

# CHAPTER 1

## INTRODUCTION

As the population in urban areas keep on growing there dependably exists a vulnerability as for time of arrival of bus at the bus stop. The bus networks in thick urban territories are frequently considered as mind boggling and hard to navigate. Additionally, infrequently transports are crossed out because of their breakdown strikes water signing on the streets or some other reasons, the workers are never educated about such cancellation of buses. The "Bus Ticketing and Tracking system" that is proposed intends to give a powerful and effective framework to enable facilities to track buses, know evaluated time of entry of any bus, capacity to book tickets ahead of time and to keep up a database for ticketing transaction by means of a flexible application on the android platform.

Real time vehicle tracking to follow routes and locations driven by a bus is done with the help of global positioning system (GPS). Users can likewise see bus routes on the map with their geographic and nongeographic qualities. GPS and Google maps are utilized for showing current areas of buses on the maps together with the related course data [8]. The RFID tags are utilized to recognize individuals and other data like payment, validity etc. Bidirectional sensor on the door is utilized to take the people count tally in the bus. Arduino UNO is a microcontroller to program with real time clock (RTC). A real time clock is a computer clock that monitors the present time. In light of IOT the travelers can get to this data of a transport in view of user source and destination through the android application [5].

The challenge here is to give all the required tracking and landing data and in addition the ticketing facilities to the traveler. In spite of the fact that these facilities can be given at a bus stop, it crushes the point of giving adaptability which is an essential piece of this task. Subsequently the android application is created as a User Interface. The principle reason for this application is to give a simple interface to get to all the different highlights of the system and to make full utilization of its functionality. A large portion of the facilities provided by this framework will wipe out the issues looked because of vulnerability of arrival of buses.

## **1.1 Scope**

The IoT system for bus passengers, has the ability to seamlessly interconnect bus passengers with the real-world public bus infrastructure. The navigation system relies on a distributed IoT system comprising an embedded bus computing smartphone system to detect the presence of passengers on buses, backend computing infrastructure and a mobile smartphone app for passengers provide continuous real-time navigation over the complete course of a bus journey.

## **1.2 Motivations**

In the way people move around their communities' public transportation systems is the main problem which play an increasingly important role. It is a very cost effective mode of transport. Due to cause of heavy traffic and roadwork etc., most of the buses are delayed in time. At the bus terminus people have to wait for long time without even knowing when the bus will arrive. Anybody who want to use the public transportation system, can't find the time of arrival of particular bus at the particular destination even at their homes and plan their departure from home accordingly. But due to unexpected delays in traffic congestion the bus arrival time cannot be guaranteed. Our main focus is to provide such a system to remote user which will reduce waiting time for bus and will provide him with all necessary details regarding the arrival/departure time of the bus, its real location and expected waiting time.

## CHAPTER 2

### LITERATURE SURVEY

Research on open transportation has generally centered around techniques to enhance the effectiveness of the physical transport framework. For example, service scheduling is considered as a critical issue for productive transport activity [7]. Be that as it may, Camacho et al. [3] contend that this point of view is simply inspired by the interests of transport administrators and misses the mark regarding the data needs of travellers in the advanced age. Rather than settling operational transport issues, they recommend to outline traveller driven data framework that can enhance the traveller's journeys.

A critical change of public transport data accessibility has been the improvement of portable transport applications. One Bus Away is the main portable application that brought estimates arrival times of transports on cell phones [6]. The creators indicated experimentally that pervasive access to expected waiting times essentially expanded fulfilment with public transport administrations. Then, various versatile transport applications have been produced for transport frameworks in numerous urban areas around the globe. Some of these applications use on the inherent sensors of cell phone gadgets to give personalization and setting mindfulness, example by utilizing GPS for proposing transport stops in the surroundings of a user. This gives knowledge into the crude context of the user, for example, his present area, however does not catch a more extensive thought of transport setting including the collocated physical transport system, example the bus vehicle and bus line on which a user is riding.

Moreover, current versatile transport applications regularly depend on the vehicle data that is distributed by transport administrators as open information, example in form of the Google's general transit feed specification (GTFS). This information incorporates a depiction of the vehicle network including routes, schedules, and landing times while subjective travel data is absent as a reason for educated travel decisions of transport users. Crowding on public transport framework is a measurement of subjective travel data that is known to have a big effect on travel satisfaction and cause an abnormal state of pressure and inconvenience [6]. To recognize crowd levels on public transport frameworks, some public transport offices embrace mechanized toll gathering framework which can give measurements about the quantity of travellers with advanced tickets [5]. In any case, because of considerable speculation costs

these frameworks are just conveyed in chosen urban communities and frequently need combination with explorer data frameworks. Without committed following foundation bolstered by transport specialists, crowdsourcing applications have been produced to secure extra continuous transport data. The possibility of these application is to permit transport clients act in a cooperative way and gather content information about true transport conditions experienced amid their journeys. For example, Tiramisu empowers transport traveller's to utilize their cell phones for physically sharing travel encounters, for example, regardless of whether adequate free seats are accessible on transports. In any case, to detect real time crowd data across large-scale public transport networks, completely programmed approaches are required which can work without steady, manual mediations of transport users. Current traveller data system offer help in singular travel context, example before an excursion is begun, while scanning for a nearby stop or while waiting for a bus. With a specific end goal to offer continuous direction for movement exercises, the possibility of route frameworks has been effectively connected in different portability situations, specifically for people on foot and auto drivers. The equivalent of a navigation system for public transport systems is currently missing as of now. There has been inquire about into cell phone based frameworks to identify the method of transport of voyaging users. These frameworks can separate transport ride movement from other transport modes by perceiving designs in sensor information acquired from the user's cell phones. However, a route benefit which continually goes with an public transport client and chooses how to best offer help from begin to end of his adventure is yet an open research challenge [3]. The WI Rover System (WRS) has been running on the buses since April 2010 giving Wi-Fi hotspots to the utilization of traveller's. On the review, 17,567 novel customer gadgets were associated with the WRS. This framework gives outline, administration systems for the system. This demonstrates individuals are towards the use of network facilities. The area construct administrations are finished focusing with respect to mobile guides, transport support, assistive innovation and so forth. The necessities openings in every zone are gathered, assessed and applications are spoken to in view of the request of open travel benefit. Those applications that beat the specialized and monetary difficulties are executed for public utilization.

The exploration in Dublin downtown area having two goals i.e., to analyze the feeling of anxiety and to decide solace and reliability level of public transport administrations. This overview demonstrated that pressure was high and solace and reliability level diminished

impressively. Thus to defeat this multinomial logic model was utilized to kill crowded and untrustworthy services [2]. Components influencing the reliability of urban transport administrations thinks about twelve urban areas regarding vitality use and ozone depleting substance outflow [GGE]. Every city audited had expanded vitality utilization, GGE, and private vehicle use. Here creator says that private vehicle utilization is expanding since individuals don't bargain on solace and unwavering quality of public transport framework.

Transportation Research Record says that there are a number of factors like traffic congestion, weather conditions etc. which affects the predetermined bus arrival schedule and hence results in increasing passenger waiting time. So there is a need for the system to predict the accurate bus arrival time and hence reduce the passenger waiting time.

Models for anticipating transport delays, says that travel period dependability is a noteworthy factor in public transport framework and led an investigation on estimating of reliability and standard travel times utilizing transport information from Chicago region focuses on nature of administration and reliability offered by open travel administrations. From information gathered for a situation investigation of automated vehicle location (AVL), it displays the Transit Capacity and Quality of Service Manual (TCQSM) strategy for level-of-benefit (LOS) estimation. Other than usability and effortlessness, once in a while the framework is conflicting since the framework does not consider defer sum, does not states the impacts of early flights. The Urban Bus Navigation (UBN) [1] framework has been incorporated into the metropolitan transport foundation in Madrid, Spain, and is accessible to general society as a free cell phone application that has been utilized by many transport riders. The technical components of the UBN system, give insights into user experiences from real-world field trials and share important lessons have learned in providing a connected transport application for bus passengers. Subjective criticism from genuine transport clients demonstrated diminished intellectual exertion for overseeing bus journeys, increased motivation of using bus transport and better accessibility of travel information.

## **2.1 Problem Statement**

The public transport like bus, car etc. networks are dense, complex and difficult to navigate. In contrast to private modes of transport the existing modes of transport offer only a low level of comfort and convenience. The bus journey lacks the contextual information about the arrival time, departure time, crowd alert, and micro information about the progress of the journey. The public mode of transport does not have the novel information about the crowd or traffic jam as in private mode of transport example: Ola cabs. Time is the precious thing but the fine information about the arrival, delay and approach time of the bus is not provided in the existing public transport system.

## **2.2 Gaps in Existing System**

The existing public bus transport systems have the capacity to absorb large masses of urban travelers, their public image often suffers from a negative perception. From a passenger's point of view, bus networks in dense urban areas are often considered as complex and difficult to navigate. In contrast to private modes of transport, traveling on busses offers only a low level of comfort and convenience. Bus journeys lack a sense of personal control and ownership that is valued by car users.

### **Disadvantages of existing public transport system:**

- Lacks real-time passenger information
- Lacks Automatic Vehicle location
- Lacks service alerts
- Lacks crowd aware and route recommendation system
- Lacks bus arrival notification systems
- Lacks Micro-navigation system

## **2.3 Proposed System**

A key challenge for rapidly growing cities of today is to provide effective public transport services to satisfy the increasing demands for urban mobility. Toward this goal, the Internet of Things (IoT) connected bus ticket generator has great potential to overcome existing deficiencies of public transport systems given its ability to embed smart technology into real-life urban contexts. an IoT enabled system for urban bus riders provides three novel

information services for bus users: 1) micro-navigation and 2) crowd-aware route recommendation.3) Bus travel time estimation. We have presented a navigation system for bus passengers that has the ability to seamlessly interconnect bus passengers with the real-world public bus infrastructure. This relies on a distributed IoT system comprising an embedded bus computing smartphone system to detect the presence of passengers on buses, backend computing infrastructure and a mobile smartphone app for passengers provide continuous real-time navigation over the complete course of a bus journey. Here there is also a web application for admin in the head office for managing the buses, bus routes and non app users.

## CHAPTER 3

# SYSTEM REQUIREMENT SPECIFICATION

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. It is the first and foremost work of a software developer to study the system to be developed and specify the user requirements before going to the designing part.

It includes a set of use cases that describes all the interactions the users will have with software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains the non-functional (or supplementary) requirements. Nonfunctional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, Quality standards, or design constraints).

### **The basic requirements are:**

- User should register in the app by creating an account and the non app user should register in the head office with the admin.
- Login facility.
- The source location is determined through GPS; user needs to select his desired destination.
- The server should provide the buses details in real time, which are arriving in next 15-20 minutes.
- The retrieved details may contain single bus info or multiple buses to reach the destination. Once these details are displayed to the passenger, he/she can choose among them which suits them better.
- When one of the option is selected: server should display current bus location on map, current crowd information, bus arrival time details. In case of multiple buses, it should provide time gap between the buses, and total time to reach the destination.



- Once the passenger selects an option, E-payment from the wallet is to be done. If the passenger cancels the ticket the deducted amount should be refunded.
- The network enabled ticket machine has its own unique identification and fixed routes details about bus stops.
- The network enabled ticket machine is connected to the server in real time sharing useful information to the driver. It should have retrieved the number of passengers booked the tickets and display into the driver screen, he can cross verify the total count of passengers and sensor count to find non ticket taken passengers.
- Website is provided at the head office for the admin.
- RFID cards are provided to the passengers at the head office which will be read by the RFID reader at the entrance of the bus. Driver will get to know whether the passenger has done the payment or not when the card is swiped by the passenger.
- The passengers who don't have smart phones and have registered at the head office and have recharged the RFID card can do on spot ticket bookings with the driver's app, the amount will be deducted from the passenger's RFID card.
- The passengers can register any complaints about the problems faced by them while travelling in respective buses, any complaints with the bus facility or feedback about the proposed system which will be reported at the website in the head office.

