Project Report

of

REALTIME BUS TRACKING AND SEAT BOOKING SYSTEM

R J L M Herath

715903311

Department Computer Science

Faculty of Natural Sciences

The Open University of Sri Lanka

FOR

THE OPEN UNIVERSITY OF SRI LANKA

Project Supervisor : Mrs. Wasana Uduwela

Name : R J L M.Herath  
Reg. No. : 715903311  
Address : 333/A, Pitawela, Hewadiwela, Rambukkana.  
Contact No. : 071 7932194/076 3683238  
Email : [judelakshithaousl@gmail.com/](mailto:judelakshithaousl@gmail.com/) s15008693@ousl.lk

**DECLARATION**

The undersigned hereby recommend to the Department of Computer Science acceptance of the final project report,

Realtime Bus Tracking and Seat Booking System

…………………………

Signature

R J L M.Herath

**Project Supervisor:**

Mrs. Wasana Uduwela ………………………… ……………….…..

Signature Date

**DEDICATION**

***Special dedicated to Department of Computer Science, university academic staff, my family, my friends,***

***For all your care and the support.***

*Sincerely.*

*R J L M Herath*

**ACKNOWLEDGEMENTS**

I would like to express an enormous gratitude to my project supervisor/coordinator Mrs. Wasana Uduwela, the Lecturer, Department of Computer Science, Open university of Sri Lanka. For all her help, her continuous guidance, motivation, and encouragement. Working with her has been a great honor for me and an excellent professional and personal experience. She always gives me valued perspective from the expert point of view and keeping the project firmly oriented towards the end goal.

I am also thankful to Mr. Sathsara Gunarathne, visiting Lecturer, Department of Computer Science, Open University of Sri Lanka for his invaluable help during my project period and his industrial experience brought my Project to the success.

I would also like to thank Mrs. Champika Megasooriya, the Lecturer of Kandy Regional Center and all the academic staff of the Open University for giving me encouragement. I could also not forget my colleagues always support to finish my project in a nice way.

**TABLE OF CONTENT**

[Chapter 01 1](#_Toc57981323)

[1 INTRODUCTION 1](#_Toc57981324)

[1.1 Introduction 1](#_Toc57981325)

[1.2 Problem 2](#_Toc57981326)

[1.3 Background and Motivation 3](#_Toc57981327)

[1.4 Aim 3](#_Toc57981328)

[1.5 Objectives 3](#_Toc57981329)

[1.6 Chapter Breakdown 4](#_Toc57981330)

[1.6.1 Introduction 4](#_Toc57981331)

[1.6.2 System Analysis 4](#_Toc57981332)

[1.6.3 System Design 4](#_Toc57981333)

[1.6.4 System Development 4](#_Toc57981334)

[1.6.5 Conclusion and further development 4](#_Toc57981335)

[1.6.6 Reference 4](#_Toc57981336)

[Chapter 02 5](#_Toc57981337)

[2 CURRENT SYSTEM ANALYSIS 5](#_Toc57981338)

[2.1 Introduction 5](#_Toc57981339)

[2.2 Collect Requirements 5](#_Toc57981340)

[2.3 Use case of the manual System 5](#_Toc57981341)

[2.4 Use case Catalogue 6](#_Toc57981342)

[2.5 User Requirements 9](#_Toc57981343)

[2.6 Summary 10](#_Toc57981344)

[Chapter 03 11](#_Toc57981345)

[3 SYSTEM DESIGN 11](#_Toc57981346)

[3.1 Introduction 11](#_Toc57981347)

[3.2 Literature Survey 11](#_Toc57981348)

[3.3 System Architecture of the proposed System 13](#_Toc57981349)

[3.4 Software requirement specification 13](#_Toc57981350)

[3.5 Functional requirements 13](#_Toc57981351)

[3.6 Non-functional requirements 14](#_Toc57981352)

[3.7 Use Case 14](#_Toc57981353)

[3.7.1 Actors 15](#_Toc57981354)

[3.7.2 Use case narratives 15](#_Toc57981355)

[3.7.3 Use case Catalogue 15](#_Toc57981356)

[3.8 Class Diagrams 22](#_Toc57981357)

[3.9 ER Diagram (Entity Relationship Diagram) 23](#_Toc57981358)

[3.10 Activity Diagrams 24](#_Toc57981359)

[3.10.1 Passenger Login 24](#_Toc57981360)

[3.10.2 Driver/Conductor Login 25](#_Toc57981361)

[3.10.3 Passenger /Driver Registration 26](#_Toc57981362)

[3.10.4 Booking a Seat 27](#_Toc57981363)

[3.11 Sequence Diagram of Login 28](#_Toc57981364)

[3.12 Sequence Diagram of Booking 29](#_Toc57981365)

[Chapter 4 30](#_Toc57981366)

[4 SYSTEM DEVELOPMENT 30](#_Toc57981367)

[4.1 Introduction 30](#_Toc57981368)

[4.2 Database Implementation 30](#_Toc57981369)

[4.2.1 Tables 30](#_Toc57981370)

[4.2.2 Android code sample for creating firebase 33](#_Toc57981371)

[4.3 System Implementation 34](#_Toc57981372)

[4.3.1 Signup Activity android code 34](#_Toc57981373)

[4.3.2 Login Activity android code 37](#_Toc57981374)

[4.3.3 Map Activity android code 39](#_Toc57981375)

[4.4 Interface Implementation 45](#_Toc57981376)

[4.4.1 Login Window 45](#_Toc57981377)

[4.4.2 Registration Window 46](#_Toc57981378)

[4.4.3 Bus Registration 47](#_Toc57981379)

[4.4.4 Bus Registration List 48](#_Toc57981380)

[4.4.5 Driver Map Window 49](#_Toc57981381)

[4.4.6 Seat update Window 50](#_Toc57981382)

[4.4.7 Passenger Map Window 51](#_Toc57981383)

[Chapter 5 52](#_Toc57981384)

[5 CONCLUSION AND FURTHER DEVELOPMENT 52](#_Toc57981385)

[5.1 Introduction 52](#_Toc57981386)

[5.2 Project 52](#_Toc57981387)

[5.3 System 52](#_Toc57981388)

[5.4 Conclusion 52](#_Toc57981389)

[5.5 Future Modifications 53](#_Toc57981390)

[Chapter 6 54](#_Toc57981391)

[6 References 54](#_Toc57981392)

**List of Abbreviations**

CSS Cascading Style Sheet

DB Database

GPS Global Position System

HTML Hypertext Markup Language

IOS Apple Operating System

OS Operating System

OUSL Open University of Sri Lanka

RAM Random Access Memory

UI User Interface

Chapter 01

# INTRODUCTION

## Introduction

Mobile applications are commonly named by people as apps and they are run on smart devices like phones and tablet Pcs. These apps use to access the features that provided by the websites but in a more secure and more reliable way. Mostly apps are smaller in size and consume less memory to store in smart phones and tablets. People can access the services that already provide by the website but in more quickly because they do not need to run the browser, type the website address, and get the service. They just need to launch the app, then all the services are load to the mobile window. Basically, there are two type of mobile apps divided by the developing technology.

Most of apps are built for the specific OS like Android, Windows, iOS, and BlackBerry, called as Native Apps. These apps have better performance with attractive UI. Developers have to use their sharp knowledge with skill to design and develop a bug free app. Because of the complexity, these apps require some memory from the device storage and from the RAM.

Other kind of apps are identified as Web Apps, use the web technology (HTML5, CSS) to provide the service. These apps consume less memory because services run through the browser. Once a user opens a Web App, all the information saved on a server-based database. Because of this matter, user should have a stable internet connection to use the service from Web Apps.

