Usama Rao
161120

Sami Ahmed
161132

CarPool

Bachelor of Science in Computer Science

Supervisor: Shoaib Malik

Co-Supervisor: Dr. Tauseef

Department of Computer Science

Air University, Islamabad

June 2020

Certificate

We accept the work contained in the report titled 'CarPool', written by Mr. Usama Rao AND Mr. Sami Ahmed as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by...:

Supervisor:

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator:

_____

Head of the Department:

_____

June, 2020

Table of Contents

List of Figures

Abstract

In general, people have a hard time conciliating their schedules because of the way they move from one location to another. and students suffer from this the most especially since transportation between cities is not that great, As students, we think there should exist more suitable transportation solutions to places where transportation networks are short and cheap and helpful for students.
This report proposes a platform to help improve students mobility through carpooling, a way for vehicle owning students to share their private vehicle with non vehicle owning students in order to splitting and reducing costs. carpooling may be one of the best solutions when there is no other mean of transportation to a specific location but naturally it is not the only one. mobile applications take more and more part of everyone's lives, different services for carpooling with different features begin to compete with existing transportation solutions. some people start to prefer using new carpooling services over the traditional services represented by taxi services. carPool aims to promote carpooling by targeting students making it easier for them to adhere and use this system. by targeting students people will more likely join the service since its users are primarily other people form the same environment. to put the carpooling system in place, we have designed and developed an android mobile application with backend servers for users to access the carpooling service through their smartphones, additionally the application involves some features that are critical to the service. by using android development tools and libraries and efficient backend solutions we have managed to make the application simple but powerful as well, which makes this application very useful for the students to use.
The combination of the smart phone and the internet service is the trend of the future information development and software applications. mobile phones are the most commonly used communication tools. using mobile phones to obtain information is not only quick, but also more convenient shortcut to improve people's lives. in the paper, we propose the software development architecture based on web services. this framework introduces the three-layer architecture of web development into mobile phone software development. based on the threelayer architecture, the android based carPool system is developed.
Our app CarPool will be a unique carpooling application that would take benefits of the advantages of carpooling and try to improve and eliminate the disadvantages, all while focusing on making it a good carpooling experience for students. the realization of our project will go through the conceptual phase and then development phase. since making a good application requires good planning first.

Acknowledgments

We wish to thank various people for their contribution to this project.we would like to express our deep gratitude to Sir Shoaib Malik, our Project supervisor, for their patient guidance, enthusiastic encouragement and useful critiques of this project work. we would also like to thank Dr Touseef, our project co-supervisor for his advice and assistance in keeping our progress on schedule. we would also like to extend our thanks to the librarians of the Air University for their help in offering us the resources in running the program.
Finally, we wish to thanks our parents for their support and encouragement throughout our study.

Usama, Sami Ahmed Malik
Islamabad, Pakistan

June 2020

Chapter 1

Introduction

What is CarPool?

1.1 Project Background/Review

Carpooling (also car-sharing, ride-sharing and lift-sharing) is the sharing of car journeys so that more than one person travels in a car, and prevents the need for others to have to drive to a location themselves.
Drivers and passengers offer and search for journeys through one of the several mediums available. after finding a match they contact each other to arrange any details for the journey(s). costs, meeting points and other details like space for luggage are agreed on. they then meet and carry out their shared car journey(s) as planned.
By having more people using one vehicle, carpooling reduces each person's travel costs such fuel costs, tolls and the stress of driving. authorities often encourage carpooling, especially during periods of high pollution or high fuel prices. car sharing is a good way to use up the full seating capacity of a car, which would otherwise remain unused if it were just the driver using the car.
In 2009, carpooling represented 43.5% of all trips in the United States and 10% of commute trips. the majority of carpool commutes (over 60%) are "fam-pools" with family members. in 2011, an organization called Greenock created a campaign to encourage others to use this form of transportation in order to reduce their own carbon footprint.
Carpooling, or car sharing as it is called in British English, is promoted by a national UK charity, carplus, whose mission is to promote responsible car use in order to alleviate financial, environmental and social costs of motoring today, and encourage new approaches to car dependency in the UK. carplus is supported by transport for London, the British government initiative to reduce congestion and parking pressure and contribute to relieving the burden on the environment and to the reduction of traffic related air-pollution.
Cabbing All the Way is a book written by author Jatin Kuberkar that narrates a success story of a carpool with twelve people on board. based in the city of Hyderabad, India, the book is a real-life narration and highlights the potential benefits of having a carpool

1.2 Problem description

Many vehicle-owning Students who commute on daily basis often have unoccupied seats in their vehicles. many non-vehicle owning students find it very difficult sometimes to find ride for travelling to and from university.

1.3 Project Objectives

Objective

- To Allow vehicle owning students to share their rides with other students for traveling to and from their institutes and cut down their fuel bills.
- To facilitate non-vehicle owning students for travelling to and from university easier and cheaper.

Goals

- Cost Effective: much Cheaper than cab services.
- Ease of getting ride: riders are easy approachable, which reduces the tension of finding and catching of local transport right on time.
- Fewer Cars on the road will have reduced fuel consumption which will make environment eco-friendly.

1.4 Project Scope

This project (CarPool) aims to develop an android based application for carpooling for students, this application allows vehicle owning students to submit rides for specific targets and allows passengers to reserve/request rides from drivers all while being secure and having a simple interface.
This application will help students save money and also reduce the pollution of the environment and effects of vehicles, this application focuses on serving needs of students. CarPool will be intended for the students in Air University and it will support android phones and tablets, users will need internet connection to use the application to offer or find a common route to travel.
The application will have a simple and easy interface, Users must register at first before using the application, after that they must choose between a driver or a passenger, a driver can offer a drive to a specific location while a passenger can find or request a ride to a location.

1.5 The Degree of Project Report

In our FYP-1, we presented our idea that how CarPool would be beneficial. the only purpose of FYP-1 was to present and defend the idea. We have completed both tasks successfully and we also developed some mockup screens to present our

idea.
However, in fyp-2, the task assigned to us was to develop a working application for two users: driver and rider along with the implementation of the core feature of our application, which was location tracking of driver and rider, fetching current location, use firebase it's real-time database which is a NoSQL fast databaseand displaying that location on the map using Google Map API.

Chapter 2

Literature Review

Carpooling

2.1 Definition and general principle

Carpooling (also car-sharing, ride-sharing and lift-sharing) is the sharing of car journeys so that more than one person travels in a car, and prevents the need for others to have to drive to a location themselves.
By having more people using one vehicle, carpooling reduces each person's travel costs such as: fuel costs, tolls, and the stress of driving. carpooling is also a more environmentally friendly and sustainable way to travel as sharing journeys reduces air pollution, carbon emissions, traffic congestion on the roads, and the need for parking spaces. authorities often encourage carpooling, especially during periods of high pollution or high fuel prices. car sharing is a good way to use up the full seating capacity of a car, which would otherwise remain unused if it were just the driver using the car.
In 2009, carpooling represented 43.5% of all trips in the United States and 10% of commute trips. the majority of carpool commutes (over 60%) are "fam-pools" with family members.
Carpool commuting is more popular for people who work in places with more jobs nearby, and who live in places with higher residential densities. carpooling is significantly correlated with transport operating costs, including fuel prices and commute length, and with measures of social capital, such as time spent with others, time spent eating and drinking and being unmarried. however, carpooling is significantly less likely among people who spend more time at work, elderly people, and homeowners.
Carpooling usually means to divide the travel expenses equally between all the occupants of the vehicle (driver or passenger). the driver does not try to earn money, but to share with several people the cost of a trip he would do anyway. the expenses to be divided basically include the fuel and possible tolls. But if we include in the calculation the depreciation of the vehicle purchase and maintenance, insurance and taxes paid by the driver, we get a cost around 20RS/km. There are platforms that facilitate carpooling by connecting people seeking respectively passengers and drivers. usually there is a fare set up by the car driver and accepted by passengers

because they get an agreement before trip start.

2.2 Carpooling Types

2.2.1 Regular

The car is often perceived as an extension of the personal space, the driver, alone in his vehicle is in a closed space; he is free to do what he likes: listen to the radio, sing, call with headsets etc. carpooling regularly is to share a dialogue, experiences, stories. in the United States an intermediate concept has developed between carpooling and the public transport line: the vanpool. These are minibuses chartered by an employer, a public authority or a private company and made available to a group of people who regularly make the same journey.

2.2.2 Occasional

This type of carpooling is mainly used for leisure or last minute departures. the linking is often done through websites or mobile applications, which can significantly reduce travel costs, but usually requires to carpool with one or more unknown. this type of carpooling is mainly used for leisure or last minute departures. the linking is often done through websites or mobile applications, which can significantly reduce travel costs, but usually requires to carpool with one or more unknown.

2.2.3 Eventual

Participants in an event (music festival, sporting event, wedding, associative or institutional meeting ...) can organize to carpool to the venue of the event. this one-time carpool has a special feature: all participants travel to the same place on the same date.carpooling is also used for departures on holidays or weekends, savings on a trip being even larger than the trip is long. So carpooling becomes an alternative of affordable and accessible transportation.
There are also "cultural" carpooling platforms to visit a cultural site: castles, museums, exhibitions, artists' studios, religious places, festivals, etc.

2.3 Advantages and disadvantages of carpooling applications

2.3.1 Advantages of carpooling

- The main advantage of rideshare solutions is cost saving, of course. depending on the agreement, a ride can be twice as cheap than traveling by conventional way.
- The car is not a gas cost only. there are some cost items for the maintenance, repair, parts replacement in your vehicle. If you reduce the time of car utilization, you reduce these costs.

- A fewer number of cars on the road can reduce the CO2 emission in the roads and make the air we breathe cleaner.
- Saving time. fewer cars - fewer traffic jams. that is to say, it is possible to reach a destination point faster and find a parking place.
- Meeting new people. traveling together allows you to find good friends.

