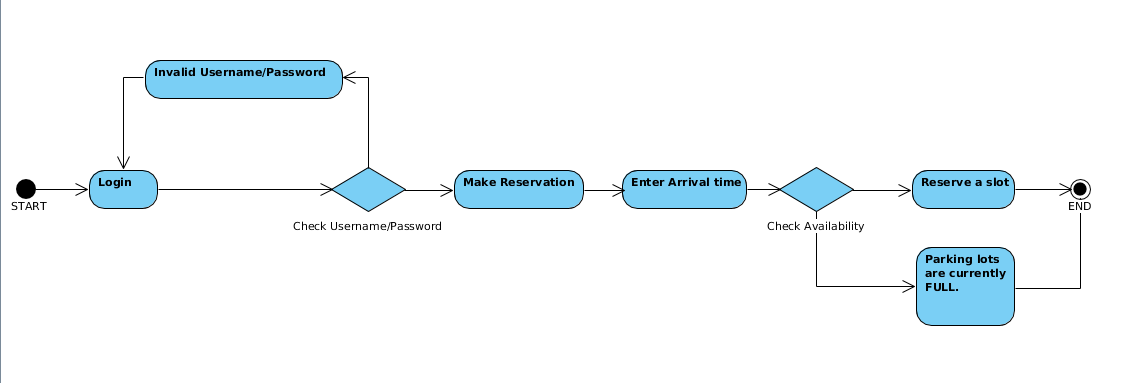


Fig.14: Reservation activity.

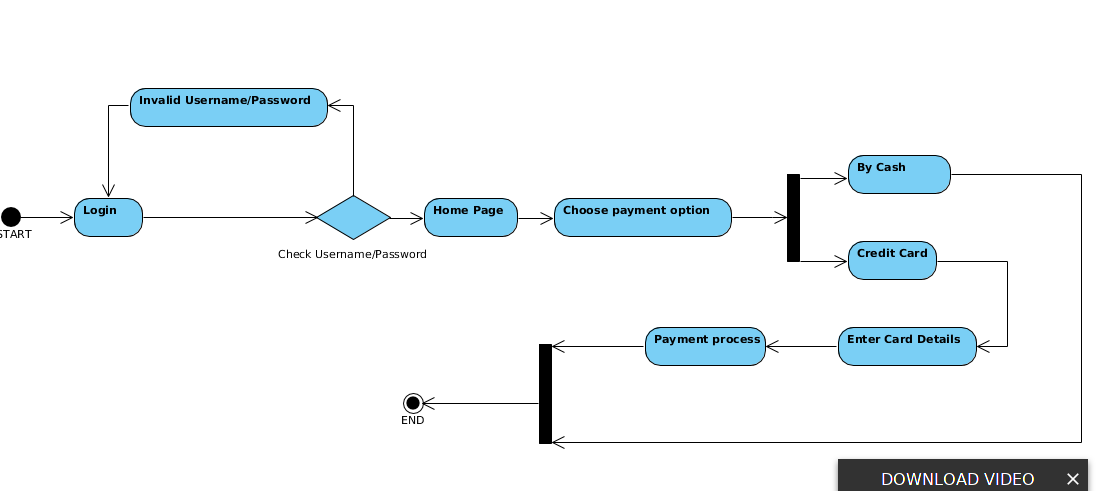


Fig.15: Payment activity.

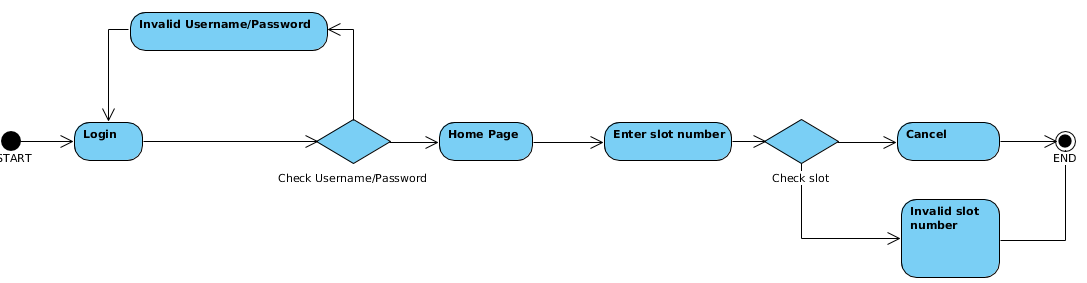


Fig.16: Cancel activity.

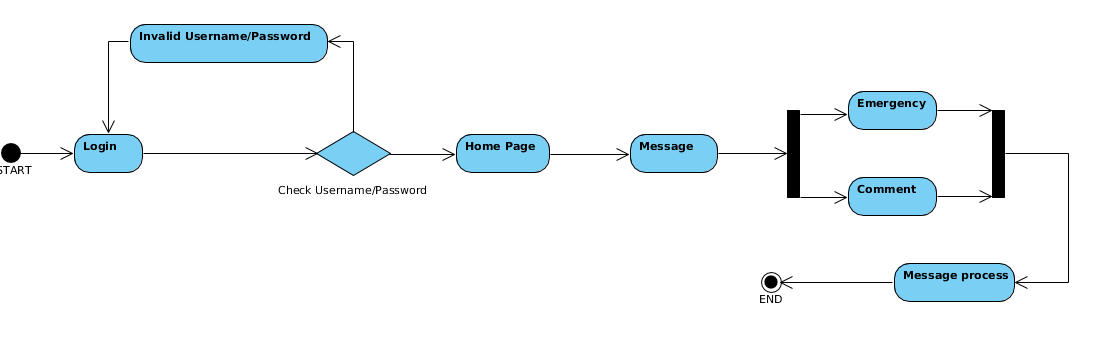


Fig.17: Message activity

#### 5.2.2 Sequential Diagrams of the system

Here I show the illustration of sequential diagrams which show the different parts of the system interaction with each other to carry out a certain function. The interaction is basically of objects in each single use case.

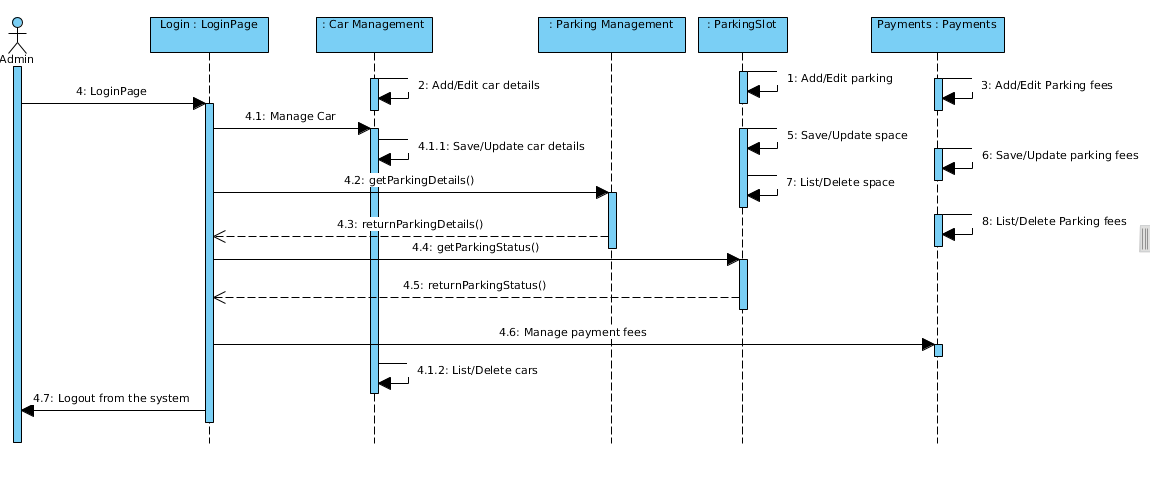


Fig.18: Admin home page diagram.