The role of the road sector in a country plays a cautious role in boosting the economy of a country significantly. There is a significant improvement in the road sector in Sri Lanka with the development of the construction industry after the 30-year of civil war. Hence, the national road master plan’s focus is on building expressways, improving expressways, rehabilitating roads in particular rural area and building expressway. Meanwhile, the expressway was a new experience for Sri Lanka which would bring great benefits to the lives of the people. The Southern expressway from Kottawa to Galle was introduced to Sri Lanka in 2011. Since then, the expansion of the southern expressway to Hambanthota, the Outer circular expressway, and the Katunayake expressway has been seen in line with the country’s demand for the Sri Lankan road sector. Moreover, the Central expressway is in the proposal of the country development plans. With the advantages of the expressway, including increased safety, comfort and convenience for drivers and passengers and the low vehicle operating costs, people are constantly using expressway.

The government has introduced a bus service to improve passenger use of the expressway as it is limited to four-wheeled vehicles. However, it is limited only for few major destination points of the expressways (ex: - Kottawa to Galle, Kottawa to Matara etc.). The use of the expressway sometimes reduces the distance and time of the journey rather than the normal routes. For example, it is better to use the expressway than to use either the railway system or the Kandy road (to go to Petah and take the Maharagama bus, approximately the distance is 16.8 Km and the time to go to the Maharagama around 50 minutes). Meanwhile, the private bus sector absorbs this opportunity by introducing more routes with the demand of the passengers. By now the private bus sector has taken the opportunity to introduce more routes to meet the demand of passengers. In fact, it is a cost-effective and time-saving way to travelers to reach their destinations, especially for the office crowd.

Thus, there is a great demand for an automated solution to track the buses and to reserve the seats, both from bus owners and passengers. Since this service requested when the passengers are on the roads, the best solution is to go for a mobile application rather than a web application. Further, Mobile application is considered as the most suitable solution as the usage of mobile devices in Sri Lanka is very high. In the year 2020 January, there were 31.80 million mobile connections in Sri Lanka; out of them the use of the smartphones also has a high percentage. The complete system enables the passenger to track the bus using GPS technology and reserve the seat as well making it easier for drivers and conductors too. Due to the large use of Android devices in Sri Lanka, the proposed system is built for Android users.

## Problem

Both private and government expressway bus system operates on a timely basis and the people also have adapted to the schedules. Due to the huge demand, some people on the road cannot enter the express way buses on the road, especially during the office hours, unless they are at the starting point. This has led to create a group of unsatisfied customers who attempt to find the alternatives. Alternatively, drivers and conductors receive phone calls to reserve passenger seats and to know the current location of the bus, which is a major obstacle to their job performance. Moreover, due to the dynamic bus schedule, customers need to maintain a large number of telephone numbers and calls until they receive the bus number that matches with their schedules as well as at the request of passengers, the drivers must find a different bus schedules too. Therefore, it was discouraged to pick up the phone to reserve seats for passengers by drivers. Also, it is very difficult for drivers to find a reserved passenger (the location and the right passenger), as there is a huge demand for limited seats at bus stops and there is a problem with passenger identification. The headaches of these options caused the passenger group to abandon this valuable service, which was a financial loss for bus owners.

## Background and Motivation

There is a great demand for an automated solution to track the buses and to reserve the seats, both from bus owners and passengers. Since this service requested when the passengers are on the roads, the best solution is to go for a mobile application rather than a web application. Further, Mobile application is considered as the most suitable solution as the usage of mobile devices in Sri Lanka is very high. In the year 2020 January, there were 31.80 million mobile connections in Sri Lanka; out of them the use of the smartphones also has a high percentage. The complete system enables the passenger to track the bus using GPS technology and reserve the seat as well making it easier for drivers and conductors too. Due to the large use of Android devices in Sri Lanka, the proposed system is built for Android users.

## Aim

The aim of the Vehicle Tracking and seat Booking automated system is to develop a mobile application with facility of monitor the real-time location of the bus and reserve seats as the request of the passenger.

## Objectives

This project developed a mobile application to automate the manual process of seat booking for the public transport service. The existing system is running with mobile phone conversation and passengers call to the conductor and asking for the current location and available seats on the bus at the current moment.

Objectives: -

-To investigate and analyze the current booking system and identify the real requirements of the passengers and drivers/conductors.

-To propose a solution for passengers using the modern mobile phone technology

-To design and develop the proposed mobile system, using the android technology

-To validate the proposed system that the passenger can reverse seat using the mobile phone technology.

Further, ultimate results for the transport systems are,

Using the mobile application people do not need to call to the driver or conductor, the real time location of the bus is display on the mobile phone screen.

Not only the real-time location, but the application also displays the available seat count, and the passenger wants a book a seat, there is an option to request a seat.

## Chapter Breakdown

### Introduction

The introduction chapter consists with a brief introduction of the project. What is the problem I address and what are the major functionalities of the project is mentioned. Finally add a chapter breakdown to get rough idea about the content of the thesis.

### System Analysis

This is the chapter I talk about the literature survey and feasibility study for the current system. And I explain the existing use case with suitable diagram.

### System Design

System design is the chapter I discussed about the design of the proposed system. It is explained using system architecture and mentioning functional and nonfunctional requirement. All the diagrams related to the proposed system are also added in this chapter. Use case diagram, Class Diagram, ER Diagram, and a sequence diagram for the proposed system is attached with System design chapter.

### System Development

This chapter discussed about the development strategy for proposed system. In this chapter I talk about android application, database and interface development using related diagrams. I add several interfaces of the developed system with simple explanation.

### Conclusion and further development

This is the chapter I conclude the android application for the vehicle tracking and seat booking system. And it is discussed about the further development of this system.

### Reference

References are mentioned in this chapter.

Chapter 02

# CURRENT SYSTEM ANALYSIS

## Introduction

in this chapter, analyze the current system and collect the requirements from the passengers, drivers, conductors, and bus owners to get a broad idea of the existing system. The use-case diagram has drawn to get a clear image of the existing system.

## Collect Requirements

To understand the existing system, I interviewed passengers which the route was from Kadawata to Gampaha. My project supervisor, Miss. Wasana Uduwela gave me a broad idea about the existing system and the functionality of the system.

## Use case of the manual System

Diagram

Description automatically generatedThe existing system has followed the below procedure done by the passenger and conductor. Following figure is the Use case diagram for the existing seat booking system.

Figure 2.3.1- Use case diagram of the existing manual booking system

Figure .4.1 - Use Case Diagram for Existing seat booking system

## Use case Catalogue

|  |  |  |
| --- | --- | --- |
| **Use case:** | Make a phone call | |
| **Actors:** | Passenger | |
| **Purpose:** | Know the location of the bus | |
| **Overview:** | Using the mobile phone and make a call to the driver or conductor | |
| **Type:** | Essential | |
| **Preconditions:** | should have the driver or conductor mobile phone number | |
| **Post conditions:** | Got to know the bus is running on the route on the day | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Passenger Actions** | | **Driver/Conductor Response** |
| 1. Call to the driver/conductor   3.Asking the bus is running on the route, time to reach to the bus stop | | 2.Answer to the phone call  4.inform whether running or not, reaching time |

|  |  |  |
| --- | --- | --- |
| **Use case:** | Ask the current location | |
| **Actors:** | Passenger | |
| **Purpose:** | To know the current location of the bus | |
| **Overview:** | Using the phone call, asking about the exact location of the bus | |
| **Type:** | Essential | |
| **Preconditions:** | Passenger wants to know the current location of the bus to get into the bus from when it reaches to the passenger’s regular bus stop. | |
| **Postconditions:** | Passengers get to know the current location of the bus. | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Passenger Actions** | | **Driver/conductor Response** |
| 1. Call and ask about the current location | | 1. Give the location information of the bus |

|  |  |  |
| --- | --- | --- |
| **Use case:** | Request a seat | |
| **Actor:** | Passenger | |
| **Purpose:** | Booking a seat of the bus | |
| **Overview:** | Passenger wants to book a seat | |
| **Type:** | Essential | |
| **Preconditions:** | There should be availability of seats to book | |
| **Post conditions:** | Seats booked by the passenger | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Passenger Actions** | | **Driver/conductor Response** |
| 1. Asking for a reservation a seat | | 1. Check the availability |