### **3.1 Functional Requirements**

The proposed system must provide access information of all the buses in the respective area along with the number of seats available and the current location of the bus. The system requires the passengers to have a RFID card issued at the head office, an account (not mandatory) and passenger's login with their respective login names and passwords. The proposed system must allow the passengers do the ticket booking and payment online. The proposed system should allow the website admin to access the passenger's details. The system also should display the list of the all the buses arriving in 15-20 minutes and must provide notification to passengers regarding their destination bus stop arrival in the app. The proposed

system also should display the occupied and unoccupied seats to the driver on the screen. System should decrement the number of seats available in each bus whenever a passenger gets out of the bus.

### 3.2 Non-Functional Requirements

- **Performance Requirements:** The proposed system is used for online ticket booking and tracking the bus in real time to make the bus system easy and comfortable. The passenger's database in the website help the admin in managing the passenger's information. Therefore, it is expected that the database would perform functionally all the requirements that are specified.
- **Safety Requirements:** The database containing passengers, drivers as well as bus information may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup.
- **Security Requirements:** We are going to develop a secured database. Depending upon the category of user the access rights are decided. If the user is an administrator then he/she can modify the data, append etc. All other users only have the rights to retrieve the information from database. The hardware will check for the details and based on the sensors the user will be allowed into the system. It secures the system from unauthorized entry.
- **Availability:** The RFID system must be functional all the time. The system must be up and running whenever the passenger arrives into the bus and gets down of the bus. Website should be always visible whenever the admin tries to extract the stored data.
- **Reliability:** The system must be reliable or dependable. The system must provide accurate and correct information to the passengers, driver and the head office so that they trust the system.
- **Scalability:** The system must be elastic and must be able to handle more number of requests. As the number of passengers and buses increases the system must be able to support their ever increasing needs.

### 3.3 Hardware Requirement

- Arduino Uno ATmega328 microcontroller board
- RFID reader RFID-RC522 (MF-RC522) 13.56MHz
- Bluetooth serial module HC-05
- Ultrasonic sensor module HC-SR04
- Power bank 20000 mAh
- RFID cards
- Arduino Uno cable
- Android tablet for Driver console

### 3.4 Software Requirement

- Operating System : Windows 8 and other higher versions.
- IDE : Notepad++, Arduino, Android SDK & Studio, WAMP  
Apache Server.
- Database : MySQL.
- Languages : HTML, CSS, PHP, JS, JAVA, XML, Arduino C
- API : Google Maps geocoding API, Google places API web service, Google Places API for Android, Google Maps JavaScript API, Google Maps Geolocation API, Google Maps Directions API, Google Maps Distance Matrix API, Google Maps Android API

### 3.5 User Requirements

The system is provided with the intension to provide easy to use simple system so that no cumbersome and elaborate training for operation is required. The system has 4 types of users.

**The system should allow them to do the following:**

#### **Admin**

- Adding / removing buses
- Adding / removing bus routes
- Feedback / complaints view
- Users card information
- Update bus, route and card Info

## **App User**

- Register
- Login
- Search buses
- Track buses
- Book tickets
- Wallet
- Payment details
- Feedback/complaints
- Logout

## **Non-App User**

- Register
- Buy card
- Travel
- Update card

## **Bus Driver**

- Login
- passenger counting
- View tickets
- Update data
- Logout

## **CHAPTER 4**

### **SYSTEM ANALYSIS**

#### **4.1 Social Analysis**

This is the study that checks the acceptance of the system by the user. The users will be able use the proposed system easily. The users will be able to easily navigate throughout the system and be able to access all its services as easily as possible. The system should have a user friendly interface that provides a basic and easy to understand version of the system functionalities. Users should be able to trust the system and should be able to use the system without worrying about anything.

#### **4.2 Performance Analysis**

The system provides an easy to use and understand interface provides three main information services for bus users: 1) micro-navigation and 2) crowd-aware route recommendation. 3) Bus travel time estimation. The navigation system relies on a distributed IoT system comprising an embedded bus computing smartphone system to detect the presence of passengers on buses, backend computing infrastructure and a mobile smartphone app for passengers provide continuous real-time navigation over the complete course of a bus journey.

#### **4.3 Economic Analysis**

The study provides us an overview of the expenses that incur in setting up and running the system. The developed system is well within the budget as most of the technologies used are freely available, software's used are free or of low cost and involves very less hardware components.

## CHAPTER 5

# SYSTEM DESIGN

System design implies a systematic approach to the design of a system. It may take a bottom up approach or a top down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout the framework. System design then overlaps with system analysis, system engineering and system architecture.

### 5.1 General System Architecture for IoT System

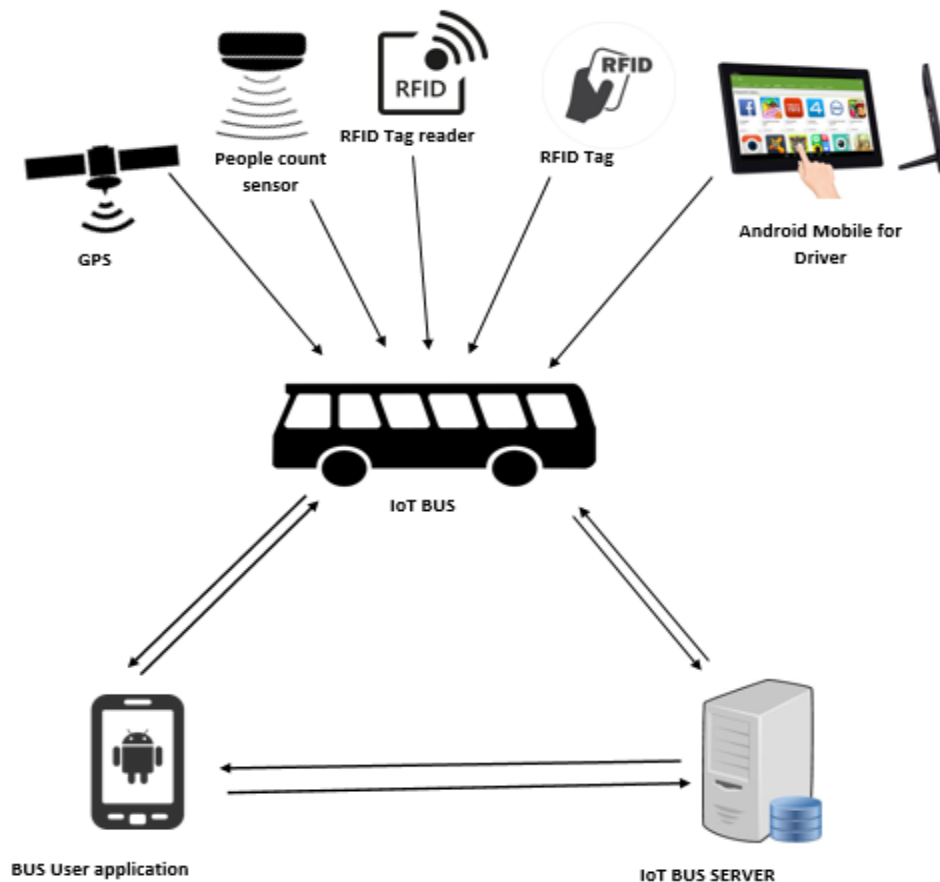


Figure 5.1 General System Architecture for IoT System

A IoT system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation

of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

## 5.2 System Architecture for User

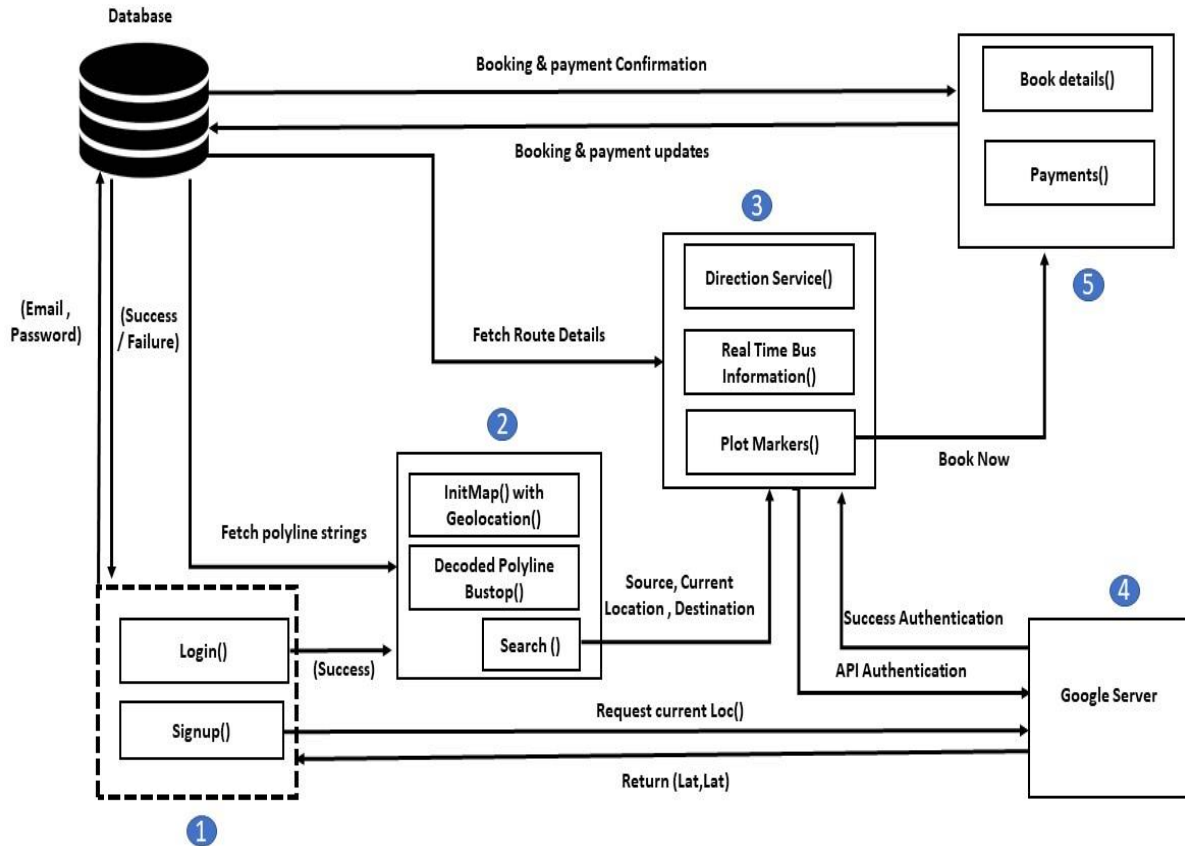


Figure 5.2: Block diagram of user module

The above figure depicts the real time implementation of the location module of the user. This architecture has a login page that fetches the current location of the login user and this is developed using google geolocation API. The fetched location (lat,lng) will be stored in the database and the fetched when a call is sent from a marker script to show on the map.

## 5.3 System Architecture for Admin

The figure 5.2 depicts the real-time implementation of the admin module. The admin has to login onto the login page successfully. Once the admin has logged in successfully he/she has to build polyline, assign routes and register the buses. He/she has to plot the markers for the bus stops and this data will be authenticated with the Google server.

