**Carpool Application**



**Submitted By**

Nazar Muhammad

(2018-2022)

**Department of Computer Science**

**Khwaja Fareed University of Engineering & Information Technology**

**Rahim Yar Khan**

**2022**

**Carpool Application**

**Submitted to**

**Mr. Shadab Alam Hashmi**

**Department of Computer Science**

**In partial fulfilment of the requirements**

**For the degree of**

**Bachelor of Science in Computer Science**

**By**

Nazar Muhammad

Cs181060

**Khwaja Fareed University of Engineering & Information Technology**

**Rahim Yar Khan**

**2022**

**DECLARATION**

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at Khwaja Fareed University of engineering & Information Technology or other institutions.

|  |  |
| --- | --- |
| RegNo : | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Name : | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Signature : | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Date : | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**APPROVAL FOR SUBMISSION**

I certify that this project report entitled **Carpool Application** was prepared by **Nazar Muhammad** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science in Computer Science at Khwaja Fareed University of Engineering & Information Technology.

Approved by:

**Signature :** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Supervisor :** Mr. Shadab Alam Hashmi

**Date :** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MEETING LOG**

**ACKNOWLEDGEMENT**

I would like to thank everyone who had contributed to this project. I would like to express my gratitude to my Project supervisor, **Mr. Shadab Alam Hashmi** for his invaluable advice, guidance, and his enormous patience throughout the development of the project.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement.

**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.

TABLE OF CONTENTS

[Chapter 1 1](#_Toc103958184)

[INTRODUCTION 1](#_Toc103958185)

[1.1. What is Carpool? 1](#_Toc103958186)

[1.2. Problem Statement 2](#_Toc103958187)

[1.3. Project Objective 2](#_Toc103958188)

[1.3.1. Objective 2](#_Toc103958189)

[1.3.2. Goals 2](#_Toc103958190)

[1.4. Project Scope 2](#_Toc103958191)

[1.5. Chapter Summary 3](#_Toc103958192)

[Chapter 2 4](#_Toc103958193)

[EXISTING SYSTEMS 4](#_Toc103958194)

[2.1. Existing Systems 4](#_Toc103958195)

[2.2. Examples of Existing Systems 4](#_Toc103958196)

[2.2.1. Website 4](#_Toc103958197)

[2.2.2. Mobile Applications 4](#_Toc103958198)

[2.3. Chapter Summary 4](#_Toc103958199)

[Chapter 3 6](#_Toc103958200)

[REQUIREMENT ENGINEERING 6](#_Toc103958201)

[3.1. Proposed System 6](#_Toc103958202)

[3.2. Understanding the System 6](#_Toc103958203)

[3.2.1. User Involvement 6](#_Toc103958204)

[3.2.2. Stakeholders 6](#_Toc103958205)

[3.2.3. Domain 6](#_Toc103958206)

[3.3. Requirement Specifications 6](#_Toc103958207)

[3.3.1. Functional Requirements 7](#_Toc103958208)

[3.3.2. Non-Functional Requirements 9](#_Toc103958209)

[3.3.3. Requirements Baseline 11](#_Toc103958210)

[3.3.4. Need to Feature Mapping 11](#_Toc103958211)

[3.4. Gantt Chart 11](#_Toc103958212)

[3.5. Hurdles in Optimizing the Current System 11](#_Toc103958213)

[3.6. Chapter Summary 11](#_Toc103958214)

[Chapter 4 12](#_Toc103958215)

[DESIGN 12](#_Toc103958216)

[4.1. Software Process Model 12](#_Toc103958217)

[4.1.1. Benefits of Selected Model 12](#_Toc103958218)

[4.1.2. Limitations of Selected Model 12](#_Toc103958219)

[4.2. Design 12](#_Toc103958220)

[4.2.1. Methodology of the Proposed System 12](#_Toc103958221)

[4.2.2. Entity Relationship Diagram 12](#_Toc103958222)

[4.2.3. UML Diagrams 12](#_Toc103958223)

[4.2.3.1. Use Case Diagram of the System 12](#_Toc103958224)

[4.2.3.2. Class Diagram of the System 12](#_Toc103958225)

[4.2.3.3. Activity Diagram of the System 12](#_Toc103958226)

[4.2.3.4. Sequence Diagram of the System 12](#_Toc103958227)

[4.2.3.5. Component Diagram of the System 12](#_Toc103958228)

