Bus Guiding System

By  
W J A J K Wijesinghe  
SE/2013/059

A report submitted in partial fulfillment of the requirements for the degree of Bachelor of Science Honours in Software Engineering (B.Sc. SE)

Name of the Supervisors: Dr. Sachintha Pitigala

Department of Industrial Management  
Faculty of Science  
University of Kelaniya  
Sri Lanka  
2017

# DECLARATION

I hereby certify that no part of this assignment has been copied from any other student work or from any other source except where due acknowledgement is made in the assignment. No part of this assignment has been written/ produced for me by any other person except where such collaboration has been authorized by the subject lecturer/tutor concerned. I hold a copy of this assignment that I can produce if the original is lost or damaged.

**Student Number Signature**

SE/2013/059 ……………………………………..

**Note: An examiner or lecturer/tutor has the right not to mark this assignment if the above declaration has not been signed.**

Supervisor: Dr. Sachintha Pitigala

**Signature**

……………………………………..

**Only for use by Lecturer/Tutor:**

**ASSESSMENT MARKS AND COMMENTS**

|  |
| --- |
| Marking scheme: |
| 1. |
| 2. |
| 3. |
| 4. |
| Comments: |
| Total marks out of 100%: |
| Weighting considered for the continuous assessment: % |

# ACKNOWLEDGEMENT

First and foremost, I would like to extend my heartfelt gratitude to my project supervisor, Dr. Keerthi Wijayasiriwardhane for his immense support and guidance given to me from the beginning of the project.

I would also like to convey my thanks to Dr. Dilani Wickramarachchi, the project Coordinator, for initial understanding and guidance we were given and for her constant support given throughout the project.

I thank all the staff members, especially the panel of judges who gave valuable inputs and feedback on the poster presentation of the system I was proposing to do. Those comments helped immensely to identify points to be improved and to have a clear idea on the system scope.

My sincere thanks go to my friends and batch mates who have been sharing knowledge and extending support in case of need and to each and every individual who have been supportive by all mentioned and non-mentioned means. Last but not the least; my gratitude is always extended to my parents and family who is behind everything I am today and for being there with me in everything since the beginning.

# ABSTRACT

Bus guiding system is a mobile application which helps drivers to maintain an optimal gap between buses. It’ll reduce clashes among buses and reduce negative consequences happening to the passengers. The main target is override the current inconsistent bus service and form a consistent bus service.

Introduction explains objectives of the project and scope of the project. System analysis includes facts gathering, Requirements, SRS and BSOs. System design shows Use case diagrams, activity diagrams, class diagram, ER diagram and user interfaces. Conclusion is included at last.

Table of Contents

[DECLARATION i](#_Toc527262299)

[ACKNOWLEDGEMENT ii](#_Toc527262300)

[ABSTRACT iii](#_Toc527262301)

[List of Figures 7](#_Toc527262302)

[List of Tables 9](#_Toc527262303)

[Abbrevations 10](#_Toc527262304)

[Introduction 11](#_Toc527262305)

[01. Terms of References 11](#_Toc527262306)

[01.01. Project Background 11](#_Toc527262307)

[01.01.01. Existing System 11](#_Toc527262308)

[01.01.02. Motivation 11](#_Toc527262309)

[01.01.03. Problems in the existing system 11](#_Toc527262310)

[01.01.04. Clients 11](#_Toc527262311)

[01.02. Aims 12](#_Toc527262312)

[01.02.01. Project is intended to achieve 12](#_Toc527262313)

[01.02.02. What is not part of the project 12](#_Toc527262314)

[01.02.03. Gain from the project 12](#_Toc527262315)

[01.03. Objectives 12](#_Toc527262316)

[01.04. Features of the prototype 13](#_Toc527262317)

[01.05. Project Deliverables 14](#_Toc527262318)

[01.06. Resource Requirements 14](#_Toc527262319)

[01.07. Activity Schedule 14](#_Toc527262320)

[02. Poster 15](#_Toc527262321)

[03. Organization of the Dissertation 16](#_Toc527262322)

[Summery 16](#_Toc527262323)

[Introduction 17](#_Toc527262324)

[01. Exsisting System 17](#_Toc527262325)

[01.01. Use cases for existing system 17](#_Toc527262326)

[01.02. Activity diagrams for the current system 19](#_Toc527262327)

[02. System Requirements Specifications 22](#_Toc527262328)

[02.01. Functional Requirements 22](#_Toc527262329)

[02.02. Non Functional Requirements 23](#_Toc527262330)

[03. Business system options (BSOs) 24](#_Toc527262331)

[03.01. BSOs 24](#_Toc527262332)

[03.01.1. Using a GPS GSM Tracker 24](#_Toc527262333)

[03.01.2. Using a Arduino GPS shield 25](#_Toc527262334)

[03.01.3. Using the Android mobile phone 26](#_Toc527262335)

[03.02. Functional Requirements vs. BSOs 27](#_Toc527262336)

[03.03. Non-functional requirements vs. BSOs 29](#_Toc527262337)

[03.04. Selecting a BSO 29](#_Toc527262338)

[Summary 29](#_Toc527262339)

[Introduction 30](#_Toc527262340)

[01. Use case diagrams 30](#_Toc527262341)

[01.01. Actors 30](#_Toc527262342)

[01.02. Register (UC01) 31](#_Toc527262343)

[01.03. Display icon (UC02) 31](#_Toc527262344)

[01.04. Display details (UC03, UC04, UC05, UC06) 32](#_Toc527262345)

[01.05. Reports (UC07, UC08, UC09) 32](#_Toc527262346)

[01.06. Search (UC10, UC11) 33](#_Toc527262347)

[01.07. GPS (UC12, UC13) 33](#_Toc527262348)

[02. Use case descriptions 34](#_Toc527262349)

[03. Activity diagrams 39](#_Toc527262350)

[03.01. Register a bus 39](#_Toc527262351)

[03.02. Display icon of a bus 40](#_Toc527262352)

[03.03. Display details 41](#_Toc527262353)

[03.04. Search bus/buses 42](#_Toc527262354)

[03.05. GPS 43](#_Toc527262355)

[03.06. Monthly Report 44](#_Toc527262356)

[04. High Level Architecture 45](#_Toc527262357)

[05. Class Diagram 46](#_Toc527262358)

[06. Model, View and Controller classes 47](#_Toc527262359)

[06.01. Model Classes 47](#_Toc527262360)

[06.02. View Classes 47](#_Toc527262361)

[06.03. Controller Classes 47](#_Toc527262362)

[07. Entity Relationship Diagram 48](#_Toc527262363)

[08. Sequence diagrams 49](#_Toc527262364)

[08.01. Register a bus 49](#_Toc527262365)

[08.02. Show bus icon 50](#_Toc527262366)

[08.03. Search a bus 50](#_Toc527262367)

[08.04. Search buses 51](#_Toc527262368)

[08.05. Update GPS coordinates 51](#_Toc527262369)

[08.06. Get GPS coordinates 52](#_Toc527262370)

[08.07. Monthly Report 52](#_Toc527262371)

[08.08. Get static details 53](#_Toc527262372)

[08.09. Get dynamic details 54](#_Toc527262373)

[09. Interfaces 55](#_Toc527262374)

[09.01. Main Interface 55](#_Toc527262375)

[09.02. Driver/Conductor Menu 55](#_Toc527262376)

[09.03. Login Menu 56](#_Toc527262377)

[09.04. Register Menu 56](#_Toc527262378)

[09.05. Owner Menu 57](#_Toc527262379)

[09.06. Report Menu 57](#_Toc527262380)

[09.07. Report 58](#_Toc527262381)

[09.08. User Menu 59](#_Toc527262382)

[09.09. Search Menu 59](#_Toc527262383)

[09.10. Search by Number 60](#_Toc527262384)

[09.11. Search by filter 60](#_Toc527262385)

[09.12. Display 61](#_Toc527262386)

[Summary 61](#_Toc527262387)

[Introduction to fourth chapter 65](#_Toc527262388)

[01. Degree of objectives met 65](#_Toc527262389)

[02. Usability, accessibility, reliability and friendliness 65](#_Toc527262390)

[03. User’s response 65](#_Toc527262391)

[04. Limitations and drawbacks 66](#_Toc527262392)

[05. Further modifications, improvements and extensions possible 66](#_Toc527262393)

[References 67](#_Toc527262394)

# List of Figures

[Figure 1 - Prototype 13](#_Toc475360421)

[Figure 2 - Schedule 14](#_Toc475360422)

[Figure 3 - Poster 16](#_Toc475360423)

[Figure 4 – Use case – Call over phone 18](#_Toc475360424)

[Figure 5 – Use case – Oral Communication 19](#_Toc475360425)

[Figure 6 – Use case – Oral Communication II 19](#_Toc475360426)

[Figure 7 – Activity diagram – Call over phone 20](#_Toc475360427)

[Figure 8 – Activity diagram – Oral communication 21](#_Toc475360428)

[Figure 9 – Activity diagram – Oral communication II 22](#_Toc475360429)

[Figure 10 – BSO-1 High level architecture 25](#_Toc475360430)

[Figure 11 – BSO-2 High level architecture 26](#_Toc475360431)

[Figure 12 – BSO-3 High level architecture 27](#_Toc475360432)

[Figure 13 – Use case - Actors 31](#_Toc475360433)

[Figure 14 – Use case - Register 32](#_Toc475360434)

[Figure 15 – Use case – Display icon 32](#_Toc475360435)

[Figure 16 – Use case – Display details 33](#_Toc475360436)

[Figure 17 – Use case - Reports 34](#_Toc475360437)

[Figure 18 – Use case - Search 34](#_Toc475360438)

[Figure 19 – Use case - GPS 35](#_Toc475360439)

[Figure 20 - Activity diagram – Register a bus 41](#_Toc475360440)

[Figure 21 – Activity diagram – display icon of a bus 42](#_Toc475360441)

[Figure 22 – Activity diagram – Display details 43](#_Toc475360442)

[Figure 23 – Activity diagram – Search bus/buses 44](#_Toc475360443)

[Figure 24 – Activity diagram - GPS 45](#_Toc475360444)

[Figure 25 – Activity diagram - Monthly report 46](#_Toc475360445)

[Figure 26 – High level Architecture 47](#_Toc475360446)

[Figure 27 – Class diagram 48](#_Toc475360447)

[Figure 28 - ER diagram 50](#_Toc475360448)

[Figure 29 – Sequence diagram – Register a bus 51](#_Toc475360449)

[Figure 30 – Sequence diagram – Show bus icon 52](#_Toc475360450)

[Figure 31 – Sequence diagram – Search a bus 52](#_Toc475360451)

[Figure 32 – Sequence diagram – Search buses 53](#_Toc475360452)

[Figure 33 – Sequence diagram – Update GPS coordinates 54](#_Toc475360453)

[Figure 34 – Sequence diagram – Get gps coordinates 55](#_Toc475360454)

[Figure 35 - Sequence diagram – Monthly report 55](#_Toc475360455)

[Figure 36 – Sequence diagram – Get static details 56](#_Toc475360456)

[Figure 37 – Sequence diagram – Get dynamic details 57](#_Toc475360457)

[Figure 38 – Main Interface 58](#_Toc475360458)

[Figure 39 – Driver/Conductor Menu 58](#_Toc475360459)

[Figure 40 – Login Menu 59](#_Toc475360460)

[Figure 41 – Register Menu 59](#_Toc475360461)

[Figure 42 – Owner Menu 60](#_Toc475360462)