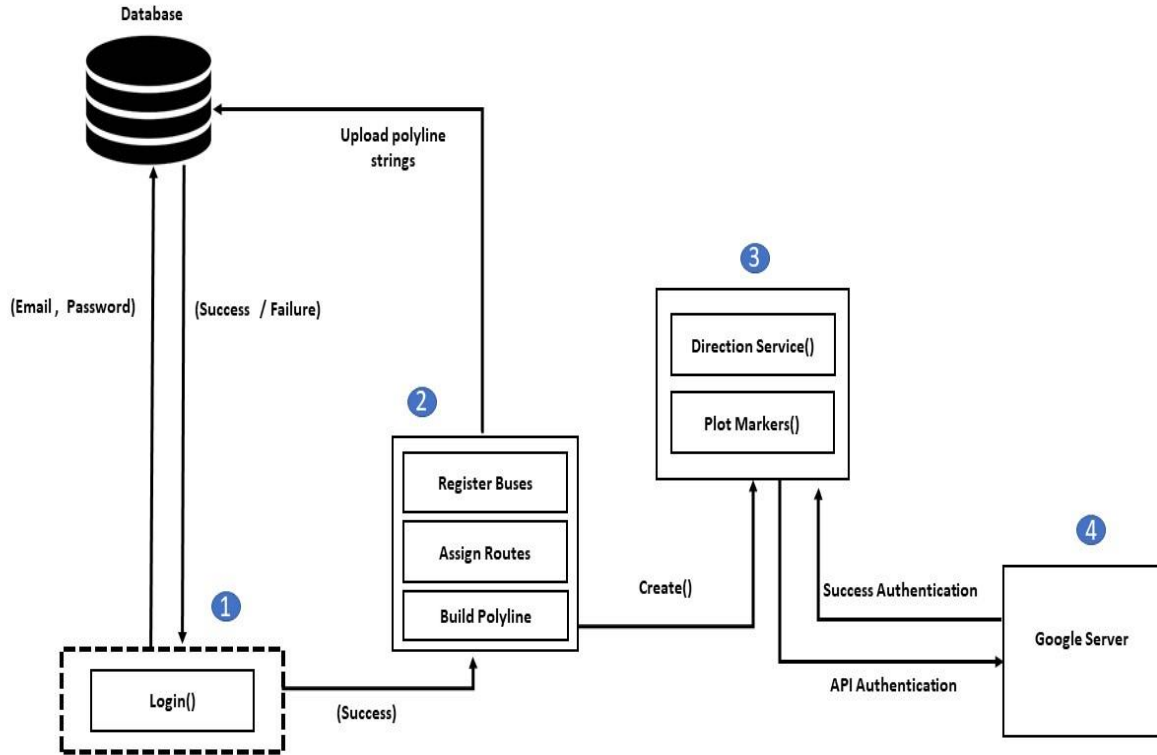


Figure 5.3: Block diagram of admin module

## 5.4 Data Flow Diagram

This data flow diagram (DFD) is a graphical representation of the flow of data through information system. With a flow diagram, users will be able to visualize how the system will operate, what the system will accomplish and how the system will be implemented. A DFD shows the flow of data values from their sources to their destinations and the transformations of data by the processes. A DFD also known as bubble char has the purpose of clarifying the system requirements and identifying major transformations that will become programs in system design. Hence it is the starting point of the design phase that functionally decomposes the requirement specifications down to the lowest level of detail. The bubble represents the data transformation and the line represents the data flow in the system. The DFD consists of the processes that transform data, data flows that moved data, actor objects that produce and consume data and data store objects that store data.



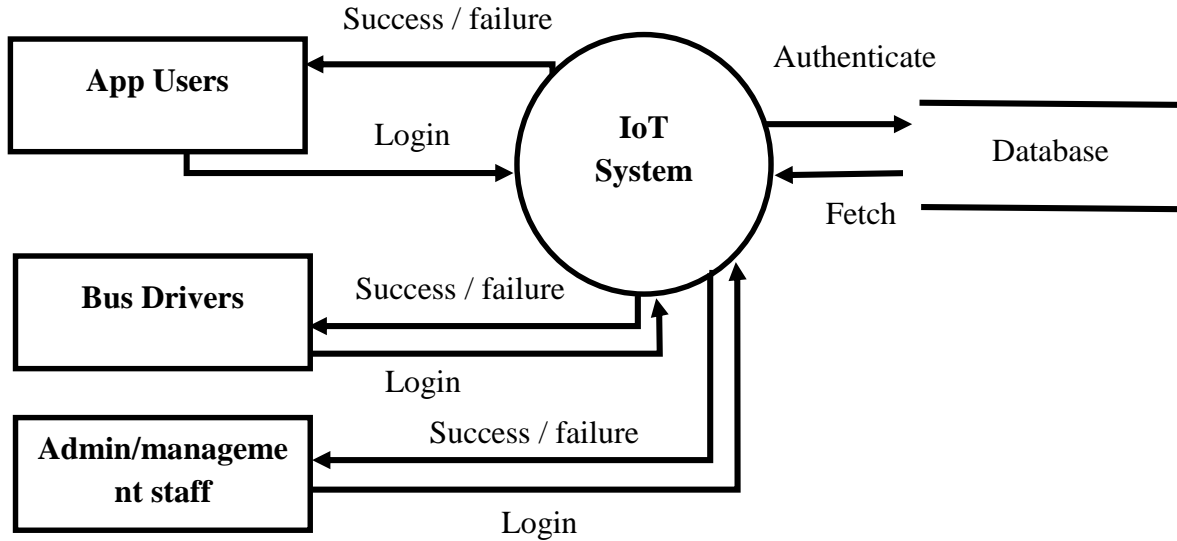


Figure 5.4 DFD for login

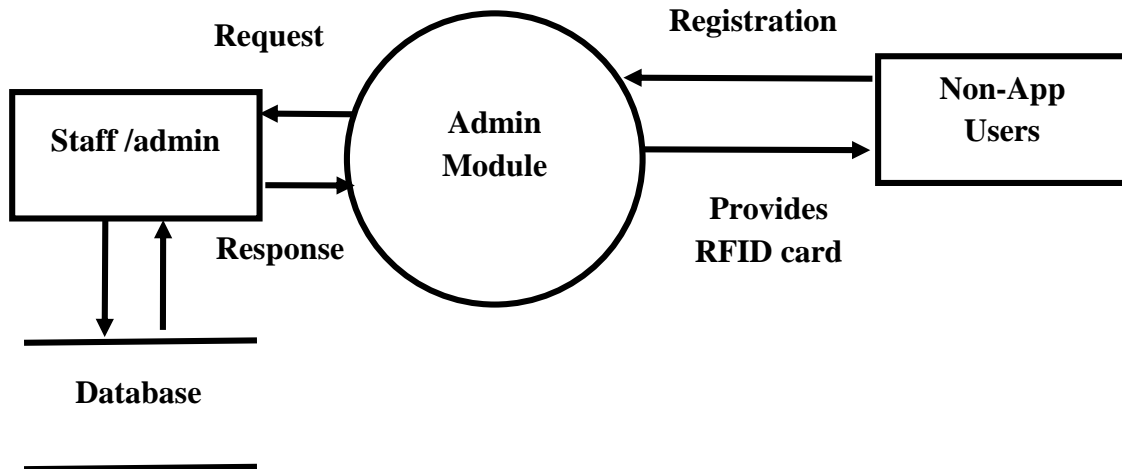


Figure 5.5 DFD for Non-App Users Registration

## 5.5 Use Case Diagram

The use case diagrams in figure 5.6 and figure 5.7 are the representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram portrays the different types of users of a system and the various ways that they interact with the system. In software and systems engineering, a use case is a list of steps that define the interactions between a user and a system to achieve a goal. The actor can be a human or an external system.