|  |  |  |
| --- | --- | --- |
| **Use case:** | Reserve a seat | |
| **Actors:** | Driver/conductor | |
| **Purpose:** | Reserve a seat to the passenger | |
| **Overview:** | Passengers request a seat and driver/conductor reserve a seat if available | |
| **Type:** | essential | |
| **Preconditions:** | There should be available seats on the bus at the requesting moment | |
| **Post conditions:** | Driver/conductor reserve a seat to the passenger | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Actor Actions** | | **Driver/conductor Response** |
| 1. Passenger request a seat | | 2.check the availability of seats  3.if available reserve seat for the passenger  4.if not available inform it to the passenger |

|  |  |  |
| --- | --- | --- |
| **Use case:** | Call back to the passenger | |
| **Actors:** | Driver/conductor | |
| **Purpose:** | Call back and inform the location of the bus | |
| **Overview:** | Driver/conductor make a phone call and let the passenger know the current location of the bus | |
| **Type:** | essential | |
| **Preconditions:** | Passenger must be reserved a seat | |
| **Post conditions:** | Passenger knows the bus is near to the bus stop | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Passenger Actions** | | **Driver/conductor Response** |
| 2.make a confirmation of the bus location | | 1.call back to the passenger and letting know the current location of the bus |

## User Requirements

* Passenger wants to know the real-time location of the bus
* Passenger wants to know are there any available seats on the bus
* Passenger wants to book seats if there are seats to book
* Passenger wants to know the time of the bus that reach to the bus stop

## Summary

In this chapter we talked about the system analysis of the projected system. Identification of the past attempts of this kind of systems and collect requirement from the current system is complete in this phase.

According to the analysis of the existing system, I drown create a use case diagram for the existing system.

The next chapter is System design, and we are talking about the system architecture and use case diagram and class diagram for the proposed System.

.

Chapter 03

# SYSTEM DESIGN

## Introduction

The proposed system is totally depending on the mobile device’s services and Firebase functions. The system design will discuss in this chapter using use case diagrams, some use case narratives, some Activity diagrams, and system architecture. Functional requirements and non-functional requirements of the system are listed under requirement specification section.

## Literature Survey

There are some apps on Play Store and Apple Apps Store for reservation seats but currently do not function as well as related purpose. One of these apps support booking a seat before only 24 hours from the departure time. “SLTB EXPRESS” and “Mybooking.lk” android apps only support to long distance (Inter provincial) travelling and it should book seats before 24 hours. The National Transport Board has recently launched an app MYBUS-SL, and its only shows the real time location of the bus and cannot find available seats on the bus. Because of that there is a good marketplace to launch a mobile app with real time tracking and seat booking facility in Sri Lanka.

GPS is now available in smart phones and supports for the maps to find places and directions to the destinations. Using this service, we can find out current location of the device more accurately. This service is widely use in the world for the transport and online delivery systems.

Android owned by Google and Google Map support the GPS function for the map functions. Firebase is a database maintain by the Google and it is most suitable to store and retrieve location data more accurately and timely.

The usage of android devices is larger than IOS devices, because there are many brands using android as the OS, can purchase affordable prices. The proposed system is designed according to the usage of mobile devices in the country.

Android Studio is the tool supports to android app development.

The National Transport Board has established GPS tracking system to the public transport system to track the real-time location of the buses, but it is not applied to all the busses around the country. Also, there is a practical problem regarding the application of the GPS system to the bus, if a bus breakdown on the road, another bus should replace to cover the route and maintain a stable transport system on the route. In this circumstance the replaced bus has systemized for another route. When tracking the location of the replaced bus, it shows different location. So, the main database should be updated according to the route.

|  |  |  |  |
| --- | --- | --- | --- |
| **REQUIREMENTS** | **SLTB EXPRESS** | **MYBUS-SL** | **Mybooking.lk** |
| Tracking the real-time location of the bus | No | No | No |
| Provide booking Service | Yes | Yes | Yes |
| Provide real-time booking service | No | No | No |
| View the real-time available seat count on the bus | No | No | No |

Figure .2.1- Systematic Review of current Apps

## System Architecture of the proposed System

Figure .2.2- Architecture of the Proposed System

## Software requirement specification

This section describes the services which are expected by passengers from the driver/conductor also from the system.

## Functional requirements

Functional requirements define the basic system behavior.

Followings are the functional requirements of the proposed system.

* App shall be able to register the bus, passengers, conductors, and drivers.
* App shall be able to approve login to the registered passengers, conductors, and drivers.
* App shall be able to collect real-time location data on the firebase.
* App shall be able to retrieve bus details to the passenger mobile device.
* App shall be able display bus icon and the real-time location on the map.
* App shall be able to view the available seat count of the related bus.
* App shall be able to request a seat from the related bus.
* App shall be able to allow or deny the request sent by the passenger.

## Non-functional requirements

Non-functional requirements define the system attributes.

Following are some of the non-functional requirements of the proposed system.

* App should be able to run on android 5.0 and above.
* App should be able to install without any error.
* App should be able to access and operate easily.
* App should be a usable and user friendly.

## Diagram Description automatically generatedUse Case

Figure 3.7.1- Use case diagram of the proposed booking system

### Actors

There are three actors on the system, Passenger, GPS, and Bus. The actors always connect through the mobile app with the Firebase.

### Use case narratives

Use case narratives are the description about the use cases and it will describe the cases with respect to the Actors action and system response.

### Use case Catalogue

|  |  |  |
| --- | --- | --- |
| **Use case:** | View the real time location on the map | |
| **Actors:** | Passenger, GPS | |
| **Purpose:** | Showing the real time location of the bus | |
| **Overview:** | Indicate the location of the bus in google map using an icon | |
| **Type:** | essential | |
| **Preconditions:** | Passenger must be reserved a seat | |
| **Post conditions:** | Passenger knows the bus is near to the bus stop | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Actor Actions** | | **System Response** |
| 1. Passenger selects the bus 2. GPS signals send to the firebase | | 3.System shows the location of the bus using an icon |

|  |  |  |
| --- | --- | --- |
| **Use case:** | View the passenger’s location | |
| **Actors:** | Bus, GPS | |
| **Purpose:** | Get the passenger’s location to the system | |
| **Overview:** | Using GPS signals, system getting information about the passenger location | |
| **Type:** | Essential | |
| **Preconditions:** | passenger wants to reserve a seat from a selected bus stop | |
| **Post conditions:** | Bus can view the passenger location in the route | |
| **Special Requirements:** | The location of the passenger is the selected bus stop in the related bus route. | |
| **Flow of Event** | | |
| **Actor Actions** | | **System Response** |
| 1.Get the GPS location of the requested bus stop in the route | | 2.update the firebase using the longitudes and latitudes of the passenger location |

|  |  |  |
| --- | --- | --- |
| **Use case:** | Register Driver & Conductor | |
| **Actors:** | Bus | |
| **Purpose:** | Register drivers/conductors and buses to the system | |
| **Overview:** | Entering the driver/conductor/bus details to the database using the registration process | |
| **Type:** | Essential | |
| **Preconditions:** | Driver/conductor/bus should have license | |
| **Post conditions:** | System database update with information of driver/conductor/bus | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Actor Actions** | | **System Response** |
| 1.enter driver/conductor/bus details using the registration window | | 2.update the firebase using the related data entered by the driver/conductor |

|  |  |  |
| --- | --- | --- |
| **Use case:** | Register Passenger | |
| **Actors:** | Passenger | |
| **Purpose:** | Register the passenger to the system | |
| **Overview:** | Entering passenger details to the database | |
| **Type:** | Essential | |
| **Preconditions:** | Passenger wants to register to the app | |
| **Post conditions:** | Passenger register to the system successfully | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Actor Actions** | | **System Response** |
| 1. Enter the passenger details in registration window | | 1. Save passenger information in firebase |