2.3.2 Disadvantages of carpooling

- Indecent passengers. some people can try to discount the price or even ask the driver to visit the place that is not on the scheduled route. and it is important to screen out such individuals at the very beginning of traveling.
- Indecent drivers. unfortunately, some drivers can play an unfair game as well. for example, drivers who take 5-6 people in a small car and ask for a high price.
- In some cases, a driver has to pick up each companion separately. it increases the time of traveling.
- Passengers can be not satisfied with the driving style of a car owner. in turn, the driver can be annoyed with an excessive volubility of companions, their untidiness or lack of manners

Chapter 3

Requirement Specifications

3.0.1 Existing System for carpooling

Many carpoolings applications and websites have been developed around the world. a similar carpooling system was developed in Massey University New Zealand by a group of students to allow students of Massey University, Albany campus to share their vehicle with non-vehicle owning students.
Following some examples of carpooling systems around the globe :

3.0.2 Websites

- New Zealand: https://www.asa.ac.nz/carpool
- Algeria: www.nroho.com, www.m3aya.com, www.nsogo.net
- Europe: BlaBlaCar.com, carpooling.com, GoMore.com
- France: covoiturage.fr
- USA: car.ma, www.rdvouz.com
- World: Outpost.travel, joinntravel.com, www.letsride.in

3.0.3 Mobile Applications

- New Zealand: ASA
- Algeria: YAssir, Nsogo, AMIR
- World: Uber, sRide, RideShare,
- USA: Uber, Lyft
- France: Karos, Wever, BlaBlaCar, OuiHop

3.1 Proposed System

Our purposed system is a “Carpool” application which is a ride sharing application designed just for students. students can login or signup to this application only via university email id to make sure that only enrolled students in a university used this application.
Vehicle owning students can share their rides with other students for traveling to and from their institutes and earn money.

3.2 Requirement Specifications

It involves functional and non-functional functionalities that must be performed by the system:

3.2.1 Functional Requirements

3.2.1.1 Table 1: Functional Requirement - 01

Identifier	FR-01
Title	Create Account
Requirement	Registered New User
Source	Supervisor, M. Shoaib Malik
Rationale	To registered new users
Restrictions and Risk	User can only be registered via university email
Dependencies	Android phone, Google API, Firebase server, Firebase Authentication
Priority	High

3.2.1.2 Table 2: Functional Requirement - 02

Identifier	FR-02
Title	Sign In
Requirement	Already registered
Source	Supervisor, M. Shoaib Malik
Rationale	It’s essential to use this application
Restrictions and Risk	User must be registered on this application
Dependencies	Google API, Firebase server, Firebase Authentication
Priority	High

3.2.1.3 Table 3: Functional Requirement - 03

Identifier	FR-03
Title	Reset Password
Requirement	Already registered on this application.
Source	Supervisor, M. Shoaib Malik
Rationale	Reset user account password
Restrictions and Risk	Have access to university email
Dependencies	Firebase server, Firebase Authentication, Google Map Api
Priority	Low

3.2.1.4 Table 4: Functional Requirement - 04

Identifier	FR-04
Title	Switch to Driver / Rider
Requirement	Sign In
Source	Supervisor, M. Shoaib Malik
Rationale	Confirm the role of user
Restrictions and Risk	User have to choose only one role at a time.
Dependencies	Firebase server
Priority	Medium

3.2.1.5 Table 5: Functional Requirement - 05

Identifier	FR-05
Title	Vehicle Details (Driver)
Requirement	Choose vehicle type car / bike
Source	Supervisor, M. Shoaib Malik
Rationale	To make sure the vehicle type and detail
Restrictions and Risk	Nil
Dependencies	Firebase server
Priority	Medium

3.2.1.6 Table 6: Functional Requirement - 06

Identifier	FR-06
Title	Create Ride (Driver)
Requirement	Choose role of a driver
Source	Supervisor, M. Shoaib Malik
Rationale	Select riders from list
Restrictions and Risk	Driver have to choose only certain number of riders according to free seating capacity.
Dependencies	Firebase server, Google Map Api
Priority	Medium

3.2.1.7 Table 7: Functional Requirement - 07

Identifier	FR-07
Title	End Ride (Driver)
Requirement	Driver ends the ride.
Source	Supervisor, M. Shoaib Malik
Rationale	To make sure all riders dropped.
Restrictions and Risk	Nill
Dependencies	firebase server, Google Api
Priority	Medium

3.2.1.8 Table 8: Functional Requirement - 08

Identifier	FR-08
Title	Book Ride (Rider)
Requirement	Choose role of a rider.
Source	Supervisor, M. Shoaib Malik
Rationale	Select destination and pickup point
Restrictions and Risk	Rider have to choose only available pickup point.
Dependencies	Firebase server, Google Map Api
Priority	Medium

3.2.1.9 Table 9: Functional Requirement - 09

Identifier	FR-09
Title	Ride End (Rider)
Requirement	Driver pick the rider
Source	Supervisor, M. Shoaib Malik
Rationale	To make sure that driver drops a rider to destination.
Restrictions and Risk	Nil
Dependencies	Firebase server
Priority	Medium

3.2.2 Non-Functional Requirements

3.2.2.1 Table 10: Non-Functional Requirement - 01

Identifier	NFR-01
Title	User Authentication
Requirement	User must be registered via university email id

3.2.2.2 Table 11: Non-Functional Requirement - 02

Identifier	NFR-02
Title	Real time location tracking
Requirement	Mobile phone must be provide accurate GPS location of device.

3.2.2.3 Table 12: Non-Functional Requirement - 03

Identifier	NFR-03
Title	Multi-user system
Requirement	Efficient use of the system when the user increases.

3.2.2.4 Table 13: Non-Functional Requirement - 04

Identifier	NFR-04
Title	Internet connection
Requirement	User device must have a internet connection.

3.2.2.5 Table 14: Non-Functional Requirement - 05

Identifier	NFR-05
Title	Device compatibility
Requirement	Use of latest APIs and application must support most of android phone versions

3.2.2.6 Table 15: Non-Functional Requirement - 06

Identifier	NFR-06
Title	User friendly application
Requirement	Easy interface of application, not makes a user to think twice.

3.2.2.7 Table 16: Non-Functional Requirement - 07

Identifier	NFR-07
Title	Application limited to university students
Requirement	Only students with university email able to use bus system

3.3 Use Cases

3.3.1 Use Case Diagram

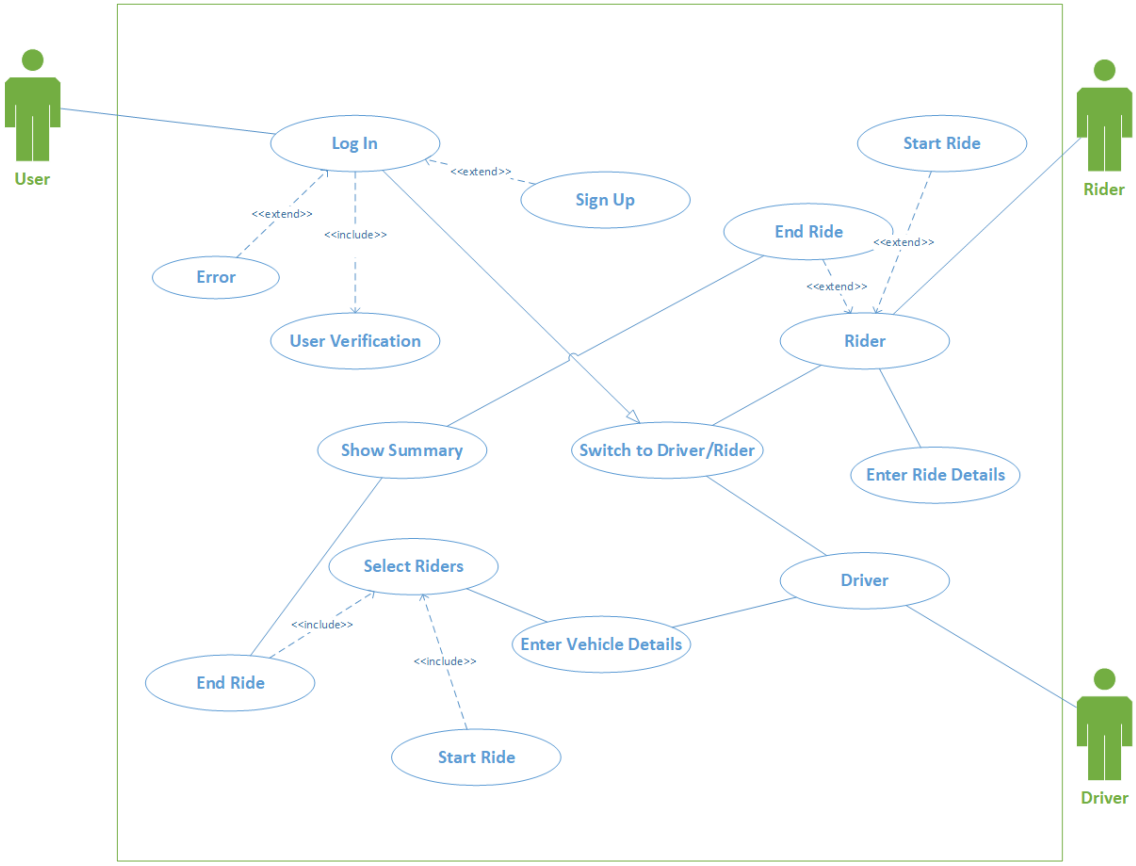


Figure 3.1: Use Case Diagram

3.3.2 Use Case Description

3.3.2.1 Table 17: Use Case - 01

Use Case ID:	UC-01
Use Case Name:	Login
Actors:	Rider, driver
Description:	Signing in to use the application
Trigger:	Perform certain tasks based on user type
Preconditions:	Splash screen
Postconditions:	Home screen based on user type
Normal Flow:	Successful sign in
Alternative Flows:	Sign in failed, try again or reset password
Exceptions:	The account doesn't exist
Includes:	Verify password
Assumptions:	User has an account
Notes and Issues:	nil