[Figure 43 – Report Menu 60](#_Toc475360463)

[Figure 44 – Report 61](#_Toc475360464)

[Figure 45 - Report table 61](#_Toc475360465)

[Figure 46 – Report graph 62](#_Toc475360466)

[Figure 47 – User Menu 62](#_Toc475360467)

[Figure 48 – Search Menu 63](#_Toc475360468)

[Figure 49 – Search by number 64](#_Toc475360469)

[Figure 50 – Search by filter 65](#_Toc475360470)

[Figure 51 - Display 65](#_Toc475360471)

# List of Tables

[Table 1 – Use case - Call over the phone 18](#_Toc475360472)

[Table 2 – Use case - Oral Communication 19](#_Toc475360473)

[Table 3 – Use case - Oral Communication II 19](#_Toc475360474)

[Table 4 - Functional Requirements vs BSOs 29](#_Toc475360475)

[Table 5 - Non-functional requirements vs BSOs 30](#_Toc475360476)

[Table 6 - Use case - Register a bus 36](#_Toc475360477)

[Table 7 -Use case - Show icon of a bus 36](#_Toc475360478)

[Table 8 -Use case - Determine distance gap 36](#_Toc475360479)

[Table 9 - Use case - Display details 37](#_Toc475360480)

[Table 10 - Use case - Determine time gap 37](#_Toc475360481)

[Table 11 - Use case - Determine current speed 38](#_Toc475360482)

[Table 12 - Use case - Determine average speed 38](#_Toc475360483)

[Table 13 - Use case - Monthly reports 38](#_Toc475360484)

[Table 14 - Use case - Determine average time 38](#_Toc475360485)

[Table 15 - Use case - Search a bus 39](#_Toc475360486)

[Table 16 - Use case - Search buses 39](#_Toc475360487)

[Table 17 - Use case - Upload GPS coordinates 40](#_Toc475360488)

[Table 18 -Use case - Get GPS coordinates 40](#_Toc475360489)

# Abbrevations

* GPS - Gloabal Positioning Sytem
* GSM - Global System for Mobile communication
* UML – Unified Modelling Language

CHAPTER I – INTRODUCTION

# Introduction

This chapter explains about the project background, current practices, aims and objectives of the project, scope of the project with clear boundaries and organization of dissertation. This will help to understand project background and the need of a software system for the problems identified.

# Terms of References

## Project Background

### Existing System

Even though the current practice conductors and drivers are using mobile phones to communicate with each other the distance between the buses. However, in some situations this cannot be done. Apart from that, sometimes there is a separate person to communicate the times that the buses that pass him to the bus drivers and conductors where he gets a Rupees 50 as a tip for communicating these messages. Nevertheless, sometimes even he is not accurate in reporting them.

### Motivation

There are so many difficulties to both bus drivers, conductors and passengers. Thus, there is a need of better way to do this. For the sake of both parties, solution is real time tracking system.

### Problems in the existing system

1. Less accuracy
2. Clashes among bus drivers/conductors
3. Cost (For phone calls)
4. Difficulties which have to face by passengers
5. Inefficient bus service

### Clients

1. Bus Conductors
2. Bus Owners
3. Bus Companies
4. Bus Drivers
5. Passengers
6. Bus Auditors
7. Time Keepers

## Aims

### Project is intended to achieve

1. Provide the facility of tracking buses
2. Provide the facility of filtering buses in many ways
3. Provide monthly reports on average time, speed and etc.
4. Keep records of average time, speed and etc.
5. Provide details about

* Current distance
* Current speed
* Current time gap
* Average speed
* Average time

### What is not part of the project

This project will not providing facilities to count passengers, count amount of fuel consumes, count amount of income.

### Gain from the project

This project is firstly to improve my academic knowledge. One of the outcomes is learning about GPS tracking and create a useful thing out of it. Using advertisements on the app can be a benefit, if the app gets popular.

## Objectives

1. Analyzing requirements
2. Categorizing requirements
3. Identifying requirements of different customer types
4. Designing the main facility – Get the real time GPS coordinates to the server
5. Designing sub facilities (calculating the current speed, etc.)
6. Designing reports (monthly reports, etc.)
7. System Design
8. Logical Design
9. Creating UML diagrams
10. Creating documentation
11. Dealing with Google maps API
12. Integrate google maps into app

## Features of the prototype



Figure 1 - Prototype

1. Select customer type (Driver/Conductor, Passenger, Auditor)

The first screen you have to select customer type.

1. Different UI for different customer types

There are 4 different UIs based on customer type.

1. Real time tracking

GPS coordinates are updated regularly within same periods of time and uploaded to sever and the user can get them frequently and have real time information.

1. Average Speed

Difference between distance within a portion of time taking to account and speed is calculated.

1. Provide identification information of other buses (Name, Number)

In the map user can see other buses details with their indicator

1. Gap between your bus and another bus

With the indicator of another bus we can see in details how far our bus is from that bus.

1. Other buses speed

With the indicator of other bus we can see their average speed.

1. Next buses

If you are a passenger type customer you can see the buses coming in the road which are going for your destination.

1. Average time taken for the route

With the statistics the app calculate average time taken for a single route by a particular bus.

1. All the buses in one area

If you are an auditor you can see the all buses in a particular area.

1. All the buses of single owner/company

If you’re an owner you can see where your buses are.

1. Monthly reports (on Average time, speed)

The app will calculate based on statistics and it’ll generate useful reports for owner and other customers.

## Project Deliverables

1. The Prototype
2. Project Report
3. Project Presentation
4. Test case

## Resource Requirements

1. Android Studio
2. Google Maps API
3. Smart Phones
4. SQL Server

## Activity Schedule

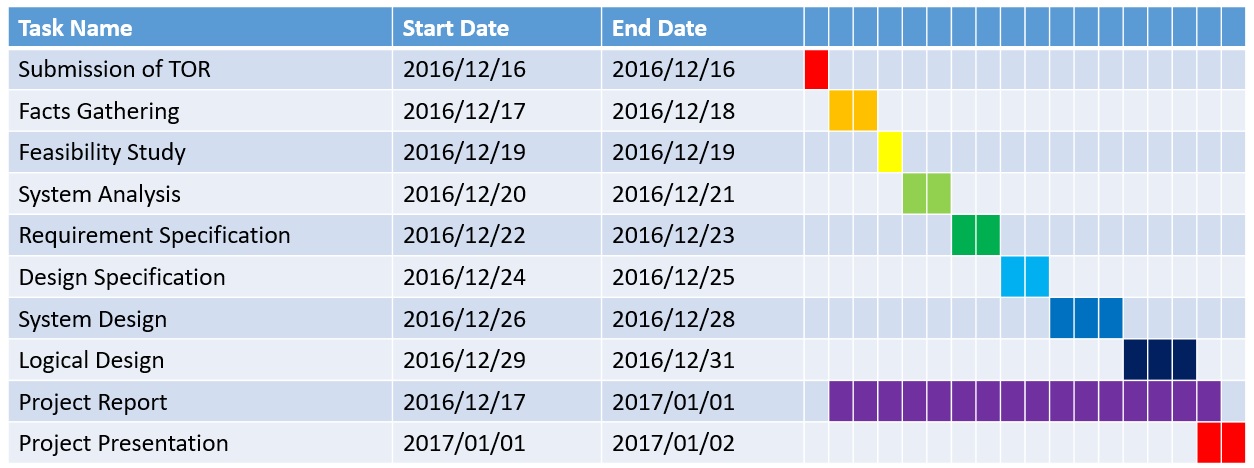


Figure 2 - Schedule

# Poster



Figure 3 - Poster

# Organization of the Dissertation

Chapter II

Chapter II include details about system analysis. Furthermore it include use case diagrams and activity diagrams for currents system. Then it include system requirements specification. Business system option and selecting a business option is included at last.

Chapter III

Chapter III include complete system design documentation including use case diagrams, activity diagrams, high level architecture class diagram, ER diagram, sequence diagrams and software interfaces.

Chapter IV

Chapter IV include the conclusion, including degree of objectives covered, further modifications and improvements, Reliability, extensibility and Limitations and drawbacks of the system.

# Summery

Now we have the project background and identified the problems in current system. It is good to know your problem well before you are going to give a solution for it. Furthermore, we identified the need of a new software system to increase the accuracy and efficiency of the current bus services. Next we have to analyize the current system to identify requirements.

CHAPTER II - SYSTEM ANALYSIS

# Introduction

In this chapter the first goal is to evaluate the current system and identify requirements. We use use-case diagrams and activity diagrams to evaluate current system. Then we need to analyze the requirements and derive functional and non-functional requirements. Business system options – the alternative are identified and to be evaluated. Appropriate BSO is identified through cost-benefit analysis and other analysis methods.

# Exsisting System

## Use cases for existing system

This diagram explain a driver call to another driver in another bus and asking for information.



Figure 4 – Use case – Call over phone

|  |  |
| --- | --- |
| **Thumbnail** | Call over phone |
| **Description** | Driver/Conductor in one bus can call over phone to another bus and get their location |
| **Pre-Conditions** | Credit on phone  Network coverage |
| **Post-Conditions** | Registered  Mobile data enabled |
| **Actors** | Driver/Conductor |
| **Use Case Relationships** | - |

Table 1 – Use case - Call over the phone

This diagram explain the special situation where an informer plays and role among bus drivers to inform them about other buses.



Figure 5 – Use case – Oral Communication

|  |  |
| --- | --- |
| **Thumbnail** | Oral Communication |
| **Description** | Driver/Conductor is getting information by an informer |
| **Pre-Conditions** | Have to pay informer |
| **Post-Conditions** | Paid |
| **Actors** | Driver/Conductor, Informer |
| **Use Case Relationships** | - |

Table 2 – Use case - Oral Communication

This diagram shows when two buses are passing by exchange information to each other.



Figure 6 – Use case – Oral Communication II

|  |  |
| --- | --- |
| **Thumbnail** | Oral Communication II |
| **Description** | Driver/Conductor can orally communicate with another Driver/Conductor when the two buses are met. |
| **Pre-Conditions** | Road have to be clear  No policeman in the area |
| **Post-Conditions** | - |
| **Actors** | Driver/Conductor |
| **Use Case Relationships** | - |

Table 3 – Use case - Oral Communication

## Activity diagrams for the current system

This diagram explain a driver call to another driver in another bus and asking for information.

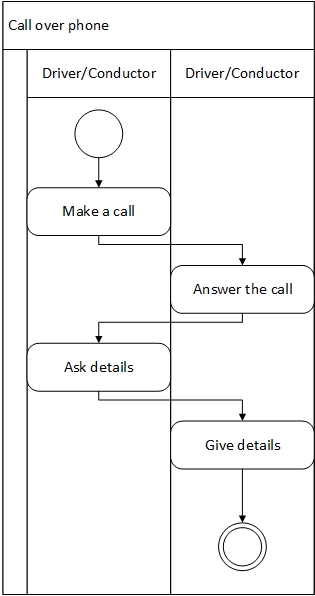


Figure 7 – Activity diagram – Call over phone

This diagram explain the special situation where an informer plays and role among bus drivers to inform them about other buses.