[4.3. Chapter Summary 12](#_Toc103958229)

[Chapter 5 13](#_Toc103958230)

[DATABASE 13](#_Toc103958231)

[5.1. Database Introduction 13](#_Toc103958232)

[5.2. Selected Database 13](#_Toc103958233)

[5.2.1. Reasons for Selection of the Database 13](#_Toc103958234)

[5.2.2. Benefits of the Selected Database 13](#_Toc103958235)

[5.2.3. Limitations of the Selected Database 13](#_Toc103958236)

[5.3. Database Queries 13](#_Toc103958237)

[5.4. Database Tables 13](#_Toc103958238)

[5.5. Database Schema Diagram 13](#_Toc103958239)

[5.6. Chapter Summary 13](#_Toc103958240)

[Chapter 6 14](#_Toc103958241)

[DEVELOPMENT AND IMPLEMENTATION 14](#_Toc103958242)

[6.1. Development of the Computer Program 14](#_Toc103958243)

[6.2. Implementation Strategy 14](#_Toc103958244)

[6.3. Tools Selection 14](#_Toc103958245)

[6.3.1. Hardware 14](#_Toc103958246)

[6.3.2. Software 14](#_Toc103958247)

[6.4. Coding 14](#_Toc103958248)

[6.5. User Interface 14](#_Toc103958249)

[6.5.1. Description 14](#_Toc103958250)

[6.5.2. Interface Screenshots 14](#_Toc103958251)

[6.6. Program Deployment 14](#_Toc103958252)

[6.7. Chapter Summary 14](#_Toc103958253)

[Chapter 7 15](#_Toc103958254)

[TESTING 15](#_Toc103958255)

[7.1. Introduction 15](#_Toc103958256)

[7.2. Testing Methods 15](#_Toc103958257)

[7.3. Comparison 15](#_Toc103958258)

[7.4. Software Evaluation 15](#_Toc103958259)

[7.4.1. Testing Strategy 15](#_Toc103958260)

[7.4.2. Test Plans 15](#_Toc103958261)

[7.4.3. Test Cases 15](#_Toc103958262)

[7.4.4. Test Report 15](#_Toc103958263)

[7.5. Chapter Summary 15](#_Toc103958264)

[REFERENCES 16](#_Toc103958265)

[APPENDIX (Optional) 17](#_Toc103958266)

LIST OF FIGURES

[Figure ‎1.1 Figure Caption 2](#_Toc80019892)

LIST OF TABLES

[Table ‎1.1 Table Caption 2](#_Toc80019893)

[Table ‎3.1 User Needs of System 5](#_Toc80019894)

[Table ‎3.2 Functional Requirement 01 5](#_Toc80019895)

# 

## INTRODUCTION

### What is Carpool?

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.

### Problem Statement

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.

### Project Objective

#### 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.

### 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 KFUEIT 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 to.

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.

### Chapter Summary

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 mock-up 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 Database and displaying that location on the map using Google Map API.

# 

## EXISTING SYSTEMS

### Existing Systems

Many carpooling 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.

### Examples of Existing Systems

#### Website

* 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](http://www.letsride.in)

#### Mobile Applications

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

### Chapter Summary

This project aims to develop an Android based application for carpooling for students, this application allows nonprofessional drivers 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 elders that may have disabilities and illnesses.

# 

## REQUIREMENT ENGINEERING

### 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.

### Understanding the System

#### User Involvement

* Create an account
* Login
* Registration new user

#### Stakeholders

* Students
* Business analyst
* Driver
* Passengers
* Web developer

#### Domain

* Transport

### Requirement Specifications

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

#### Functional Requirements

Table 3.1: Functional Requirement 01

|  |  |
| --- | --- |
| Identifier | FR-01 |
| Title | Create Account |
| Requirement | Registered New User |
| 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 3.2: Functional Requirement 02

|  |  |
| --- | --- |
| Identifier | FR-02 |
| Title | Sign In |
| Requirement | Already registered |
| 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 3.3: Functional Requirement 03

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

Table 3.4: Functional Requirement 04

|  |  |
| --- | --- |
| Identifier | FR-04 |
| Title | Switch to Driver / Rider |
| Requirement | Sign In |
| 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 3.5: Functional Requirement 05