|  |  |  |
| --- | --- | --- |
| **Use case:** | Indicate available seat count | |
| **Actors:** | Passenger, Bus | |
| **Purpose:** | Notify the available seat count to passengers | |
| **Overview:** | show to passenger, the current seat count on the bus | |
| **Type:** | Essential | |
| **Preconditions:** | Passenger wants to know the available seat count on the bus | |
| **Post conditions:** | Passenger can view the available seat count in real time | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Actor Actions** | | **System Response** |
| 1.log into the map window | | 2.show the seat count on the bus icon as a label |

|  |  |  |
| --- | --- | --- |
| **Use case:** | Request a seat | |
| **Actors:** | Passenger | |
| **Purpose:** | Booking a seat of the bus | |
| **Overview:** | Passenger wants to book a seat | |
| **Type:** | Essential | |
| **Preconditions:** | There should be availability of seats to book | |
| **Post conditions:** | Seats booked by the passenger | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Actor Actions** | | **System Response** |
| 1.click the seat count label  3. click the confirmation | | 2. showing a window to confirm the reservation  4.Reserved a seat and change the total available count |

|  |  |  |
| --- | --- | --- |
| **Use case:** | Mange the booking request | |
| **Actors:** | Bus | |
| **Purpose:** | booking a seat according to the passenger request | |
| **Overview:** | Driver/conductor put a notice to show other passengers about the booked seats on the bus | |
| **Type:** | Essential | |
| **Preconditions:** | A request sent by the passenger | |
| **Post conditions:** | Allow or deny the request according to the availability | |
| **Special Requirements:** |  | |
| **Flow of Event** | | |
| **Actor Actions** | | **System Response** |
| 1. Allow or deny the request sent by the passenger | | 2.update the available seat count when the driver/conductor allow the reservation |

## Class Diagrams

The proposed system has several classes that represent on the below class diagram, Driver, Conductor, and the Passenger classes are inherited from the User class. They have equal attributes inherit from the User class

Class diagram for the proposed system is attached here.

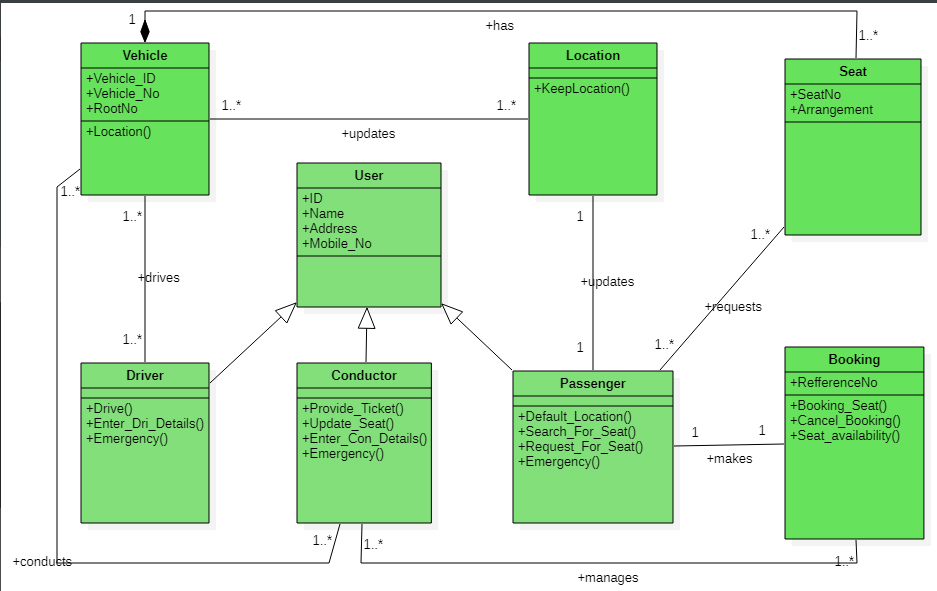


Figure .8.1 - Class Diagram for Proposed Material Management System

## ER Diagram (Entity Relationship Diagram)

Figure .9.1 - ER Diagram for Proposed System

## Activity Diagrams

There is some of Activity diagrams are given below.