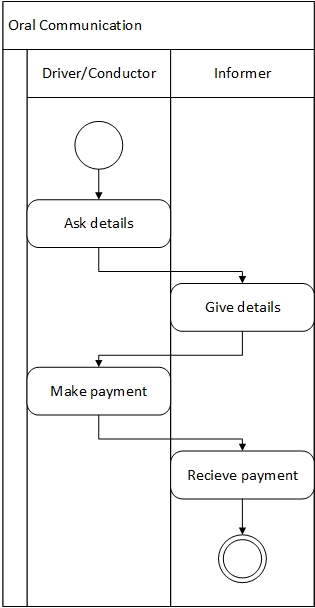


Figure 8 – Activity diagram – Oral communication

This diagram shows when two buses are passing by exchange information to each other.

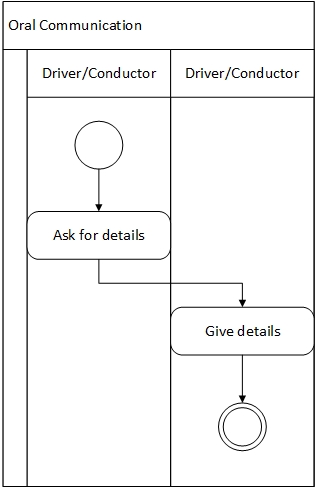


Figure 9 – Activity diagram – Oral communication II

# System Requirements Specifications

## Functional Requirements

1. Shall be able to register a bus using registration number (ex. WP JH 3928), route number (ex. 01 Kandy-Colombo), category (Slow/Normal/Semi Luxury/Intercity) and name (optional).
2. Shall be able to display an icon for a bus in the google map.
3. Shall facilitate the user (driver/conductor) to start the execution of app by tapping the start button on GUI.
4. Shall facilitate the user (driver/conductor) to end the execution of app by tapping the end button on GUI.
5. Shall be able to determine the current speed using GPS coordinates and time and display in kmpH format.
6. Shall be able to determine average speed using distance traveled divided by time taken for the travel and display in kmpH format.
7. Shall be able to show registration number, current speed, how far from the user, time gap with the user at the current speed of the bus and the name of the bus (if any).
8. Shall be able to keep start time, average speed, time taken, distance travelled, distance to go during the trip.
9. Shall be able to keep records about average speed during the trip, average time taken for a trip.
10. Shall be able to show the ability to filter buses depending on route number, road number, area, trip, owner name, going towards a particular bus stop and category.
11. Shall be able to search a bus by registered number
12. Shall be able to provide different user interfaces to driver/conductor, passenger, time keeper, owner and auditor.
13. Shall be able to generate reports monthly containing average speed (this month), average time (this month), average speed (overall) and average time (overall).
14. Shall be able to keep records about the bus station to other bus stops on average time taken from a bus station to another bus stop. This records will keep in both ways. (ex. Kandy to Colombo and Colombo to Kandy)
15. Shall be able to show the time taken for and average speed for bus stop to another bus stop in the row.
16. The passenger shall be able to see how far the next buses and time they take to reach him/her.

## Non Functional Requirements

1. Performance Requirements
2. Shall be able to update GPS coordinates every 30 seconds.
3. Shall be able to provide 98% accuracy on distance gaps.
4. Shall be able to provide 80% accuracy on average speeds.
5. Shall be able to provide 80% accuracy on current speeds.
6. Shall be able to provide 75% accuracy on time gaps.
7. Shall be able to use battery as low as possible
8. Shall be able to keep the screen without sleeping
9. Safety Requirements
10. Shall be able to survive without crashing or lagging when location or/and Mobile data service is/are gone off.
11. Security Requirements
12. Shall be able to ignore other apps which are trying to use the database.
13. Software Quality Attributes
14. Shall be able to indicate when mobile data or GPS services are gone offline.
15. Shall be able to display disconnected buses where they are connected last time with disconnected status.
16. Business Rules
17. Shall be able to use the nearest bus stop as the bus stop for determining locations.

# Business system options (BSOs)

## BSOs

### Using a GPS GSM Tracker

Using a GPS GSM tracker sensor get the GPS coordinates and upload them to the database. Here we use GPS GSM tracker to get the location. Users use their android phones to use the services.

Architectural design



Figure 10 – BSO-1 High level architecture

Costs:

* + GPS GSM Tracker – Rs. 5500/=

Drawbacks

* + Cost
  + Maintenance
  + Integration

Benefits

* + Tracker always inside the bus
  + Can use bus’s battery

### Using a Arduino GPS shield

Using a GPS sensor we can get the location using GPS sensor and using Arduino process the task and transmit using GSM shield transmit data to the server.

Architectural design



Figure 11 – BSO-2 High level architecture

Costs:

* + Arduino GPS shield – Rs. 4500
  + Arduino Mega – Rs. 1500
  + Arduino GSM shield – Rs. 4500
  + Total – Rs. 10500/=

Drawbacks

* + Cost
  + Maintenance
  + Integration

Benefits

* + Can use bus’s battery
  + Gadget always inside the bus

### Using the Android mobile phone

In this case Driver’s or conductor’s android mobile phone used to locate the bus and transmit data to the server using mobile data connection.

Architectural design

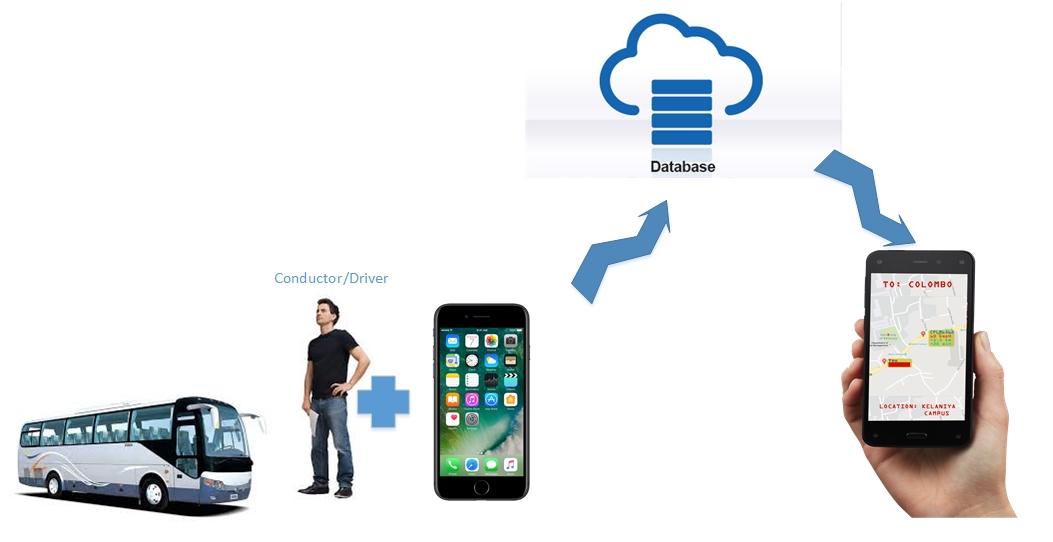


Figure 12 – BSO-3 High level architecture

Costs:

* + Driver or the conductor should own an Android mobile with GPS and GSM services

Drawbacks

* + If conductor’s phone is the device in use he has to remain with the bus all the time during a journey
  + Phone may run out of battery, so need to charge

Benefits

* + No cost
  + Need no maintenance
  + Need no integration

## Functional Requirements vs. BSOs

|  |  |  |  |
| --- | --- | --- | --- |
| Functional Requirements | BSO-1 | BSO-2 | BSO-3 |
| 1. Shall be able to register a bus using registration number (ex. WP JH 3928), route number (ex. 01 Kandy-Colombo), category (Slow/Normal/Semi Luxury/Intercity) and name (optional). | x | x | x |
| 1. Shall be able to display an icon for a bus in the google map. | x | x | x |
| 1. Shall facilitate the user (driver/conductor) to start the execution of app by tapping the start button on GUI. | x | x | x |
| 1. Shall facilitate the user (driver/conductor) to end the execution of app by tapping the end button on GUI. | x | x | x |
| 1. Shall be able to determine current speed using GPS coordinates and time and display in kmpH format. | x | x | x |
| 1. Shall be able to determine average speed using distance traveled divided by time taken for the travel and display in kmpH format. | x | x | x |
| 1. Shall be able to show registration number, current speed, how far from the user, time gap with user at the current speed of the bus and name of the bus (if any). | x | x | x |
| 1. Shall be able to keep start time, average speed, time taken, distance travelled, distance to go during the trip. | x | x | x |
| 1. Shall be able to keep records about average speed during the trip, average time taken for a trip. | x | x | x |
| 1. Shall be able to show able to filter buses depending on route number, road number, area, trip, owner name, going towards a particular bus stop and category. | x | x | x |
| 1. Shall be able to search a bus by registered number | x | x | x |
| 1. Shall be able to provide different user interfaces to driver/conductor, passenger, time keeper, owner and auditor. | x | x | x |
| 1. Shall be able to generate reports monthly containing average speed (this month), average time (this month), average speed (overall) and average time (overall). | x | x | x |
| 1. Shall be able to keep records about bus station to other bus stops on average time taken for bus station to another bus stop. This records will keep in both ways. (ex. Kandy to Colombo and Colombo to Kandy) | x | x | x |
| 1. Shall be able to show the time taken for and average speed for bus stop to another bus stop in the row. | x | x | x |
| 1. Passenger shall be able to see how far the next buses and time they take to reach him/her. | xx | x | x |

Table 4 - Functional Requirements vs BSOs

## Non-functional requirements vs. BSOs

|  |  |  |  |
| --- | --- | --- | --- |
| Non-functional requirements | BSO-1 | BSO-2 | BSO-3 |
| 1. Shall be able to update GPS coordinates every 30 seconds. | x | x | x |
| 1. Shall be able to provide 98% accuracy on distance gaps. | x | x | x |
| 1. Shall be able to provide 80% accuracy on average speeds. | x | x | x |
| 1. Shall be able to provide 80% accuracy on current speeds. | x | x | x |
| 1. Shall be able to provide 75% accuracy on time gaps. | x | x | x |
| 1. Shall be able to use battery as low as possible | x | x | x |
| 1. Shall be able to keep the screen without sleeping | x | x | y |
| 1. Shall be able to survive without crashing or lagging when location or/and Mobile data service is/are gone off. | x | x | x |
| 1. Shall be able to ignore other apps which are trying to use the database. | x | x | x |
| 1. Shall be able to indicate when mobile data or GPS services are gone offline. | x | x | x |
| 1. Shall be able to display disconnected buses where they are connected last time with disconnected status. | x | x | x |
| 1. Shall be able to use the nearest bus stop as the bus stop for determining locations. | x | x | x |

Table 5 - Non-functional requirements vs BSOs

## Selecting a BSO

It’s pretty much clear when considering cost it’s easy to use driver’s or conductor’s android mobile phone. The only drawback is if conductor’s android mobile phone is used he need to remain with the bus. As it’s not a major disadvantage we can proceed with BSO-1.

Accuracy is equal in all 3 cases. We need to consider maintainability so the smart phone need no maintenance such as other alternatives do. So the BSO-1 is the conclusion.

# Summary

In this chapter we selected and appropriate solution for the problem form alternatives. Next chapter will show hoe the appropriate solution to be deigned. Set of requirements identified by this chapter will be convert into a design in the next chapter.

CHAPTER III – DESIGN

# Introduction

In this chapter the diagrams and designs will explain the complete design for the solution identified by the second chapter. Use-cases, activity diagrams are used to identify the structural aspects of the system. Also the class diagram explains the overall structure. Sequence diagram explains the dynamic behavior of the system. Entity relationship diagram explains the database design.

# Use case diagrams

## Actors

These are the actors we have to consider in system design.

