Usama Rao
161120

Sami Ahmed
161132

CarPool

Bachelor of Science in Computer Science

Supervisor: Shoaib Malik

Co-Supervisor: Imran Ihsan

Department of Computer Science

Air University, Islamabad

June 2020

Copyrights ©Usama Rao and Sami Ahmed, 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

List of Tables

Abstract

Generally, people find it very difficulty to schedules their rides because of the way they move from one place to another, and students suffer from this the most especially since the travelling among cities in private cab services is not that great. As students, we think there should exist more suitable travelling solutions to places where transportation networks are efficient and cost effective and helpful for students. This project aims to provide a platform for students to improve student mobility via carpooling allowing vehicle owning students to share their vehicle with non vehicle owning students. Ride sharing could also be one among the simplest solutions when there's no other mean of transportation to a particular location.

Mobile applications are playing a vital role in everyone's lives. Multiple ride sharing services with different features compete with existing transportation facilities. Some people prefer new transportation services over the traditional services like private cab services. Our application "CarPool" aims to promote carpooling by targeting only students, making it easier for them to stick and use this application. To place the ride sharing system in place, we have designed and developed an android application with backend servers for users to access the ride sharing service through their smartphones. Additionally, our 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 create the application simple but powerful as well, which makes this application very useful for the students.

Our app, CarPool, will be a unique ride sharing application that would take benefits of the advantages of carpooling and try to improve and eliminate the disadvantages while focusing on making it a good carpooling experience for students. In this report, we will go through the conceptual phase and the development phase since making developing a good application requires good planning first. We describe the software development architecture supported by web services. The framework uses the three-layer architecture of web development into mobile software development.

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 Javed, 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

1.1 Project Background/Review

Carpooling (also car-sharing, ride-sharing and lift-sharing) is sharing of car journeys so that more than one person travels in a car, and fulfills the need of others instead of driving to a location themselves. Drivers and passengers find a partner for journeys through multiple platforms available. After finding a match, they can communicate with each other to rearrange any details for the journey. Cost, meeting points and other details like space for luggage are agreed on. Then they meet and do their shared car journey as planned.

Ride sharing usually means to divide the travel expenses equally between all the occupants of the vehicle (driver or riders). Vehicle owner doesn't want to earn money, but to cut down their fuel bills with the occupants of vehicle. The expenses divided include the fuel bills and tolls taxes. Multiple platforms are available that provide the facility of carpooling. Usually there's a fare founded by the car driver and accepted by passengers sicne it is important they get an agreement before trip start.

By using the maximum capacity of a vehicle, ride sharing reduces each person's travel costs such as fuel costs, tolls taxes, and also the stress of driving. Carpooling is environmental friendly and better way to travel as it helps reduces pollution, carbon dioxide emission, traffc on the roads, and need of parking spaces. Ride sharing is a great way to fritter away the total spaciousness of a car, which might otherwise remain unused if it were just the one person travelling in the car.

Carpooling or ride sharing is more popular among the folks who work in same place or nearby places, and who board city area. However, carpooling is less likely among people those spend longer at work, elderly people, and homeowners.

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 some-

times to find ride for travelling to and from university. This project aims to provide a ride sharing application where non vehile owning students can share a ride with vehicle owning students.

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 easily and cheaply.

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. The application allows vehicle owning students to share rides with non-vehicle owning students and allows passengers to search for a ride while being secure.

This application will help students save money and also reduce the pollution of the environment. This application focuses on serving needs of students. CarPool is intended for the students of Air University and it supports android phones and tablets. Users will need internet connection to use the application to offer or find a common route to travel.

The application has a simple and easy interface. Users must register at first before using the application and after that they must choose between a driver or a rider. A driver can select riders while a rider 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.

Chapter 2

Literature Review

In 2009, carpooling represented 43.5% of all trips within the U.S and 10% of commute trips. In 2011, a corporation called Greenock created a campaign to encourage others to use this type of transportation so as to reduce the emission of carbon dioxide. Carpooling, or car sharing, is promoted by a national UK charity, carplus, whose mission is to market responsible car use so as to alleviate financial, environmental and social costs of motoring. This contributes to British government's initiative to scale back 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 story of a carpool with twelve people on board. Based in the city of Hyderabad, India, the book is on a real-life narration and highlights the potential benefits of a carpool.

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 are some examples of carpooling systems around the globe:

2.1 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

2.2 Mobile Applications

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

2.3 Carpooling Types

2.3.1 Regular

In this type of carpooling, the driver sits in the car in a closed space and (s)he is free to do what (s)he likes such as listen to the songs or call with headsets etc. In the United States, an intermediate concept has developed between carpooling and the public transport line, called the vanpool. These are minibuses chartered by an employer, a public authority or a private company and availabe to a group of people who regularly travel to the same route.

2.3.2 Occasional

This type of carpooling is especially used for freedom, peace, pleasure or last minute departures. The linking is usually done through websites or mobile applications, which may significantly reduce travel costs, but usually requires to carpool with one or more unknown persons. This type of carpooling is mainly used for freedom or last minute departures. The linking is commonly done through websites or mobile applications, which might significantly reduce travel costs, but usually requires to carpool with one or more unknown persons.

2.3.3 Eventual

Participants in an event (music festival, sporting event, wedding, associative or institutional meeting) can organize ride sharing to the venue of the event. This one-time carpool includes a special feature: all participants commute on the same route and to the same place on the same date. The type of carpooling is also used for departures on holidays or weekends.

There are also "cultural" carpooling platforms which provide facility to go to a cultural site: castles, museums, exhibitions, religious places, festivals, etc.

2.4 Advantages and Disadvantages of Carpooling Applications

2.4.1 Advantages of carpooling

- Cost saving: A ride may be twice as cheap than traveling in own vehicle or in a private cab.
- A fewer number of cars on the road can reduce the emission of carbon dioxide and make the air cleaner.
- Saving time: Fewer cars - fewer traffic jams. It becomes possible to reach at destination faster and solve the issue to find a parking place.
- Meeting new people. Traveling together allows you to find out good friends.

2.4.2 Disadvantages of carpooling

- Indecent riders: Some people can attemp to discount the price or even ask the driver to pickup or to go to the place that's not on the scheduled route. And it is important to point out to riders at the very beginning of trip.
- Indecent drivers: Unfortunately, drivers may overload the car and ask for a high price.
- In some cases, a driver has to pick up the each passenger separately. It increases the time of traveling.
- Passengers may be not satisfied with the driving. In turn, the drivers are often annoyed with an excessive volubility of companions, their untidiness or lack of manners.

Chapter 3

Requirement Specifications

Our purposed system is a "Carpool" application wic is a ride saring application designed just for students. Students can login or signup to tis application only via university email. We made sure tat only enrolled students in a university use tis application. Te application involves functional and non-functional functionalities tat must be performed by te system.