### Passenger Login

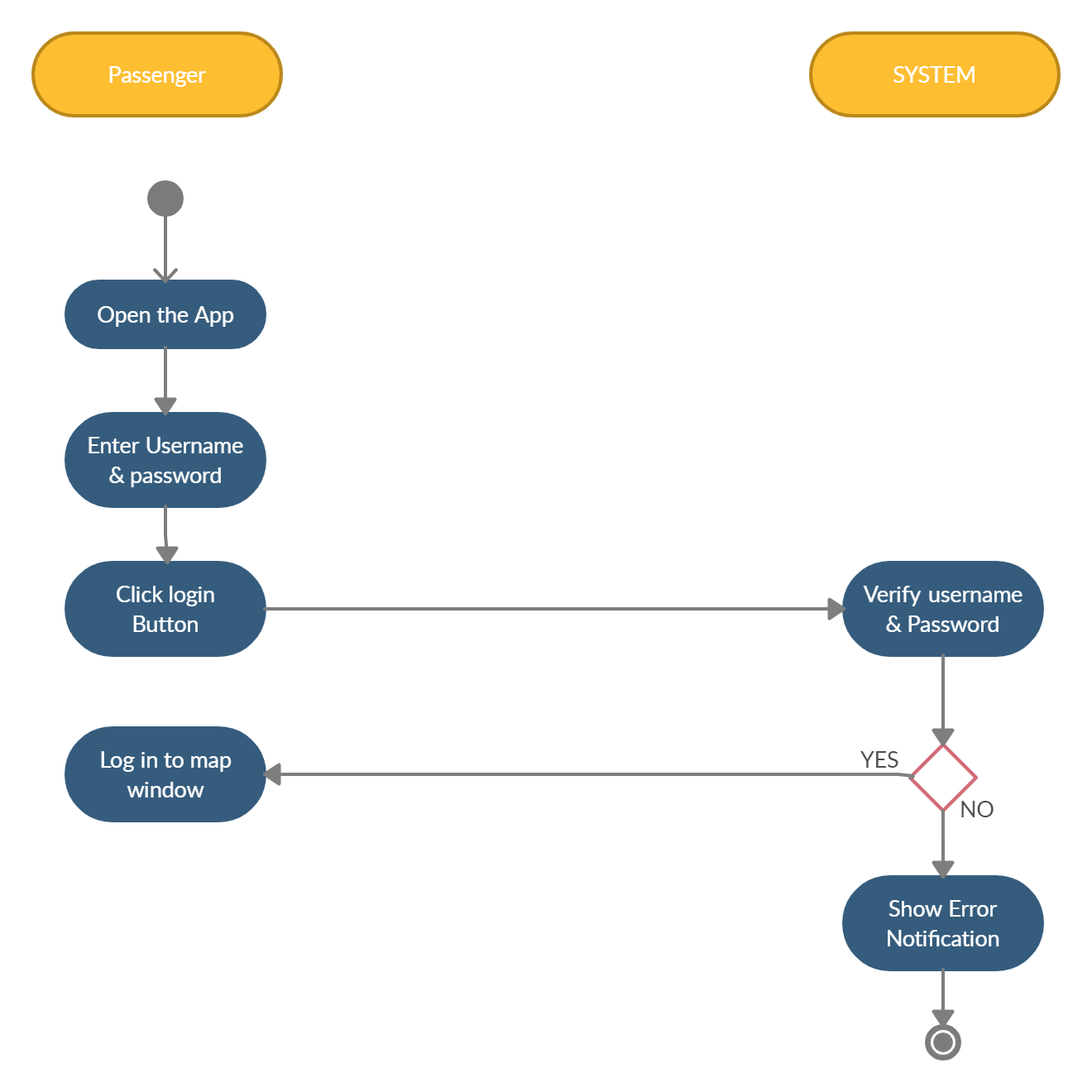


Figure 3.10.1 - Activity Diagram – Passenger Login

### Driver/Conductor Login

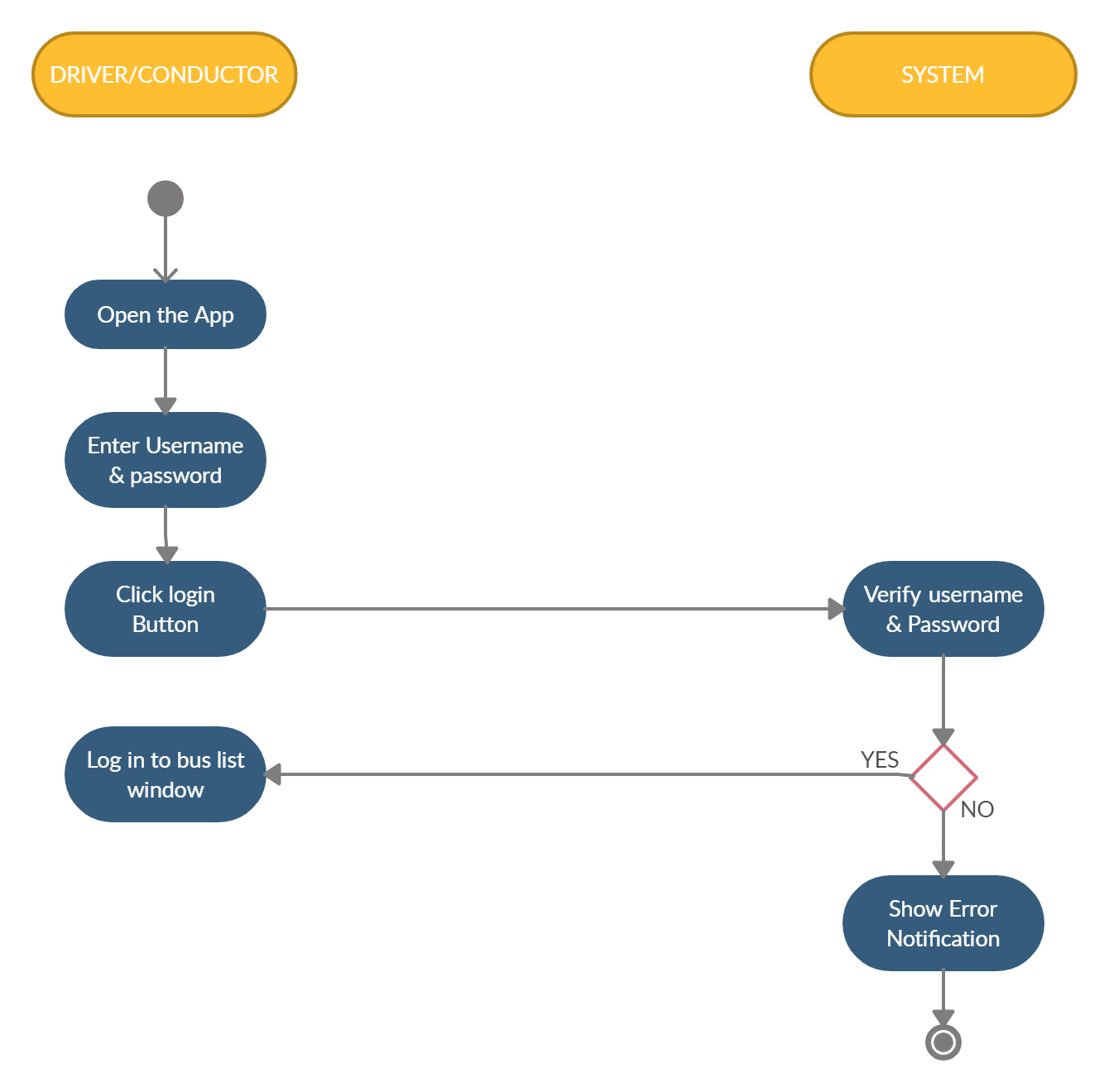


Figure 3.10.2 - Activity Diagram – Driver/Conductor Login

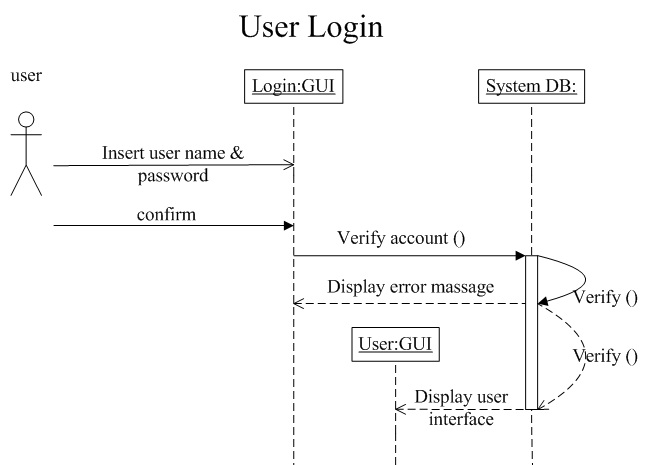
### Passenger /Driver Registration

Figure 3.10.3 - Activity Diagram – Registration Process

### Booking a Seat

Figure 3.10.4 - Activity Diagram – Booking a seat

## Sequence Diagram of Login

****

..,

Figure .10- Sequence Diagram of Login

## Sequence Diagram of Booking

Diagram

Description automatically generated

Figure .11- Sequence Diagram of booking a seat

Chapter 4

# SYSTEM DEVELOPMENT

## Introduction

This chapter will discuss about the system development of the proposed system. That will discuss through the technologies, UI, DB, and how the proposed system was developed.

## Database Implementation

The database was implemented by using the Google Firebase technology which is the most suitable real-time database for the android mobile apps. The tables are arranged by hard-coding other than the drawing tables, using the android studio and the generated data sends to the data at real-time.

### Tables

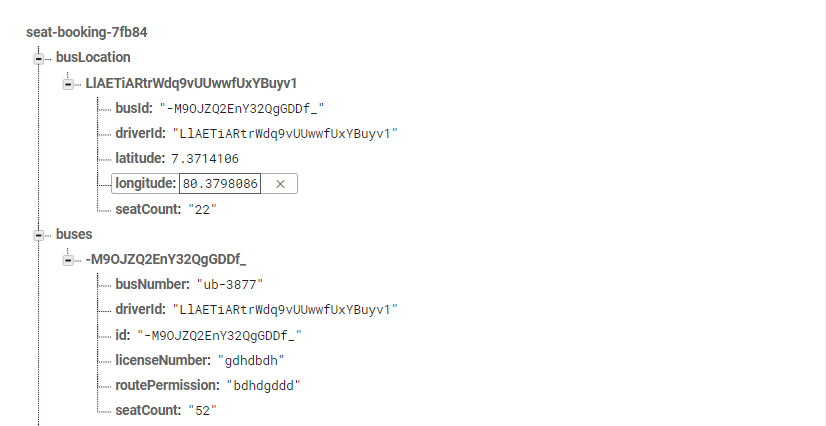


Figure .2.1. -Table of registered buses and bus Locations



Figure 4.2.1.2 - Table of registered Users and booking request

Graphical user interface, text, application

Description automatically generated

Figure 4.2.1.3 - Table of registered Users

### Android code sample for creating firebase

The database is a critical component of this system because Firebase used as the database. This is a real-time database, and the tables are created using the android coding. Tables updates when the data sent to the firebase.