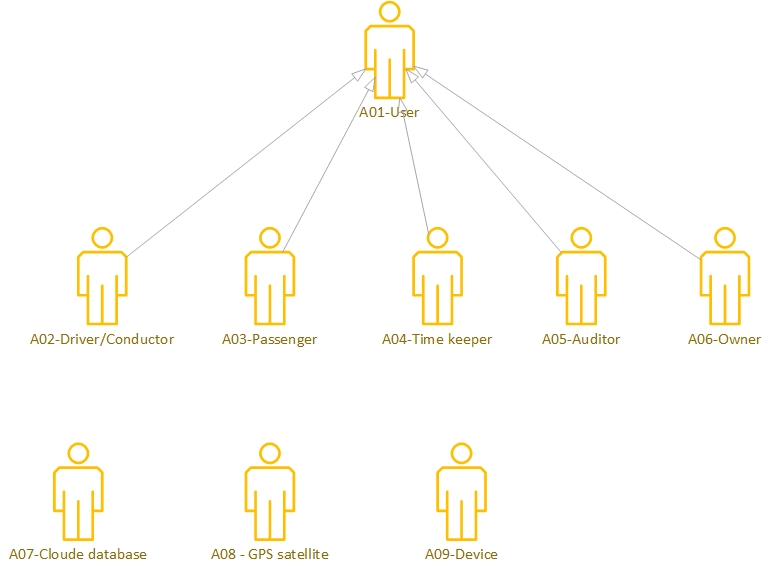


Figure 13 – Use case - Actors

## Register (UC01)

This use case diagram explain how a bus is registered on new system.



Figure 14 – Use case - Register

## Display icon (UC02)

This use case diagram explain how to show the icon of the bus in google map.

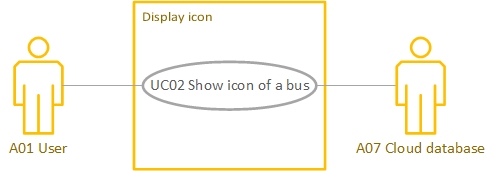


Figure 15 – Use case – Display icon

## Display details (UC03, UC04, UC05, UC06)

This use case diagram explains how to show the relevant details to the user.

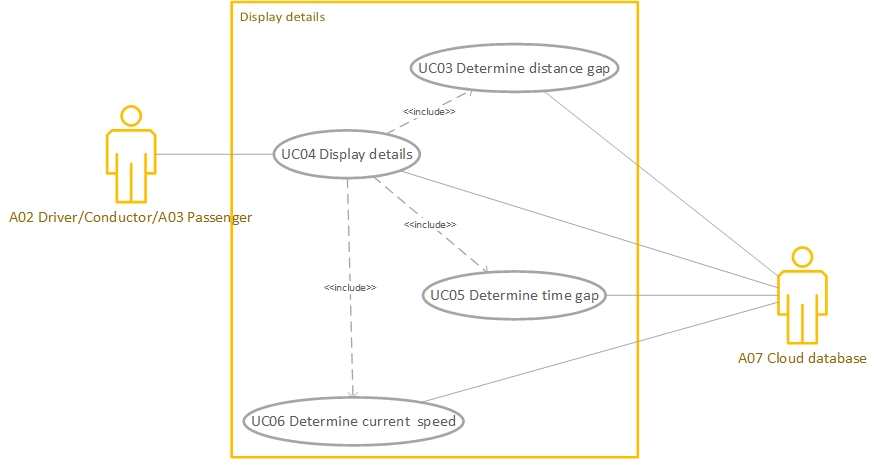


Figure 16 – Use case – Display details

## Reports (UC07, UC08, UC09)

This use case diagram explains how the records are calculated and reports are generated.

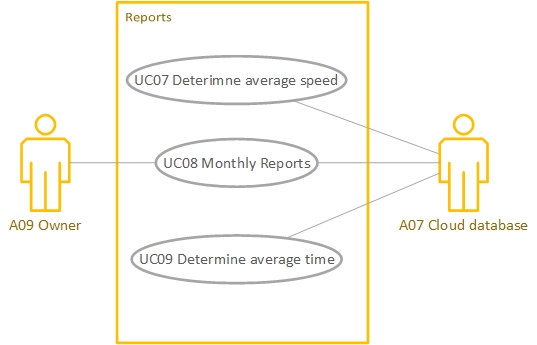


Figure 17 – Use case - Reports

## Search (UC10, UC11)

This use case explains the search functionality of the system.

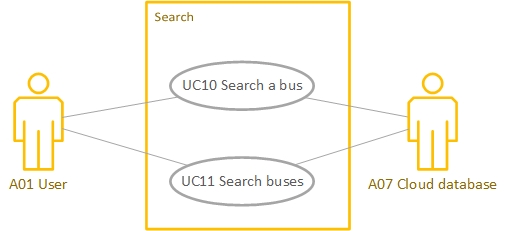


Figure 18 – Use case - Search

## GPS (UC12, UC13)

This use case explains how the GPS coordinates are get and how they passed to the cloud database.

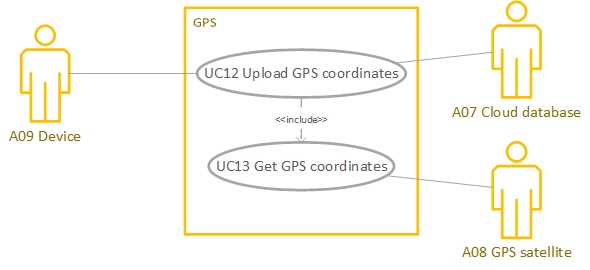


Figure 19 – Use case - GPS

# Use case descriptions

|  |  |
| --- | --- |
| **Thumbnail** | UC01 Register a bus |
| **Description** | A02 Driver/Conductor shall be able to register a bus in Bus Guiding System using registration number (ex. WP JH 3928), route number (ex. 01 Kandy-Colombo), category (Slow/Normal/Semi Luxury/Intercity) and name (optional). |
| **Pre-Conditions** | Not registered  Mobile data enabled |
| **Post-Conditions** | Registered  Mobile data enabled |
| **Actors** | A02 Driver/Conductor  A07 Cloud database |
| **Use Case Relationships** | - |

Table 6 - Use case - Register a bus

|  |  |
| --- | --- |
| **Thumbnail** | UC02 Show icon of a bus |
| **Description** | System shall be able to show an icon for a bus on google map using its GPS coordinates. |
| **Pre-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Post-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Actors** | A01 User  A07 Cloud database |
| **Use Case Relationships** | - |

Table 7 -Use case - Show icon of a bus

|  |  |
| --- | --- |
| **Thumbnail** | UC03 Determine distance gap |
| **Description** | System shall be able to calculate distance between two buses using their GPS coordinates. |
| **Pre-Conditions** | Registered  GPS enabled  Mobile data enabled |
| **Post-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Actors** | A07 Cloud database |
| **Use Case Relationships** | Included in UC04 Display details |

Table 8 -Use case - Determine distance gap

|  |  |
| --- | --- |
| **Thumbnail** | UC04 Display details |
| **Description** | System shall be able to show registration number, current speed, how far from the user, time gap with user at the current speed and name (if any). Registration number, and name (if any) are directly get from the cloud database. Others are determined using other use cases. |
| **Pre-Conditions** | Registered  GPS enabled  Mobile data enabled |
| **Post-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Actors** | A02 Driver/Conductor  A03 Passenger |
| **Use Case Relationships** | UC03 Determine distance gap(include)  UC05 Determine tie gap(include)  UC06 Determine current speed(include) |

Table 9 - Use case - Display details

|  |  |
| --- | --- |
| **Thumbnail** | UC05 Determine time gap |
| **Description** | System shall be able to determine time gap using current speed and distance gap and show in minutes. |
| **Pre-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Post-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Actors** | A07 Cloud database |
| **Use Case Relationships** | Included in UC04 Display details |

Table 10 - Use case - Determine time gap

|  |  |
| --- | --- |
| **Thumbnail** | UC06 Determine current speed |
| **Description** | System shall be able to determine current speed using distance gap and time taken and show in kmpH. |
| **Pre-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Post-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Actors** | A07 Cloud database |
| **Use Case Relationships** | Included in UC04 Display details |

Table 11 - Use case - Determine current speed

|  |  |
| --- | --- |
| **Thumbnail** | UC07 Determine average speed |
| **Description** | System shall be able to determine average speed by using records available on cloud database. Average speed during every run in a month and overall average speed are available on database. |
| **Pre-Conditions** | Registered  Mobile data enabled |
| **Post-Conditions** | Registered  Mobile data enabled |
| **Actors** | A07 Cloud database |
| **Use Case Relationships** | Included in UC08 Monthly reports |

Table 12 - Use case - Determine average speed

|  |  |
| --- | --- |
| **Thumbnail** | UC08 Monthly reports |
| **Description** | System Shall be able to generate reports monthly containing average speed (this month), average time (this month), average speed (overall) and average time (overall). |
| **Pre-Conditions** | Registered  Mobile data enabled |
| **Post-Conditions** | Registered  Mobile data enabled |
| **Actors** | A09 Owner |
| **Use Case Relationships** | UC09 Determine average time(include)  UC07 Determine average speed(include) |

Table 13 - Use case - Monthly reports

|  |  |
| --- | --- |
| **Thumbnail** | UC09 Determine average time |
| **Description** | System shall be able to determine average time by using records available on cloud database. Average time for every run in a month and overall average time are available on database. |
| **Pre-Conditions** | Registered  Mobile data enabled |
| **Post-Conditions** | Registered  Mobile data enabled |
| **Actors** | A07 Cloud database |
| **Use Case Relationships** | Included in UC08 Monthly reports |

Table 14 - Use case - Determine average time

|  |  |
| --- | --- |
| **Thumbnail** | UC10 Search a bus |
| **Description** | A01 User shall be able to search a bus using its registered number or name(optional) |
| **Pre-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Post-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Actors** | A01 User  A07 Cloud database |
| **Use Case Relationships** | - |

Table 15 - Use case - Search a bus

|  |  |
| --- | --- |
| **Thumbnail** | UC11 Search buses |
| **Description** | A01 User Shall be able to show able to filter buses depending on route number, road number, area, trip, owner name, going towards a particular bus stop and category. |
| **Pre-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Post-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Actors** | A01 User  A07 Cloud database |
| **Use Case Relationships** | - |

Table 16 - Use case - Search buses

|  |  |
| --- | --- |
| **Thumbnail** | UC12 Upload GPS coordinates |
| **Description** | A09 Device shall be able to upload GPS coordinates to the A07 Cloud database. |
| **Pre-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Post-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Actors** | A09 Device  A07 Cloud database |
| **Use Case Relationships** | UC13 Get GPS coordinates(include) |

Table 17 - Use case - Upload GPS coordinates

|  |  |
| --- | --- |
| **Thumbnail** | UC13 Get GPS coordinates |
| **Description** | System shall be able to get GPS coordinates from A08 GPS satellite and provide them to A09 Devise |
| **Pre-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Post-Conditions** | Registered  Mobile data enabled  GPS enabled |
| **Actors** | A08 GPS satellite |
| **Use Case Relationships** | Included in UC12 Upload GPS coordinates |

Table 18 -Use case - Get GPS coordinates

# Activity diagrams

## Register a bus

This activity diagram explain how the registering process will happen



Figure 20 - Activity diagram – Register a bus

## Display icon of a bus

This activity diagram explains how the process of showing icon of the bus is happening.