3.3.2.2 Table 18: Use Case - 02

Use Case ID:	UC-02
Use Case Name:	Sign Up
Actors:	Driver, Rider
Description:	To use this application registration is required
Trigger:	Registered on this application
Preconditions:	Enrolled in a university
Postconditions:	Have access to university email for account verification
Normal Flow:	Log in to application
Alternative Flows:	Not eligible for registration
Exceptions:	University Email is not verified
Includes:	Must have access to university email
Assumptions:	User is enrolled in a universirty
Notes and Issues:	User can be registered only via university email

3.3.2.3 Table 19: Use Case - 03

Use Case ID:	UC-03
Use Case Name:	Select User Type
Actors:	Driver, Rider
Description:	Switch according to your need
Trigger:	User Switch to particular user type
Preconditions:	Login
Postconditions:	User either act as a rider or driver
Normal Flow:	Switch to splash screen according to user type
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	User have to choose user type
Assumptions:	Both type users are university students.
Notes and Issues:	Nil

3.3.2.4 Table 20: Use Case - 04

Use Case ID:	UC-04
Use Case Name:	Driver
Actors:	Driver
Description:	User can acts as a driver
Trigger:	User acts as a driver
Preconditions:	User type switch screen
Postconditions:	User select riders
Normal Flow:	User is now acts as a driver
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	User must have vehicle
Assumptions:	Driver must be student
Notes and Issues:	Nil

3.3.2.5 Table 21: Use Case - 05

Use Case ID:	UC-05
Use Case Name:	Rider
Actors:	Rider
Description:	User can acts as a rider
Trigger:	User acts as a rider
Preconditions:	User type switch screen
Postconditions:	Enter ride details
Normal Flow:	Now user acts as a driver
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	Rider is non-vehicle owning person
Assumptions:	Rider is a student
Notes and Issues:	Nil

3.3.2.6 Table 22: Use Case - 06

Use Case ID:	UC-06
Use Case Name:	Enter vehicle details
Actors:	Driver
Description:	User acts as a driver and enter the vehicle detail
Trigger:	Enter vehicle details
Preconditions:	Switch to driver
Postconditions:	Select riders
Normal Flow:	Select vehicle type and enter vehicle details
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	User vehicle details update to database server
Assumptions:	Nil
Notes and Issues:	Nil

3.3.2.7 Table 23: Use Case - 07

Use Case ID:	UC-07
Use Case Name:	Enter ride detail
Actors:	Rider
Description:	Show current location of rider and rider have to select pickup point and drop off points.
Trigger:	Display user current location on map and pickup points list .
Preconditions:	User switch to rider mode.
Postconditions:	Wait for driver
Normal Flow:	Display current location from GPS and show list of drop off and pick up points.
Alternative Flows:	Nil
Exceptions:	Device GPS location access not given by user
Includes:	Loading current location from GPS and update data to server
Assumptions:	Nil
Notes and Issues:	Nil

3.3.2.8 Table 24: Use Case - 08

Use Case ID:	UC-08
Use Case Name:	Start ride (Driver)
Actors:	Driver
Description:	After selection of passengers, driver has to start a ride
Trigger:	Start ride
Preconditions:	Select riders from list
Postconditions:	Drop riders and end ride
Normal Flow:	Ride is start and notified detail of driver to riders
Alternative Flows:	Ride cancel
Exceptions:	Riders not available
Includes:	Must have to select riders to start ride
Assumptions:	Riders available
Notes and Issues:	Nil

3.3.2.9 Table 25: Use Case - 09

Use Case ID:	UC-09
Use Case Name:	Start Ride (Rider)
Actors:	Rider
Description:	Start ride when driver pick up the rider
Trigger:	Start ride
Preconditions:	Enter ride details
Postconditions:	End ride
Normal Flow:	Start ride and display driver details to rider
Alternative Flows:	Nil
Exceptions:	Driver is not availabe
Includes:	Must enter ride details
Assumptions:	Driver available
Notes and Issues:	Nil

3.3.2.10 Table 26: Use Case - 10

Use Case ID:	UC-10
Use Case Name:	End ride (Driver)
Actors:	Driver
Description:	After drops the riders end the ride and summary of current ride is shown.
Trigger:	Ride End and show summary
Preconditions:	Start Ride
Postconditions:	Show summary
Normal Flow:	Succesfully end ride and show summary of current ride
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	Driver drops all riders to their drop off locations
Assumptions:	Nil
Notes and Issues:	Nil

3.3.2.11 Table 27: Use Case - 11

Use Case ID:	UC-11
Use Case Name:	End ride (Rider)
Actors:	Rider
Description:	Ride is end by rider and show the summary of ride
Trigger:	End ride and show ride details
Preconditions:	Pick up by driver
Postconditions:	Show summary of ride
Normal Flow:	Succesfully ride End and show summary of ride.
Alternative Flows:	Nil
Exceptions:	Nil
Includes:	Driver drops the rider to their drop off location
Assumptions:	Nil
Notes and Issues:	Nil

Chapter 4

Design

4.1 System Architecture

4.1.1 High Level Context Diagram

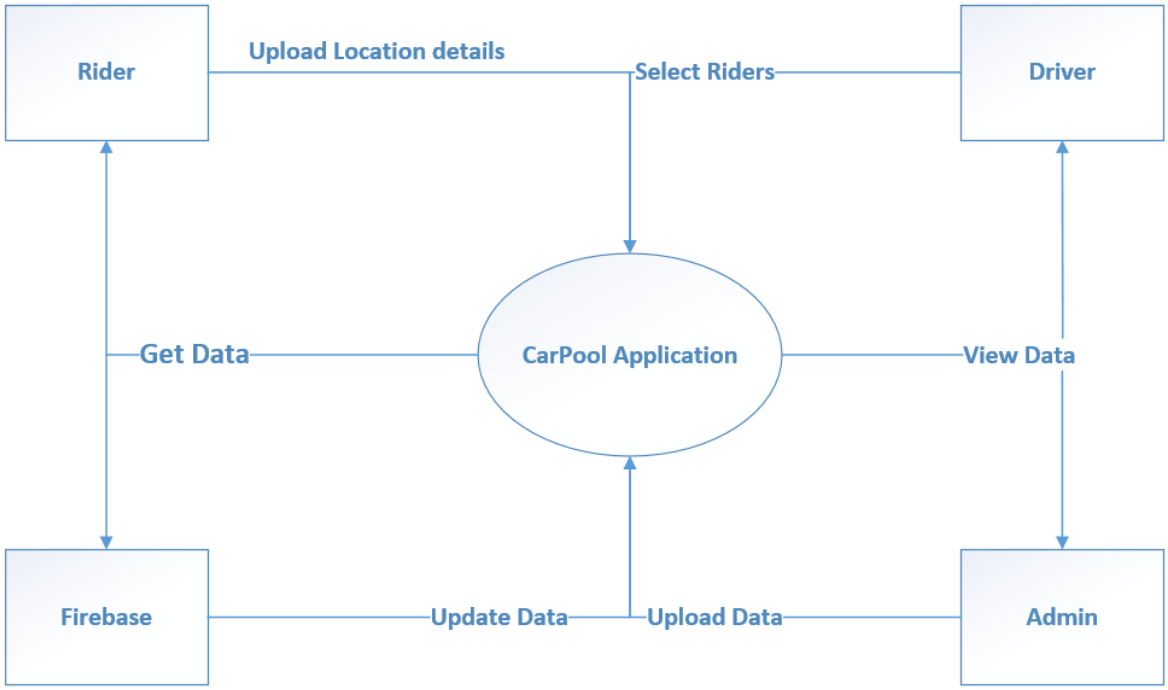


Figure 4.1: Context Diagram

4.2 Design Constraints

In our application we add a list of pick up points for the ease of drivers. This is made easier when clients can select pick up points and drivers reach the destination