**public class** MyFirebase {  
**public static** FirebaseDatabase *firebaseDatabase* = FirebaseDatabase.*getInstance*();  
**public static** FirebaseAuth *auth*;

*// create tables of the firebase*

**public static** String *tableUsers* = **"users"**;  
**public static** String *tableBuses* = **"buses"**;  
**public static** String *tableBusLocation* = **"busLocation"**;  
**public static** String *tableSeatRequest* = **"seatRequest"**;  
**public static void** writeNewUser(User user) {  
 DatabaseReference mDatabase = *firebaseDatabase*.getReference();  
 String userId = user.**id**;  
 mDatabase.child(*tableUsers*).child(userId).setValue(user);  
 }  
  
**public static void** writeNewBus(Bus bus) {  
 DatabaseReference mDatabase = *firebaseDatabase*.getReference();  
 bus.**id** = mDatabase.push().getKey();  
 mDatabase.child(*tableBuses*).child(bus.**id**).setValue(bus);  
 }  
  
**public static void** writeBusLocation(BusLocation busLocation) {  
 DatabaseReference mDatabase = *firebaseDatabase*.getReference();  
 mDatabase.child(*tableBusLocation*).child(busLocation.**driverId**).setValue(busLocation);  
 }  
  
**public static void** writeSeatRequest(SeatRequest seatRequest) {  
 DatabaseReference mDatabase = *firebaseDatabase*.getReference();  
 seatRequest.**id** = mDatabase.push().getKey();  
 mDatabase.child(*tableSeatRequest*).child(seatRequest.**id**).setValue(seatRequest);  
 }  
  
**public static void** updateSeatRequest(SeatRequest seatRequest) {  
 DatabaseReference mDatabase = *firebaseDatabase*.getReference();  
 mDatabase.child(*tableSeatRequest*).child(seatRequest.**id**).setValue(seatRequest);  
 }  
}

Figure .2.2–Android code for create the database

## System Implementation

The passenger, driver and the conductor use the android platform as the OS. The UI, DB and all other implementation is done by using the Android Studio.

### Signup Activity android code

**public class** SignupActivity **extends** AppCompatActivity {  
  
**private** EditText **etEmail**, **etPassword**, **etName**, **etContactNumber**;  
**private** Button **btnSignIn**, **btnSignUp**, **btnresetPassword**;  
**private** ProgressBar **progressBar**;  
**private** FirebaseAuth **auth**;  
 ProgressDialog **dialog**;  
 Spinner **spUserType**;  
  
  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
**super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_signup***);  
  
**auth** = FirebaseAuth.*getInstance*();  
**dialog** = **new** ProgressDialog(**this**);

// Initialize Buttons and input variables

**btnSignIn** = (Button) findViewById(R.id.***sign\_in\_button***);  
**btnSignUp** = (Button) findViewById(R.id.***sign\_up\_button***);  
**etEmail** = (EditText) findViewById(R.id.***etEmail***);  
**etPassword** = (EditText) findViewById(R.id.***etPassword***);  
**etName** = (EditText) findViewById(R.id.***etName***);  
**etContactNumber** = (EditText) findViewById(R.id.***etContactNumber***);  
**progressBar** = (ProgressBar) findViewById(R.id.***progressBar***);  
**spUserType** = findViewById(R.id.***spUserType***);  
  
 ArrayList<StringWithInt> arrayList = **new** ArrayList<>();  
 arrayList.add(**new** StringWithInt(0, **"Passenger"**));  
 arrayList.add(**new** StringWithInt(1, **"Driver"**));  
 ArrayAdapter<StringWithInt> adapter = **new** ArrayAdapter<>(getApplicationContext(), R.layout.***display\_spinner***, arrayList);  
**spUserType**.setAdapter(adapter);

//Create the button functions   
**btnSignIn**.setOnClickListener(**new** View.OnClickListener() {  
@Override  
**public void** onClick(View v) {  
 Intent intent = **new** Intent(getApplicationContext(), LoginActivity.**class**);  
 startActivity(intent);  
 finish();  
 }  
 });  
  
**btnSignUp**.setOnClickListener(**new** View.OnClickListener() {  
@Override  
**public void** onClick(View v) {  
  
**final** String contact = **etContactNumber**.getText().toString().trim();  
**final** String name = **etName**.getText().toString().trim();  
**final** String email = **etEmail**.getText().toString().trim();  
**final** String password = **etPassword**.getText().toString().trim();  
  
// check whether the inputs are empty  
**if** (TextUtils.*isEmpty*(name)) {  
  
 Toast.*makeText*(getApplicationContext(), **"Enter Email address!"**,  
 Toast.***LENGTH\_SHORT***).show();  
**return**;  
 }  
  
**if** (TextUtils.*isEmpty*(contact)) {  
  
 Toast.*makeText*(getApplicationContext(), **"Enter Email address!"**,  
 Toast.***LENGTH\_SHORT***).show();  
**return**;  
 }  
**if** (TextUtils.*isEmpty*(email)) {  
  
 Toast.*makeText*(getApplicationContext(), **"Enter Email address!"**,  
 Toast.***LENGTH\_SHORT***).show();  
**return**;  
 }  
**if** (TextUtils.*isEmpty*(password)) {  
  
 Toast.*makeText*(getApplicationContext(), **"Enter Email address!"**,  
 Toast.***LENGTH\_SHORT***).show();  
**return**;  
 }  
  
**if** (password.length() <6) {  
 Toast.*makeText*(getApplicationContext(), **"Password is too short, Enter minimum 6"** +  
**" characters!"**, Toast.***LENGTH\_SHORT***).show();  
**return**;  
 }  
**progressBar**.setVisibility(View.***VISIBLE***);

*// Create a user*

**auth**.createUserWithEmailAndPassword(email, password).addOnCompleteListener(SignupActivity.**this**, **new** OnCompleteListener<AuthResult>() {  
@Override  
**public void** onComplete(@NonNull Task<AuthResult> task) {  
 Toast.*makeText*(getApplicationContext(), **"Registration successful"**,  
 Toast.***LENGTH\_SHORT***).show();  
**progressBar**.setVisibility(View.***GONE***);  
**if** (!task.isSuccessful()) {  
 Toast.*makeText*(getApplicationContext(), **"Registeration failed"**, Toast.***LENGTH\_SHORT***).show();  
 } **else** {  
 FirebaseUser firebaseUser = FirebaseAuth.*getInstance*().getCurrentUser();  
  
 StringWithInt stringWithInt = (StringWithInt) **spUserType**.getSelectedItem();  
**int** userType = stringWithInt.getId();  
  
 User user = **new** User(firebaseUser.getUid(),name,email,contact, userType);  
  
 MyFirebase myFirebase = **new** MyFirebase();  
 myFirebase.*writeNewUser*(user);  
 Common.*currentUser* = user;  
**if** (user.**userType** == 0) {  
 Intent intent = **new** Intent(getApplicationContext(), PassengerActivity.**class**);  
 startActivity(intent);  
 finish();  
 } **else** {  
 Intent intent = **new** Intent(getApplicationContext(), DriverActivity.**class**);  
 startActivity(intent);  
 finish();  
 }  
 }  
 }  
 });  
 }  
 });  
  
 }

### Login Activity android code

**public class** LoginActivity **extends** AppCompatActivity {  
  
**private** EditText **inputEmail**, **inputPassword**;  
**private** ProgressBar **progressBar**;  
**private** Button **btnSignup**, **btnLogin**, **btnReset**;  
  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
**super**.onCreate(savedInstanceState);  
  
*// set the view now*

setContentView(R.layout.***activity\_login***);  
  
**inputEmail** = findViewById(R.id.***email***);  
**inputPassword** = (EditText) findViewById(R.id.***password***);  
**progressBar** = (ProgressBar) findViewById(R.id.***progressBar***);  
**btnSignup** = (Button) findViewById(R.id.***btn\_signup***);  
**btnLogin** = (Button) findViewById(R.id.***btn\_login***);  
**btnReset** = (Button) findViewById(R.id.***btn\_reset\_password***);  
  
**btnSignup**.setOnClickListener(**new** View.OnClickListener() {  
@Override  
**public void** onClick(View v) {  
 startActivity(**new** Intent(LoginActivity.**this**, SignupActivity.**class**));  
 }  
 });  
  
**btnLogin**.setOnClickListener(**new** View.OnClickListener() {  
@Override  
**public void** onClick(View v) {  
 String email = **inputEmail**.getText().toString();  
**final** String password = **inputPassword**.getText().toString();  
  