Fig. 18, shows a sequential diagram for accessing the home page as an Admin. It includes Admin(actor) and some few other classes which are Login, Payments, ParkingSlots, etc. First the administrator provides his/her login details which are username and password. After login in, the user gets taken to the home page where he/she manages details like editing car details, listing of deleted cars, Save or update car details. Then the user moves on to manage the parking lots. Here the user manages parking lots by either adding or editing parking fees, saving or updating space, and listing or deleting old parking space details. Finally, if the user is done with it, they can logout of the system.

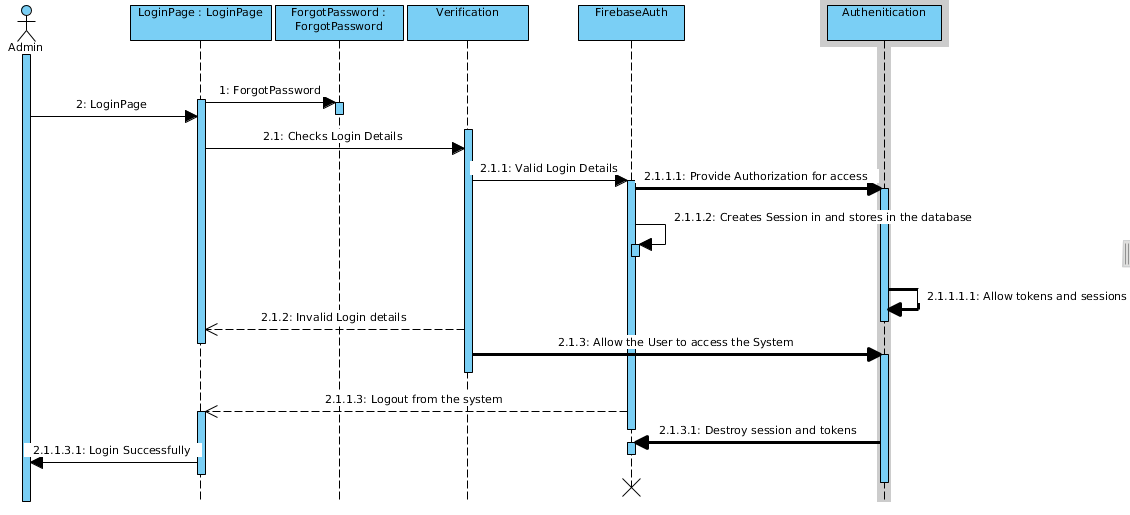


Fig.19: Admin login page diagram

Fig.19 displays the login page for the Administrator, where the user will provide their login detail. Then the details will get verified and return an error if there is something wrong with the username or password provided. But, if the details are ok, then the user will move on to home page. There is and option that if user doesn’t recall his/her password, the user can press the forget password option in order to receive a new password.

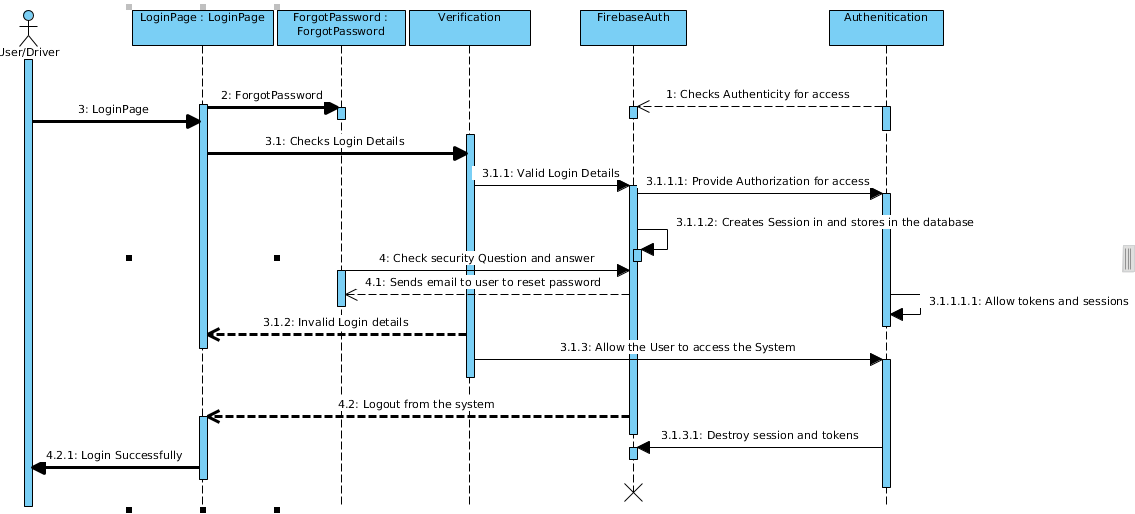


Fig.20: Driver login page diagram

Fig.20 shows the sequential diagram for driver login page. The driver is the actor here. You provide your login details and if there is no error for authentication, then the user will be taken to the users home page. But the is any error during validation of username and password, then a prompt message of “invalid login details”. If the user can’t remember his/her password, the user must take the forgot password option in order to receive a new one. Or if user is logging in for the first time, then

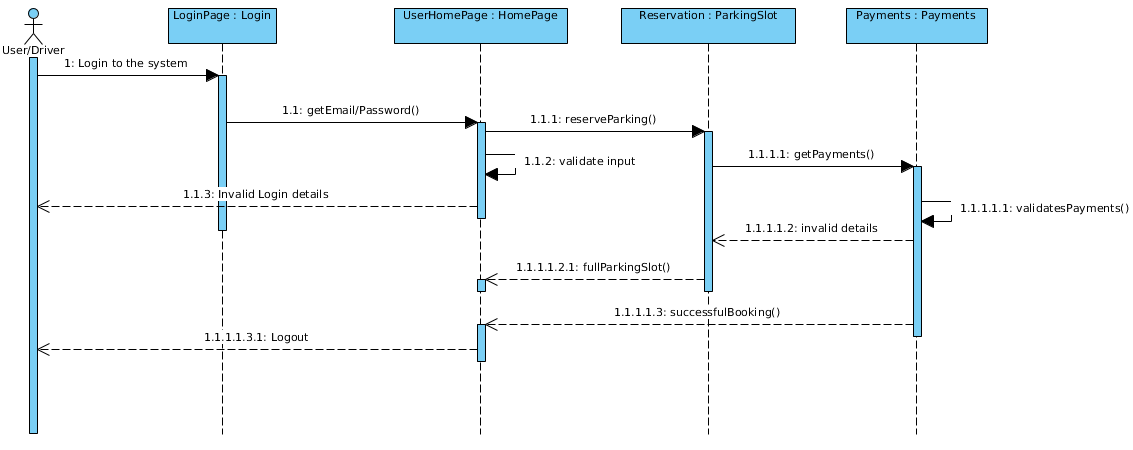


Fig.21: Reservation page diagram.

Here on Fig.21 above, the user gives username and password as it in order to login, then user is taken to the home page after successful login. The user makes the reservation, then the system checks for available parking slots. Next is the Payment home page, where a user must pay for the parking fees. After successful payment, the system gets updated.