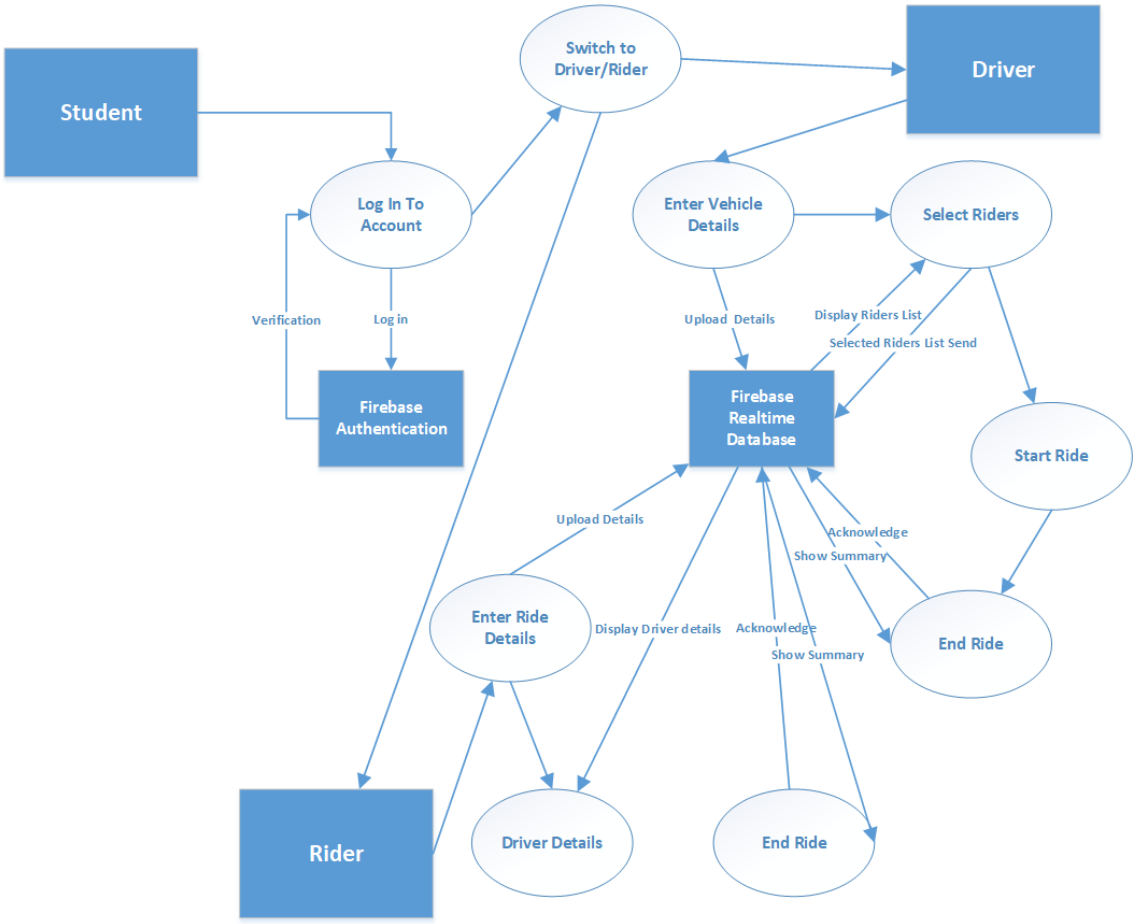


Figure 4.2: Sub System

without hassle. due to the COVID-19 pandemic, we could not be supervised and guided properly. hence this process was not added and executed.

4.3 Design Methodology

The design of a project is important for the structure of the application by using UML (Unified Modeling Language) which is a general purpose modelling language, that aims to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.
Reasons to use UML for project analysis and design are:

- Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- A lot of time is saved down the line when teams are able to visualize processes, user interactions and static structure of the system. these project designs will try and make the overall idea of the project more understandable and clear by identifying actors and functional / non-functional needs, and also all the diagrams needed to give a clear view about this project.