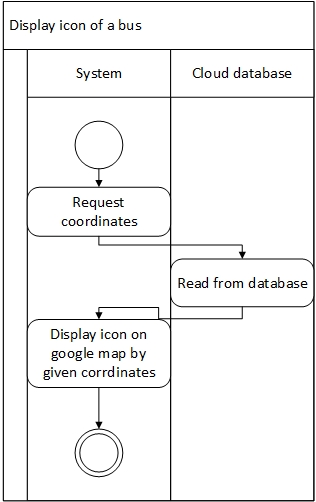


Figure 21 – Activity diagram – display icon of a bus

## Display details

This activity diagram shows the process of displaying relevant details.

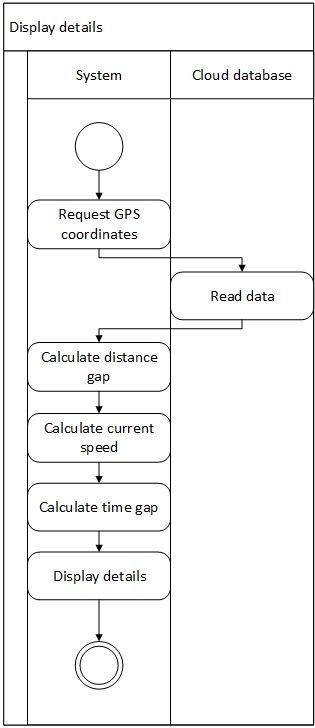


Figure 22 – Activity diagram – Display details

## Search bus/buses

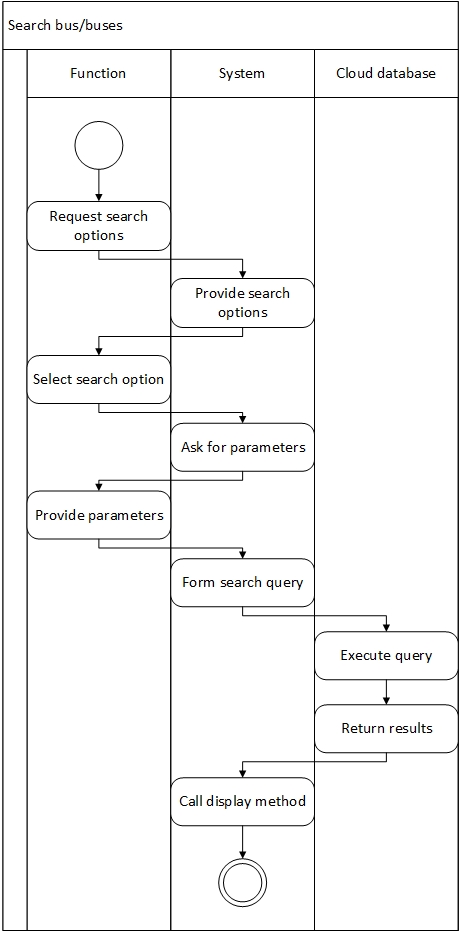


Figure 23 – Activity diagram – Search bus/buses

## GPS

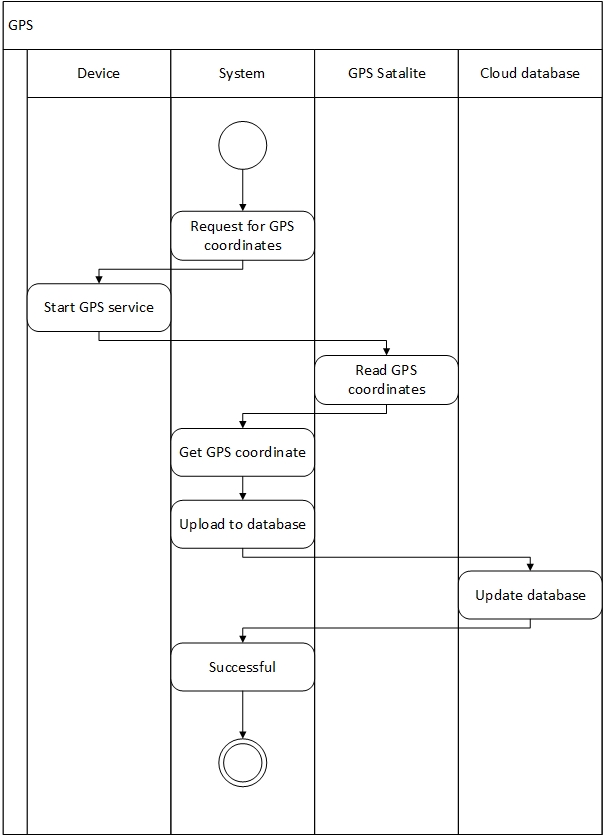


Figure 24 – Activity diagram - GPS

## Monthly Report

This activity diagram shows how the monthly reports are generated.

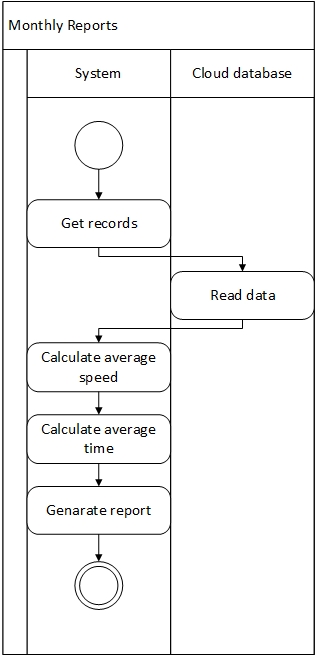


Figure 25 – Activity diagram - Monthly report

# High Level Architecture

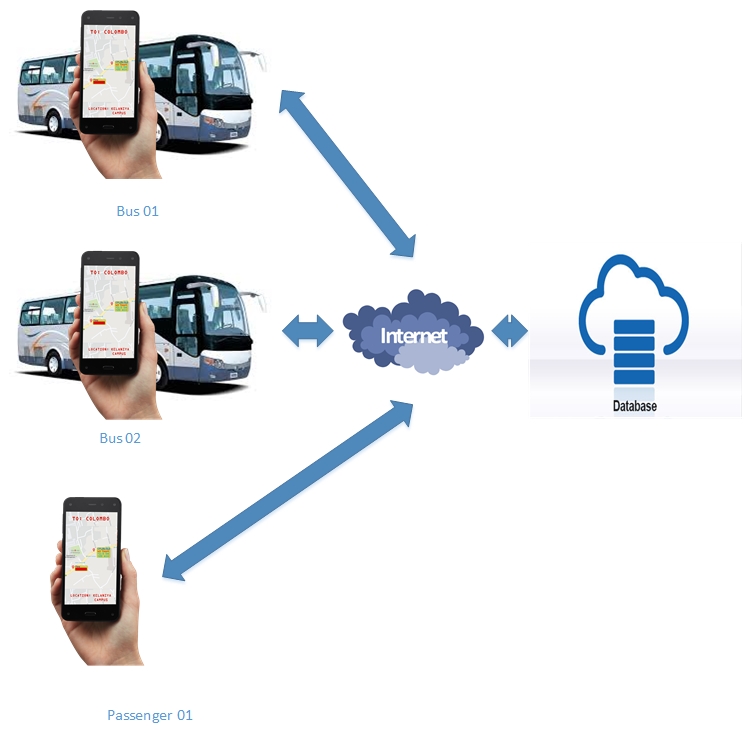


Figure 26 – High level Architecture

Proposed system do not want a load balancer because the system do not have many users. Front end is the mobile application runs on user’s smart phone. Back end is the cloud database. The system is support maintainability with both front and back ends. If it’s required we can have a firewall before cloud database. The system can extend for many users, then we have to use load balancers. System is designed accordingly to 2- tier architecture. Observer design pattern is used, because this system users are the observers of the system. The design is to be implemented using java, it will follow Object Oriented Principles, and so the system will support Maintainability, Extensibility and Flexibility.

# Class Diagram

This is the overall class diagram of the proposed system, which helps to implement the system.

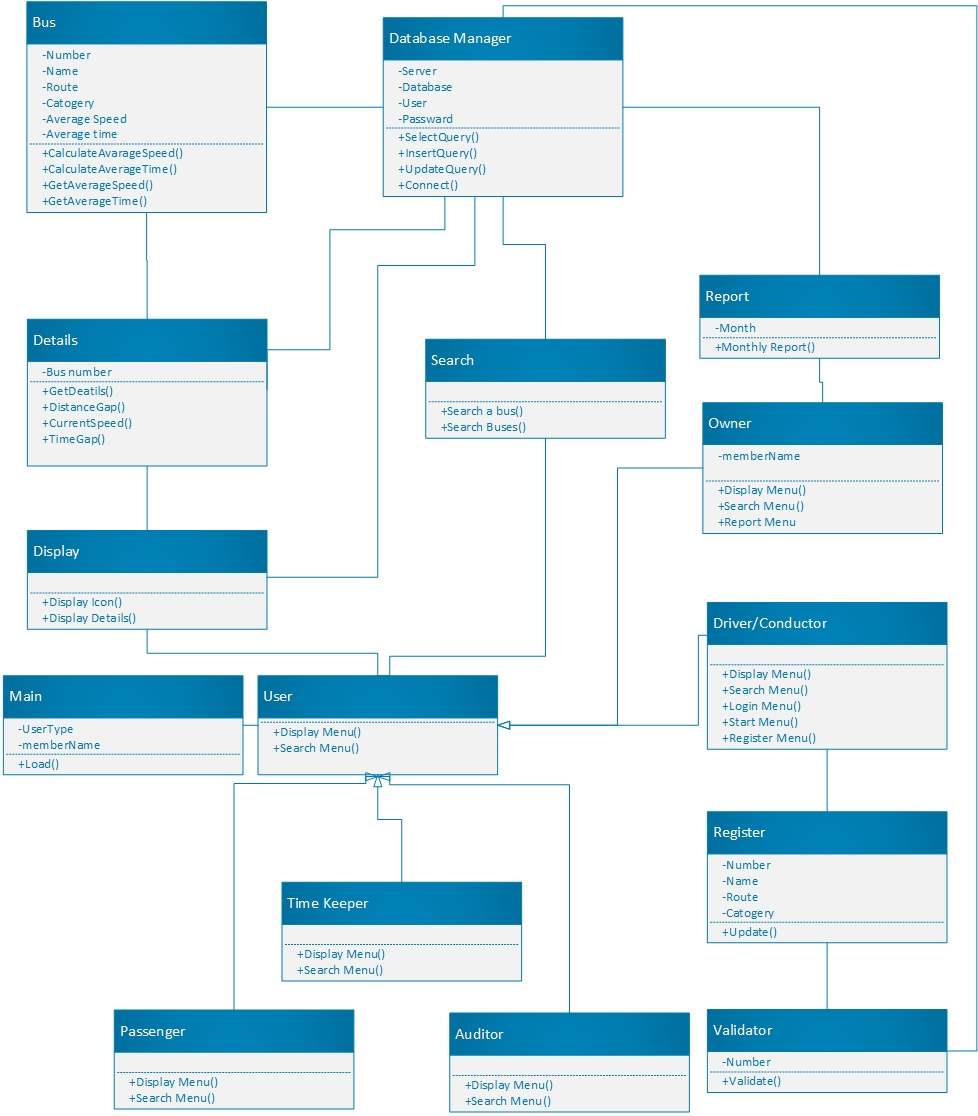


Figure 27 – Class diagram

# Model, View and Controller classes

## Model Classes

* Owner
* Driver/Conductor
* Auditor
* Passenger
* User
* Time Keeper
* Bus

## View Classes

* Display
* Details
* Search
* Report

## Controller Classes

* Validator
* Database Manager
* Main
* Register

# Entity Relationship Diagram

This is the ER diagram which helps to implement database for the new system.