3.0.1 Functional Requirements

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

Table 3.1: Functional Requirement - 01

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

Table 3.2: Functional Requirement - 02

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

Table 3.3: Functional Requirement - 03

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

Table 3.4: Functional Requirement - 04

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

Table 3.5: Functional Requirement - 05

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

Table 3.6: Functional Requirement - 06

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

Table 3.7: Functional Requirement - 07

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

Table 3.8: Functional Requirement - 08

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

Table 3.9: Functional Requirement - 09

3.0.2 Non-Functional Requirements

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

Table 3.10: Non-Functional Requirement - 01

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

Activate

Table 3.11: Non-Functional Requirement - 02

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

Table 3.12: Non-Functional Requirement - 03

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

Table 3.13: Non-Functional Requirement - 04

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

Table 3.14: Non-Functional Requirement - 05

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

Activate

Table 3.15: Non-Functional Requirement - 06

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

Table 3.16: Non-Functional Requirement - 07

3.1 Use Cases

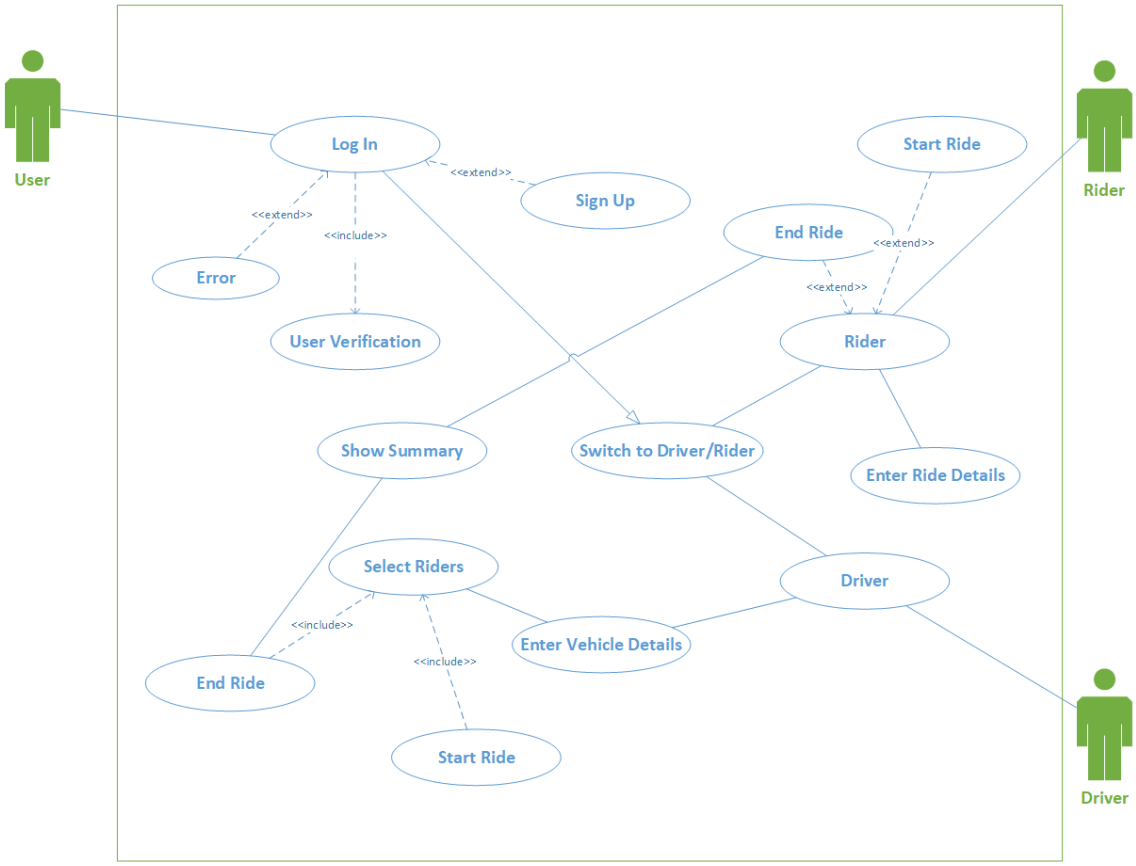


Figure 3.1: Use Case Diagram

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

Table 3.17: Use Case - 01

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

Table 3.18: Use Case - 02

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

Table 3.19: Use Case - 03

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

Table 3.20: Use Case - 04

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

Table 3.21: Use Case - 05

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

Table 3.22: Use Case - 06

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

Table 3.23: Use Case - 07

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

Table 3.24: Use Case - 08

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

Table 3.25: Use Case - 09

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

Table 3.26: Use Case - 10

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

Table 3.27: Use Case - 11

Chapter 4

Design

4.1 System Architecture

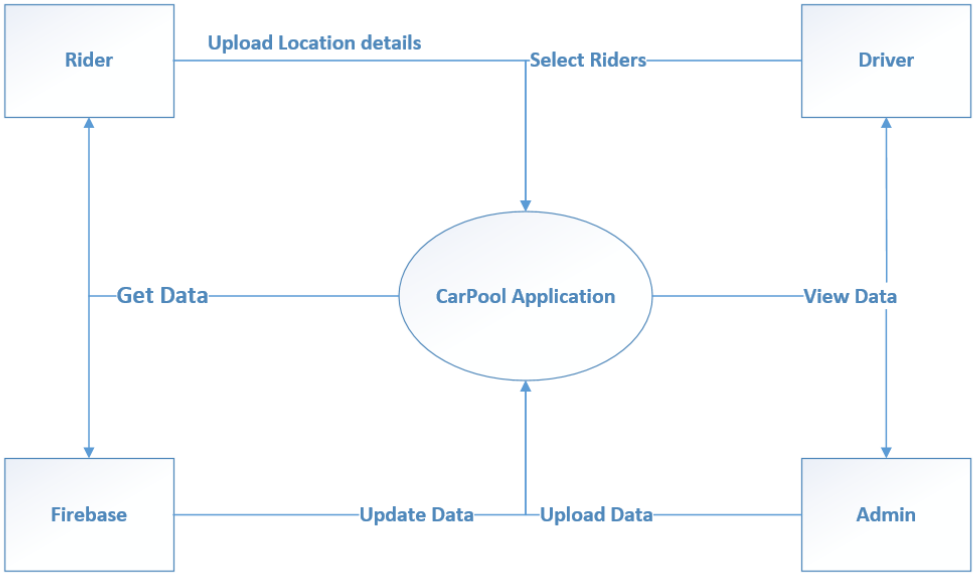


Figure 4.1: Context Diagram

4.2 Design Constraints

In our application, we added 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 without hassle. Due to the COVID-19 pandemic, we could not be supervised and guided properly. Hence this process was not added and executed.