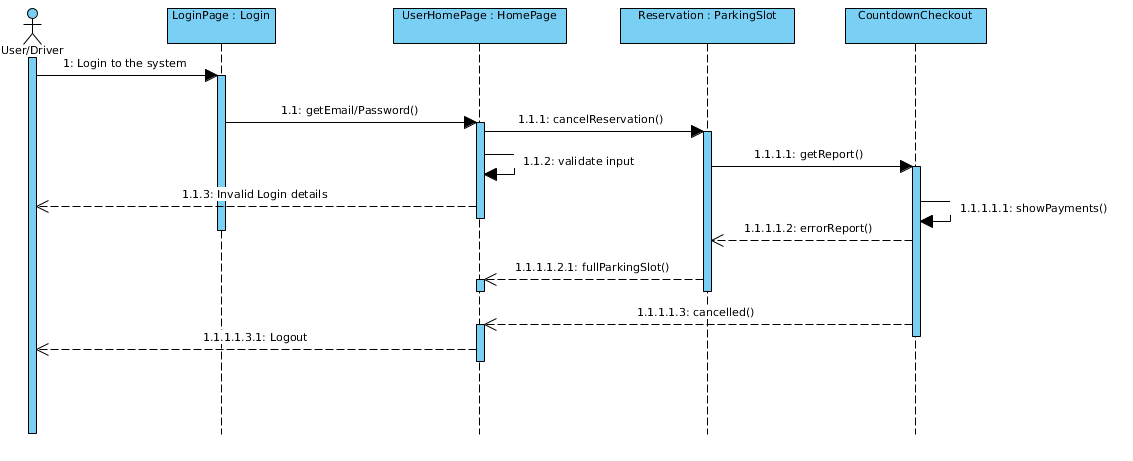


Fig.22: Cancel reservation page diagram

The above figure (22), shows cancelation of reservation, when user cancels the vacation. The system will check whether you made a reservation or not using the History Review. Then after validation, your reservation will be cancelled and a message will be sent to the user that verifies the cancelation. Then the user will have options to either go back to home page or logout.

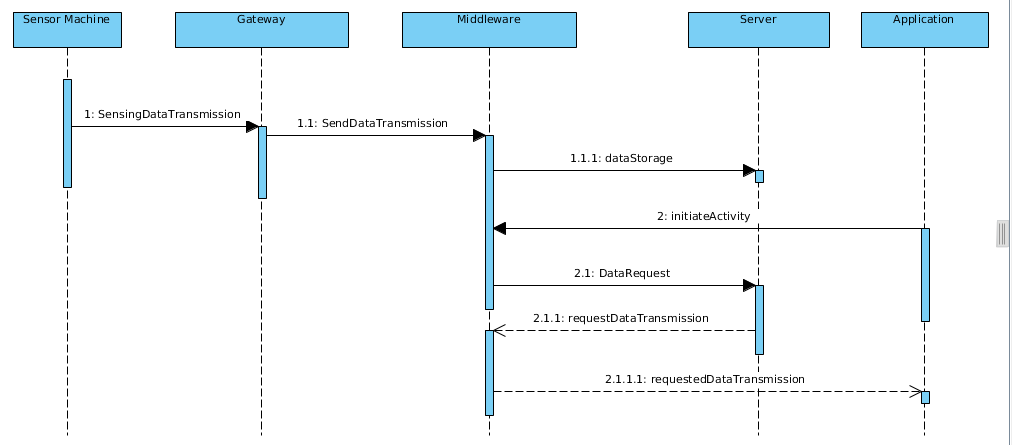


Fig.23: Sensor page diagram.

The diagram above (Fig.23), is a sequential demonstration of how a sensor functions within the system. The sensor sends out data transmission in terms of using a pulse and an echo, when a car is detected within the entrance boom gate, the pulse sent will bounce back to the echo pin and data will be sent to the system (manly to the servo motor). The servo motor will then open the boom gate and let the car in. Afterwards the car will then park at specified parking lot, and that sensor on that parking slot will send a message to the firebase and that data will get stored on the database.

##### 5.2.3 **Class Diagrams for the system design**

It’s the Unified Modelling Language that is the type of static structure diagram that describes the structure of a system by showing their system’s classes, attributes, operations or methods and the relationship amongst the relationship among objects.

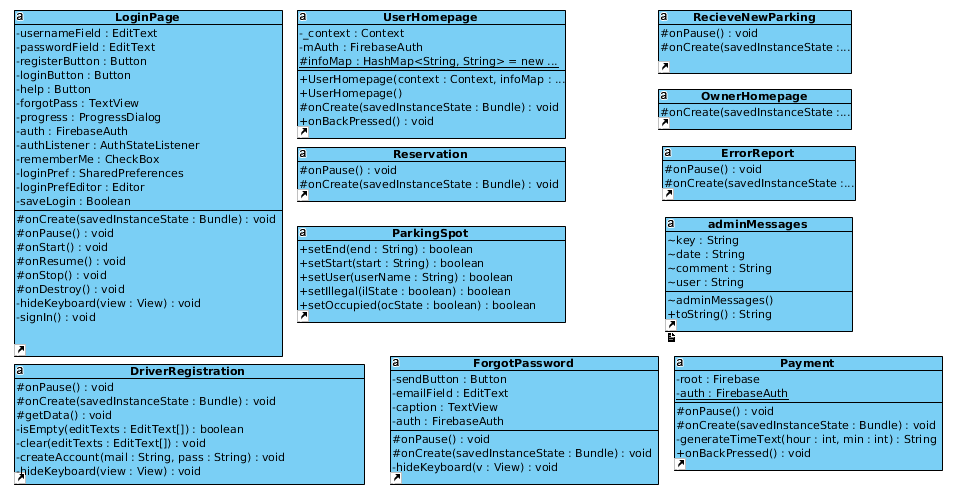


Fig.24: First part of the class diagram

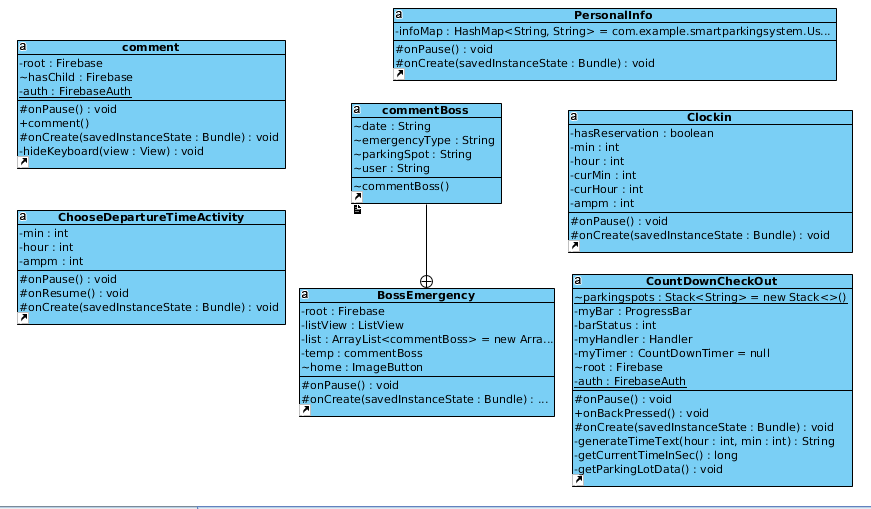


Fig.25: Second part of class diagram

# Section 6 – Database Design

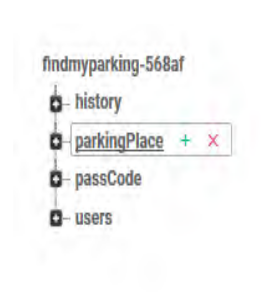
Database design is what translates data objects defined as part of the analysis model into:

* Data structure at the software component level.
* A possible database architecture at the application level.

It mainly focuses on the representation of data structures that are directly accessed by one or more software components.