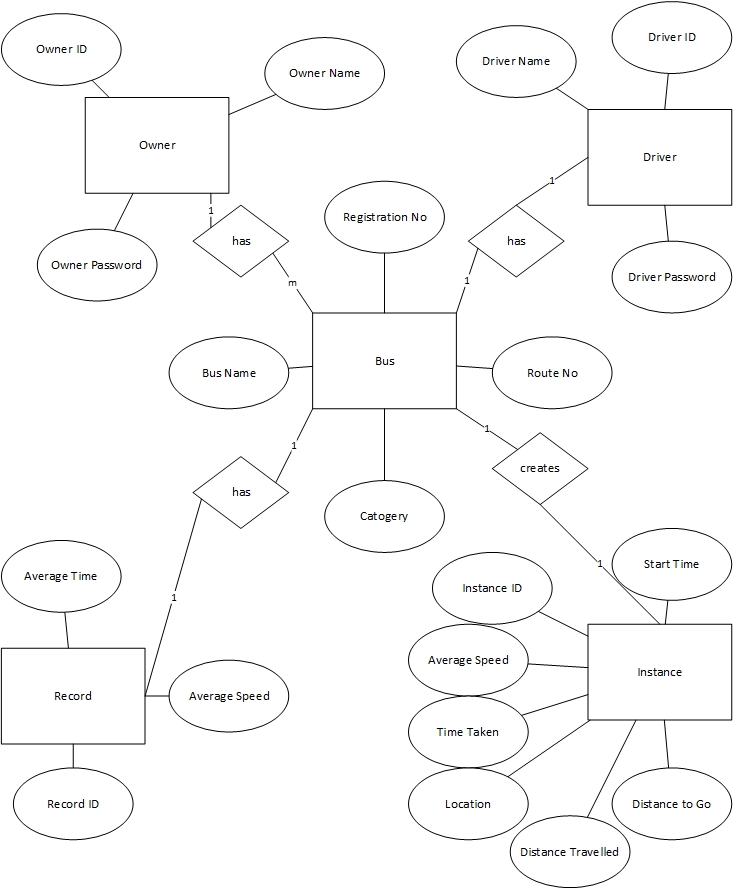


Figure 28 - ER diagram

# Sequence diagrams

These diagrams explain the behavior of the functions. Helps to identify runtime behaviors.

## Register a bus

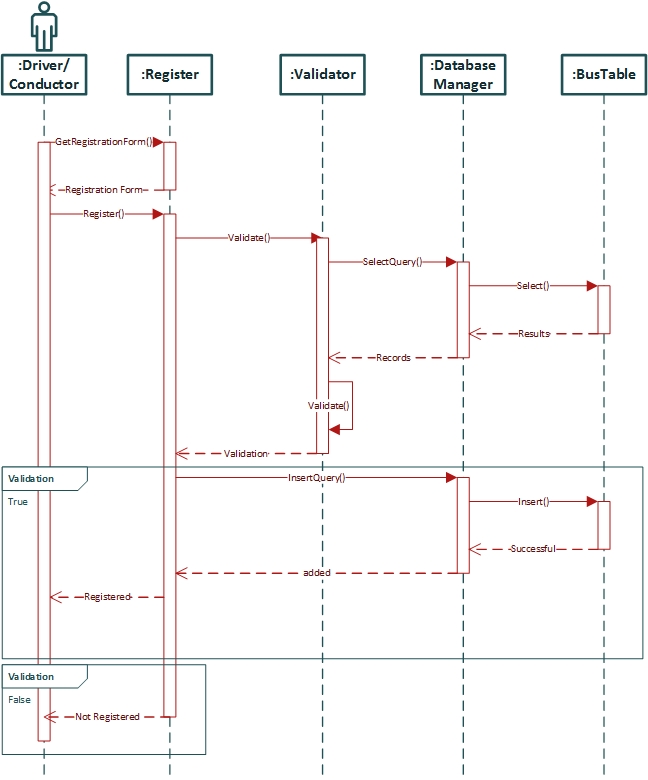


Figure 29 – Sequence diagram – Register a bus

## Show bus icon

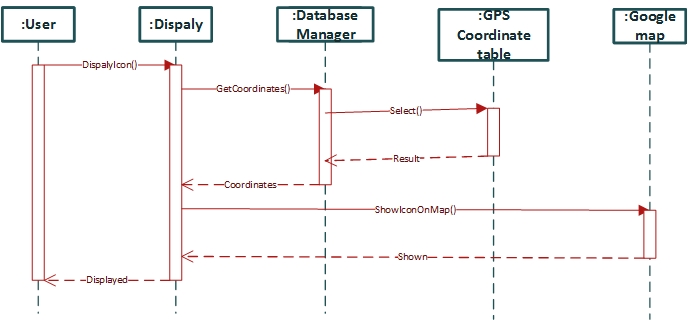


Figure 30 – Sequence diagram – Show bus icon

## Search a bus

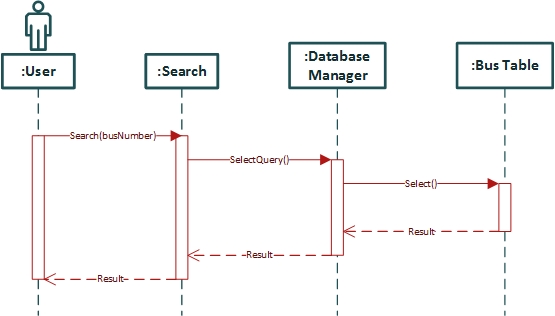


Figure 31 – Sequence diagram – Search a bus

## Search buses

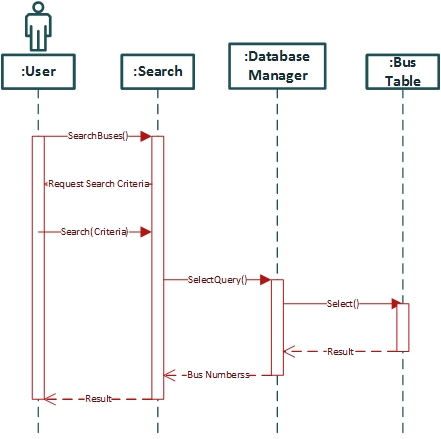


Figure 32 – Sequence diagram – Search buses

## Update GPS coordinates

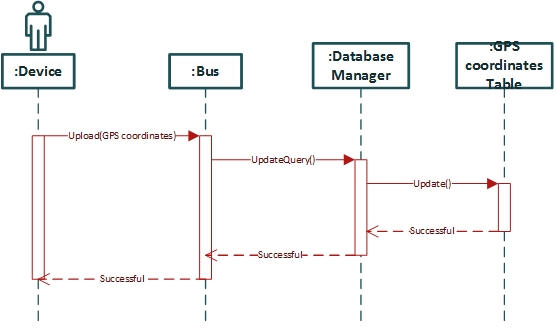


Figure 33 – Sequence diagram – Update GPS coordinates

## Get GPS coordinates

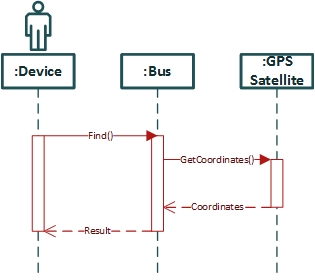


Figure 34 – Sequence diagram – Get GPS coordinates

## Monthly Report

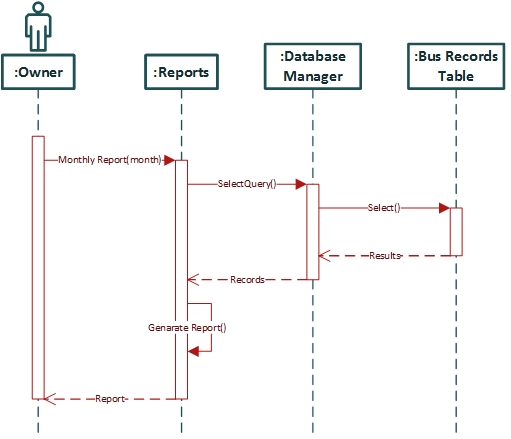


Figure 35 - Sequence diagram – Monthly report

## Get static details

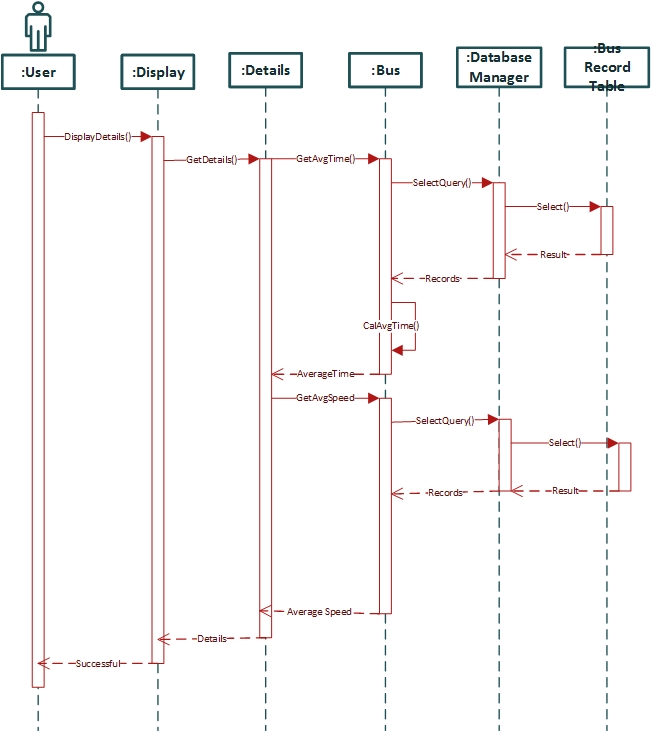


Figure 36 – Sequence diagram – Get static details

## Get dynamic details

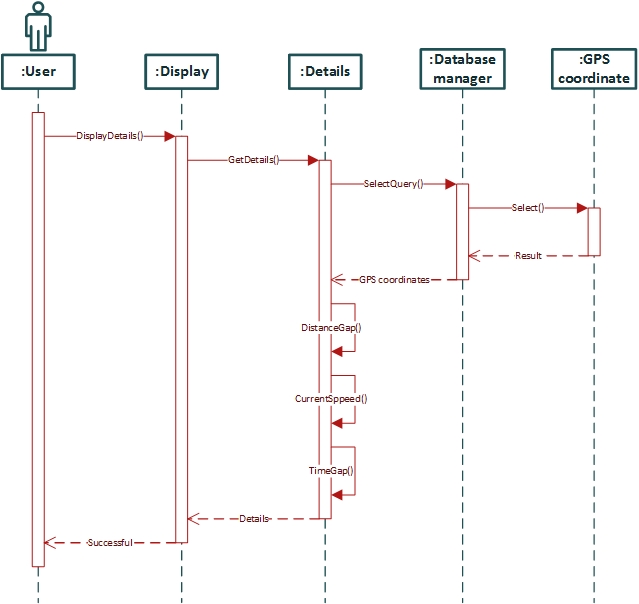
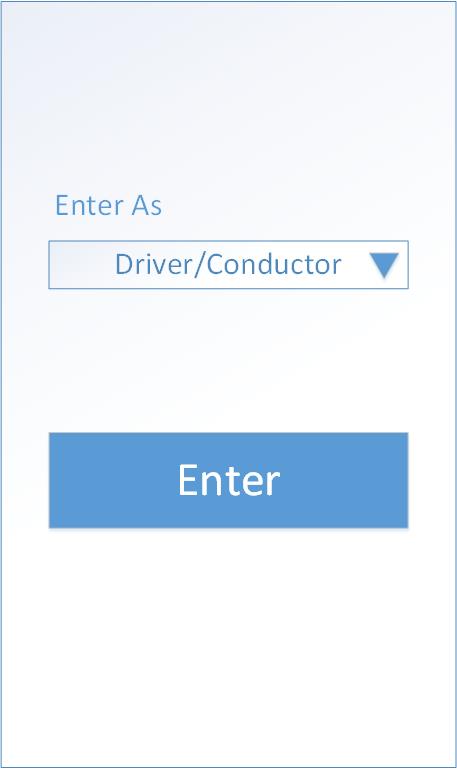


Figure 37 – Sequence diagram – Get dynamic details

# Interfaces

These are the sketches of the interfaces to be implemented.