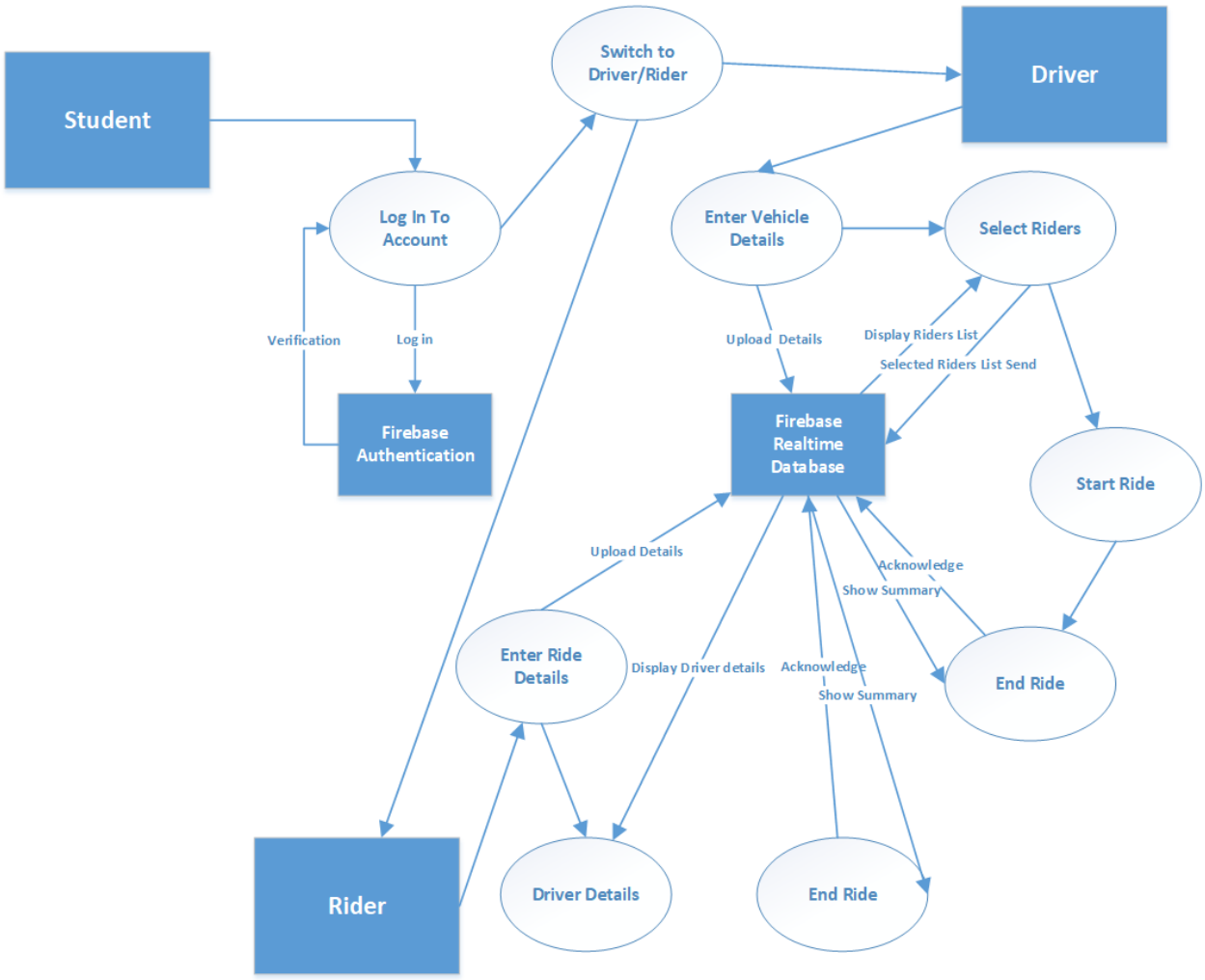


Figure 4.2: Sub System

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 needs collaboration and planning, hence require a transparent and concise way to communicate amongst them.
- A lot of time is saved down the road when team is able to visualize the processes, user interactions and static structure of the system. These project designs will try to make the general idea of the project clearly understandable by identifying actors and functional / non-functional requirements, and also all the diagrams provide a transparent view about this project.