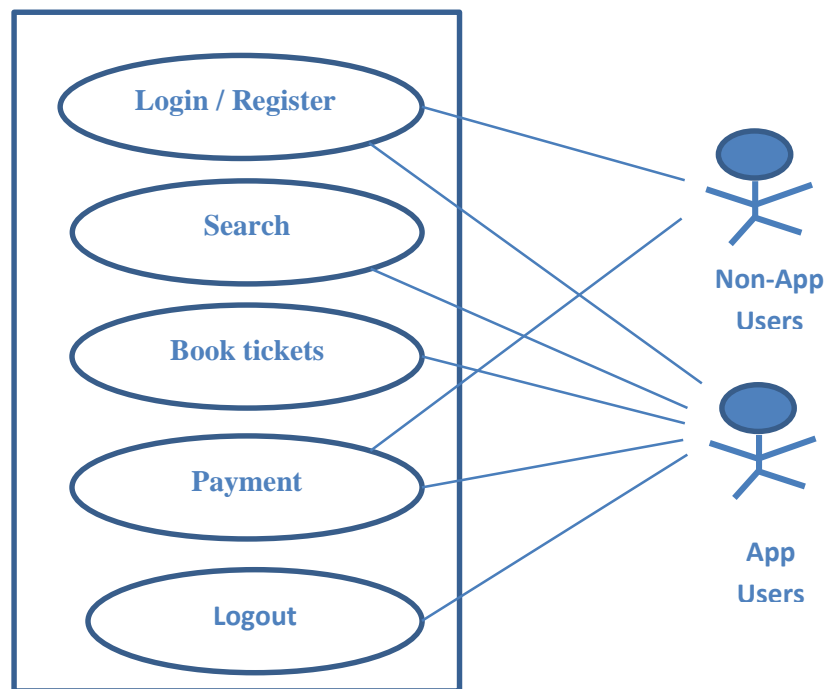


Figure 5.6 use case diagram for passengers

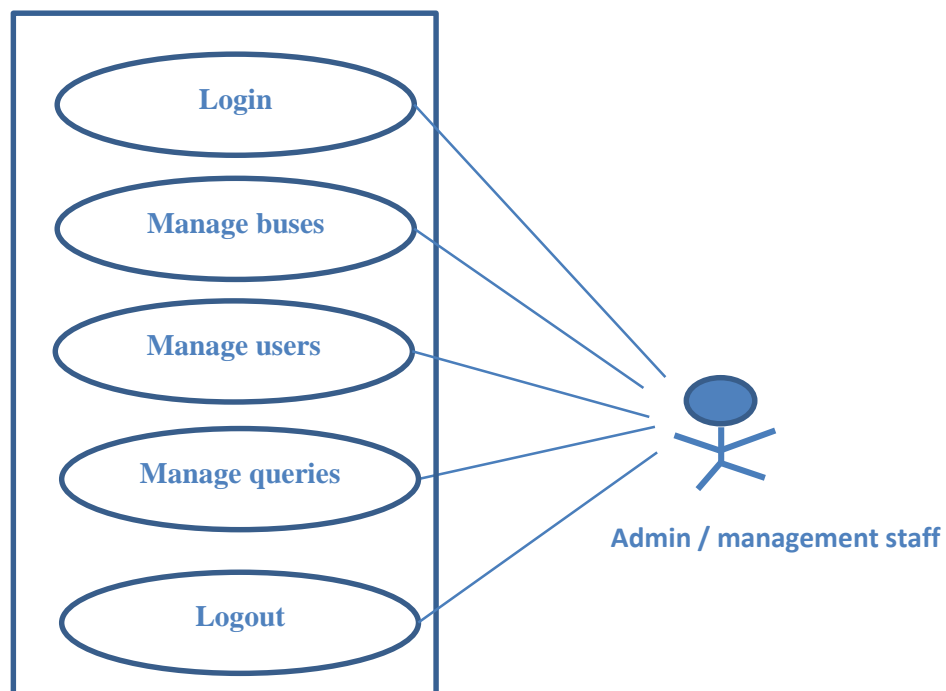


Figure 5.7 use case diagram for management staff/admin

## 5.6 Sequence Diagram

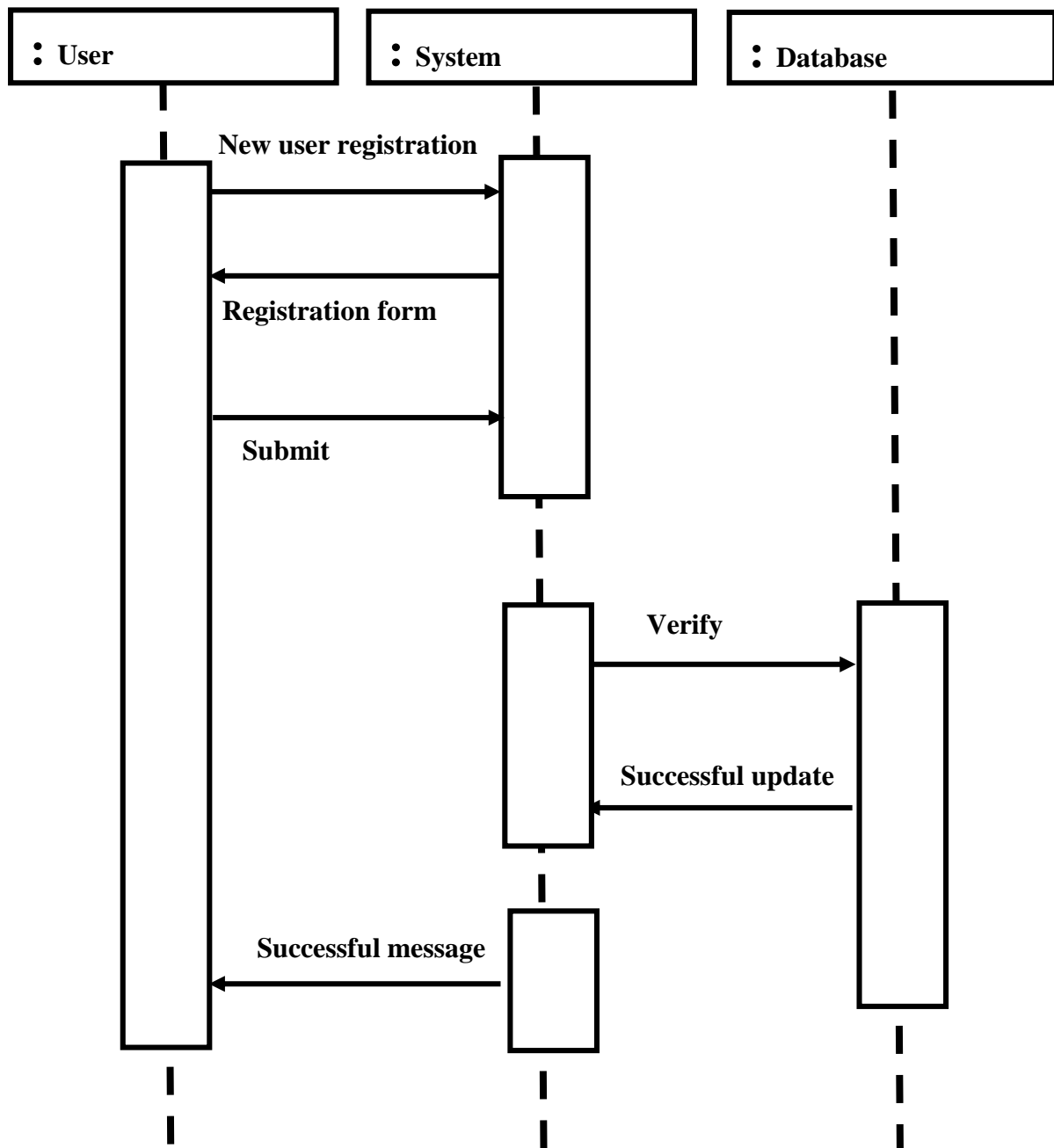


Figure 5.8 sequence diagram for passenger's registration

The above sequence diagram is for registration of passengers. The user has to fill the details in the registration form and submit. The data will be verified and successfully updated.

## 5.7 Activity Diagram

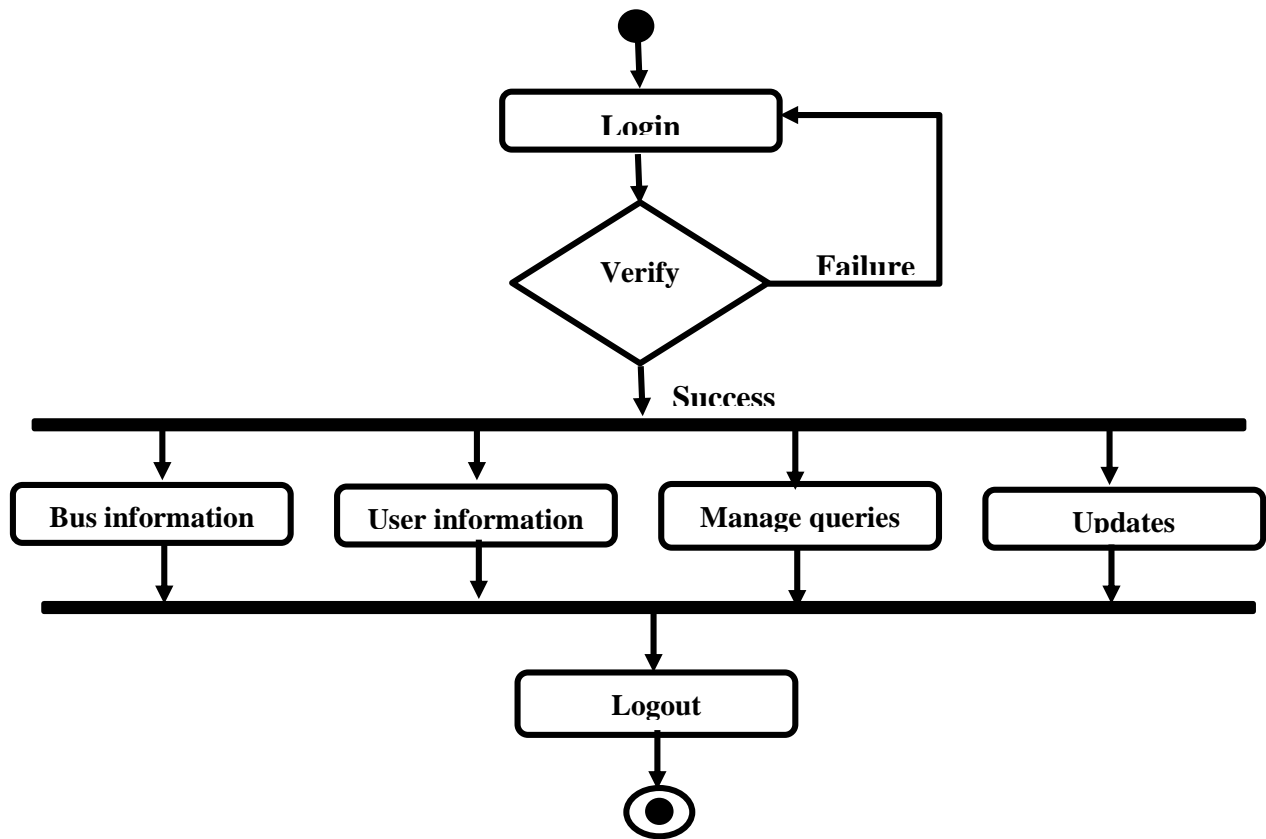


Figure 5.9 activity diagram management staff /admin

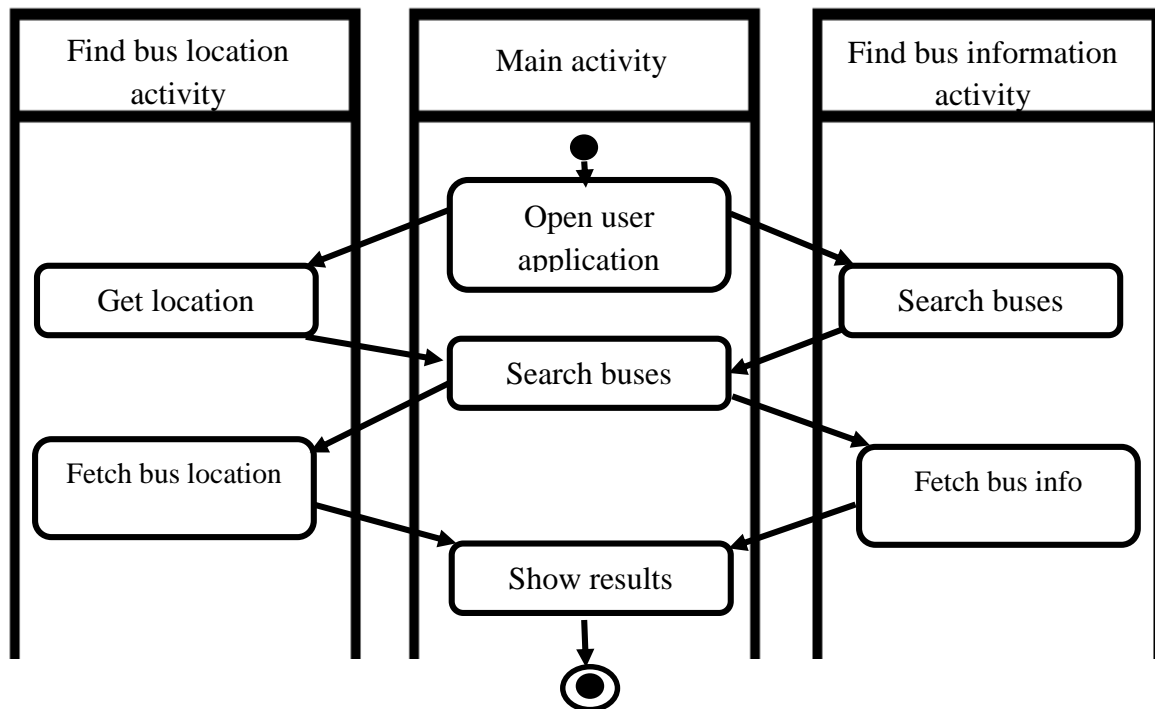
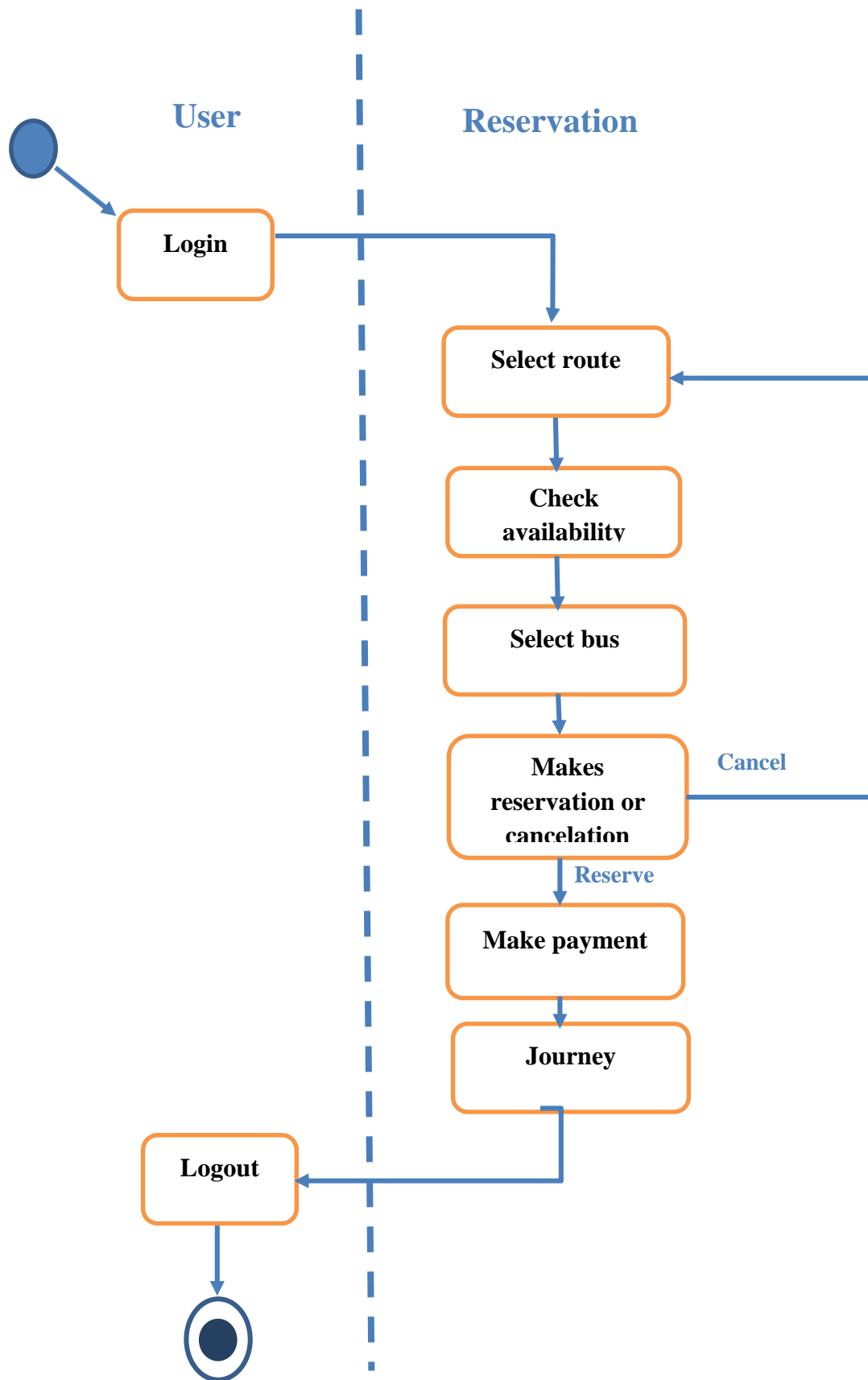