## Main Interface



This is the first screen. Here user has to select as what kind of user going to use the application, and press enter button.

Figure 38 – Main Interface

## Driver/Conductor Menu



This is the start screen of the driver/conductor. ‘Start’ is there for start the journey. ‘Login’ is there for to login. Accordingly ‘Search’, ‘Display’ and Register functions are there.

Figure 39 – Driver/Conductor Menu

## Login Menu

This is the login Menu. User has to enter username in the ‘Username’ field and password in the ‘Password’ field and press ‘Login’.

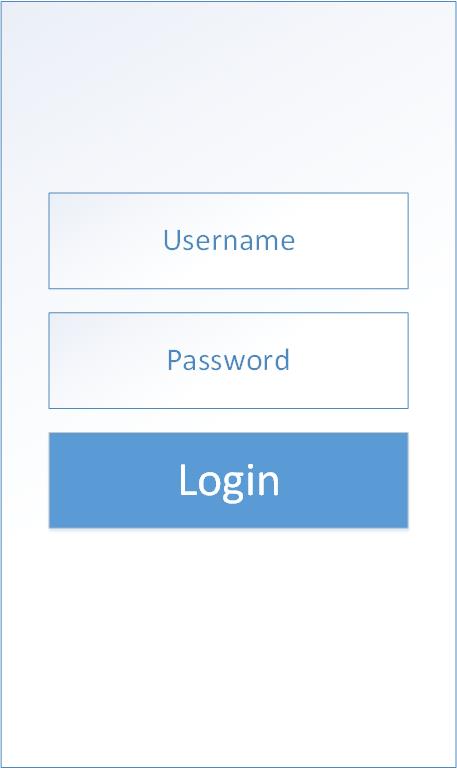


Figure 40 – Login Menu

## Register Menu

This is the Register Menu. User have to fill the four data fields and press ‘Register’.

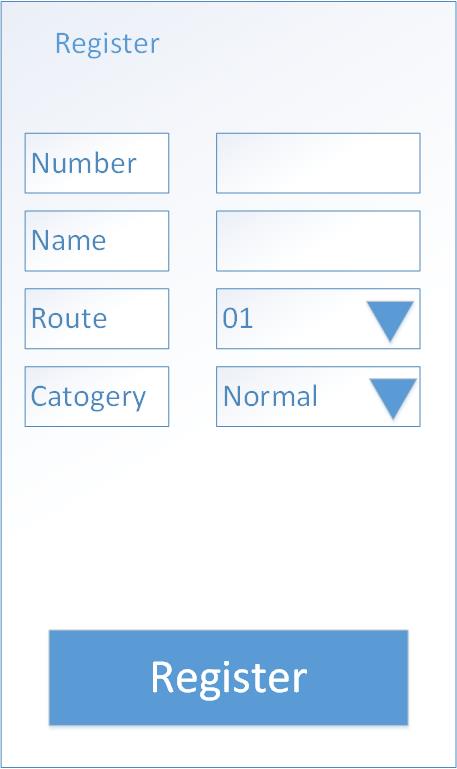


Figure 41 – Register Menu

## Owner Menu

This is the owner Menu. Owner can access these features, ‘Search’, ‘Display’ and Report.



Figure 42 – Owner Menu

## Report Menu



Figure 43 – Report Menu

## Report



Figure 44 – Report

|  |  |  |
| --- | --- | --- |
| Bus Number | Average speed (kmph) | Average time (h) |
| CPJH 4525 | 38.5 | 3.15 |
| CPGF 1256 | 37 | 3.2 |
| CPJH 3525 | 35.5 | 3.3 |

Figure 45 - Report table

Figure 46 – Report graph

## User Menu



Figure 47 – User Menu

## Search Menu



Figure 48 – Search Menu

## Search by Number

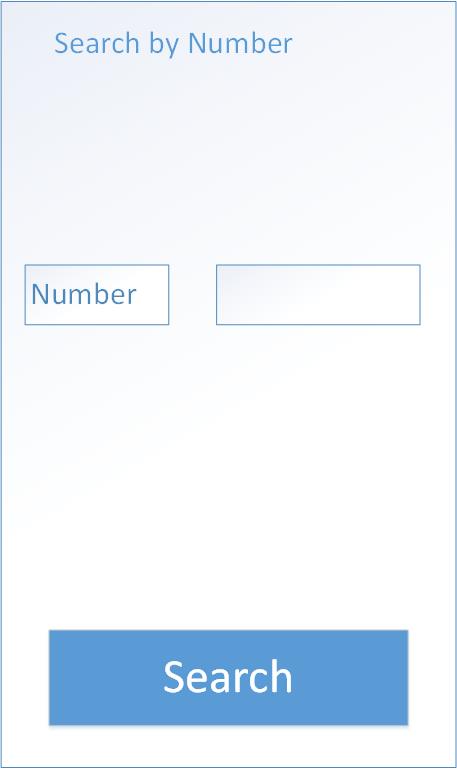


Figure 49 – Search by number

## Search by filter

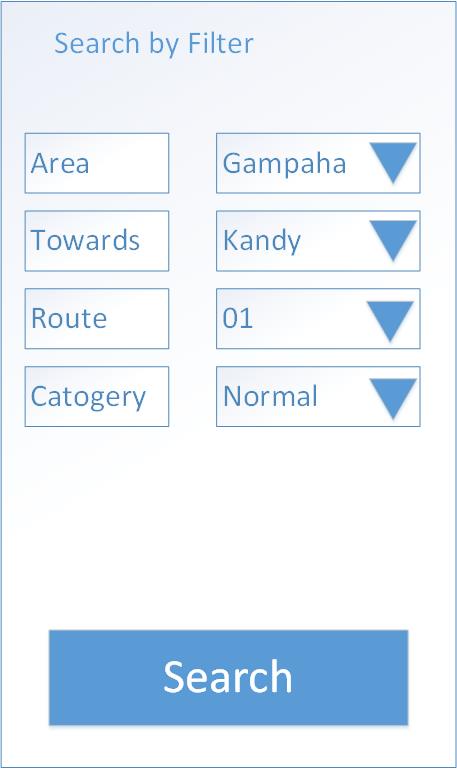


Figure 50 – Search by filter

## Display

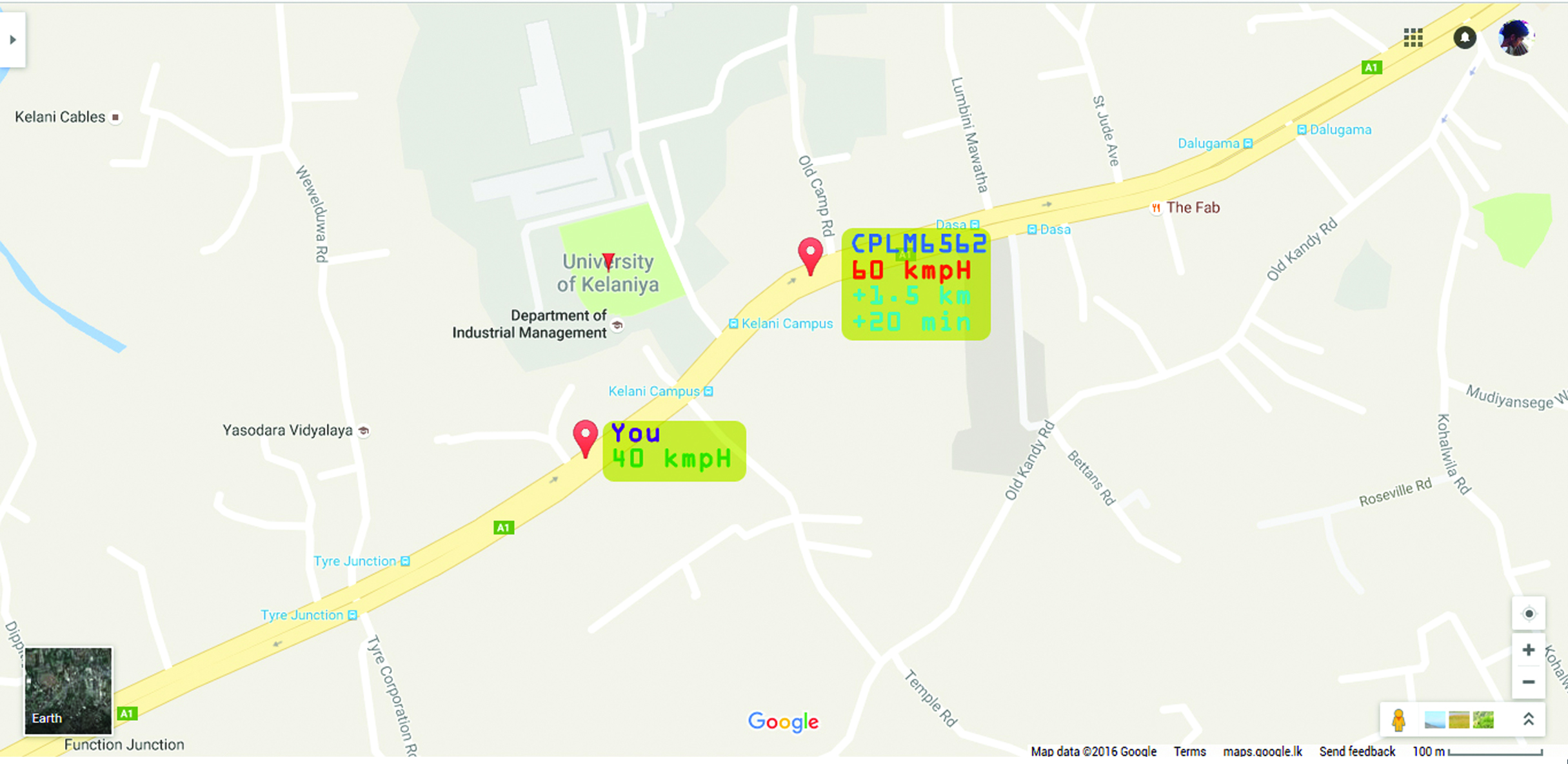


Figure 51 - Display

# Summary

This chapter explains the High level architecture, structural design and dynamic behaviors of the system. It also include the class diagram and entity relationship diagram. System design part is completed and let’s have to go to a conclusion.