4.4 High-Level Design

4.4.1 Package Diagram

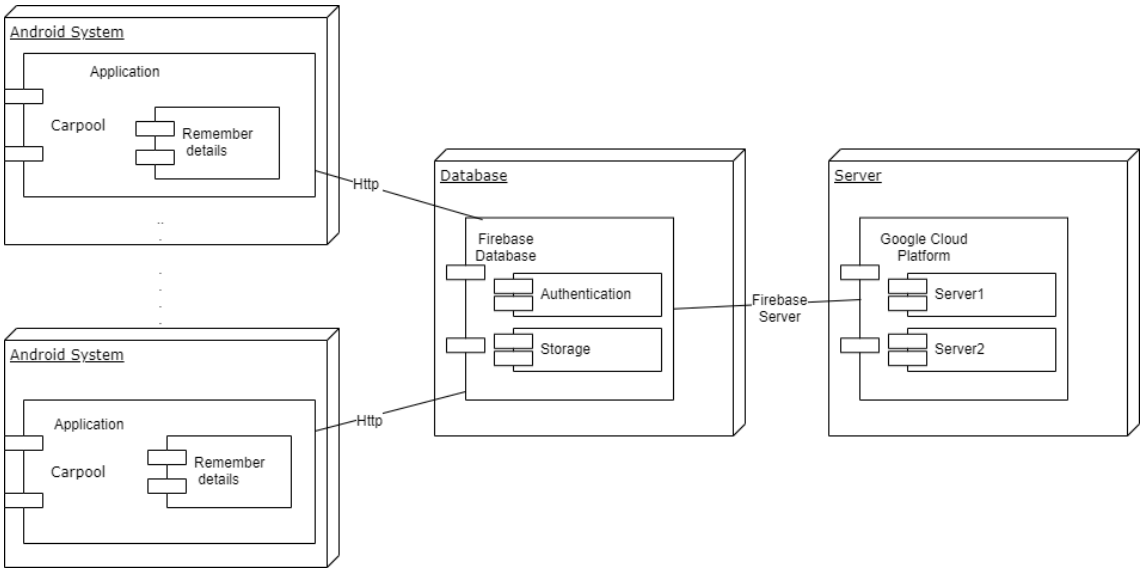


Figure 4.3: Package Diagram

4.4.2 Interaction Diagram

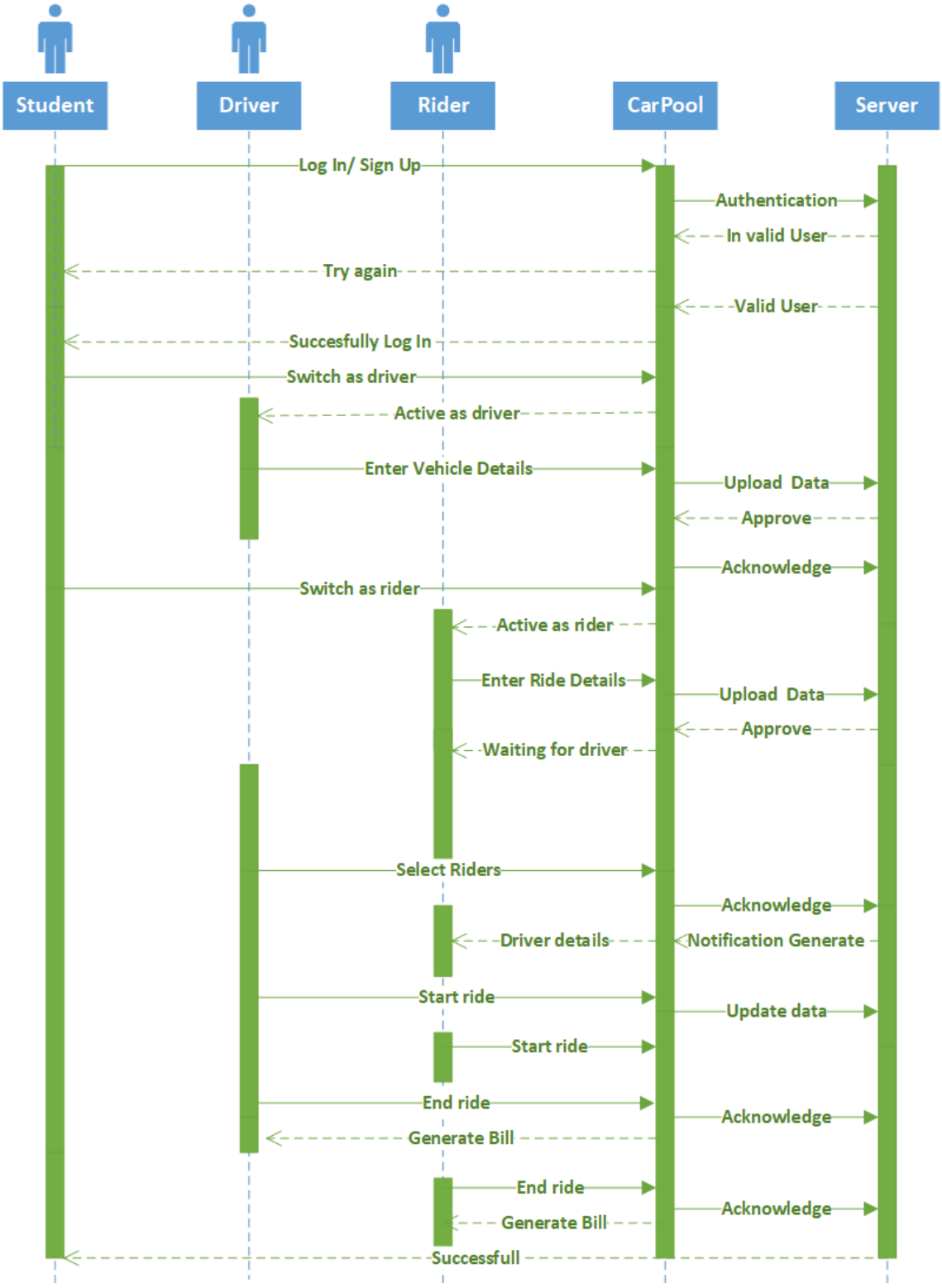


Figure 4.4: Process Interaction Diagram

4.4.3 Deployment Diagram

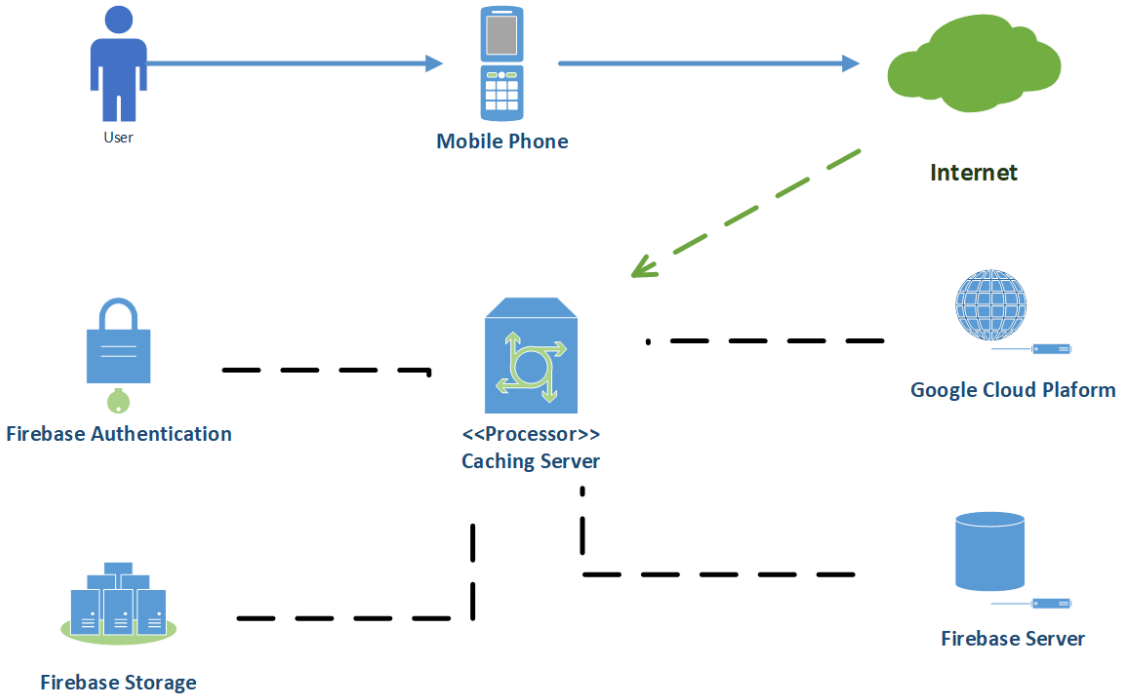


Figure 4.5: Deployment Diagram

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 enter his/her 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 is used to avoid malpractice as this module is there to verify student details.

4.4.4.2 Driver

In this module, the user acts as a driver. This module consists 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 can use passenger list fragment to see the list of passengers. Driver module also contains the activities to enter vehicle details and see summary of the ride.

4.4.4.3 Rider

In this module, the user acts as a rider. This module consists 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 can use driver detail fragment to see the detail of driver. Rider module also contains the activities to enter trip detail and see summary of the ride.

4.4.5 Security

This application does not disclose any personal information of any user and does not collect any sensitive information from it's own users.

4.5 Low Level Design

4.5.1 Database Design

Firebase platform provides a cloud-hosted real time 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

Figure 4.6: Low Level Design

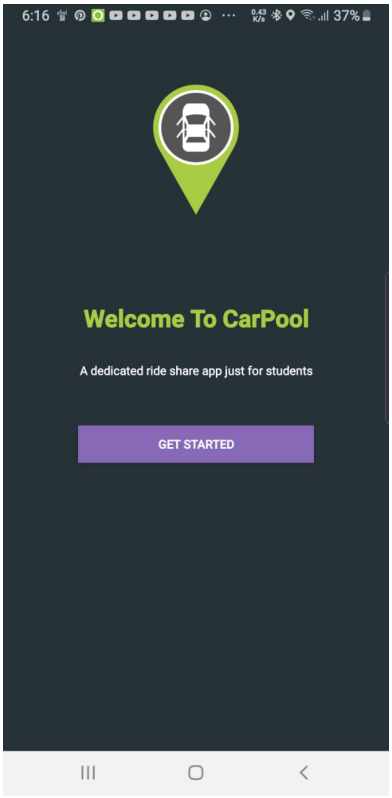
and documents. There is no relationship between classes, so there won't be any schema of our database. Data gets stored as JSON Objects. it is more like a cloud-hosted JSON tree. When a person adds data to the JSON tree, it becomes a node in the existing JSON structure with an associated key. The best 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.