The challenge is to store and retrieve data in such a manner that useful information can be extracted from the data environment. Reasons to use Firebase are:

* **Real-time:** Insteadof typical HTTP requests, Firebase Realtime Database uses data synchronization. Every time data changes, any connected device receives that update within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.
* **Offline:** Firebase apps remain responsive even when offline because the Firebase real-time Database SDK persists your data to disk. Once connectivity is re-established, the user device receives any changes it missed, synchronizing it with the current server state.
* **Accessibility from User devices:** The Firebase Real-time database can be accessed directly from a mobile device or web browser, there’s no need for an application server. Security and data validation are available through the Firebase Real-time Database security rules, expression-based rules that are executed when data is read or written.

Fig.26: Database root tree

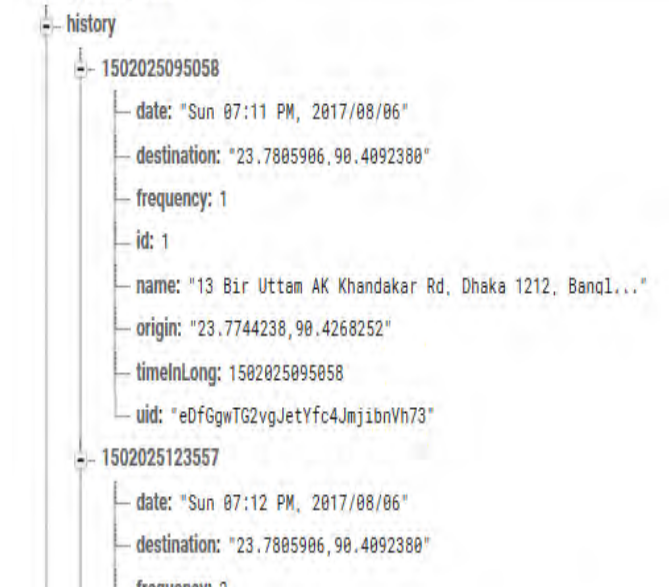


Fig.27: Expansion of the history tree sub-root.

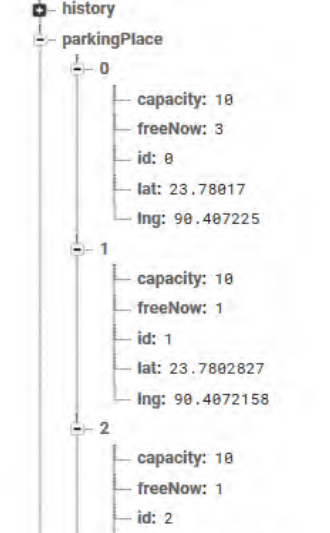
Fig.28: Parking slot Info tree sub-root



Fig.29: PassCode tree sub-root



Fig.30: Users tree sub-root

## 6.1 System Implementation

Below are a few snippets of important code of my software:

### 6.1.1 Android Manifest

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.smartparkingsystem">

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".LoginPage">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<action android:name="android.intent.action.VIEW" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity android:name=".DriverRegistration" />

<activity android:name=".Payment" />

<activity android:name=".Clockin" />

<activity android:name=".activity\_review" />

<activity android:name=".CountDownCheckOut" />

<activity android:name=".PersonalInfo" />

<activity android:name=".CommentBoss" />

<activity android:name=".BossEmergency" />

<activity android:name=".BossMap" />

<activity android:name=".comment" />

<activity android:name=".Reservation" />

<activity android:name=".MapDirectional" />

<activity android:name=".ReportIllegal" />

<activity android:name=".Emergency" />

<activity android:name=".Settings" />

<activity

android:name=".PopUp"

android:theme="@style/AppTheme" />

<activity android:name=".RecieveReports" />

<activity android:name=".Messages" />

<activity android:name=".ErrorReport" />

<activity android:name=".ForgotPassword" />

<activity android:name=".RecieveNewParking" />

<activity android:name=".Resume">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

</intent-filter>

</activity>

<activity android:name=".UserHomepage" />

<activity android:name=".History\_2\_user" />

<activity android:name=".UserReviewHistory" />

<activity android:name=".MessageUsersActivity" />

<activity android:name=".HELP\_USER" />

<activity android:name=".OwnerHomepage" />

<activity android:name=".ChooseDepartureTimeActivity" />

<activity android:name=".UserReviewHistoryPage2">

</activity>

</application>

</manifest>

### 6.1.2 App Dependencies



### 6.1.3 Critical Coding

Calculate total time parked and total to pay

totHours = bundle.getInt("departHour") - bundle.getInt("arriveHour");

totMins = bundle.getInt("departMin") - bundle.getInt("arriveMin");

totPay = (totHours + (double) totMins / 60.0) \* rate;

TextView startTimeText = (TextView) findViewById(R.id.startTimeText);

TextView endTimeText = (TextView) findViewById(R.id.endTimeText);

TextView totalPayText = (TextView) findViewById(R.id.totalToPay);

TextView currPrice = (TextView) findViewById(R.id.currentPrice);

double currentPrice = iLink.getDefaultPrice();

currPrice.setText("Current rate is: " + "R" + currentPrice + "/hour");

// generate the Clockin and Clockout times

startTimeText.setText(generateTimeText(bundle.getInt("arriveHour"), bundle.getInt("arriveMin")));

endTimeText.setText(generateTimeText(bundle.getInt("departHour"), bundle.getInt("departMin")));

// format total amount to pay

totalPayText.setText(String.format("R%.2f", totPay));

final TextView total = (TextView) findViewById(R.id.totalToPay);

payButt.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

String rate = total.getText().toString();

// generate the Clockin and Clockout time

String clockInTime = generateTimeText(bundle.getInt("arriveHour"), bundle.getInt("arriveMin"));

String clockOutTime = generateTimeText(bundle.getInt("departHour"), bundle.getInt("departMin"));

long clockInTimeInSec = bundle.getInt("arriveHour") \* 60 \* 60 + bundle.getInt("arriveMin") \* 60;

long clockOutTimeInSec = bundle.getInt("departHour") \* 60 \* 60 + bundle.getInt("departMin") \* 60;

String spotAssign = iLink.getSpot(clockInTimeInSec, clockOutTimeInSec);

if (spotAssign == null) {

AlertDialog.Builder respond = new AlertDialog.Builder(Payment.this);

respond.setTitle("PARKING LOT FULL");

respond.setMessage("Sorry, all spots are currently full. Please try again later.");

respond.setPositiveButton("Ok", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

dialog.cancel();

}

});

AlertDialog alertDialog = respond.create();

alertDialog.show();

} else {

// get the current date

SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");

Date date = null;

try {

date = sdf.parse(sdf.format(new Date()));

} catch (ParseException e) {

e.printStackTrace();

}

Firebase resChild = root.child("Reservation");

Firebase hasChild = resChild.child(date + " " + clockInTime);

// create child fields for reservation in Firebase

Firebase rateChild = hasChild.child("Rate");

Firebase clockInChild = hasChild.child("Clockin");

Firebase clockOutChild = hasChild.child("Clockout");

Firebase dateChild = hasChild.child("Date");

Firebase userChild = hasChild.child("User");

// populate reservation in Firebase

rateChild.setValue(rate);

clockInChild.setValue(clockInTime);

clockOutChild.setValue(clockOutTime);

dateChild.setValue(sdf.format(date));

userChild.setValue(userName);

// Set order for new parking spot

iLink.setOrder(spotAssign, clockInTimeInSec, clockOutTimeInSec);