**if** (TextUtils.*isEmpty*(email)) {  
 Toast.*makeText*(getApplicationContext(), **"Enter email address!"**, Toast.***LENGTH\_SHORT***).show();  
**return**;  
 }  
  
**if** (TextUtils.*isEmpty*(password)) {  
 Toast.*makeText*(getApplicationContext(), **"Enter password!"**, Toast.***LENGTH\_SHORT***).show();  
**return**;  
 }  
  
**progressBar**.setVisibility(View.***VISIBLE***);  
  
*//authenticate user*

MyFirebase.*auth*.signInWithEmailAndPassword(email, password)  
 .addOnCompleteListener(LoginActivity.**this**, **new** OnCompleteListener<AuthResult>() {  
@Override  
**public void** onComplete(@NonNull Task<AuthResult> task) {  
**progressBar**.setVisibility(View.***GONE***);  
**if** (!task.isSuccessful()) {  
*// there was an error***if** (password.length() <6) {  
**inputPassword**.setError(getString(R.string.***minimum\_password***));  
 } **else** {  
 Toast.*makeText*(LoginActivity.**this**,  
 getString(R.string.***auth\_failed***),  
 Toast.***LENGTH\_LONG***).show();  
 }  
 }

**else** {  
FirebaseUser firebaseUser = FirebaseAuth.*getInstance*().getCurrentUser();  
  
 MyFirebase.*firebaseDatabase*.getReference().child(**"users"**).child(firebaseUser.getUid()).addValueEventListener(**new** ValueEventListener() {

@Override  
**public void** onDataChange(DataSnapshot dataSnapshot) {  
 Common.*currentUser* = dataSnapshot.getValue(User.**class**);

//show different activities for passenger and bus   
**if** (Common.*currentUser*.**userType** == 0) {  
 Intent intent = **new** Intent(getApplicationContext(), PassengerActivity.**class**);  
 startActivity(intent);  
 finish();  
 } **else** {  
Intent intent = **new** Intent(getApplicationContext(), DriverActivity.**class**);  
 startActivity(intent);  
 finish();  
 }  
 }

@Override  
**public void** onCancelled(DatabaseError error) {  
 Intent intent = **new** Intent(getApplicationContext(), LoginActivity.**class**);  
 startActivity(intent);  
finish();  
 }  
 });  
 }  
 }});  
 }  
 });  
 }  
}

### Map Activity android code

The map is the most important component of the system. This is the driver map activity android code.

**public class** MapsActivity **extends** FragmentActivity **implements** OnMapReadyCallback,  
 LocationListener,  
 PlaceSelectionListener,  
 GoogleMap.OnPolygonClickListener, GoogleMap.OnInfoWindowClickListener {  
  
**private static final** String ***TAG*** = **"MapsActivity"**;  
  
 ArrayList<BusLocation>**buslist**;  
**private** GoogleMap **mMap**;  
 LocationManager **locationManager**;  
**double myLatitude**, **myLongitude**;  
**private static final int *PERMISSION\_REQUEST*** = 100;  
  
 PlacesClient **placesClient**;  
  
 ArrayList **markerPoints** = **new** ArrayList();  
  
**private** LocationCallback **locationCallback**;  
  
**private** GeofencingClient **geofencingClient**;  
  
**private static final int *PERMISSIONS\_REQUEST*** = 100;  
**private** Button **btnRequestBtn**;  
**private** BusLocation **selectedBus**;  
  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
**super**.onCreate(savedInstanceState);  
  
*// Retrieve the content view that render the map.*

setContentView(R.layout.***activity\_maps***);  
  
 String apikey = **"AIzaSyAxTT9aUEfPapeWd-98PIdeuiIjuvVJHVk"**;  
  
**btnRequestBtn** = findViewById(R.id.***btnRequestBtn***);  
**btnRequestBtn**.setOnClickListener(**new** View.OnClickListener() {  
@Override  
**public void** onClick(View v) {  
 requestSeat();  
 }  
 });  
  
**buslist** = **new** ArrayList<>();  
 loadbus();  
  
**if** (!Places.*isInitialized*()) {  
 Places.*initialize*(getApplicationContext(), apikey);  
 }  
  
**placesClient** = Places.*createClient*(**this**);  
  
**final** AutocompleteSupportFragment autocompleteSupportFragment =  
 (AutocompleteSupportFragment) getSupportFragmentManager().findFragmentById(R.id.***autocomplete\_fragment***);  
  
 autocompleteSupportFragment.setPlaceFields(Arrays.*asList*(Place.Field.***ID***,  
 Place.Field.***ID***.***LAT\_LNG***, Place.Field.***NAME***));  
  
 ActivityCompat.*requestPermissions*(**this**,  
**new** String[]{Manifest.permission.***ACCESS\_FINE\_LOCATION***}, ***PERMISSION\_REQUEST***);  
 autocompleteSupportFragment.setOnPlaceSelectedListener(**new** PlaceSelectionListener() {  
@Override  
**public void** onPlaceSelected(@NonNull Place place) {  
**final** LatLng latLng = place.getLatLng();  
  
 Toast.*makeText*(MapsActivity.**this**, **""** + latLng.**latitude**, Toast.***LENGTH\_SHORT***).show();  
 Log.*i*(**"PlacesApi"**, **"onPlaceSelected:"** + latLng.**latitude** + **"\n"** + latLng.**longitude**);  
 }  
  
@Override  
**public void** onError(@NonNull Status status) {  
  
 }  
 });  
SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
 .findFragmentById(R.id.***map***);  
 mapFragment.getMapAsync(**this**);  
  
 loadSeatRequestDetails();  
 }  
  
**private void** requestSeat() {  
 SeatRequest seatRequest = **new** SeatRequest();  
 seatRequest.**busId** = **selectedBus**.**busId**;  
 seatRequest.**userId** = Common.*currentUser*.**id**;  
 seatRequest.**latitude** = **myLatitude**;  
 seatRequest.**longitude** = **myLongitude**;  
 seatRequest.**status** = 1;  
  
 MyFirebase.*writeSeatRequest*(seatRequest);  
 }  
  
**private void** loadSeatRequestDetails() {  
 MyFirebase.*firebaseDatabase*.getReference().child(MyFirebase.*tableSeatRequest*).addValueEventListener(**new** ValueEventListener() {  
@Override  
**public void** onDataChange(DataSnapshot dataSnapshot) {  
**for** (DataSnapshot data : dataSnapshot.getChildren()) {  
 SeatRequest seatRequest = data.getValue(SeatRequest.**class**);  
**if** (seatRequest.**userId**.equals(Common.*currentUser*.**id**)) {  
**if** (seatRequest.**status** == 2) {  
 Toast.*makeText*(getApplicationContext(), **"Approve"**, Toast.***LENGTH\_LONG***).show();  
 seatRequest.**status** = 0;  
 MyFirebase.*updateSeatRequest*(seatRequest);  
 } **else if** (seatRequest.**status** == 3) {  
 Toast.*makeText*(getApplicationContext(), **"Reject"**, Toast.***LENGTH\_LONG***).show();  
 seatRequest.**status** = 0;  
 MyFirebase.*updateSeatRequest*(seatRequest);  
 }  
 }  
 }  
 }  
  
@Override  
**public void** onCancelled(DatabaseError error) {  
  
 }  
 });  
 }  
// show registered buses   
**private void** loadbus() {  
  
*//retrive firebase data from tablesbuslocation*

MyFirebase.*firebaseDatabase*.getReference().child(MyFirebase.*tableBusLocation*).addValueEventListener(**new** ValueEventListener() {  
@Override  
**public void** onDataChange(@NonNull DataSnapshot dataSnapshot) {  
**buslist** = **new** ArrayList<>();  
**for** (DataSnapshot data : dataSnapshot.getChildren()) {  
 BusLocation bus = data.getValue(BusLocation.**class**);  
**buslist**.add(bus);}  
 drawMarker();  
  