4.5.2 GUI Design

Our application contains more than fifteen screens. Different screens based on user type are:

4.5.2.1 Welcome Interface

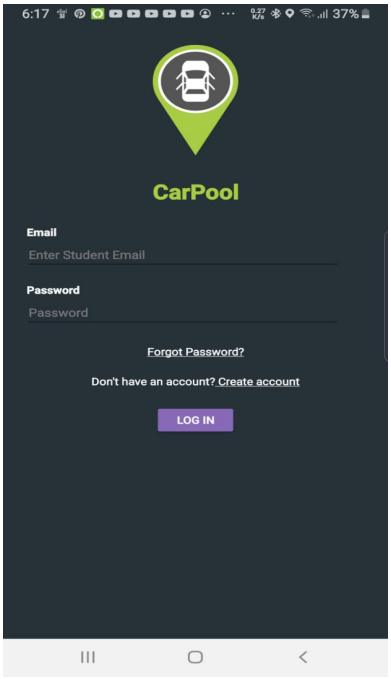
This is the interface that shows up to the user when he opens the application. This interface is composed of the application's name, a logo and a slang which describe the application alongside a "Get Started" button which directs the user to login screen.



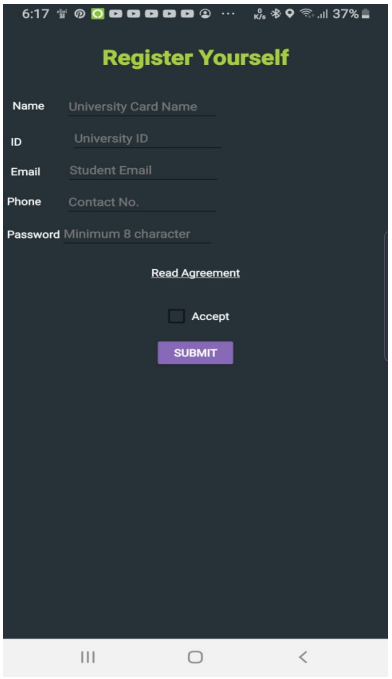
(a) Welcome Screen

4.5.2.2 Authentication Interfaces

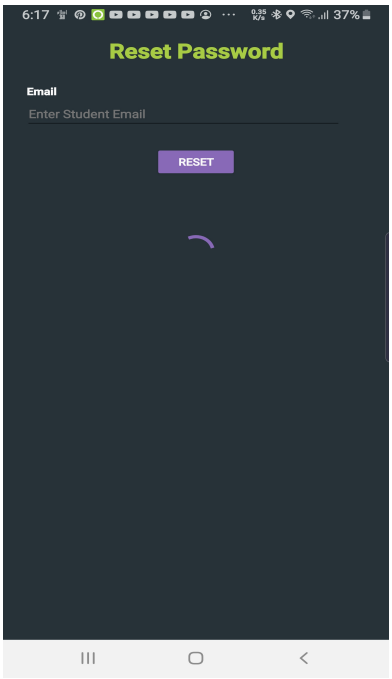
The authentication interface is made as simple as possible while also being very pretty and professional. It utilizes Firebase auth as a backend which allows for a secure connection to the database plus encrypted passwords for extra security. This interface also contains options to recover password in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



(a) Login Screen



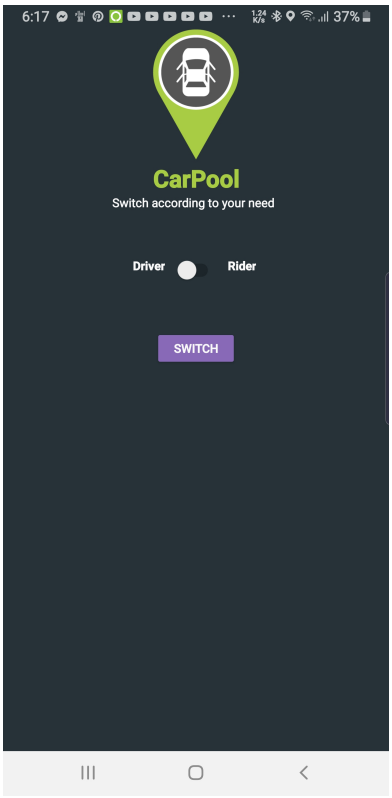
(b) Signup Screen



(c) Forgot Password Screen

4.5.2.3 Type Switching Interface

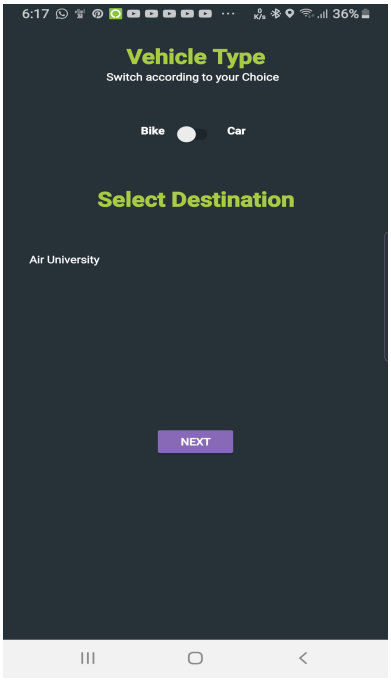
This is the interface that shows up to the user when he's logged in into the application. This interface is composed of the application's name and a logo alongside a switch button, which lets the user select his/her user type.



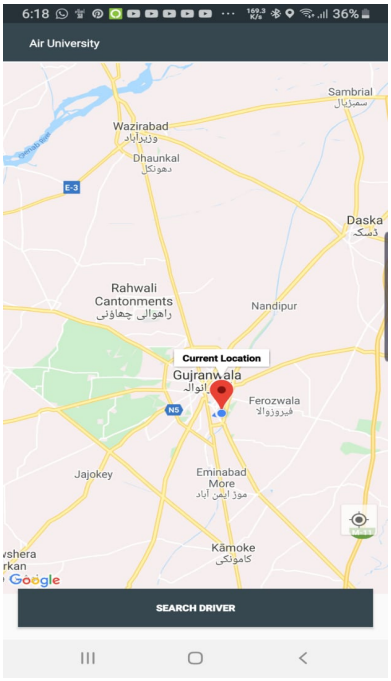
(a) Select User Type Screen

4.5.2.4 Rider Interfaces

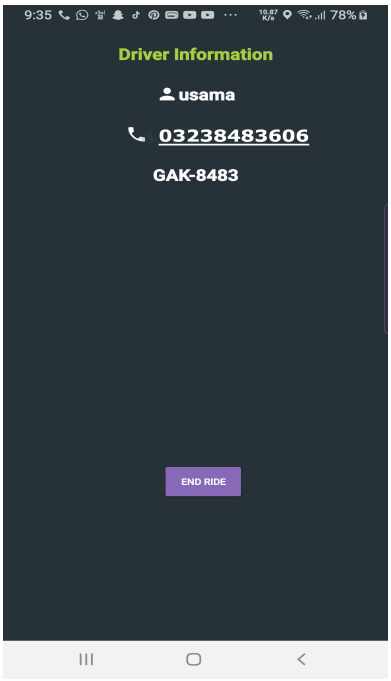
These interfaces are used by the passengers to post ride details for drivers to see and select riders. These interfaces are extremely customizable and professional and are created by using Google Places API, Google Direction API, Google Map API. We have created a very organized and efficient algorithm 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.