4.4 High-Level Design

4.4.1 Conceptual or Logical: UML Package diagram

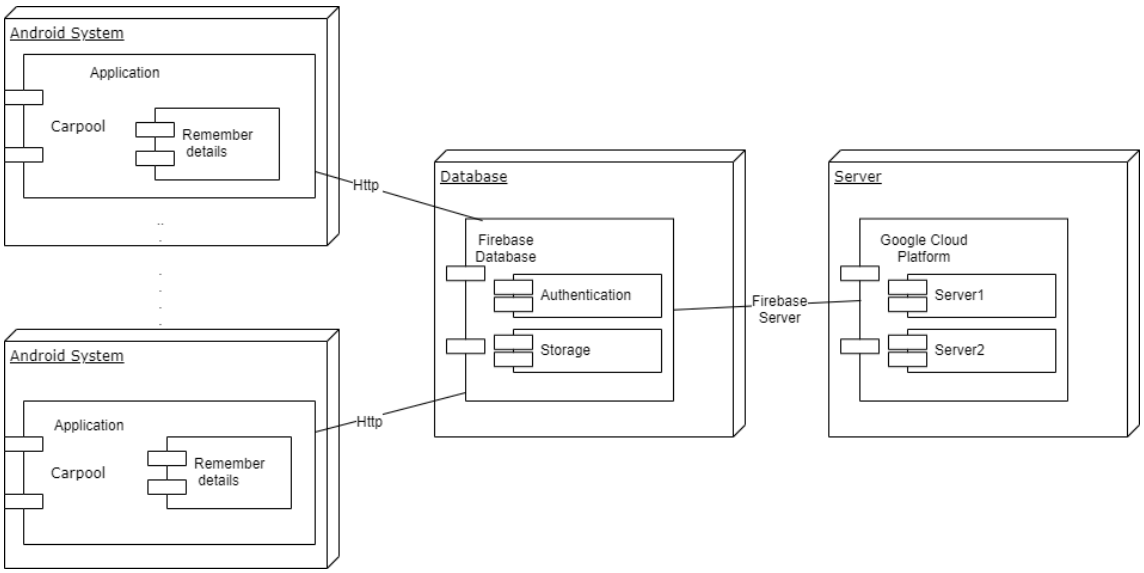


Figure 4.3: Package Diagram

4.4.2 Process Interaction Diagram

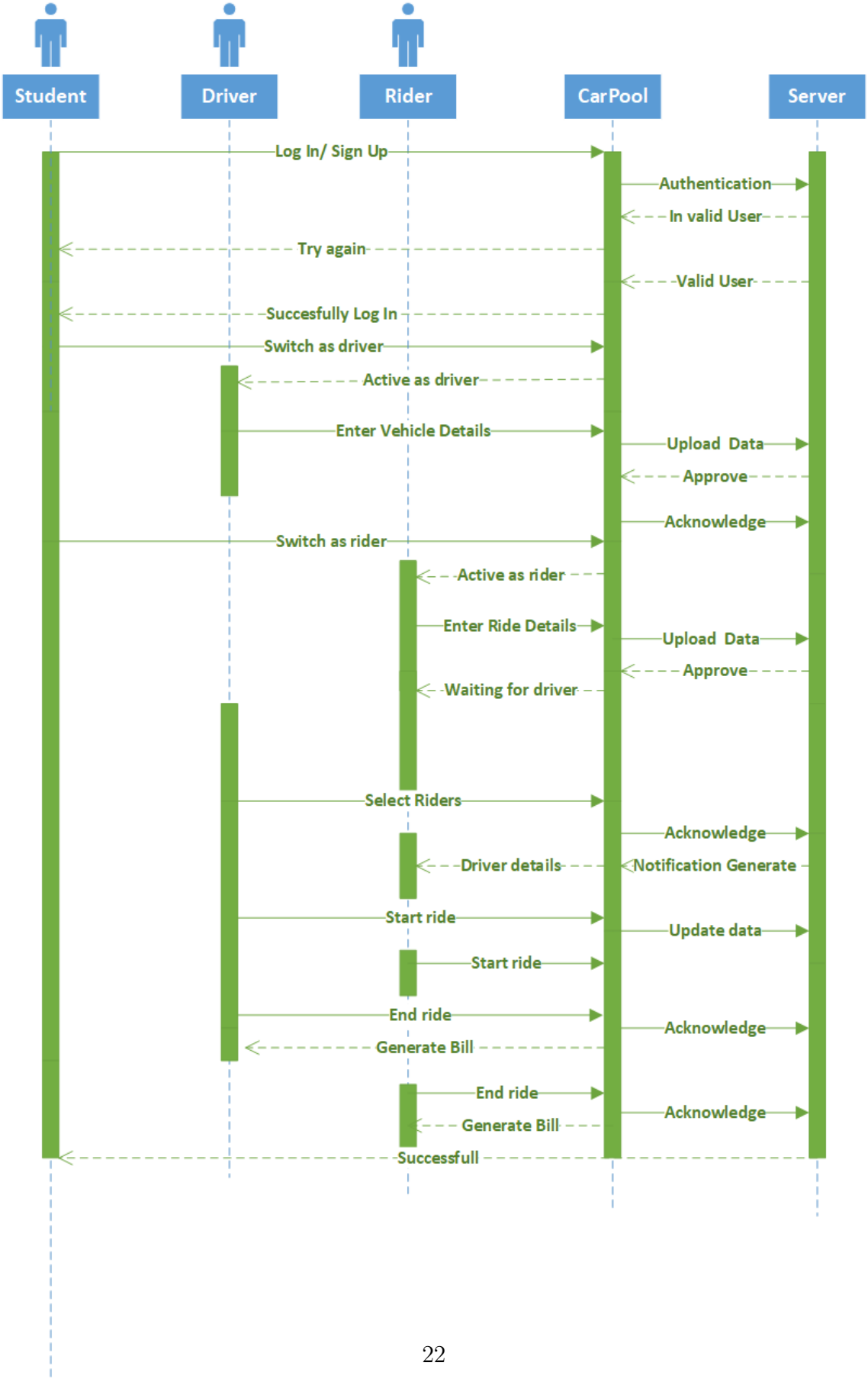


Figure 4.4: Process Interaction Diagram

4.4.3 Physical Deployment Diagram

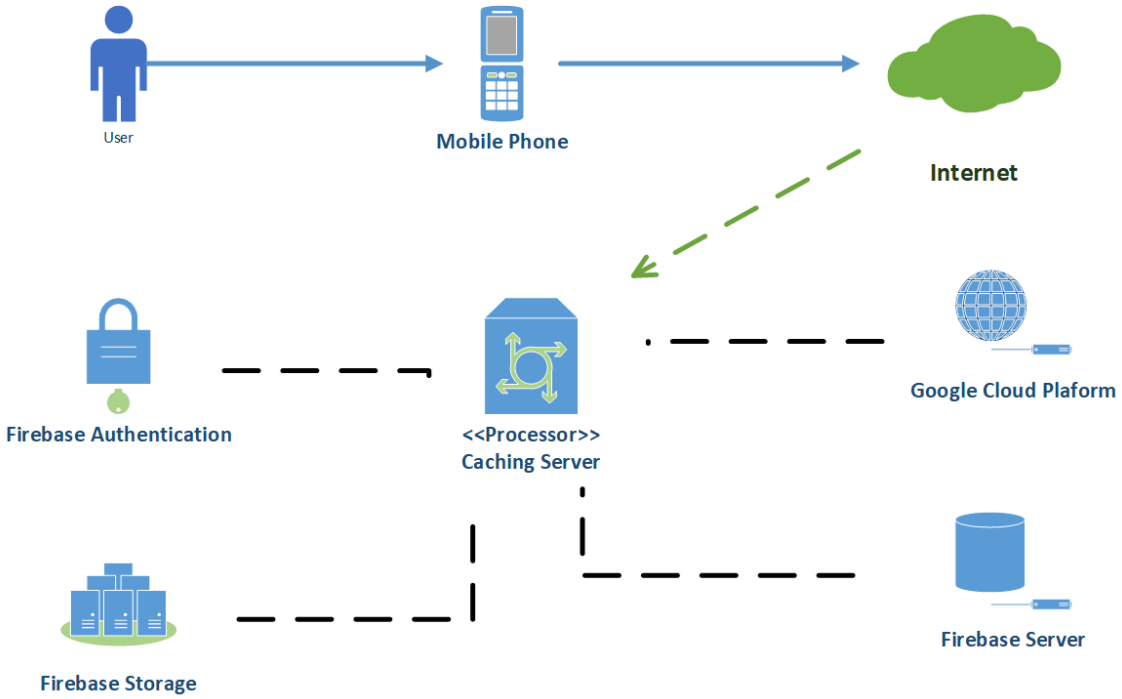


Figure 4.5: Deployment Diagram

4.4.4 Modules

The application is divided into three modules

- User Registration
- Driver
- Rider

4.4.4.1 User Registration

In this module, the user must register their credentials to use this application. users should be student and currently enrolled in Air University. the user should enter the details first name, last Name, student email, university registration id, contact no, password. so user using these registration details will be able to use application. this authentication process to avoid someone malpractice as this application is verifying student details.

4.4.4.2 Driver

In this module, the user acts as a driver. this module is consist of 8 activities which contains 2 fragments (map, passenger list), the driver can use the map fragment to see the passenger pickup points and current location and passenger list fragment to see the list of passengers. driver module also contains the activities of enter vehicle details, summary of ride.

4.4.4.3 Rider

In this module, the user acts as a rider. this module is consist of 4 activities which contains 2 fragments (map, driver details), the passenger can use the map fragment to see the pickup point and current location and driver detail fragment to see the detail of driver. rider module also contains the activities to enter trip detail, summary of ride.

4.4.5 Security

This application should never disclose any personal information of any user, and should collect no personal information from it's own users.

4.5 Low Level Design

Figure 4.6: Low Level Design

4.6 Database Design

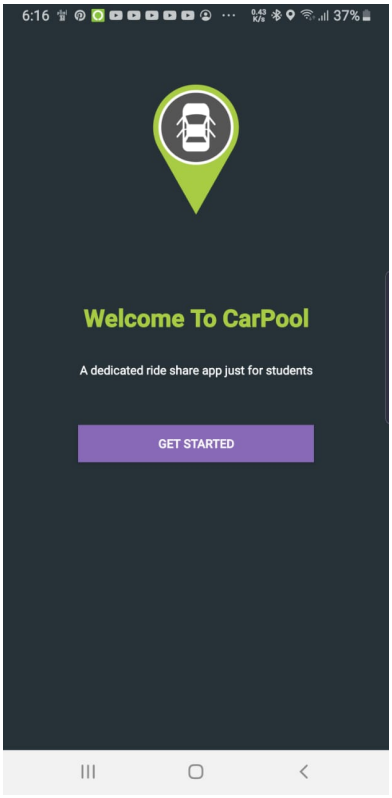
Firebase platform provides a cloud-hosted realtime database. It is a NoSQL cloud database. data synchronizes among all clients and stays available even when the app goes offline. as it is a NoSQL database, so its functionality is different as compared to a relational database. the data is stored in the form of collections and documents. there is no relationship between classes, so there won't be any schema of our database. data gets stored as JSON objects. it is more like a cloud-hosted JSON tree. when a person adds data to the JSON tree, it becomes a node in the existing JSON structure with an associated key. the best method is to use the denormalization technique; using which we split data into separate paths. it becomes easier to download the data in separate calls as required.

4.7 GUI Design

Our application contains more than fifteen screens, different screens based on user type are:

4.7.1 Welcome Interface

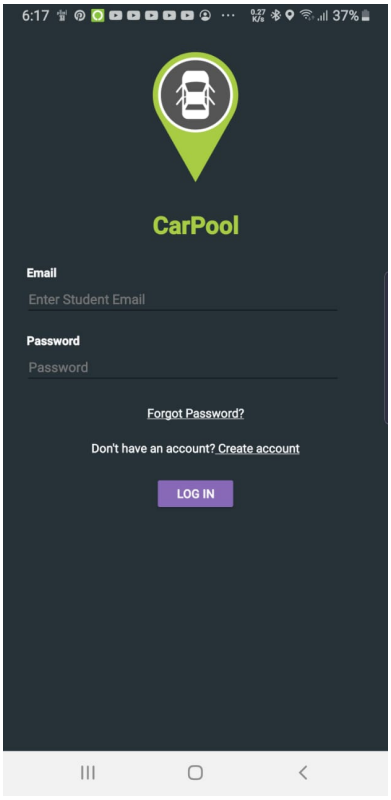
This is the interface that shows up to the user when he's open the application, this interface is composed of the application's name, a logo and a slang which describe the application alongside an "Get Started" button which lets the user to log in screen.



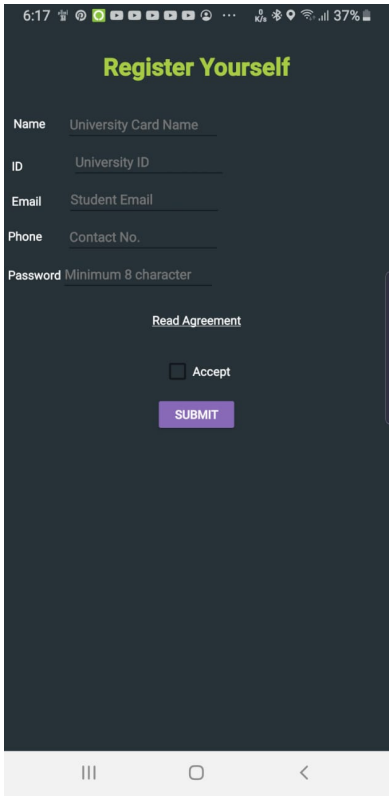
(a) Welcome Screen

4.7.2 Authentication Interfaces

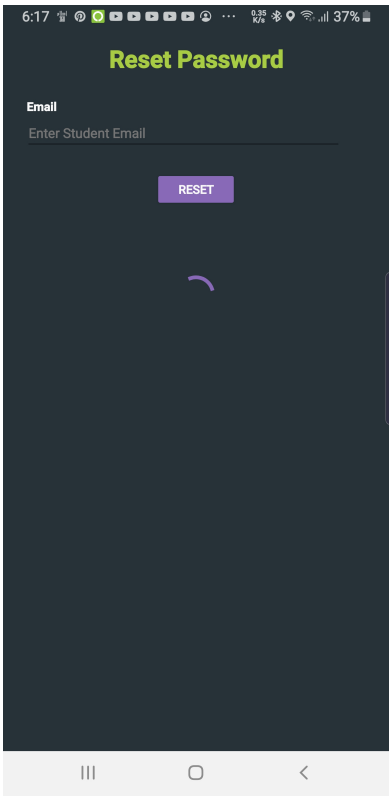
The authentication interface is made as simple as possible while also being very beautiful and professional, it utilizes firebase auth as a backend which allows for a secure connection to the database plus encrypted passwords for extra security, this interface also contains options to recover password in case the user has forgotten their password.



(a) Login Screen



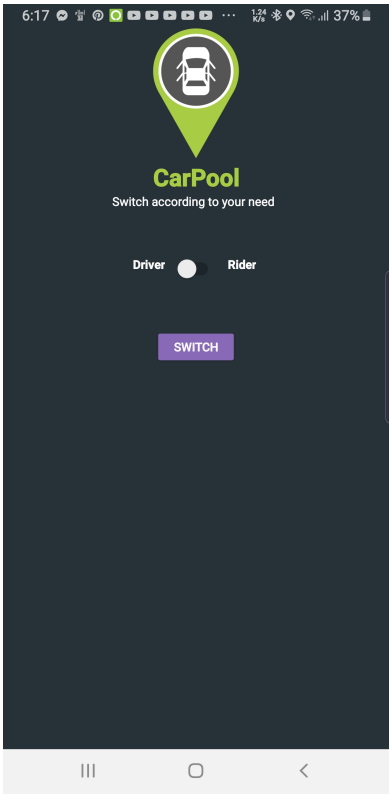
(b) Signup Screen



(a) Forgot Password Screen

4.7.3 Type Switching Interface

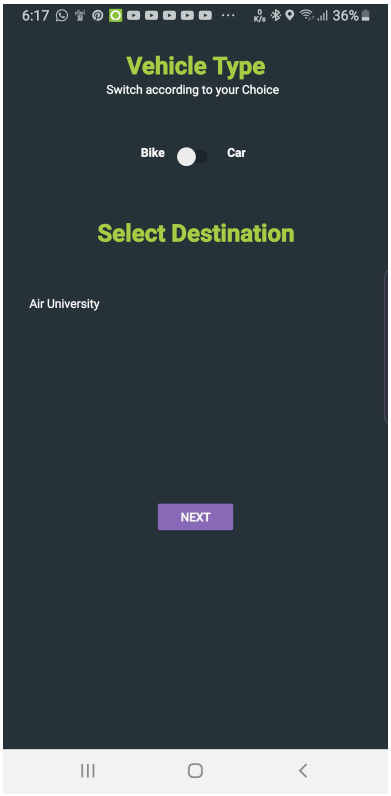
This is the interface that shows up to the user when he's logged in into the application, this interface is composed of the application's name and a logo alongside a switch button, which lets the user to their selected user type interface.