 }  
  
@Override  
**public void** onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
**void** drawMarker() {  
**if** (**mMap** != **null**) {  
**mMap**.clear();  
 } **else** {  
**return**;  
 }  
**for** (**int** i = 0; i <**buslist**.size(); i++) {  
 BusLocation bus = **buslist**.get(i);  
 LatLng busplace = **new** LatLng(bus.**latitude**, bus.**longitude**);  
 MarkerOptions markerOptions = **new** MarkerOptions()  
 .position(busplace)  
 .draggable(**true**)  
 .title(**"Available Seats "** + bus.**seatCount**)  
 .icon(BitmapDescriptorFactory.*fromResource*(R.drawable.***download***));  
 Marker marker = **mMap**.addMarker(markerOptions);  
 marker.showInfoWindow();  
 SeatRequest seatRequest = **new** SeatRequest();  
 marker.setTag(bus);  
 }  
 }  
  
@Override  
**public void** onMapReady(GoogleMap googleMap) {  
**mMap** = googleMap;  
**mMap**.getUiSettings().setMyLocationButtonEnabled(**true**);  
**mMap**.setMyLocationEnabled(**true**);  
**mMap**.setOnInfoWindowClickListener(**this**);  
 }  
  
@Override  
**public void** onLocationChanged(Location location) {  
**myLatitude** = location.getLatitude();  
**myLongitude** = location.getLongitude();  
  
 LatLng myLatLong = **new** LatLng(**myLatitude**, **myLongitude**);  
**mMap**.addMarker(**new** MarkerOptions().position(myLatLong).draggable(**true**).title(  
**"Me"**).icon(BitmapDescriptorFactory.*fromResource*(R.drawable.***download***)));  
**mMap**.moveCamera(CameraUpdateFactory.*newLatLng*(myLatLong));  
  
**mMap**.animateCamera(CameraUpdateFactory.*newLatLngZoom*(**new** LatLng(**myLatitude**,  
**myLongitude**), 12.0f));  
  
 }  
  
@Override  
**public void** onProviderDisabled(String provider) {  
 Toast.*makeText*(**this**, **"Please Enable GPS and Internet"**, Toast.***LENGTH\_SHORT***).show();  
 }  
@Override  
**public void** onStatusChanged(String provider, **int** status, Bundle extras) {  
  
 }  
@Override  
**public void** onProviderEnabled(String provider) {  
  
 }  
@Override  
**public void** onPolygonClick(Polygon polygon) {  
  
 }  
@Override  
**public void** onPlaceSelected(@NonNull Place place) {  
  
 }  
@Override  
**public void** onError(@NonNull Status status) {  
  
 }  
@Override  
**public void** onInfoWindowClick(Marker marker) {  
 Toast.*makeText*(**this**, **"Seat count updates in real time"**, Toast.***LENGTH\_SHORT***).show();  
**btnRequestBtn**.setVisibility(View.***VISIBLE***);  
**selectedBus** = (BusLocation) marker.getTag();  
 }  
}

## Interface Implementation

Users are contact and interact with the system via the UI of the app. Android Studio XML file format used to implement the UIs.XML files have the capability to add elements using the drag and drop that already built-in elements

### Login Window

Figure .4.1 – Login window UI

All the users should be login into the system via login window. User should enter his username and password to the system to login. To create a new account, user need sign up to the system by clicking “*Not a member? Get Register in Seat Booking Now! ”*and should enter user details in registration window. For the registered users, they can login to the system by entering “Username” and correct “Password” to relevant field and pressing “Login” button.

### Registration Window

Figure .4.2 – Registration window UI

To register on the system, users should enter the Name, Contact number, Email, and the Password. If the user already registered to the system, that user can login to the system using the *“Already Registered. Login Me!”*

### Bus Registration

Figure .4.3 –Bus Registration window UI

To register a bus to the system driver/ conductor should enter the bus number, License Number, Route Permission number and the total seat count of the bus. By clicking Register Bus Button all the details upload to the firebase with a unique identification number.

### Bus Registration List

Figure .4.4–Bus Registration List window UI

Bus registration list shows the registered bus of the related driver or conductor. Driver or conductor can switch to the map by clicking the bus number on the list.

### Driver Map Window

Figure .4.5–Bus Map window UI

In this window shows the bus real-time latitude and the longitude to the driver/passenger to identify the location of the bus. Start Trip and Stop Trip buttons are fixed on the window. Driver can press the start button when departing and it starts to update real-time location data to the firebase. When press the stop button it stops the update the firebase.

### Seat update Window

Figure .4.6–Seat update window UI

Driver can edit the available seat count t and update it anytime. When press the update seat count button it updates the firebase.

### Passenger Map Window

Figure .4.6–Passenger Map window UI

In passenger map window, related route bus icon shows on the passenger map.The bus icon moves when the bus traveling on the route. Passenger can send a booking request by clicking the *available seat count notification*.

Chapter 5

# CONCLUSION AND FURTHER DEVELOPMENT

## Introduction

This chapter will discuss about the conclusion of the system. This section ends up with the further modifications.

## Project

The main objective is to develop a mobile app to booking a seat of a highway bus without waiting on the road. Because of the current manual process is much more time wasting on the road and passenger should have to get a phone call to driver or conductor, verify the bus location for catch it. It is also a big issue to the driver making phone calls and answering while driving.

## System

When the app running, it is continuously connected to the firebase and update the database. The firebase has the capability to store all the information related to registration and location data. Realtime location and booking is the most important development of the system.

## Conclusion

Seat reservation is limited to buses traveling at the same time of the reservation; the service (seat reservation) will be enhanced to book in advance in any bus. The passenger can then be notified of the departure of the bus and its arrival to the passenger location. This will eliminate unnecessary waiting at the bus stops for the passenger. Ordinary passengers who receive the service on time can upgrade this service by giving more options and information. Moreover, this system can be introduced to the relevant authorities to monitor the buses with the time which is currently done by the timekeepers manually. Therefore, the bus service can be improved by the relevant authorities by providing warning timely, if the buses are waiting unnecessarily in some places.

## Future Modifications

The system can be enhanced with the automated ticketing system. A Bluetooth printer can be applied to the system and generate a ticket to the passenger after getting into the bus. This will help to update the seat count automatically too. Moreover, this will eliminate the job role of the conductor which is a cost for the bus owner. The System can be extended not only for expressway services but for regular interprovincial services.

Even android devices are cheap in the marketplace there are some people use IOS devices like iPhones and iPads. Because of the usage, the app could be developed to the IOS devices.

Chapter 6

# References

Digital 2020: Sri Lanka — Datareportal – Global Digital Insights, (2020)

Barth, M., And Farrell, J. A. (1999). The Global Positioning System & Inertial Navigation, Mcgraw-Hill, New York.

Department of Forestry and Environmental Science, University of Sri Jayawardenepura. (2016). “Proposed Central Expressway Project.”.

Expressway road. (1998). Retrieved 26 June 2020, from https://www.britannica.com/technology/expressway-road

Manners, G., (1965). Book Review: The Expressway And The City By Lewis Mumford, Seeker & Warburg, 1964, Pp. 189, Urban Studies, 2(2)

McClelland, C. (2017). 6 Things You Didn’t Know About GPS. Retrieved 26 June 2020, from https://medium.com/iotforall/6-things-you-didnt-know-about-gps-efb37e83bd02

McKinnon, A. (2006). Life without Trucks: The Impact of A Temporary Disruption of Road Freight Transport On A National Economy. Journal of Business Logistics, 27(2), 227-250. Doi: 10.1002/J.2158-1592.2006.Tb00224.X

Nuwan, G., (2020). Reservations. [online] Railway.gov.lk. Available at: <http://www.railway.gov.lk/web/index.php?option=com\_content&view=article&id=61&Itemid=68&lang=en> [Accessed 24 June 2020].

Oliver, J. (2019). “Annual Performance Report.” Hilos Tensados 1:1–476.

Sri Lanka Transport Board (2020). Retrieved 26 June 2020, from https://sltb.express.lk/