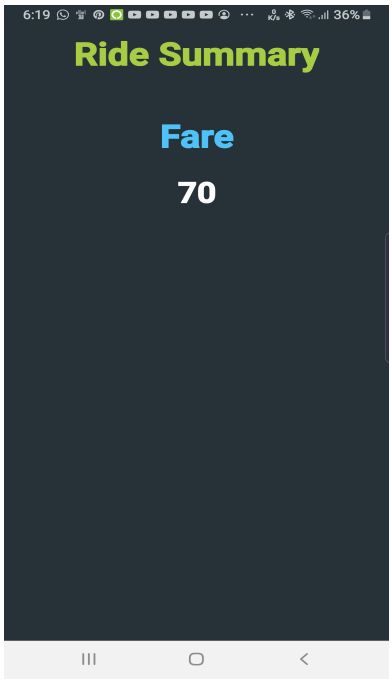
(a) Enter Ride Detail Screen



(b) Map Screen



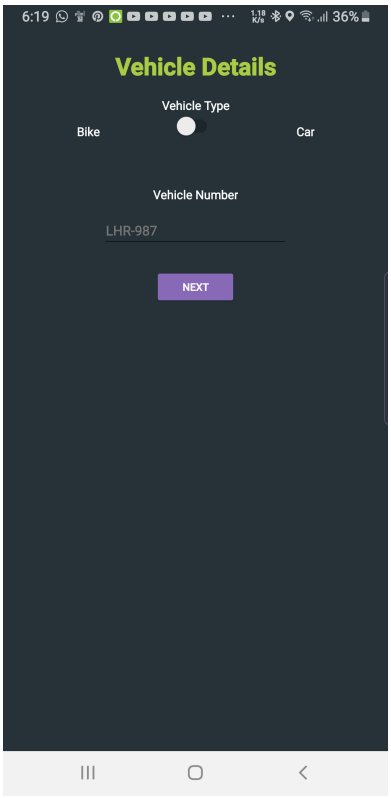
(c) Driver Detail Screen



(d) Ride Summary Screen

4.5.2.5 Driver Interfaces

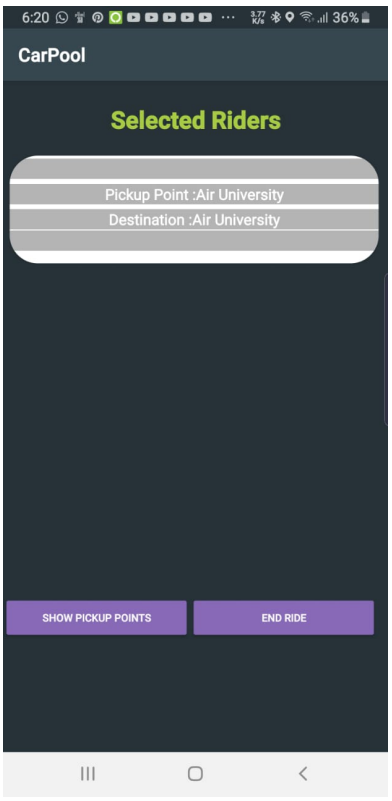
These interfaces are used by rider to enter vehicle detail, and see and select the passengers for trip. These form are extremely customizable and professional and are created by using Google Places API. In this interface, we have created a very organized and efficient algorithm to optimize the results.



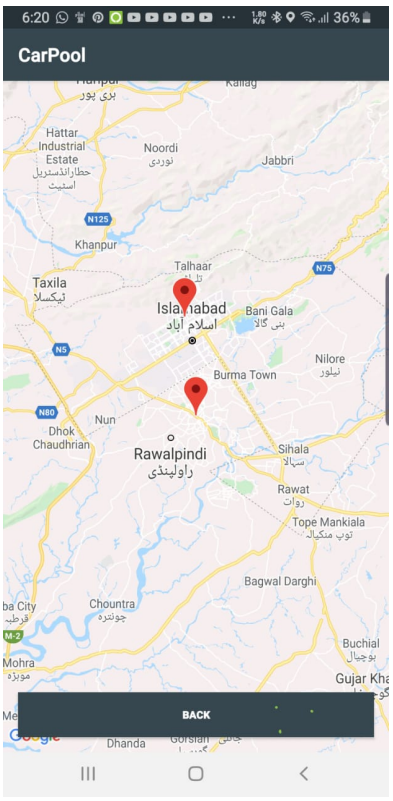
(a) Enter Vehicle Detail Screen

(b) Driver Map Screen

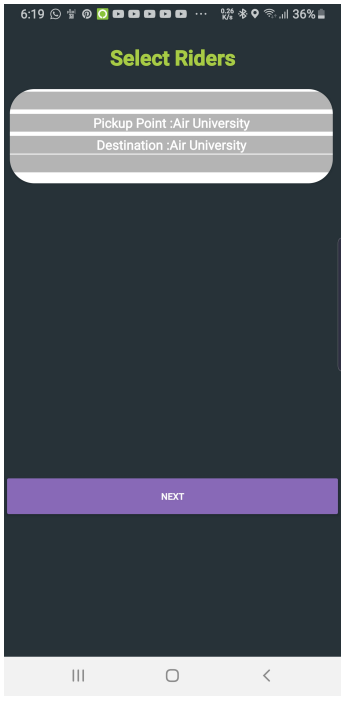
The algorithm uses a combination of Geolocation and Places API to find the full address, city and sub city.



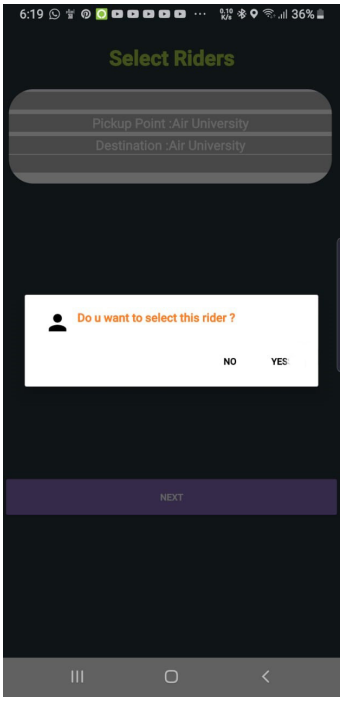
(a) Selected Passengers Screen



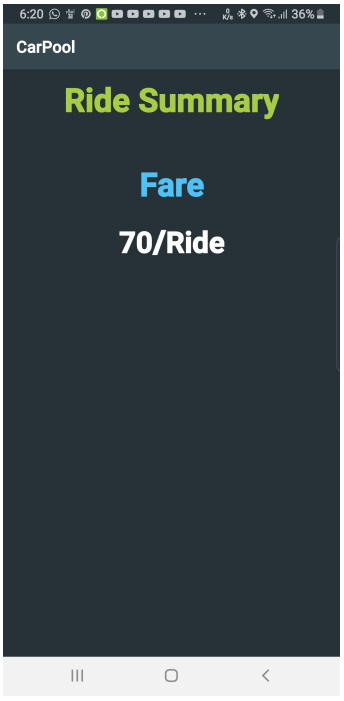
(b) Passengers Pickup Points Screen



(c) Rider List Screen



(d) Popup Selection Confirmation Screen



(e) Driver Summary Screen

Chapter 5

System Implementation

5.1 System Architecture

Application architecture is a collection of well defined technologies and models for the development of fully-structured mobile programs that supported industry and vendor-specific standards. As we develop the architecture of our application, we also consider programs that execute on wireless devices such as smartphones and tablets.