# Section 7 - User Interface Design

## 7.1 User Interface Design Overview

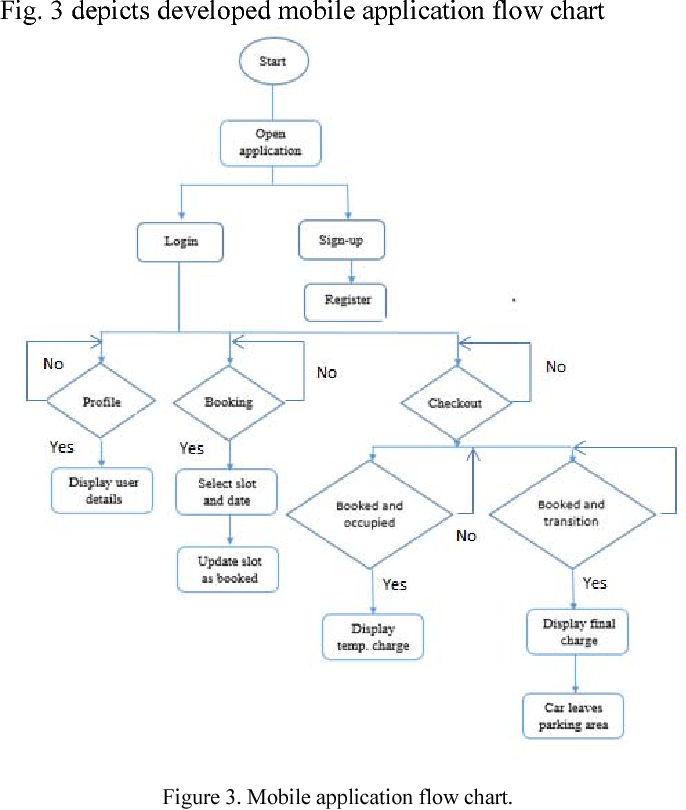
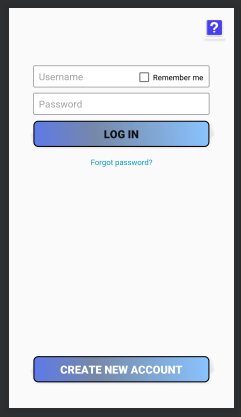
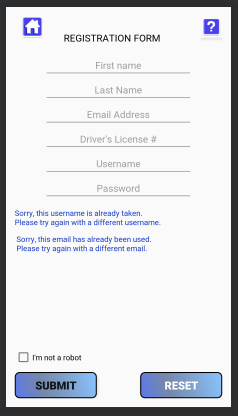


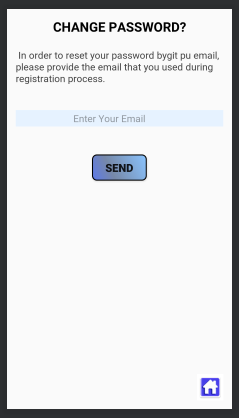
Fig.7.1: Application Flow Diagram.

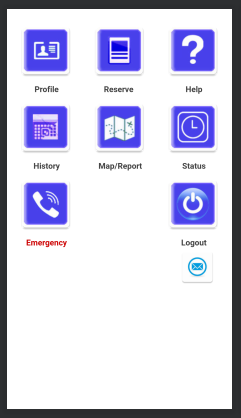
## 7.2 User Interface Navigation Flow

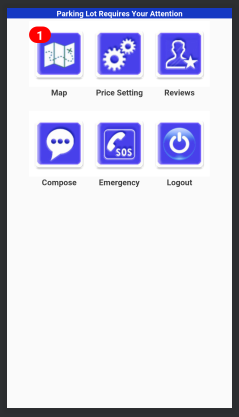
Here is the design process of making interfaces in software or other computerized devices with a focus on the looks and style. Below is a display of my application user interface.

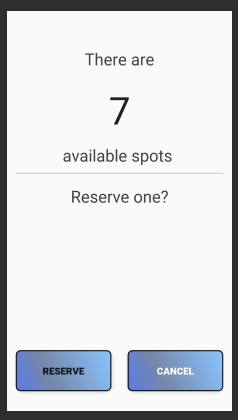
Fig.7.1: Login Page

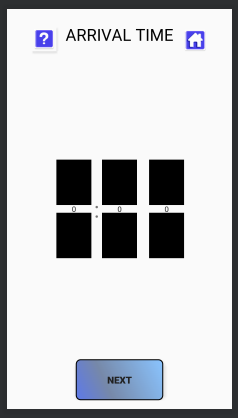
Fig.7.2: Registration Page

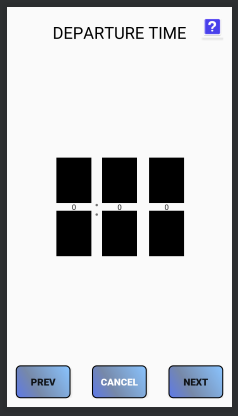
Fig.7.3: Forgot Password

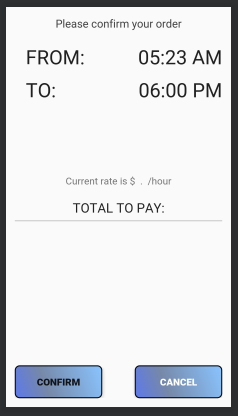
Fig.7.4: User Home Page

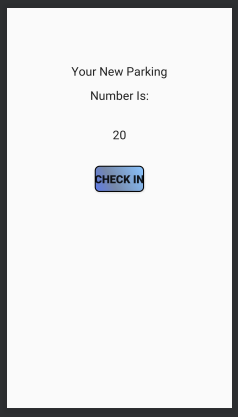
Fig.7.5: Admin Home Page

Fig.7.6: Reservation Page

Fig.7.8: Arrival Time Page

Fig.7.9: Departure Time Page

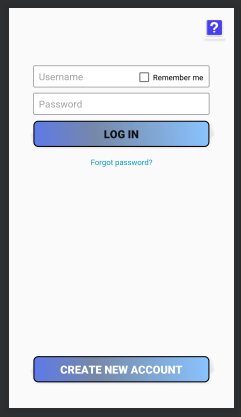
Fig. 7.9: Payment Page

Fig.7.10: Confirmation Page

## 7.3 User Function Description

The user interface is everything that the user can see and interact with. It includes a lot of features like layouts, notification overview, UI visibility, design effective navigation, toast overview, etc.

The main aim of designing this type of user interface is the fact that it is able to deliver quick and consistent solutions to vehicle parking related problems that my targeted audience grapples with. It’s a good idea to solve these issues while keeping the application looking beautiful and working smoothly. Below is the user functional description of a few UI to elaborate on the effectiveness of the app to the users.



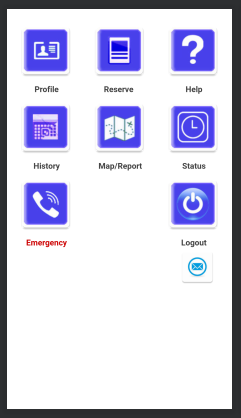
Keep typing requirements at minimum. By forms simple using autocomplete whenever applicable. Using message type query help to display a numeric keyboard when asking for a pin, displaying a search button in place of enter when searching and also include ‘@’ and ‘.com’ buttons when asking for email IDs.

Users want results quickly thus the app keeps page loading under 1-3 second by receiving real time results which Firebase Real-Time database provides with ease, giving users the happy app experience needed.

A crisp and simplified on-board flow login page, with an option to avoid login every now and then thus a “remember me” check box will prove to be highly effective to impatient Users.

The HELP button is included to give a clear explanation of what the app is all about and how it will fix the users problems and also how to use it.

Fig.7.11: Login function descriptions.



No search button/area because I intended to minimize the need to type by providing the bare essential that the user needs.

Providing a user-friendly home page with inviting colour that correspond with the system and logo of the app.

The use of vibrant images instead of normal button creates an appealing welcome to users hence it increases user on-board and engagement. Providing all the necessary functionalities for better satisfaction.