Figure 5.10 activity diagram for searching bus location and information

**Figure 5.11 Application user registration activity diagram**

## 5.8 Entity Relationship Diagram

Entity Relationship (ER) diagram is a graphical representation of entities and their relationships to each other which is used in computing in regard to the organization of data within databases or information systems. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes.

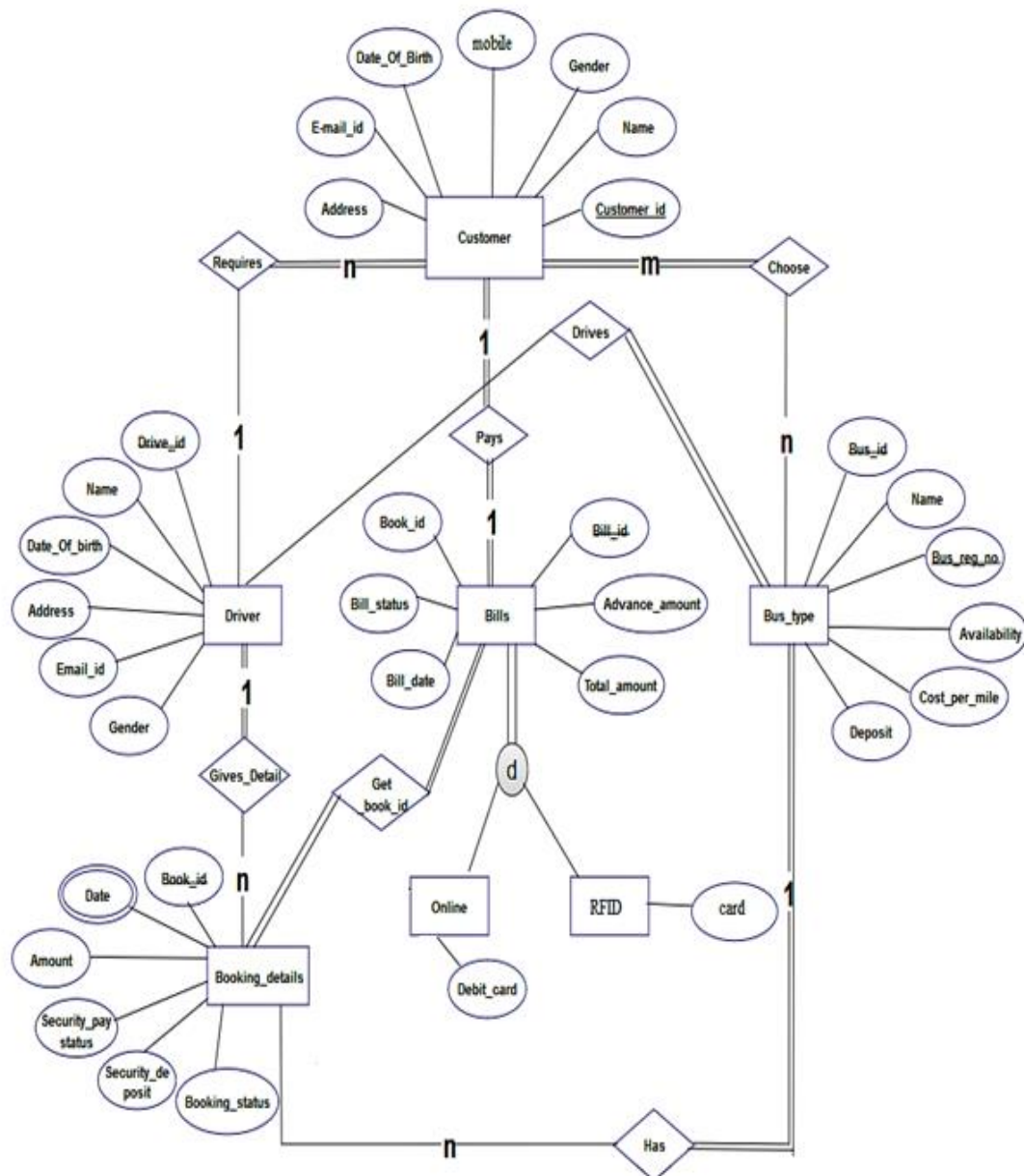


Figure 5.12 ER- diagram for IoT system

## CHAPTER 6

# SYSTEM IMPLEMENTATION

Implementation refers to conversions of a system design to an operation. An implementation plan is made before starting an actual implementation of the system. The new system may be totally new, replacing an existing manual or automated system or it may be a major modification to an existing system. It involves user training, system testing and successful running of the developed system. It is the stage where theoretical design is turned into a working system. Many preparations are involved before and during the implementation of the proposed system.

### 6.1 Google map API

It is to display the geographic location of a user or device on a Google map, using browser's HTML5 Geolocation feature along with the Google Maps JavaScript API. It refers to the identification of the geographic location of a user or computing device via a variety of data collection mechanisms. Typically, most services use network routing addresses or internal GPS devices to determine this location. It is a device-specific API which means that browsers or devices must support geolocation in order to use it through web applications. The various API used here are

#### 6.1.1 Directions API

The Google Maps Directions API is a service that calculates directions between locations using an HTTP request. A Google Maps Directions API request takes the following form:

**<https://maps.googleapis.com/maps/api/directions/outputFormat?parameters>**

where output Format may be either of the following values:

- json (recommended) indicates output in JavaScript Object Notation (JSON)
- xml indicates output as XML

### **6.1.2 Distance Matrix API**

The Google Maps Distance Matrix API is a service that provides travel distance and time for a matrix of origins and destinations.

A Google Maps Distance Matrix API request takes the following form:

**<https://maps.googleapis.com/maps/api/distancematrix/outputFormat?parameters>**

where output Format may be either of the following values:

- json (recommended), indicates output in JavaScript Object Notation (JSON); or
- xml, indicates output as XML.

### **6.1.3 Geocoding API**

Geocoding is the process of converting addresses (like "1600 Amphitheatre Parkway, Mountain View, CA") into geographic coordinates (like latitude 37.423021 and longitude -122.083739), which you can use to place markers on a map, or position the map.

A Google Maps Geocoding API request takes the following form:

**<https://maps.googleapis.com/maps/api/geocode/outputFormat?parameters>**

where output Format may be either of the following values:

- json (recommended) indicates output in JavaScript Object Notation (JSON); or
- xml indicates output in XML

To access the Google Maps Geocoding API over HTTP, use:

**<http://maps.googleapis.com/maps/api/geocode/outputFormat?parameters>**

### **6.1.4 Geolocation API**

The Google Maps Geolocation API returns a location and accuracy radius based on information about cell towers and Wi-Fi nodes that the mobile client can detect.

Geolocation requests are sent using POST to the following URL:

**[https://www.googleapis.com/geolocation/v1/geolocate?key=YOUR\\_API\\_KEY](https://www.googleapis.com/geolocation/v1/geolocate?key=YOUR_API_KEY)**



A key must be specified in the request, included as the value of a key parameter. A key is application's API key. This key identifies the application for purposes of quota management.

### **6.1.5 Roads API**

The Google Maps Roads API identifies the roads a vehicle was traveling along and provides additional metadata about those roads, such as speed limits.

## **6.2 Adding a Google Maps JavaScript API V3**

It shows to add a simple Google map with a marker to a web page which suits the beginner or intermediate knowledge of HTML and CSS, and a little knowledge of JavaScript. There are three steps to creating a Google map with a marker on the web page:

1. Create a HTML page
2. Add a map with a marker
3. Add a API key

### **6.2.1 Geolocation requests**

Geolocation requests are sent using POST to the following URL. You must specify a key in your request, included as the value of a key parameter. A key is your application's API key. This key identifies your application for purposes of quota management. Learn how to get a key.

**`https://www.googleapis.com/geolocation/v1/geolocate?key=YOUR_API_KEY`**

### **6.2.2 Load the API**

Load or include the Google Maps API using the script tag as shown below:

```
<script src ="http://maps.googleapis.com/maps/api/js">  
  
</script>
```

### **6.2.3 Navigation access**

```
navigator.geolocation.getCurrentPosition(function(position) {
```

```
var pos = {  
  
    lat: position.coords.latitude,  
  
    lng: position.coords.longitude  
  
};
```

the above code is will access the current location of the user and sends the position to the map to position variable and sets the marker to the map.

### 6.2.4 Map Options

Before initializing the map, we have to create a mapOptions object (it is created just like a literal) and set values for map initialization variables. A map has three main options, namely, centre, zoom, and maptypeid.