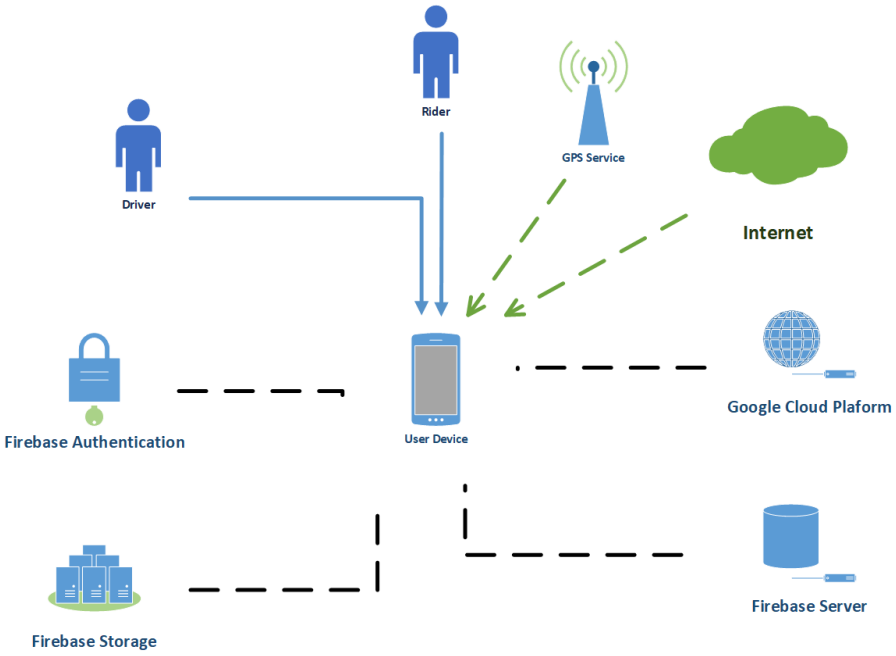


Figure 5.1: System Architecture

5.2 Work Environment

5.2.1 Hardware platforms

For the development of this, application we have used an Hp Elitebook workstation 8460w. It is fairly powerful and could handle the usage of multiple emulators at once. The specifications of the machine are as follow:

- **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 Samsung Galaxy Note 8 and 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.

- **Layout Design** : XML
- **Database Language** : NoSQL

5.2.5 Frameworks used:

- **Firebase Auth** : This framework 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 application.
- **Firebase Messaging** : This framework makes real time messaging easy and possible using firebase. It is also used for push notifications which is used for messaging function between users in our application.
- **Google Maps API** : This API enables the application to show places on the map like the origin and destination and the way between them. It is used in the map to trace the destination for users.
- **Google Places API** : This API is used to give information about places in the application and used to AutoComplete search queries. It is used to auto complete search queries for our users for easier searching. It is also used to optimize our search function.
- **Material Design Library** : This library provides components such as material buttons, material EditText etc.
- **RuntimePermission** : This library helps to manage android permissions with no problems.
- **DxLoadingButton** : This library provides a loading button used for authentication.
- **Spinner** : Spinner is a styleable drop down menu for android using the old spinner style.
- **Tooltip** : Tooltip is a simple to use customizable library based on popup window.
- **EasyValidation** : EasyValidation is a text and input validation library in Java for android. It is used to validate user input information.
- **Retrofit** : Retrofit is 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**

  Its a NoSQL database. 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 users such as name, email, number etc. We have also used it to store trip and trip request objects.

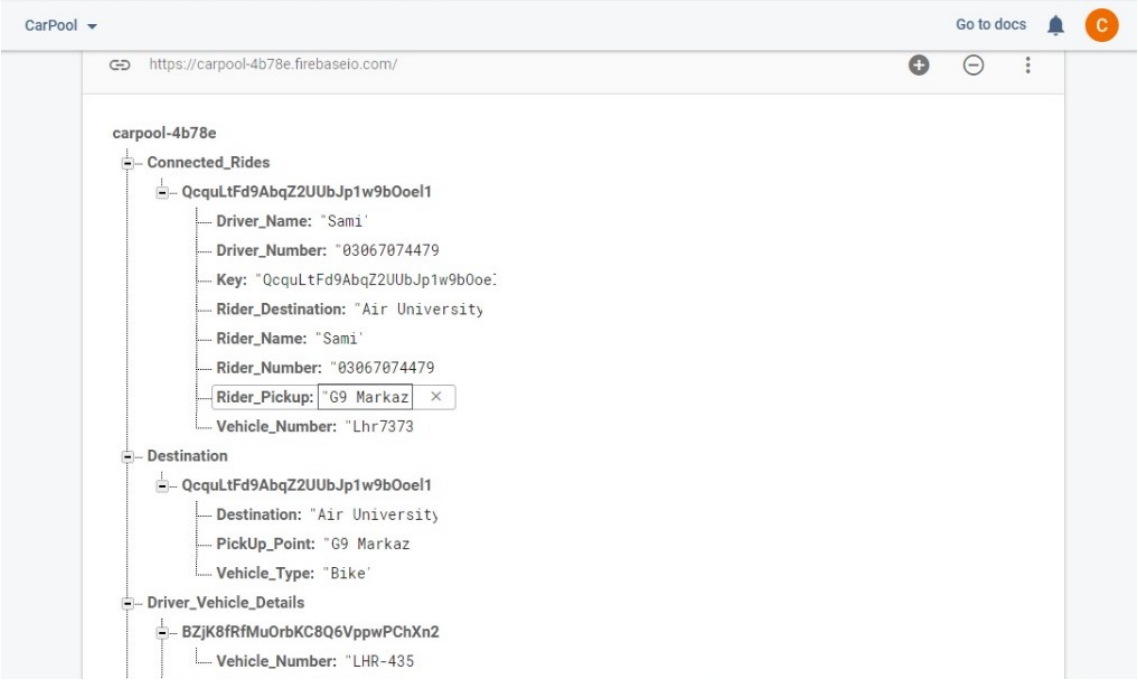


Figure 5.2: Example of a user mode in database

- **Real-time Database Usage**

  For the usage of this database, we made a separate database class. This class contains all database related functions with a listener interface. 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. 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 changes and is good for things like messages and real time updates.