Fig.7.12: Functionality description overview.

# Section 8 – Interfaces Flow

Below is the prototype of the actual system and its applications, a step by step descriptive demonstration shall follow:

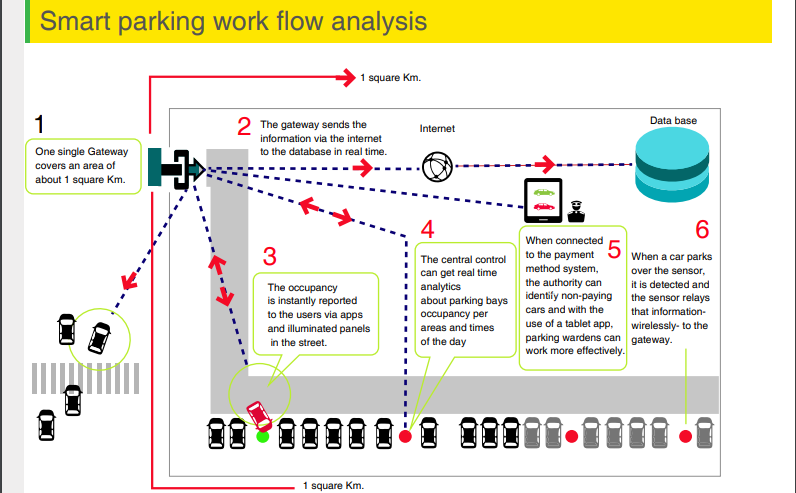


Fig.20: Work flow of the project

## 8.1 Actual Interface

* The main prototype of the Smart Parking System.

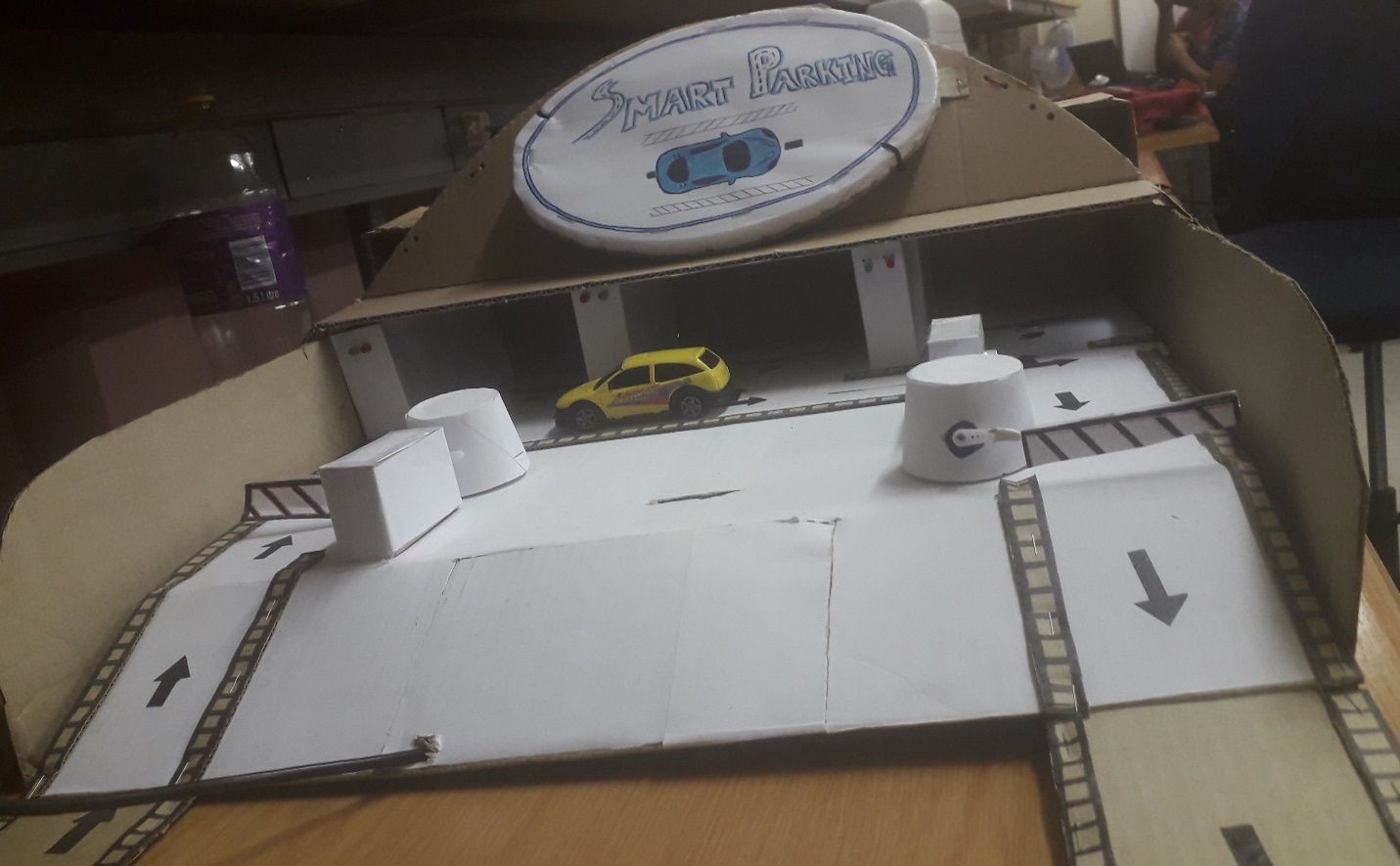


Fig.21: The Smart Parking System.

* Below is the entrance image where the car passes through the boom gate and the ultrasonic sensor.



Fig.22: Entrance

* Unoccupied parking slot is displayed with a green LED.



Fig.23: Vacant parking slot.

* Occupied parking slot is displayed with a red LED.



Fig.24: Occupied parking lot.

* The car exits the system at the boom gate and get detected by the sensor for security matters and system updating.



Fig.25: Car exiting the system.

* Back Wiring of the System.