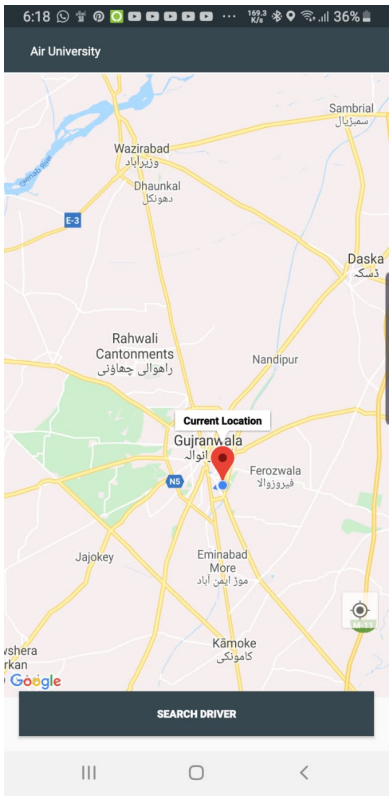
(a) Select User Type Screen

4.7.4 Rider Interfaces

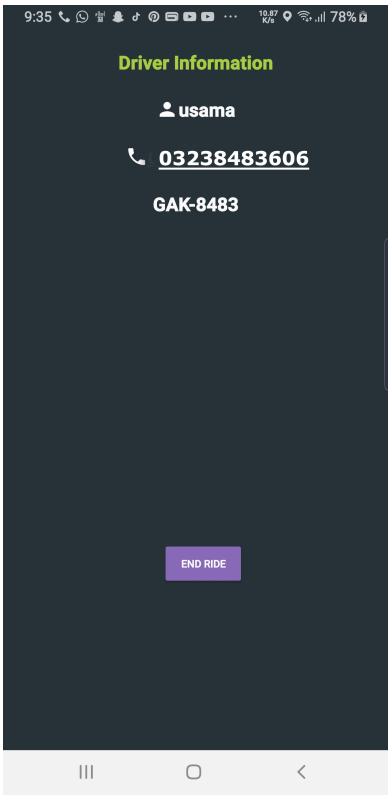
These interfaces are used by the passengers to post a ride details for drivers to see and select riders, these interfaces is extremely customizable and professional, by using Google Places API, Google Direction API, Google Map API in these interfaces. we have created a very organized and efficient algorithm to optimize the results when Passengers search for a driver, when rider picks an origin and a destination using Places AutoComplete API, the algorithm uses a combination of Geolocation and Places API to find the full address, city and sub city, then it converts the city into a number and saves all the information in the trip object for it to be saved into the database later.



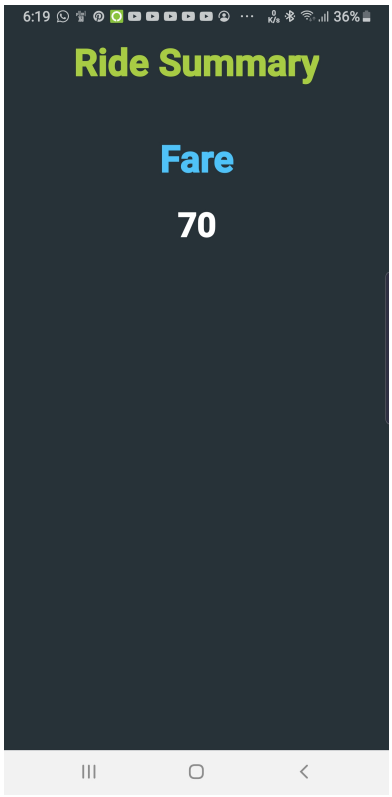
(a) Enter Ride Detail Screen



(b) Map Screen



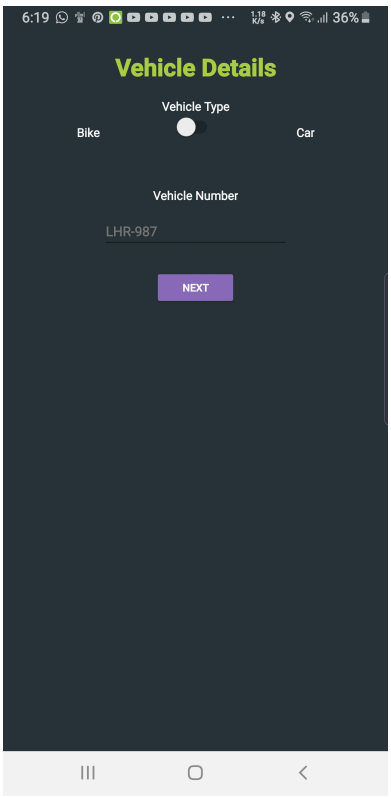
(a) Driver Detail Screen



(b) Ride Summary Screen

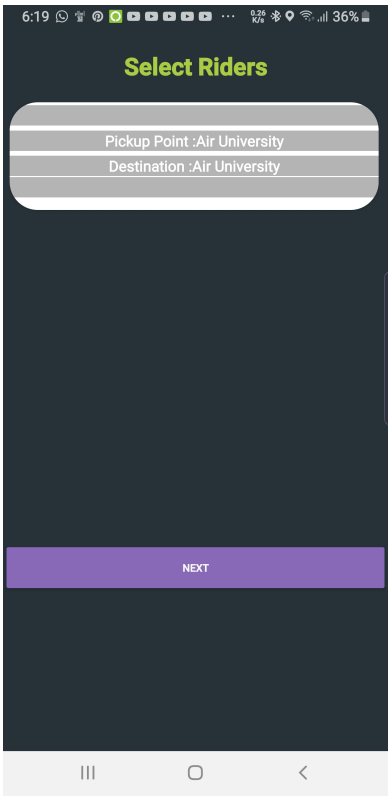
4.7.5 Driver Interfaces

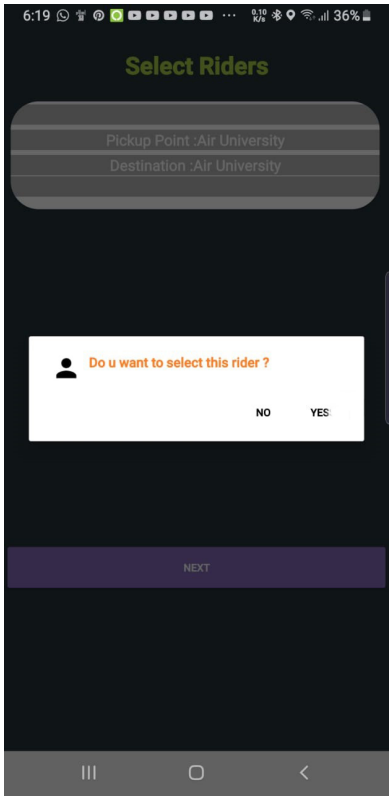
These interfaces are used by rider to enter vehicle detail, see and select the passengers for trip, these form is extremely customizable and professional, by using Google Places API in this interface we have created a very organized and efficient algorithm to optimize the results, When the driver select riders using Places AutoComplete API, the algorithm uses a combination of Geolocation and Places API to find the full address, city and sub city, then it converts the city into a number and saves all the information in the trip object for it to be saved into the database later.

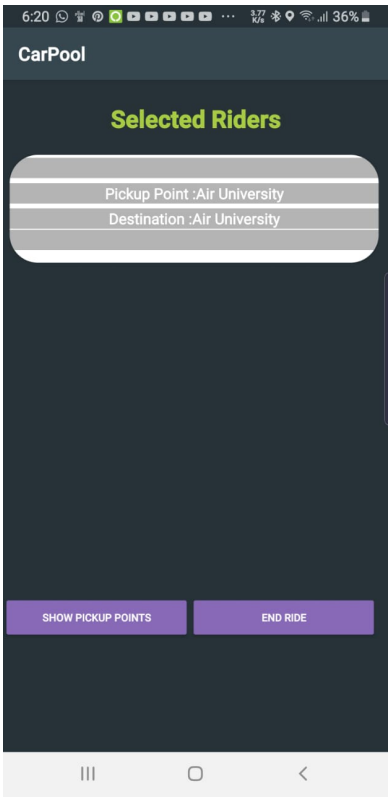


(a) Enter Vehicle Detail Screen

(b) Driver Map Screen

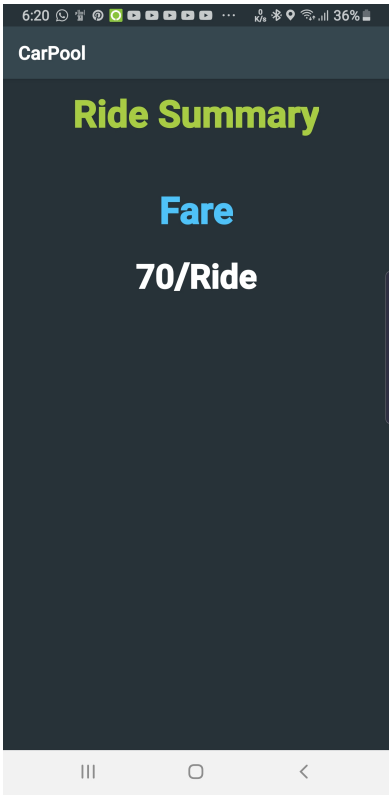






(a) Selected Passengers Screen

(b) Passengers Pickup Points Screen



(a) Driver Summary Screen

Chapter 5

System Implementation

5.1 System Architecture

application architecture is a set of technologies and models for the development of fully-structured mobile programs based on industry and vendor-specific standards. as we develop the architecture of our application, we also consider programs that work on wireless devices such as smartphones and tablets.