- **centre** – Under this property, we have to specify the location where we want to centre the map. To pass the location, we have to create a LatLng object by passing the latitude and longitudes of the required location to the constructor.
- **zoom** – Under this property, we have to specify the zoom level of the map.
- **maptypeid** – Under this property, we have to specify the type of the map we want. Following are the types of maps provided by Google –

Within a function, say, loadMap(), create the mapOptions object and set the required values for center, zoom, and mapTypeId as shown below.

```
function initMap() {  
  
    var map = new google.maps.Map(document.getElementById('map'), {  
  
        zoom: 4,  
  
        center: {lat: -33, lng: 151},  
  
        mapTypeControl: true,  
  
        mapTypeControlOptions: {  
  
            style: google.maps.MapTypeControlStyle.DROPDOWN_MENU,
```

```
mapTypeIds: ['roadmap', 'terrain']  
  
}  
  
});  
  
}
```

## 6.3 Interactive Polyline Encoder Utility algorithm

Polylines in Google Maps are formed as a set of latitude/longitude pairs. In addition, for each vertex (location) in an encoded polyline, a level can be specified indicating that the location should appear on that level and any level higher (i.e. any decrease in zoom.). If a location does not appear on a given level, then the line will go from the last visible location to the next visible location.

### 6.3.1 Encoded Polylines

The encoding process converts a binary value into a series of character codes for ASCII characters using the familiar base64 encoding scheme: to ensure proper display of these characters, encoded values are summed with 63 (the ASCII character '?') before converting them into ASCII. The algorithm also checks for additional character codes for a given point by checking the least significant bit of each byte group; if this bit is set to 1, the point is not yet fully formed and additional data must follow.

Additionally, to conserve space, **points only include the offset from the previous point** (except of course for the first point). All points are encoded in Base64 as signed integers, as latitudes and longitudes are signed values. The encoding format within a polyline needs to represent two coordinates representing latitude and longitude to a reasonable precision. Given a maximum longitude of +/- 180 degrees to a precision of 5 decimal places (180.00000 to -180.00000), this results in the need for a 32 bit signed binary integer value.

Note that the backslash is interpreted as an escape character within string literals. Any output of this utility should convert backslash characters to double-backslashes within string literals.

**The steps for encoding such a signed value are specified below.**

1. Take the initial signed value:

**-179.9832104**

2. Take the decimal value and multiply it by 1e5, rounding the result:  
**-17998321**
3. Convert the decimal value to binary. Note that a negative value must be calculated using its two's complement by inverting the binary value and adding one to the result:  
**00000001 00010010 10100001 11110001**  
**11111110 11101101 01011110 00001110**  
**11111110 11101101 01011110 00001111**
4. Left-shift the binary value one bit:  
**11111101 11011010 10111100 00011110**
5. If the original decimal value is negative, invert this encoding:  
**00000010 00100101 01000011 11100001**
6. Break the binary value out into 5-bit chunks (starting from the right hand side):  
**00001 00010 01010 10000 11111 00001**
7. Place the 5-bit chunks into reverse order:  
**00001 11111 10000 01010 00010 00001**
8. OR each value with 0x20 if another bit chunk follows:  
**100001 111111 110000 101010 100010 000001**
9. Convert each value to decimal:  
**33 63 48 42 34 1**
10. Add 63 to each value:  
**96 126 111 105 97 64**
11. Convert each value to its ASCII equivalent:  
**`~oia@**

### 6.3.2 Decoded Polyline

The decoding process converts a series of character codes for ASCII characters into a binary value using the familiar base64 decoding scheme.

The code for decoding such ASCII characters are

```
var points=[ ]
var index = 0, len = encoded.length;
var lat = 0, lng = 0;
while (index < len)
{
```

```
var b, shift = 0, result = 0;
do {
    b = encoded.charAt(index++).charCodeAt(0) - 63; //finds ascii
    //and subtract it by 63
    result |= (b & 0x1f) << shift;
    shift += 5;
} while (b >= 0x20);
var dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
lat += dlat;
shift = 0;
result = 0;
do {
    b = encoded.charAt(index++).charCodeAt(0) - 63;
    result |= (b & 0x1f) << shift;
    shift += 5;
} while (b >= 0x20);
var dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
lng += dlng;
points.push({latitude:( lat / 1E5),longitude:( lng / 1E5)})
}
return points
```

## 6.4 Google Maps distance and duration

The Distance API is a service that provides travel distance and time for a matrix of origins and destinations. The Distance API returns information based on the recommended route between start and end points, as calculated by the Google Maps API, and consists of rows containing duration and distance values for each pair.

**The code for distance and duration is as follows:**

```
var service = new google.maps.DistanceMatrixService();

service.getDistanceMatrix({

    origins: [source],
```

```
        destinations: [destination],

travelMode: google.maps.TravelMode.DRIVING,

unitSystem: google.maps.UnitSystem.METRIC,

avoidHighways: false,

avoidTolls: false

    },

function (response, status)

{

    if (status == google.maps.DistanceMatrixStatus.OK&&response.rows[0].elements[0].status !=
    "ZERO_RESULTS")

    {

        var distance = response.rows[0].elements[0].distance.text;

        var duration = response.rows[0].elements[0].duration.text;

        cost = parseInt(distance) * 1.20;

        cost1 = parseInt(distance) * 1.20;

        vardvDistance = document.getElementById("dvDistance");

        dvDistance.innerHTML = "";

        dvDistance.innerHTML += "Distance: " + distance + "<br />";

        dvDistance.innerHTML += "Duration: " + duration;

        dvDistance.innerHTML += "<br />Cost : " + cost1 + "Rs..<br />";

        document.getElementById('cost').value = cost;

    }

    Else

    {

        alert("Unable to find the distance via road.");

    }

});
```

## 6.5 Functions for direction service and rendering map

The directions (using a variety of methods of transportation) can be calculated by using the DirectionService object. This object communicates with the Google Maps API Directions Service which receives direction requests and returns an efficient path. Travel time is the primary factor which is optimized, but other factors such as distance, number of turns and many more may be taken into account. The directions results can be handled or use the DirectionRenderer object to render these results.

**The code for direction service and rendering the maps is as follows:**

```
var directionsService = new google.maps.DirectionsService();

var directionsDisplay = new google.maps.DirectionsRenderer({

    'map': map,

    'preserveViewport': true,

    'draggable': true

});

directionsDisplay.setMap(map);

var start = document.getElementById('txtstart').value;

var end = document.getElementById('txtDestination').value;

var request = {

    origin : start,

    destination : end,

    travelMode: google.maps.DirectionsTravelMode.DRIVING,

    unitSystem: google.maps.UnitSystem.METRIC

};
```

```
directionsService.route(request, function(response, status) {  
  
    if (status == google.maps.DirectionsStatus.OK) {  
  
        directionsDisplay.setDirections(response);  
  
    }  
  
});
```

## 6.6 Function for auto complete places search

Autocomplete is a feature of the Places library in the Maps JavaScript API. You can use autocomplete to give your applications the type-ahead-search behavior of the Google Maps search field. When a user starts typing an address, autocomplete will fill in the rest.

**The code for auto complete places search is as follows:**

```
<script>  
    function initAutocomplete() {  
        var map = new google.maps.Map(document.getElementById('map'), {  
  
        });  
  
        // Create the search box and link it to the UI element.  
        var input = document.getElementById('txtAddress');  
        varsearchBox = new google.maps.places.SearchBox(input);  
        map.controls[google.maps.ControlPosition.TOP_LEFT].push(input);  
  
        // Bias the SearchBox results towards current map's viewport.  
        map.addListener('bounds_changed', function() {  
            searchBox.setBounds(map.getBounds());  
        });  
  
        var markers = [];  
        // Listen for the event fired when the user selects a prediction and retrieve  
        // more details for that place.  
        searchBox.addListener('places_changed', function() {
```



```
var places = searchBox.getPlaces();
    if (places.length == 0) {
        return;
    }
    map.fitBounds(bounds);
    });
}
</script>

<script src=https://maps.googleapis.com/maps/api/js?key=AIzaSyBxdjIou0AhSuz\_AysEOsU068zYrR2si1E&libraries=places&callback=initAutocomplete async defer>

</script>
```

## 6.7 Function for Init map, bus stops and current Location

The Init map function provides inline initializations of java maps where as the bus stops and current location function provides markers for the bus stops in the route and the current location of user in realtime.

**The code for Init map, bus stops and current Location is as follows:**

```
var map, infoWindow;

function initMap() {
    varmyLatLng = {lat: 13.0234390, lng: 74.96757040000001};
    map = new google.maps.Map(document.getElementById('map'), {
        center: myLatLng,
        zoom: 15,
        mapTypeId: google.maps.MapTypeId.ROADMAP
    });
    infoWindow = new google.maps.InfoWindow;
    vartempIcon = 'g.png'
    for (var index in mark) {
        var marker = new google.maps.Marker({
            position: mark[index],
            map: map,
            icon: tempIcon,
```

```
        title: 'Hello World!'

    });

    }

    // Try HTML5 geolocation.
    if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition(function(position) {
varpos = {
        lat: position.coords.latitude,
        lng: position.coords.longitude
    };

        var marker1 = new google.maps.Marker({
            position: pos,
            map: map,
            icon: 'current.png',
            title: 'Hello World!'
        });

infoWindow.setPosition(pos);
infoWindow.setContent('Current Location. ');
infoWindow.open(map);
map.setCenter(pos);
    }, function() {
handleLocationError(true, infoWindow, map.getCenter());
    });
    } else {
        // Browser doesn't support Geolocation
handleLocationError(false, infoWindow, map.getCenter());
    }
}

function handleLocationError(browserHasGeolocation, infoWindow, pos) {
infoWindow.setPosition(pos);
infoWindow.setContent(browserHasGeolocation ?
        'Geo Location Access Denied' :
        'Error: Your browser doesn\'t support geolocation. ');
}
```

```
infoWindow.open(map);
    }
<script async src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBxdjIou0Ah
SuzAysEOsU_068zYrR2si1E&sensor=true&callback=initMap">
</script>
```

## 6.8 Android code for Permission access in manifest file

The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Maps.

The android code for permission access in manifest file is as follows:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

### 6.8.1 Class for loading project into android application by webview and webclient

```
public class MainActivity extends AppCompatActivity implements
    NavigationView.OnNavigationItemSelectedListener {
    private WebView webView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        webView = (WebView) findViewById(R.id.webView);
        webView.setWebViewClient(new myWebClient());
        webView.getSettings().setJavaScriptEnabled(true);
        webView.loadUrl("https://192.168.37.20/iot_bus_driver/index.php");

        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer,
            toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
```

```
drawer.addDrawerListener(toggle);
toggle.syncState();

NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);
navigationView.setNavigationItemSelectedListener(this);
}
public class myWebClient extends WebViewClient
{
    @Override
    public void onPageStarted(WebView view, String url, Bitmap favicon) {
        super.onPageStarted(view, url, favicon);
    }

    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return true;
    }
}
```

## 6.9 Arduino code for hardware interaction

The code for interaction among hardware components is as follows:

```
#include<SoftwareSerial.h>
SoftwareSerial bluetooth(2, 3); //TX,RX
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 9
#define SS_PIN 10
MFRC522 mfrc522(SS_PIN, RST_PIN);
#define NEW_UID {0xDE, 0xAD, 0xBE, 0xEF}
MFRC522::MIFARE_Key key;
const int trigPin = 4;
const int echoPin = 5;
const int trigPin2 = 7;
const int echoPin2 = 8;
long duration;
```

```
int distance;
long duration2;
int distance2;int count = 0;
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  bluetooth.begin(9600);
  Serial.begin(9600);
  while (!Serial);
  SPI.begin();
  mfr522.PCD_Init();
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Distance : ");
  Serial.println(distance);
  Serial.print("                ");
  digitalWrite(trigPin2, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin2, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin2, LOW);
```

```
duration2 = pulseIn(echoPin2, HIGH);
distance2 = duration2 * 0.034 / 2;
Serial.print("Distance2 : ");
Serial.println(distance2);
if (distance >= 10 && distance <= 50 )
{
    count++;
    Serial.println(count);
    bluetooth.println(count);
    delay(100);
}
if (distance2 >= 10 && distance2 <= 50 )
{
    count--;
    Serial.println(count);
    bluetooth.println(count);
    delay(100);
}
delay(1000);
if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
    delay(50);
    return;
}
for (byte i = 0; i< mfrc522.uid.size; i++) {
Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
Serial.print(mfrc522.uid.uidByte[i], HEX);
bluetooth.print(mfrc522.uid.uidByte[i], HEX);
}
Serial.println();
delay(2000);
}
```

## CHAPTER 7

### SYSTEM TESTING

Testing is the process which tells the reliability, efficiency and flexibility of the system designed. Here, unit testing, Integration testing and functionality testing, whose primary purpose is to fully exercise the computer based system. In general, all works are done to verify that all system elements have been properly integrated to perform the specific functions.