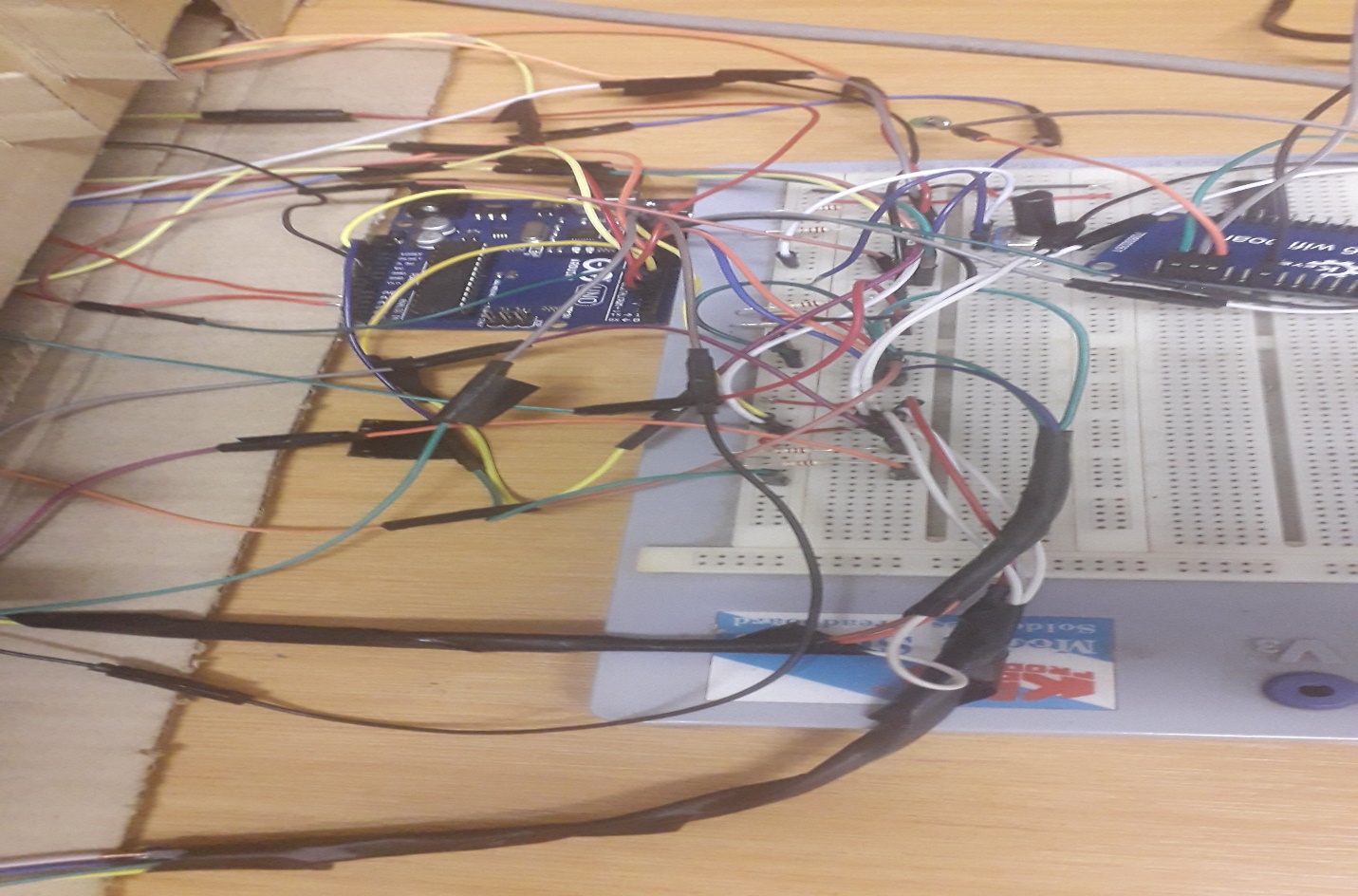
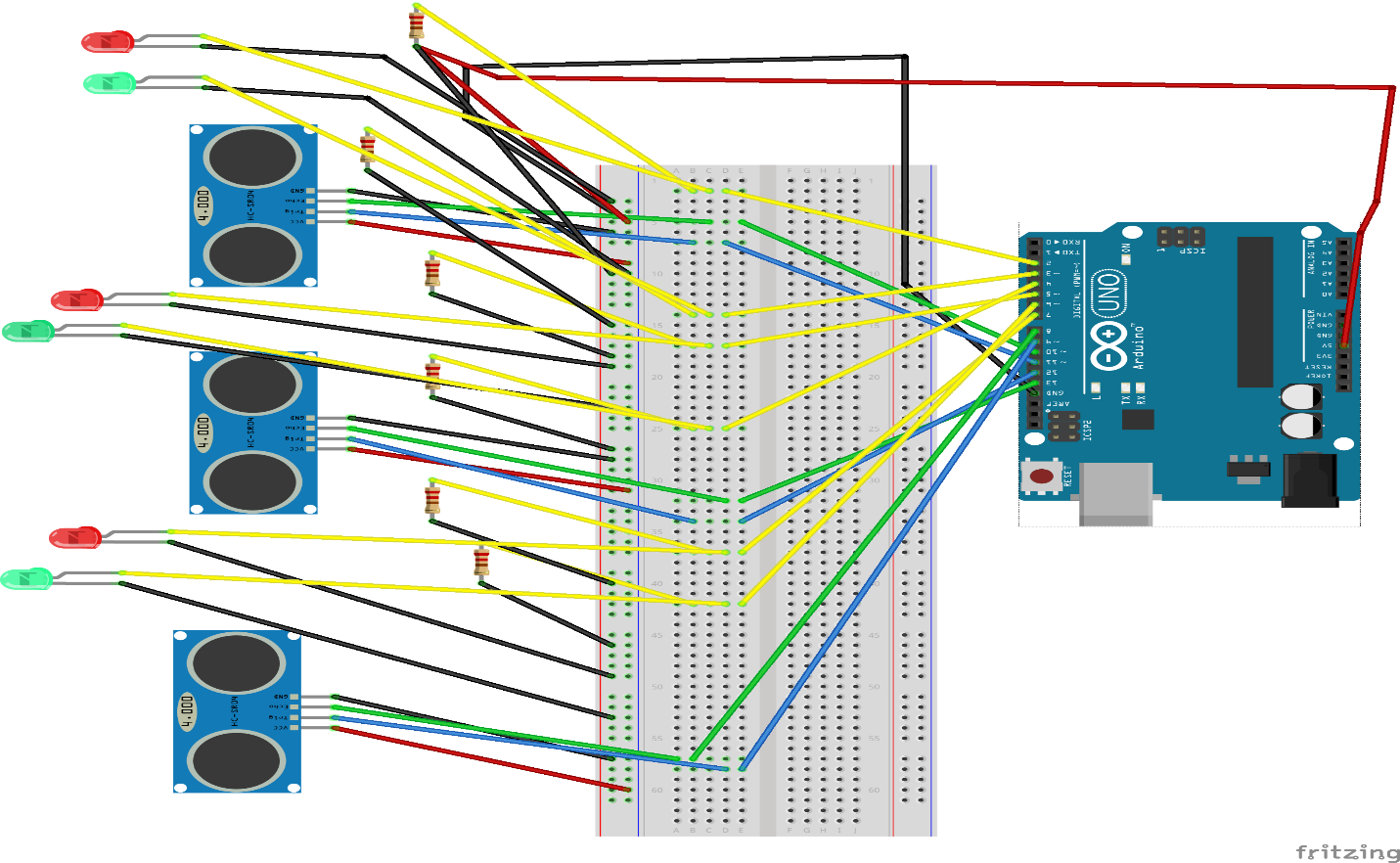


Fig.26: Back Wiring.

* A circuit diagram of the back wiring using Fritzing:



* A Schematic Circuit view to clearly display the connections of every component within SPS.