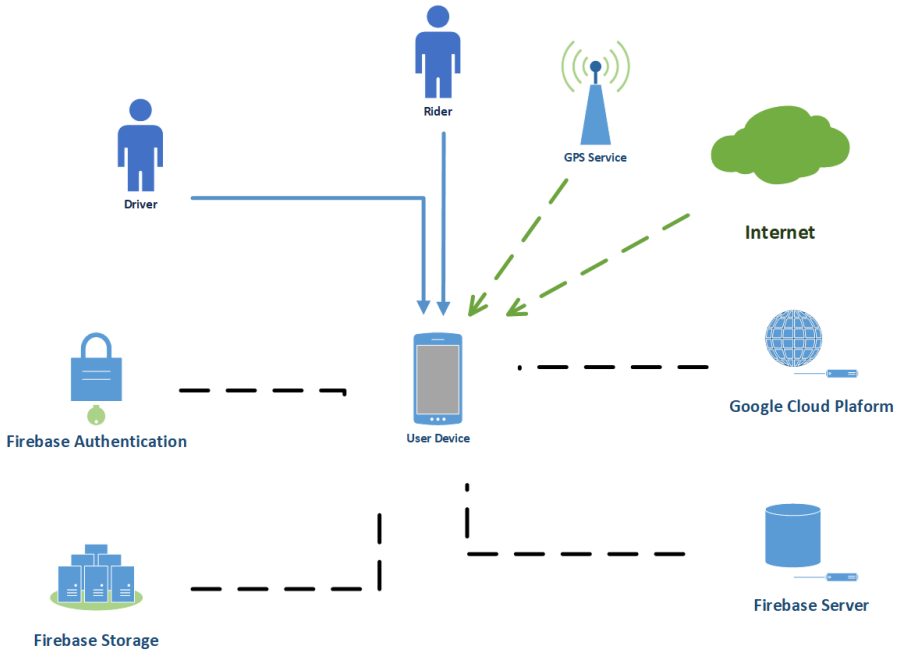


Figure 5.1: System Architecture

5.2 Work Environment

5.2.1 Hardware platforms

For the development of this application we have used an Hp Elitebook workstation 8460w it fairly powerful and could handle the usage of multiple emulators at once.

Full specifications:

- **Processor** : Intel® Core™ i7-2630QM Processor (2.0 GHz, 6 MB L3 Cache)
- **Memory** : 12GB 1333 MHz DDR3 SDRAM (2D)
- **Graphics** : AMD FirePro™ M3900 w/1 GB gDDR3
- **HDD Drive** : 1TB

For the testing of the application on a real device we have used an android phone Samsung Galaxy Note 8, Huawei Mate 10 Pro.

5.2.2 Software platforms

As for the software part of the work environment we have used several tools and frameworks alongside different versions of android.

5.2.3 Work Tools

- **Operating System** : Windows
- **IDE** : Android Studio 3.2.1
- **Emulators** : Pixel 3 Android 9 / Nexus 5 Android 6
- **Real Device OS**: Android 9 Pie (Samsung Galaxy Note 8)
- **Backend Management**: Firebase Platform
- **Places and Map Tools**: Google Cloud Platform

5.2.4 Programming languages:

- **Application Programming** : Java
  Motivation: Java code is inherently safer than Kotlin code because it prevents common programming mistakes by design, resulting in fewer system failures and application crashes. when using Kotlin, certain error causes are more likely to occur again.
- **Layout Design** : XML
- **Database Language** : NoSQL

5.2.5 Frameworks used:

- **Firebase Auth** : this framework that lets you implement easy authentication in this application email and phone number authentication was used.
- **Firebase Real-time Database** : this framework is used as our main database for this application it lets you read and write data from firebase in a NoSQL structure.
- **Firebase Storage** : this framework is used to upload and download data such as photos and videos, this is used to store user pictures for our database.
- **Firebase Messaging** : a framework that makes real time messaging easy and possible using firebase, it's also used for push notifications, this used for our messaging function between users in our application.
- **Google Maps API** a great API used for MapView that enables the application to show places on the map like the origin and destination and the way between them, it's used in the map to trace the destination on the map for users.
- **Google Places API** : a very powerful API used to give information about places in the application and used to AutoComplete search queries, it's used to auto complete search queries for our users for easier searching, it's also used to optimize our search function.
- **Material Design Library** : material design Library with all of its components such as material buttons, material EditText etc.
- **RuntimePermission** : a small library to help manage android permissions with no problems.
- **DxLoadingButton** : a small library with a loading button used for authentication.
- **StfalconImageViewer** : a simple and customizable android full-screen image viewer with shared image transition support, "pinch to zoom" and "swipe to dismiss" gestures.
- **Spinner** : a styleable drop down menu for android using the old spinner style.
- **Tooltip** : simple to use customizable android Tooltips library based on popup window.
- **EasyValidation** : a text and input validation library in Java for android. used to validate user input info.
- **Retrofit** : type-safe HTTP for android and Java.

5.3 Database Management System

To manage the backend of our application we decided to use firebase real-time database alongside with firebase Auth and Firebase Storage.

- **Firebase Real-time Database**
  it's a database structured in a NoSQL way, it's usage is very easy and very fast, we have used this database to store User objects which contain all the information about the user such as name, email, number etc., we have also used it to store trip and trip request objects . **example of the user node in the database**

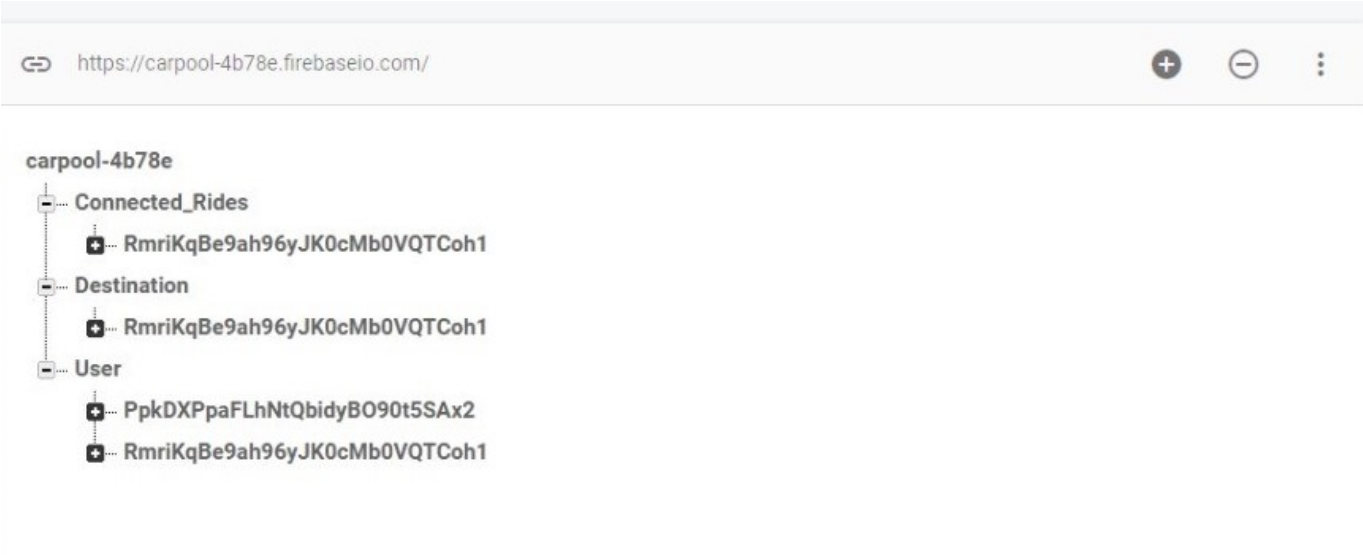


Figure 5.2: Example of a user mode in database

- **Real-time Database Usage**
  For the usage of this database a separate database class was made on Java this class contains all database related functions the functions require a listener interface that has **onStart(), onSuccess(), onFailed(),** the interface is used to listen to changes in other classes.
  The reason for creating a separate class with a listener interface is to make migration to another database easier. If we ever decide to switch to another database we will only change the content of the database class which would make changes faster, more professional and effective.
  Firebase's real-time database allows you to use different query methods to fetch data from its Database:
  The first method is using **addListenerForSingleValueEvent,** this method listens to changes in data for one time and does not trigger until it's called again, this is good for fetching data only once.
  The second method is using **addValueEventListener,** this method listens to data every time it changes and it also triggers every time it changes this is

good for things like messages and real time updates.
We have decided to use **addListenerForSingleValueEvent** for most of our data fetching, the reason is each time data has been read, Firebase charges for it, which means it the less data is fetched the better, for this we have avoided using **addValueEventListener** instead we fetch the data once and save it locally, when the data is changed the user is required to refresh the page to get updates. however in cases where it's important to get updates in real time like in messages we have used **addValueEventListener**.

5.4 Tools and Technology

- Android
- Firebase Platform
- Google Cloud Platform
- Android Studio
- Java Language
- Google Maps API
- Google Direction API
- Google Places API

5.5 System requirements

- Android phone with Minimum SDK android API 19 Kit Kat
- Google Play Services
- GPS service
- Internet Connection

5.6 Security

- **Firebase Authentication**
  Firebase authentication is a very fast and secure way to sign users into our application it is used to log in and register users into our database in our case we are using the email verification, a user must verify his email to complete the registration.

- **Authentication Usage**
  When a user registers, his basic information (Name, Email, Password) are registered in the Auth database but not the real time Database, after that users are welcomed with a finish registration activity, in this activity they have to verify their email using EMAIL verification, this method is used to prevent spam and multiple account creations, after verification and filling other information user data is saved on the real time database. as for the log in the system checks data and logs users if it's correct.

5.7 Rules

Some of the restriction and rules are mentioned below :

- User can be used one account for both driver and rider.
- Users cannot modify anyone else's data except theirs.
- Only the admin can modify the data of other users.
- The driver cannot select more than three riders.
- The rider can only contact with their driver.
- A rider have to select pick up and drop off points.
- Driver, rider cannot use the app until their accounts are verified.