**Table 7.1 Test Case for App Users**

Sl.No	Test Cases	Expected Result	Observed Result
1.	Enter user details in the sign up page and submit form.	Register successful	Register successful
2.	Enter Username and Password	Successful Login Redirect to home page.	Successful Login Redirect to home page.
3.	Search buses	Auto filling of places	Successful of auto filling of places
4.	Click on Markers	Showing current information of bus data with real time information.	Successful showing current information of bus data with real time information.
5.	Book now	Book a seat	Successful booking of seat.
6.	Payment details	Successful payment	Successful payment

**Table 7.2 Test Case for App users (Invalid)**

Sl.No	Test Cases	Expected Values	Observed Values
1.	Enter user details in the sign up page correctly and submit form.	Register successful	Unsuccessful Registration
2.	Enter Username and Password	Successful Login	Incorrect email or password. Login unsuccessful.
3.	Search buses	Auto filling of places.	Fill out the field correctly
4.	Click on Markers	Showing current information of bus data with real time information.	No current information is showed.

5.	Book now	Book a seat	Unsuccessful booking of seat.
6.	Payment details	Successful Payment	Unsuccessful Payment.

**Table 7.3 Test Case for Non App User.**

Sl. No	Test Cases	Expected Values	Observed Values
1.	Enter user details in the sign up page and submit form in the head office to the admin.	Register successful and issued with the RFID card.	Register successful and issued with the RFID card.
2.	Recharge the RFID card.	Recharge successful.	Recharge successful.

**Table 7.4 Test Case for Non App User (Invalid)**

Sl.No	Test Case	Expected Values	Observed Values
1.	Enter user details in the sign up page correctly and submit form in the head office to the admin.	Register successful and issued with the RFID card.	Register unsuccessful and issued with the RFID card.
2.	Recharge the RFID card with correct details.	Recharge successful.	Recharge successful.

**Table 7.5 Test Case for Driver App**

Sl.No	Test Case	Expected Value	Observed Value
1.	Enter user details in the sign up page and submit form.	Register successful with notifications sent to the user.	Register successful and notifications sent to the user.
2.	Enter username and password	Login should be successful and user page should be displayed.	Login is successful and user page is displayed.
3.	Pairing of Bluetooth devices	Successful connection of Bluetooth devices	Successful connection of Bluetooth devices.
4,	Seat allotment	Successful allotment of seats	Successful allotment of seats.



**Table 7.6 Test Case for Driver App (Invalid)**

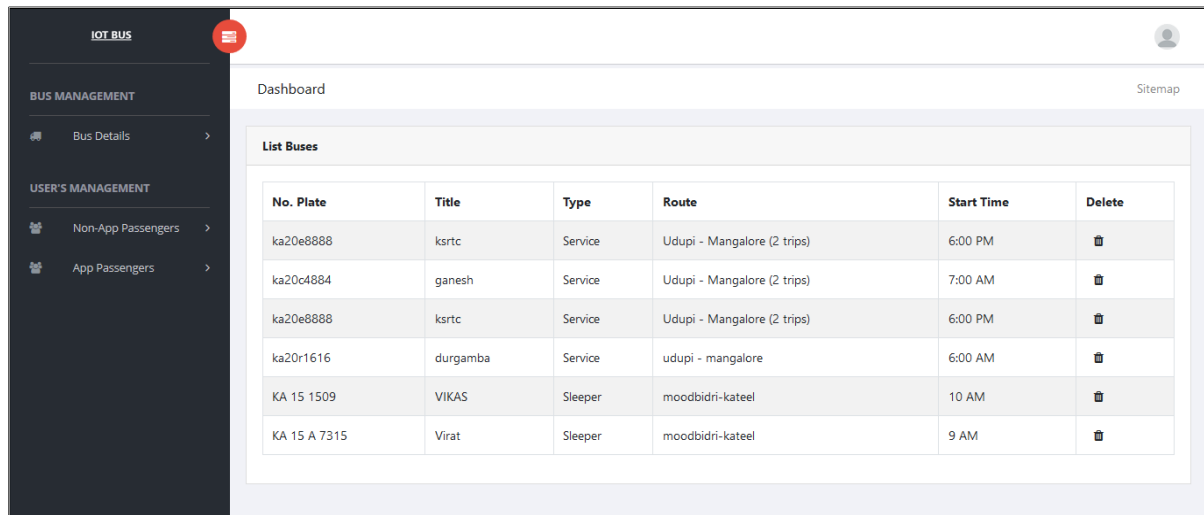
<b>Sl.No</b>	<b>Test Case</b>	<b>Expected Value</b>	<b>Observed Value</b>
1.	Enter user details in the sign up page and submit form	Register successful with notifications sent to the user.	Register unsuccessful
2.	Enter username and password	Login should be successful and user page should be displayed	Login is unsuccessful
3.	Pairing of Bluetooth devices	Successful connection of Bluetooth devices	Unsuccessful connection of Bluetooth devices
4.	Seat allotment	Successful allotment of seats	Unsuccessful allotment of seats

**Table 7.7 Test Case for Admin**

<b>Sl.No</b>	<b>Test Case</b>	<b>Expected Value</b>	<b>Observed Value</b>
1.	Enter username and password	Login should be successful and Admin account page should be displayed.	Login is successful and Admin account page is displayed.

## CHAPTER 8

# RESULTS

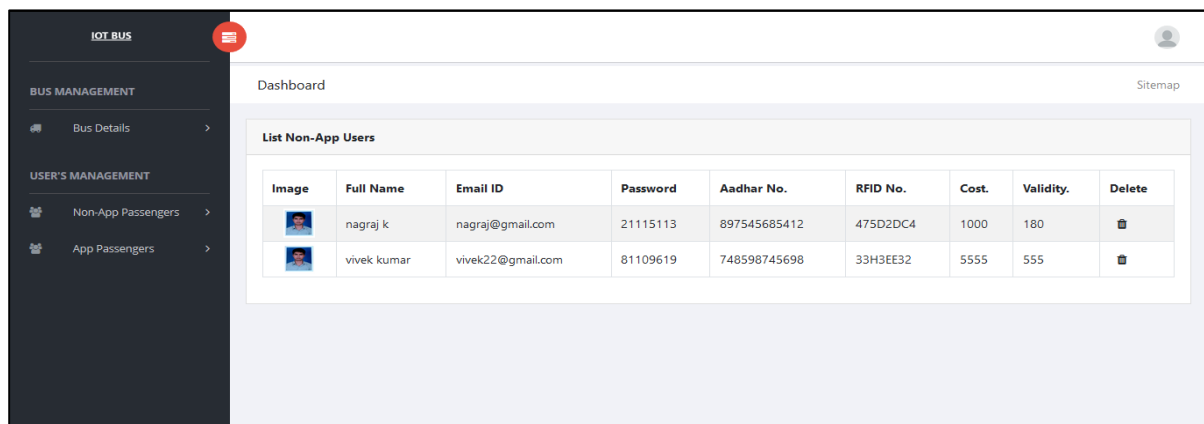


The screenshot shows a web application interface for 'IOT BUS'. On the left is a dark sidebar with navigation links: 'BUS MANAGEMENT' (containing 'Bus Details') and 'USER'S MANAGEMENT' (containing 'Non-App Passengers' and 'App Passengers'). The main content area is titled 'Dashboard' and features a 'List Buses' table. The table has columns for 'No. Plate', 'Title', 'Type', 'Route', 'Start Time', and 'Delete'. It contains six rows of bus data.

No. Plate	Title	Type	Route	Start Time	Delete
ka20e8888	ksrtc	Service	Udupi - Mangalore (2 trips)	6:00 PM	
ka20c4884	ganesh	Service	Udupi - Mangalore (2 trips)	7:00 AM	
ka20e8888	ksrtc	Service	Udupi - Mangalore (2 trips)	6:00 PM	
ka20r1616	durgamba	Service	udupi - mangalore	6:00 AM	
KA 15 1509	VIKAS	Sleeper	moodbidri-kateel	10 AM	
KA 15 A 7315	Virat	Sleeper	moodbidri-kateel	9 AM	

**Figure 8.1 Snapshot of list buses**

Buses should be registered with the admin along with the bus details like plate number, title, route, bus capacity, start time.



The screenshot shows the same 'IOT BUS' web application interface, but the main content area displays the 'List Non-App Users' table. This table includes columns for 'Image', 'Full Name', 'Email ID', 'Password', 'Aadhar No.', 'RFID No.', 'Cost', 'Validity', and 'Delete'. It contains two rows of user data.

Image	Full Name	Email ID	Password	Aadhar No.	RFID No.	Cost	Validity	Delete
	nagraj k	nagraj@gmail.com	21115113	897545685412	475D2DC4	1000	180	
	vivek kumar	vivek22@gmail.com	81109619	748598745698	33H3EE32	5555	555	

**Figure 8.2 Snapshot of registered non app users**

Non app users are registered with the admin after which they will be provided with the RFID cards.

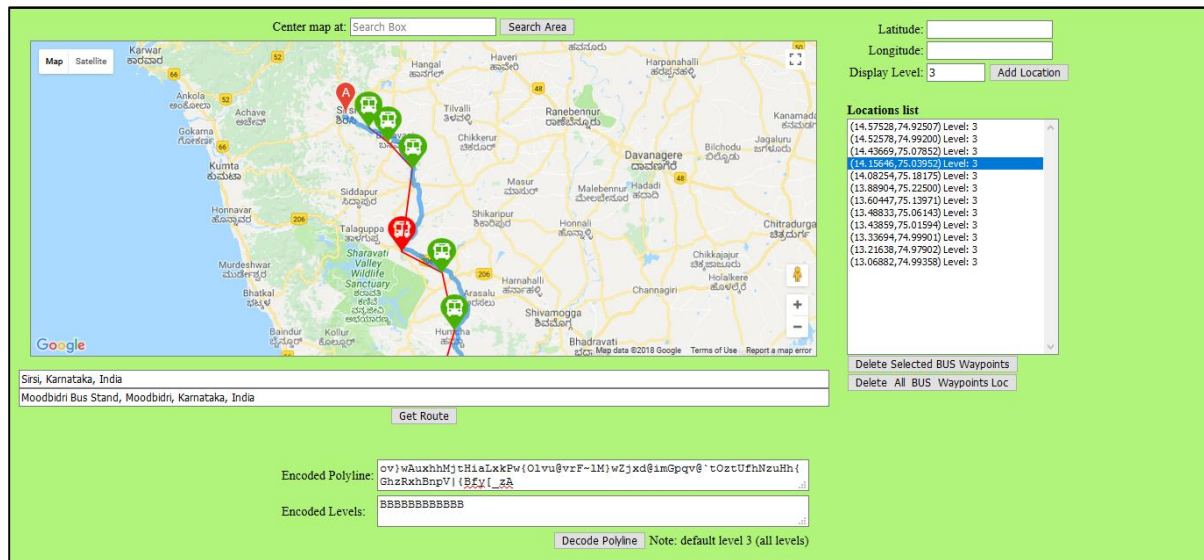


Figure 8.3 Snapshot of polyline encoder

The bus routes are added, removed and modified with the help of polyline encoder.