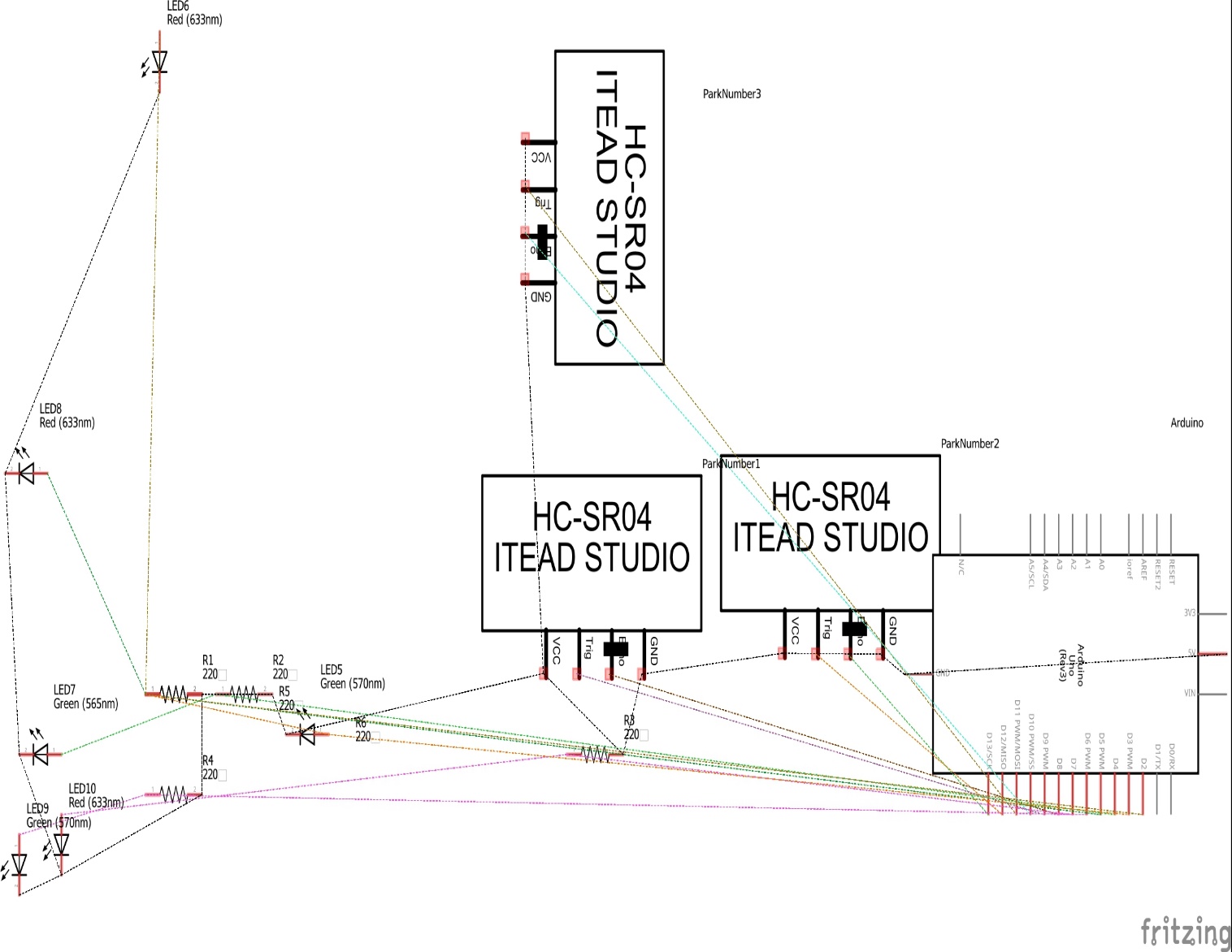


Fig.28: Circuit Wire Connections.

# Section 9 – Test Cases

A test case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. There are many types of testing that can be used to test the system. Software testing is to verify that all functions are working correctly and the overall system performance is well achieved and work properly. The purpose of testing is to reduce and minimize the risk or error of the system.

**Testing Techniques**

It is testing on a complete integrated system to evaluate the system compliance with specified requirements, in order to perform; there are some guidance and techniques that must be followed.

**Unit Testing**

In this testing we are testing a particular part and not to test the whole system. Unit testing also a method where units of source codes are tested to determine if they are fit to use. A unit is a smallest testable part of the application. Unit tests are written and ran by software developers to ensure that the code meets its design and behaves intended.

**Integration Testing**

The main goal here is to make sure that the interaction of two or more components produces results that satisfy functional requirements. This is a software development process where programmed units are combined and tested as a group in multiple ways. The purpose of integration testing is to verify functional performance, and reliability requirements placed on major items.

**System Testing**

This type of testing is very important when all modules combined from different people to verify and assess the system with its requirements. Thus, the system testing must be running in order to make sure that the logic is still strong and no error occurred. System testing is performed on the entire system in the context of system requirements specification. It tests not only design but also the behaviour and even the believed expectations of customer.

**User Acceptance Testing**

This is the final stage of testing before the system released or implementation. Acceptance testing is to demonstrate that the system can be ready to be used to the end user. It is tested with real data real/simulated environment. The acceptance testing is where products are being delivered to customer and the customer executes the acceptance test to whether the expectations of functionality fulfil their requirements.

**Test Plan**

**Login:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Test Data** | **Expected Result** | **Pass/Fail** |
| Validation | To check whether the validation is executed when required field is blank data | It displays required field information | Pass |
| Verify Username, password and access level | To check whether the Username and password and also access level is matched. | Login successful, display main page | Pass |
| Authentication | To check whether the system will perform authorization when user login | Display different main page for different access level | Pass |

**Reservation:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Test Data** | **Expected Result** | **Pass/Fail** |
| Make reservation | To check whether there is available parking lot or not | Display obtained results in the view history page | Pass |
| Add chosen slot to the history | To check whether booked parking slot is added to the list of occupied vehicles | Vehicle reservation added to database | Pass |
| Verify input data | To check whether input data has been verified by Firebase and prevent query injection | Display error message | Pass |
| View system update | To check whether occupied slot is the chosen one | Successful reservation | Pass |
|  |  |  |  |

**Admin Maintenance:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Test Data** | **Expected Result** | **Pass/Fail** |
| Validate input data | To check whether the validation is executed when the required fields are left blank or in wrong format. | Display error message | Pass |
| Edit database information | Check whether Admin can edit data | Successful message, display main page | Pass |
| Choose Departure time | Check whether the user can select departure time | Display departure time | Pass |
| Submit | To check whether departure time can be added successfully | Successful addition, message displayed on home page | Pass |

**Conclusion**

The ease of parking system is quite a challenge in modern days. Since the advent of industrialized cities, number of cars has been increasing and day by day people are facing bigger trouble while trying to manage their cars into a parking lot. This scenario of parking crisis gives rise to new solutions with the help of Internet of things (IOT) thus managing car parking systems. Our paper addresses the crisis of car parking across a remote city and comes out with an IoT based assistant mobile application system. The proposed project provides real time information of a car parking lot and is able to coordinate with the mobile application thus giving user the feasibility of booking a parking lot staying at a distance.

# Section 10 – References

1. Y. Geng and C. G. Cassandras, ‗‗A new ‗smart parking system based on optimal resource allocation and reservations, ‘‘in Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC), Oct. 2011, pp. 979–984.
2. Y. Geng and C. G. Cassandras, ‗‗New ‗smart parking system based on resource allocation and reservations, ‘‘IEEE Trans. Intell. Transp. Syst., vol. 14, no. 3, pp. 1129–1139, Sep. 2013.
3. X. Zhao, K. Zhao, and F. Hai, ‗‗An algorithm of parking planning for smart parking system ‘‘in Proc. 11th World Congr. Intell. Control Autom. (WCICA), 2014, pp. 4965–4969.
4. L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and R. Vergallo, ‗‗Integration of RFID and WSN technologies in a smart parking system, ‘‘in Proc. 22nd Int. Conf. Softw., Telecommun. Comput. Netw. (SoftCOM), 2014, pp. 104–110.
5. C. W. Hsu, M. H. Shih, H. Y. Huang, Y. C. Shiue, and S. C. Huang, ‗‗Verification of smart guiding system to search for parking space via DSRC communication, ‘‘in Proc. 12th Int. Conf. ITS Telecommun. (ITST), 2012, pp. 77–81.
6. R. E. Barone, T. Giuffrè, S. M. Siniscalchi, M. A. Morgano, and G. Tesoriere, ‗‗Architecture for parking management in smart cities, ‘‘IET Intell. Transp. Syst., vol. 8, no. 5, pp. 445–452, 2014.
7. C. Shiyao, W. Ming, L. Chen, and R. Na, ‗‗The research and implement of the intelligent parking reservation management system based on ZigBee technology, ‘‘in Proc. 6th Int. Conf.
8. C. Shiyao, W. Ming, L. Chen, and R. Na, ‗‗The research and implement of the intelligent parking reservation management system based on ZigBee technology, ‘‘in Proc. 6th Int. Conf.