Chapter 6

System Testing and Evaluation

6.1 Graphical user interface testing

6.1.0.1 Table 28: user interface testing

No	Test Objective	Test Step	Expected Result	Result
1.	Check the color and size of buttons on the home screens of different users.	1) Make boundaries visible after activating developer mode in your android phone 2) Open all the users home screen one by one and you will feel the difference	Buttons size and color must be same in all user dashboards	Affirmative
2.	Make sure that icons pixels won't fall when the app would be opened on different screen sizes.	1) Open your app on the screen of different sizes.	Icons appearance would be the same on all screens.	Affirmative
3.	Ensure the ripple effect working on all buttons.	1) Check by pressing all the buttons added in-app.	Ripple effect will generate when you will press button	Affirmative
4.	Is text visible on smaller screens?	1) Open the app on small screens.	The text is readable by the user.	Affirmative
5.	Buttons must be separated by some distance so the user won't press the wrong button.	1) Press the buttons that are placed together or have little distance	The user was able to press the desired button with his thumb	Affirmative

6.2 Graphical user interface testing

6.2.0.1 Table 29: Usability testing

No	Test Objective	Test Step	Expected Result	Result
1.	To check registered on CarPool .	1) Press the sign up button. 2) Fill the form. 3) Accept only university email for sign up. 4) Send verification email on email. 5) After verification process account has been created.	User will be able to use this application.	Affirmative
2.	Forgot Passwrod ?	1) Click the forgot password button. 2) Text box will appear enter registered email. 3) Press reset button email has been sent to your email. 4) Check your email and set your new password.	Account password reset.	Affirmative
3.	Login to Carpool application.	1) Enter registered email. 2) Enter account password. 3) Press login button.	User login to application	Affirmative
4.	Select user type driver.	1) Login in to the application. 2) Switch button to driver type. 3) Enter vehicle details and press next button to update details in database.	User act as a driver.	Affirmative
5.	Check as If a driver can select riders.	1) Login as a driver. 2) Enter vehicle details. 3) Show riders list. 4) Select riders.	Riders selected and send driver details to riders.	Affirmative
		4) Select riders.		
6.	Check can driver see pick up points of selected riders ?	1) Display list of selected riders on screen. 2) Press button show pick up points. 3) Display pick up points of riders on map.	Driver will be able to see the pickup points of riders.	Affirmative
7.	Drop off riders on different location.	1) Select rider pop up will appear. 2) Press drop off. 3) Rider eliminate from pick up list	Rider drop off on their location.	Affirmative
8.	Ride summary display.	1) Drop off all riders one by one or press the end ride button . 2) Ride summary will appear.	Display ride summary.	Affirmative
9.	Select user type rider.	1) Login in to the application. 2) Switch button to rider type. 3) Rider have to select destination and pick up point. 4) Press next button and user current location, drop off and pick up point display on map.	User act as a rider.	Affirmative

10.	Notify details of driver to rider.	1) After enter ride details a waiting screen will display. 2) When rider select by driver details of driver will be display on rider screen. 3) When driver pick up rider press start ride button.	Details of driver will be shown to rider.	Affirmative
11.	Show summary of ride to rider.	1) when driver drop off rider press end ride button. 2) Ride summary display on screen.	Show ride summary.	Affirmative

6.3 Software performance testing

6.3.0.1 Table 30: user interface testing

No	Test Objective	Test Step	Expected Result	Result
1.	To check as if the app would work smoothly when multiple riders selected by a driver.	1) Create multiple accounts and put ride details. 2) Select by single driver.	The speed of the app won't be affected.	Affirmative
2.	Check the speed of the app after registered multiple accounts .	1) The app would work as it used to work with twenty complaints.	The speed of the app won't be affected.	Affirmative
3.	To check firebase cloud speed.	1) Ask multiple drivers to enter vehicle details and select riders from list. 2) Ask multiple riders to enter ride details and selected by drivers].	App performance won't be affected.	Affirmative

6.4 Compatibility testing

6.4.0.1 Table 31: Compatibility testing

No	Test Objective	Test Step	Expected Result	Result
1.	To check as if the location of the user keeps on updating when user is moving.	Sign in to the driver and rider account.	It will keep on updating the current location.	Affirmative
2.	Ensure the readability of text on different screens.	Open your app in different screen sizes.	The user would be able to read the text on small screens	Affirmative
3.	Push Notifications keep on coming on different mobile devices.	Using the app on different android versions.	Notifications would display on the notification screen.	Affirmative

6.5 Exception handling

6.5.0.1 Table 32: Exception handling

No	Test Objective	Test Step	Expected Result	Result
1.	Only university email format required for signup; Invalid email won't be accepted.	Enter the email while registering your account.	Registration successful.	Affirmative
2.	If email is not verified by, the user won't be able to sign in the app.	Verification email has been sent to email, when he registers an account.	Account verification done.	Affirmative
3.	An invalid phone number is not allowed.	Write a valid number.	The phone number verified.	Affirmative
4.	Users not allowed to enter the same email twice.	Enter a new email if you want to register your account.	The unique email entered and account approval request sent to admin.	Affirmative
5.	The app won't crash if user do not allow location permission.	Application ask to access the user current location.	App working fine.	Affirmative
6.	Users can't edit roll number or email after signup.	Register your account.	Unable to edit email or roll number.	Affirmative
7.	Only selected riders will be notify by driver details.	Select riders from list and send detail of driver to specific riders only.	Notify the details of drivers to selected riders.	Affirmative

6.6 Security testing

6.6.0.1 Table 33: Security testing

No	Test Objective	Test Step	Expected Result	Result
1.	Eliminate the risk of password leakage when someone tries to guess the password of the user.	1) Try to guess the password of some random user.	It won't work as the user was forced to create a strong password involving numbers and symbols during the signup process.	Affirmative
2.	Prevention of a denial of service attack by an attacker.	1) Detect vulnerabilities in the application layer.	Denial of service attack passed.	Affirmative
3.	Reduce the risk of SQL injection.	1) User is unable to write a query in any text box with the use of proper validation in different forms.	Validations stopped the execution of the query.	Affirmative
4.	To validate an attacker from accessing sensitive data.	1) Try to extract extra data from the admin side.	Data won't be extracted with the proper use of validations.	Affirmative
5.	To analyze the vulnerability of file system interactions.	1) Allowing the user to only add jpg/png files while uploading fees challan.	Malicious data won't get uploaded.	Affirmative

6.7 Installation testing

6.7.0.1 Table 34: Installation testing

No	Test Objective	Test Step	Expected Result	Result
1.	Install the app on different android versions starting from 4.4 to 9.0.	Install the app using the .apk file.	The app would be installed successfully.	Affirmative
2.	To check as if an app installed with all resources.	Open the app.	App working as explained by the developer.	Affirmative
3.	Try uninstalling the app from the android phone.	Click on uninstall option		Affirmative

Chapter 7

Conclusions

7.1 Conclusion

The implementation from a concept to a real application was extremely challenging but it was very effective and full of experience, the final application turned out to be very professional and organized, it contained all necessary features to make both the driver and the passenger feel comfortable using it from design to backend features everything is checked. after a lot of testing and bug fixes the application finally reached a final state with no known bugs.
We have implemented all functional needs and non-functional needs and some of the optional needs, some additional features were added to make the application more usable but the core concept stayed the same since it offers all the mentioned features and needs in the project description it also respects the optimization needs such as good code design style, small application size (9mb) and targeting as many android devices as possible.
By using Java which is the official programming language for android we have made sure that the code was well structured as well as optimized, since android itself is built on Java, there are plenty of Java libraries to your aid. also, Java has a wide open-source ecosystem. Java apps are lighter and more compact, even when compared to Kotlin apps, resulting in a faster app experience.Java yields a faster build process too, letting you code more in less time.Thanks to the accelerated assembly with gradle, assembling large projects becomes easier in Java.
As for the backend my best choice was firebase because it's very powerful and simple to use especially for beginners, it also contains a free usage tier with good performance. the application does its best to optimize the backend for both database usage and user experience. firebase provided all necessary backend functions which made making a professional application in time possible.
By using all the tools and libraries available and by using all knowledge of data structures and object-oriented programming we were able to make a well-designed application that could be used with no problems and didn't lack any important features, this implementation was definitely helpful to my career since implementing a concept into a real Application gives good knowledge of the development environment.

Final Conclusion

By the time the application was finished we can look back and tell how much we have learned from this project. so many things that should have been done differently in terms of implementation and design concept, so many additional features that an application like this has to have to be useful and competitive.
When the application was first being designed there were so many concerns that were not considered and by the time the testing happened and after all that time developing it we have realized that we learned so many things in terms of concept design and code optimizations and because of that we modified so many things since we started working on the implementation just to make the application more efficient and optimized. this proves that working on this project made us learn so many things about software development in general, not only that but we also learned how to make better conceptual design and learned how to turn an idea into a real application.
Building the application from the ground up was a great experience, the most important thing that we learned and after doing so much work is, in this type of software/product it is difficult to reach a final state so it is very important to deliver and get a first raw version and then fast and optimize and develop side features later. Some decisions were even based on this premise, having in mind that some choices are for the short-medium term rather than the long term are just faster to deliver that way, but finally we have ended up with a satisfying application that checks the main qualities of an application, these qualities being, good design, secure and efficient backend, and finally a clean and optimized coding design style.

Appendix A

User Manual

Appendices are provided to give supplementary information, which is included in the main text may serve as a distraction and cloud the central theme.

- Appendices should be numbered using alphabets, e.g. Appendix A, Appendix B, etc.
- Tables and References appearing in appendices should be numbered and referred to at appropriate places just as in the case of chapters.
- Appendices shall carry the title of the work reported and the same title shall be written in the contents page.