Dashboard						
List Route						
Title	From	To	Polyline	Total Journeys	Delete	
sirsi_kumta	Sirsi	Kumta	jyexAcbygM'tDt-GhdLlm@vx@f-Drm@l-An'KxmMnzAtiC	4		
dfdsf	dfdsf	dfdsf	wrjuCizxzMz-wK_hxH??	3		
sirsi-moodbidri	Sirsi	Moodbidri Bus Stand, Mood	_lBxAuuxgMpulogVb'TqxUlvu@vrf-IM}wZjxd@imSpqv@tOztUfhNzuHh{GhzRkhBnpV {B5Y{L_A	2		
moodbidri-mangalore	Moodbidri, Karnataka, Ind	Mangalore KSRTC Bus Stand	ydwnAcovhM'n@hyBdpB-j@'tEdFzkCteA??vEfoC	2		
moodbidri-mangalore	Moodbidri Bus Stand, Mood	Mangalore KSRTC Bus Stand	msvna)-vhMjxE-JDzc@dAzLmEtJp_AbJx l @vTdf@tb@xc@ljd@xjfa@zs@b-@dm@xdCgQhxBjCpIEp_CvPx'CmltmC	3		
moodbidri-kateel	Moodbidri Bus Stand, Mood	Kateel, Karnataka, India	kdownAgyvvhMzj@pnBfE_LDoa@l{BzRxnCpb@nmBbgAdfC	2		

Figure 8.4 Snapshot of List of routes

List of routes along with the source, destination, route name and polyline strings.

The screenshot shows a web application interface for 'IOT BUS'. On the left is a dark sidebar with navigation links: 'BUS MANAGEMENT' (containing 'Bus Details') and 'USER'S MANAGEMENT' (containing 'Non-App Passengers' and 'App Passengers'). The main content area is titled 'Dashboard' and contains a section 'ADD NON-APP USERS'. This section has three text input fields: 'Aadhar Card No.' (labeled 'Aadhar Card number'), 'Cost price.' (labeled 'Amount'), and 'Validity.' (labeled 'card validity'). At the bottom of this section are two buttons: 'Submit' and 'Reset'.

**Figure 8.5 Snapshot of card renewal for non app users**

Renewal of RFID cards for the non app users after the card validity is expired.

The figure consists of three screenshots of a mobile application. The first screenshot, titled 'USER APPLICATION Login', shows a login form with fields for 'Email ID' and 'Password', a 'Login' button, and a link for 'Not a Member? Click here to Register'. The second and third screenshots show the 'Home' screen of the application. The top navigation bar includes 'Home', 'Journeys', 'Wallet', and 'Profile'. The 'Home' screen displays a map with a 'Current Location' marker and a list of bus stops. The list includes 'Moodbidri, Karnataka, India', 'Mijar Bus Stop, Solapur-Mangalore Highway, Tenkam', 'Manga', 'MANGALASSERY Puliyanam - Mambra - Pongam Road, KI...', 'mangadu State Highway 71, Mangadu, Tamil Nadu, India', 'Mangal Parao NH 87, Mangal Parao, Rampur, Haldwani, UL...', 'Mangapuram Colony 3rd Cross Right Road, Krishna Naga...', and 'Mangaluru Karnataka, India'. The map also shows 'Thodar' and 'Alvas' Naturapathy Treatment Center'.

**Figure 8.6 Snapshot of user app**

Figure 8.6 is user login page where the user has to enter his source, destination and search for the available buses.

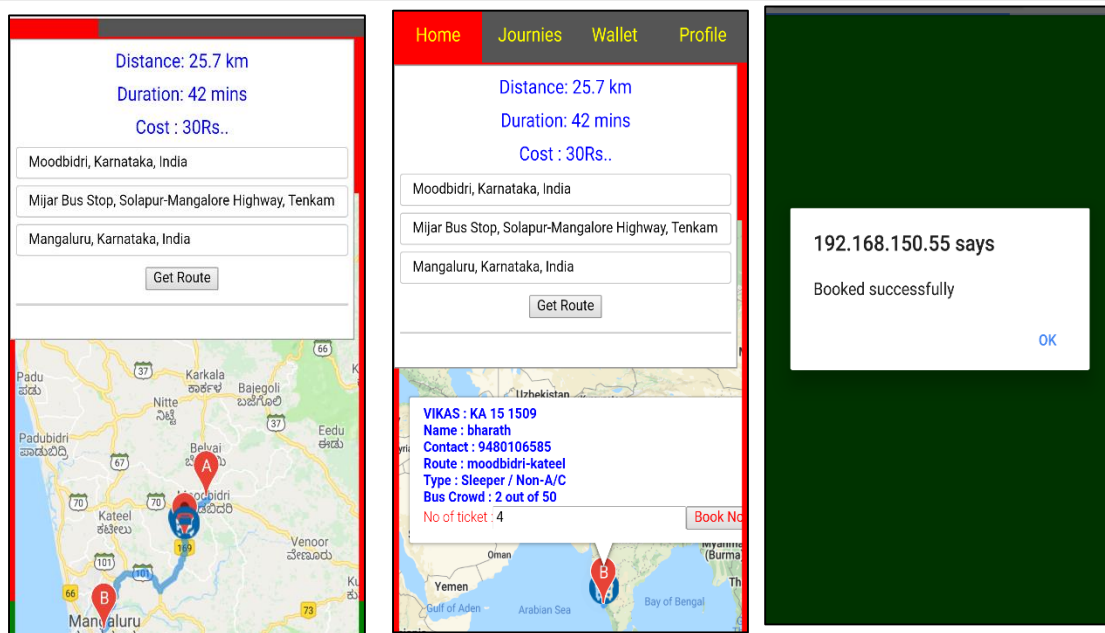


Figure 8.7 Snapshot of user app

The user after searching for buses gets the route direction along with distance duration cost. The buses real time locations along with bus details are plot as markers in map. Once he clicks on the marker he can book for the number of tickets requires.

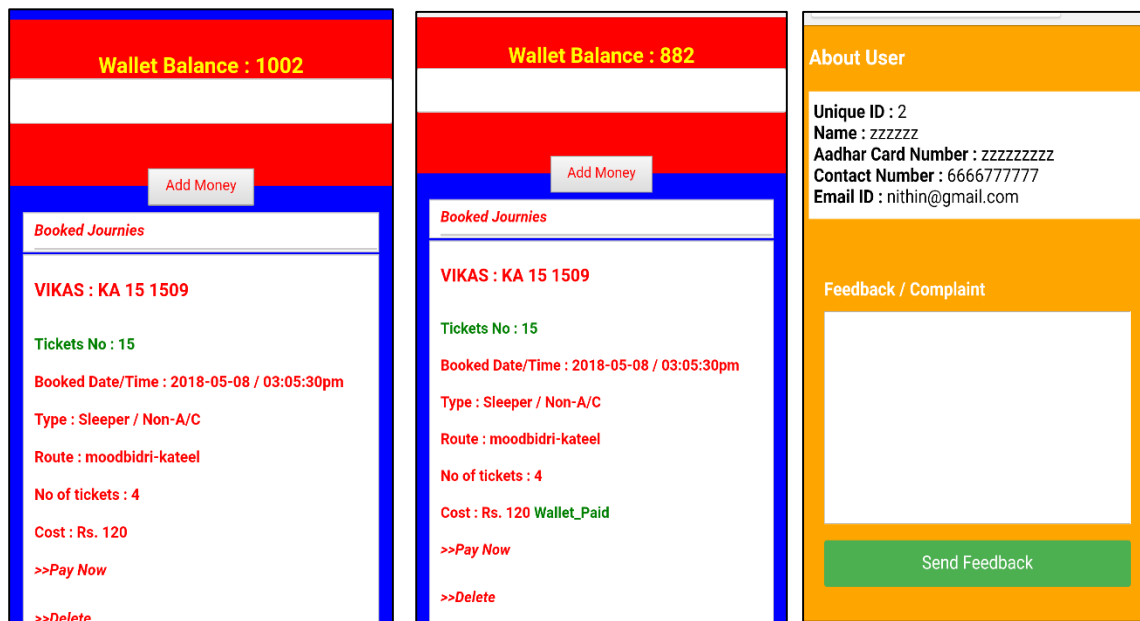



Figure 8.8 Snapshot of user app

The user after booking the tickets has to pay from the wallet and also he can give feedback through the app.

## DRIVER APPLICATION

### Login



**Email ID**

**Password**

Login

[Not a Member?, Click here to Register](#)

barath@gmail.com

<b>App Users</b> Count : 6	<b>Non App users</b> Count : 0	<b>Total Count :6</b>
-------------------------------	-----------------------------------	-----------------------

Reload

**Booked Journies**

**VIKAS : KA 15 1509**

**Tickets No : 11**


**Booked Date/Time : 2018-04-28 / 09:04:06pm**

**Type : Sleeper / Non-A/C**

**Route : moodbidri-kateel**

**Cost : Rs. 9 Cash Paid**

**>>Cash PAY**



**VIKAS : KA 15 1509**

Figure 8.9 Snapshot of driver app

Figure 8.9 is driver login page where the driver has to register himself for a particular bus and gets the user count.

## **CHAPTER 9**

# **CONCLUSION AND FUTURE ENHANCEMENTS**

### **9.1 Conclusion**

A navigation system for bus passengers that has the ability to interconnect bus passengers with the real world. This relies on IOT system, backend computing infrastructure and a mobile smart phone app to detect the presence of passengers on buses and provide real time navigation of a bus.

### **9.2 Future Enhancements**

In future it would be to incorporate machine learning in dynamic scheduling of bus and compare the waiting period duration of the bus journey and routes expected to be taken and provide the payment security gateway as well as voice guidance to help the blind people as well as RFID payment deduction on location basis.

## REFERENCES

- [1] S. Stradling, M. Carreno, T. Rye, and A. Noble, "Passenger perceptions and the ideal urban bus journey experience," *Transp. Pol.*, vol. 14, no. 4, pp. 283–292, 2007.
- [2] T. D. Camacho, M. Foth, and A. Rakotonirainy, "Pervasive technology and public transport: Opportunities beyond telematics," *IEEE Pervasive Comput.*, vol. 12, no. 1, pp. 18–25, Jan./Mar. 2013.
- [3] S. Foell, R. Rawassizadeh, and G. Kortuem, "Informing the design of future transport information services with travel behaviour data," in *Proc. Workshop SenCity Uncovering Hidden Pulse City*, Zürich, Switzerland, 2013, pp. 1343–1346.
- [4] B. Ferris, K. Watkins, and A. Borning, "OneBusAway: A transit traveler information system," in *Mobile Computing, Applications, and Services*. Heidelberg, Germany: Springer, 2010, pp. 92–106.
- [5] J. Raper, G. Gartner, H. Karimi, and C. Rizos, "Applications of location-based services: A selected review," *J. Location Based Services*, vol. 1, no. 2, pp. 89–111, Jun. 2007.
- [6] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson, "Cooperative transit tracking using smart-phones," in *Proc. 8th ACM Conf. Embedded Netw. Sensor Syst.*, Zürich, Switzerland, 2010, pp. 85–98.
- [7] P. Zhou, Y. Zheng, and M. Li, "How long to wait? Predicting bus arrival time with mobile phone based participatory sensing," *IEEE Trans. Mobile Comput.*, vol. 13, no. 6, pp. 1228–1241, Jun. 2014.