CHAPTER IV – DEVELOPMENT

# Introduction

This chapter consist of the development details of the project. It gives a brief introduction to the programing language properties required to implement the Class Diagrams described in a previous chapter and select programming languages that have used in the Easy Park Car Park Management System development. And also it will give brief description of third party components/libraries used. Finally it is consisted of examples of algorithms used in the system.

# Programming Languages and Development Tools

As I wanted a native android app which helps me to use the full power of android device I use android studio as the IDE and Java as the program language.

## Programming Language

### Java

I developed a native android app, so obviously the programming language is Java.

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

The latest version is Java 11, released on September 25, 2018, which follows Java 10 after only six months in line with the new release schedule. Java 8 is still supported but there will be no more security updates for Java 9. Versions earlier than Java 8 are supported by companies on a commercial basis; e.g. by Oracle back to Java 6 as of October 2017 (while they still "highly recommend that you uninstall" pre-Java 8 from at least Windows computers).

## Development Tools and Technologies

### Android Studio

I developed my android native app using android studio as it’s the most powerful IDE to develop native android apps.

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.2.1, which was released in October 2018.

## Third Party Components and Libraries

### Firebase

I use Firebase Authentication for Authentication and Firebase Real Time Database to store real time traffic data.

Firebase is a mobile and web application development platform developed by Firebase, Inc. in 2011, then acquired by Google in 2014.

### Google maps API

To display the map and markers on a map I used Google Maps API.

Google Maps is a web mapping service developed by Google. It offers satellite imagery, street maps, 360° panoramic views of streets (Street View), real-time traffic conditions (Google Traffic), and route planning for traveling by foot, car, bicycle (in beta), or public transportation.

Google Maps began as a C++ desktop program at Where 2 Technologies. In October 2004, the company was acquired by Google, which converted it into a web application. After additional acquisitions of a geospatial data visualization company and a realtime traffic analyzer, Google Maps was launched in February 2005. The service's front end utilizes JavaScript, XML, and Ajax. Google Maps offers an API that allows maps to be embedded on third-party websites,and offers a locator for urban businesses and other organizations in numerous countries around the world. Google Map Maker allowed users to collaboratively expand and update the service's mapping worldwide but was discontinued from March 2017. However, crowdsourced contributions to Google Maps were not discontinued as the company announced those features will be transferred to the Google Local Guides program.

Google Maps' satellite view is a "top-down" or "birds eye" view; most of the high-resolution imagery of cities is aerial photography taken from aircraft flying at 800 to 1,500 feet (240 to 460 m), while most other imagery is from satellites. Much of the available satellite imagery is no more than three years old and is updated on a regular basis. Google Maps used a variant of the Mercator projection, and therefore cannot accurately show areas around the poles.[6] However, in August 2018, the desktop version of Google Maps was updated to show a 3D globe.

The current redesigned version of the desktop application was made available in 2013, alongside the "classic" (pre-2013) version. Google Maps for Android and iOS devices was released in September 2008 and features GPS turn-by-turn navigation along with dedicated parking assistance features. In August 2013, it was determined to be the world's most popular app for smartphones, with over 54% of global smartphone owners using it at least once.

In 2012, Google reported having over 7,100 employees and contractors directly working in mapping.

### Google Play Services Location API

I used Google Play Services Location API to get the GPS location.

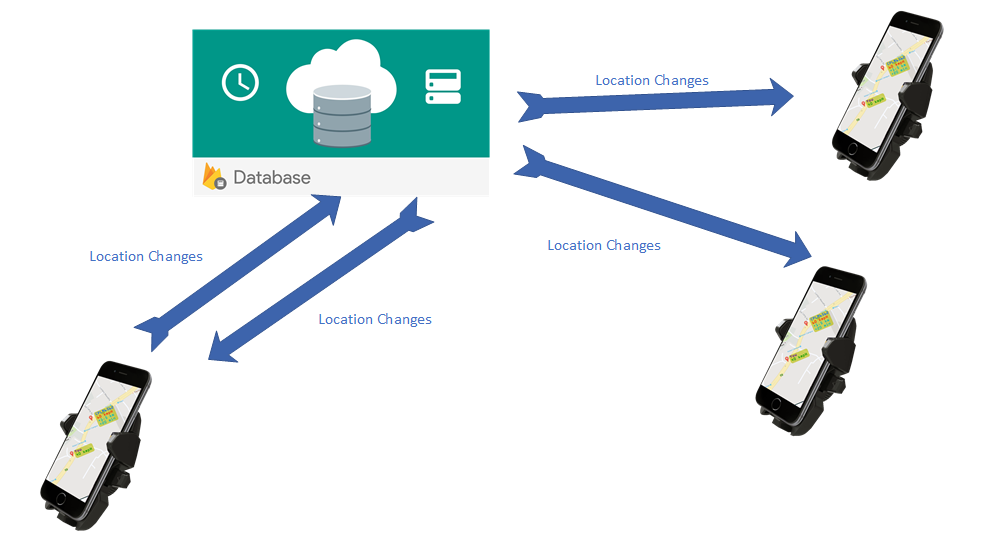
Using the Google Play services location APIs, your app can request the last known location of the user's device. In most cases, you are interested in the user's current location, which is usually equivalent to the last known location of the device.

Specifically, use the fused location provider to retrieve the device's last known location. The fused location provider is one of the location APIs in Google Play services. It manages the underlying location technology and provides a simple API so that you can specify requirements at a high level, like high accuracy or low power. It also optimizes the device's use of battery power.

According to android source code (FusedLocationProvider), Fused Location is actually a location service which combines GPS location and network location to achieve balance between battery consumption and accuracy. It has no connection to the fuse file system.

## Algorithms used in the System

The most important parts are updating the location to firebase real time database and display the marker when firebase database data entry changes.



I have put event listener to listen to location changes, whenever a location change happen the coordination will send to Firebase Real Time Database. Then the event listeners added to Firebase Real Time Database data change event listener will triggered and the code will run to display the location in connected devices.

# Summery

In this chapter, the development aspects of the system has been described. Under that initially the technologies used have been explained and justified and finally the constraints faced during the development process have been described.

CHAPTER V – IMPLEMENTATION

# Introduction

This chapter consists of the implementation details of the project. It will demonstrate the implementation requirements of the Easy park Car Park management System. In addition, this chapter mainly focuses on the implementation specifications such as installation guide and security procedures.

# System Requirements

The minimum hardware and software requirement for the installation of the system as follows.

## Hardware Requirements

Minimum hardware requirements for the system installation.

* Android device with GPS Services and and mobile Internet(3G/HSPA/4G) connection.

## Software Requirements

Operating System

* Android API Level 23 or never (Android 6.0 – Marshmallow)

Services

* Google Play Services 13.0.1 or Higher

# Summery

In this chapter, the developer has discussed about the implementation of the Bus Guiding System. It includes the implementation specifications such as software requirements and hardware requirements installation guide, User manual which guides the user to use the system easily.

CHAPTER VI - TESTING

# Introduction

Software testing aspect of the “Bus Guiding System” is discussed in this chapter. Under this chapter, the strategies and types of testing used and why and how they have been employed to test the system have been discussed. Then it will discuss about the sample test cases, followed by the test reports of the system and results illustrating the severity of the bugs identified and possible solutions.

# Test Plan and Test Strategy

Testing is done in order to align the product with the user. The functionalities and functions are tested in order to improve the quality of the product and improve the functionalities of the product.

The testing plan defines the items to be tested and the functions which are selected based on the importance of the functions, and the risk of the functions on the user’s view point.

Then the test cases were designed corresponding with the use case descriptions. They were executed manually and the results were recorded. The bugs identified were corrected and tested again. In this testing process of this system both black box and white box testing is being used.

## White Box Testing

The testing is mostly carried out while the coding is taking place. Each statement, conditions and decision structures test soon after finishing the coded. The purpose of using white box testing for this system is that white box testing enables to identify small bugs that could be created by the mistakes of coding and due to the errors in the decision logics, and it enables the developers to prevent those errors before integrating the recently developed contents together with the system.

## Black Box Testing

This has been used once after a particular functionality is completely developed. A set of random values used for the testing and the errors and bugs which are generated is identified and fixed. The gessoes over user errors can be tested at this stage.

## Test Types

Under these testing approaches four main test types has used to test this system.

* **Concept Testing**

This is used to check the compatibility of the concepts used in the system and the concepts in the actual business process.

* **Unit Testing**

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

* **Integration Testing**

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing.

* **System testing**

System testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

* **Interface Testing**

Interface testing is being used to test each and every component in the GUIs against their intended purpose.

# Testing Environment

Huwawei Y5 2017 Smartphone – Android 6.0 – Marshmellow

# Sample Test Cases

## User Login

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Test Case Description | Input Data | Expected Output | Status |
| 01.01. | User login – Sign in with Email  Steps:   1. Enter email 2. Enter password 3. Click login | 1. Email 2. Password | Accept the email and password and direct to Continue as activity | Pass |
| 01.02. | User login – Sign in with Google  Steps:   1. Click sign in with Google 2. Select the account | - | Get the user’s Google account, validate and direct to Continue as activity | Pass |

## Continue as

# Summery

In this chapter, the developer has mentioned the test types, which the developer has followed when implementing the test procedure and test cases of the system. The developer has further described test strategies used, test plan, sample test cases and presents the report of the overall test phase.

CHAPTER IV - CONCLUSION

# Introduction

This chapter explains the conclusion of the project. How the solution covers the problems identified and how appropriate the solution is.

# Degree of objectives met

The design is covered all the objectives explained in the requirements with appropriate accuracy, efficiency. This design does not cover facilities like a count of the passengers in the bus, amount of fuel in the fuel tank, total money earned by a run.

# Usability, accessibility, reliability and friendliness

The usability of this this system is high, because it is very much usable for divers, conductors, passengers, bus owners and ticket checker.

Accessibility of this system is high, because it can run on any Android smart phone which has GPS services. It only needs mobile data service and GPS service to run the application.

Reliability of the system is high, because it provides

1. 98% accuracy on distance gaps.
2. 80% accuracy on average speeds.
3. 80% accuracy on current speeds.
4. 75% accuracy on time gaps.

Friendliness of this system is high when considering other complex mobile applications, because the users are ordinary people like bus drivers and conductors. The application provides simple GUIs with lesser controls in the screen.

# User’s response

User’s response towards this app will be high, because it will help them to make their job efficient, accurate and simple. Passengers will be grateful with this app because they can have many benefits.

# Limitations and drawbacks

As explained in the objectives met by this app will not cover facilities like; a count of the passengers in the bus, amount of fuel in the fuel tank, total money earned by a run. Those are the major drawbacks. The time taken to get GPS coordinates and upload to database is a minus point it makes limitations in accuracy and efficiency.

# Further modifications, improvements and extensions possible

As we explained, before we can implement a method to count passengers in a bus using IR sensors at the footboard. Therefore, the count can be taken 100% accurately and through a Bluetooth model it can be transferred to the driver’s smart phone. Thus, the users can get the count of passengers in a bus.

With the faster 4G network we can increase the accuracy and efficiency of the app.

As an extension we can provide a mechanism to pay the bus ticket buy our app itself. So that, the income from a turn can be calculated.

With the feedback from the users we can improve our app towards developing more user friendly interfaces and functions.

From users we can improve our app towards more user friendly interfaces and functions.

# References

1. Google Maps API

<https://developers.google.com/maps/>

1. Arduino Platform

<https://www.arduino.cc/>

1. Price Lists (Arduino, GPS Tracker)

<http://aliexpress.com/>

1. Android development Basics

https://developer.android.com/index.html