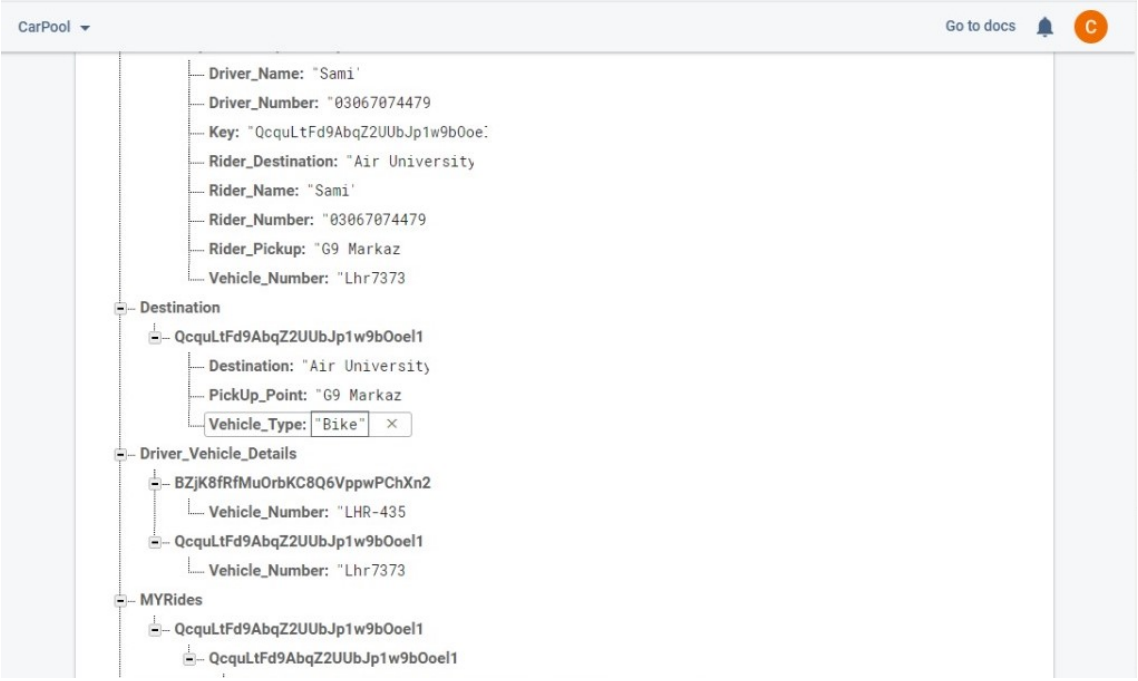


Figure 5.3: Example of a user mode in database

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 if less data is fetched, the better it is in terms of cost. For this, we have avoided using **addValueEventListener** and 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 is 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/her email to complete the registration.

- **Authentication Usage**

  When a user registers, his basic information (Name, Email, Password) are stored 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.

5.7 Rules

Some of the restriction and rules are mentioned below :

- Users can use 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 has to select pick up and drop off points.
- Driver and rider cannot use the app until their accounts are verified.

Chapter 6

System Testing and Evaluation

6.1 Graphical 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

Table 6.1: Graphical User Interface Testing

6.1.1 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.	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

Table 6.2: Usability Testing

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

6.2 Software Performance 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

Table 6.3: Software Performance Testing

6.3 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

Table 6.4: Compatibility Testing

6.4 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

Table 6.5: Exception Handling

6.5 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

Table 6.6: Security Testing

6.6 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

Table 6.7: Installation Testing

Chapter 7

Conclusions and Future Work

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 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 sized application(4mb) 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 available and Java has a wide open-source ecosystem. Java apps are lighter 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 due to which assembling large projects becomes easier in Java.

As for the backend is concerned, our 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 helped making a professional application on time.

By using all the tools and libraries available and all knowledge of data structures and object-oriented programming, we were able to make a well-designed application that could be used without bugs and didn't lack any important features. This implementation was definitely helpful to our career, since implementing a concept into a real application gives us good knowledge of the development environment.

Now that the application has been developed, we can look back and tell how much we have learned from this project. So many things could have been done differently, in terms of implementation and design concept also. Many additional features that an application like this could have to be more useful and competitive.

When the application was first being designed there were so many concerns that were not considered. By the time of developing and testing the application, we have realized so many things in terms of design and code optimization. Because of this, we modified so many things since we started working on the implementation just to make sure that application is more efficient and optimized. This proves that working on this project made us learn so many things about software development in general. Not only this, we also learned how to make better conceptual design and learned how to turn an idea into a real application.

There are some ways in which this application can be improved further. Some of the features for the application which are not yet present, but could be implemented in future development.

Our group has primarily been focusing on the functionality and simplicity of this application. If time allows, we we'll try to make the GUI more colourful and elegant, but there'll always be room for betterment in this segment.

Bibliography

[1] Wikipedia
https://en.wikipedia.org/wiki/Carpool/

[2] Google Maps documentation
https://developers.google.com/maps/documentation~uno/abcde.html

[3] Firebase Database
https://console.firebase.google.com/

[4] Android API documentation
https://developer.android.com/reference

Appendix A

Application Workflow

The user downloads and installs the application in an android device. The user opens the application, then he is welcomed with a Login/Register. New user registers via the registration form (name, university email, university registration id, contact no, password). The user must have to confirm the registration via email.

After confirmation, the user enters the application and chooses between driver/rider. If the user is a driver, he can enter vehicle details and selects riders from list of passengers.

Riders can enter trip details and request a ride. After selected by driver, it displays the details of driver on rider's screen.

After completion of ride, both driver and rider can see the summary of ride.

Appendix B

Interviews

The interviewees were three students from different departments of Air University.

(Note : Q&A with no recording, as we believe recording would make interviewees uncomfortable.)

Student 1
Name : Hassan Siddiqui
Age : 21
Gender : Male
Interview date : 10.02.2020
Duration of interview : 10 minutes
Place of interview : AUSOM building
Notes from interview : The interviewee owns a Samsng Galaxy S9 smart phone which runs android 9.0. He is not familiar with concept of programming for mobile devices but he believes that, CarPool is a very interesting concept. Hassan believes that our app is very handy and he would like to install the prototype of application, on his smart phone and become one of our test user. The interviewee would like easier access to travel to and from university. He believes that, it's not easy to find out local transportation. He also mentioned that he would be happier, that he can share his ride expense and cut his fuel bills.

Student 2
Name : Ahmed Shahid
Age : 29
Gender : Male
Interview date : 10.02.2020
Duration of interview : 8 minutes
Place of interview : Department of Mechatronics
Notes from interview : The interviewee owns a IPhone 7 smart phone which runs IOS. He is not familiar with concept of programming for mobile devices but he believes that, it is a very interesting concept. Ahmed believes our app is very useful and he wishes a similar application for iPhone. The interviewee would like to use this application through his Iphone. He would like to share ride with non-vehicle

owning students.

Student 3
Nickname : Amna Asim
Age : 19
Gender : Female
Interview date : 10.02.2020
Place of interview : Department of Electrical Engineering
Duration of interview : 5 minutes
Notes from interview : The interviewee owns a Oppo K1 smart phone which runs Android 8.1. She is not familiar with concept of programming for mobile devices and she believes that, it is a very interesting concept and finally someone is doing this for students. She complained a lot about the local transportation in Islamabad, which frustrate her every time, when she wants to travel to and from university. She also believes that it is a good idea to share ride with non-vehicle owning students. She also mentioned to include the feature which facilitates female students to share ride with only female students.

Appendix C

Steeple Analysis

Social

- Increase social interaction and solidarity.
- Meet new people during rides and make new friends.
- Driving with people is better than driving alone since it involves less stress.

Technology

- Smartphone penetration is increasing day after day.
- Use of technology to create matches between drivers and passengers.
- The application is accessible from anywhere using a smartphone real time communication between actors.

Environmental

- Increase of high occupancy vehicles which will lead to a decrease of CO2 emissions.
- Less cars on the roads that leads to safer roads and fluent traffic.

Economical

- Savings as the price of gas and highways is shared among the travelers in a context of an increasing gas price.

Political

- Increase of support for initiatives that decreases greenhouse gas emission support from government.

Legal

- Insurances of drivers and passengers: In case of accidents, if the owner of the car has insurance, it covers any medical expense.
- Ride sharing serves an important role in enhancing mobility in low-income, hostel students who are more likely to be unable to afford personal automobiles.

Ethical

- Client confidentiality should be kept: all information related to trips' history should only be communicated to their respective use.