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

Table 3.6: Functional Requirement 06

|  |  |
| --- | --- |
| Identifier | FR-06 |
| Title | Create Ride (Driver) |
| Requirement | Choose role of a driver |
| 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 3.7: Functional Requirement 07

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

Table 3.8: Functional Requirement 08

|  |  |
| --- | --- |
| Identifier | FR-08 |
| Title | Book Ride (Rider) |
| Requirement | Choose role of a rider. |
| 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 3.9: Functional Requirement 09

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

#### Non-Functional Requirements

Table 3.10: Non-Functional Requirement 01

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

Table 3.11: Non-Functional Requirement 02

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

Table 3.12: Non-Functional Requirement 03

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

Table 3.13: Non-Functional Requirement 04

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

Table 3.14: Non-Functional Requirement 05

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

Table 3.15: Non-Functional Requirement 06

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

### Gantt Chart

# 

Figure 3.1: Gantt Chart

### Chapter Summary

In this era, number of students in universities increasing drastically, this will also rise problem of their transportation from university to their home or vice versa. To conquer this problem purposed solution helps to utilize private vehicles to facilitate others at very economical way. It is a very practical solution for current many problems. Functional and non-functional requirement along other important components discussed to make sure that this system can be implemented in real world.

# 

## DESIGN

### Software Process Model

A software process model is an abstraction of the actual process, which is being described. It can also be defined as a simplified representation of a software process. There are many development life cycle models that have been developed in order to achieve different required objectives. So in this project we are going to use V-model. The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.

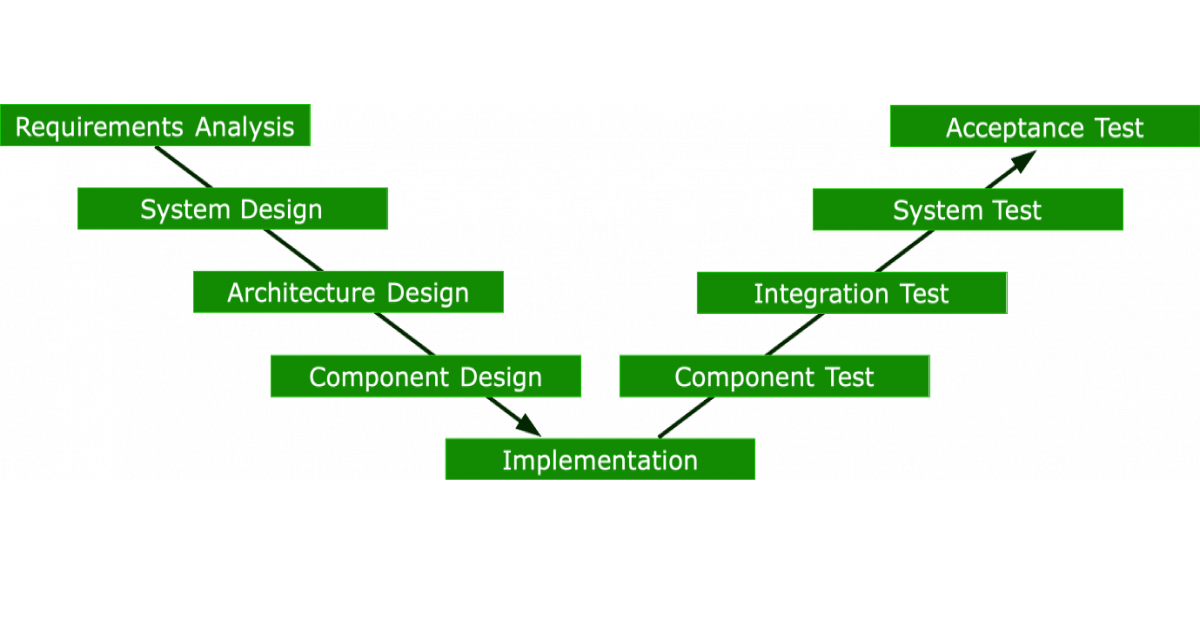


Figure 4.1: V-Model

#### Benefits of Selected Model

* + **Uncomplicated Use**: The framework of the V model is highly uncomplicated. The environment is quite user-friendly and provides a solid base for or software development for small-scale developers.
  + **Time saver:** V model is simple to use, and trial activities such as designing, planning, and development take place before coding the software.
  + **Straightforward design:** The design of the V Model is quite simple. The developers of the V model work to make this model straightforward to understand; hence they have made the architecture simple yet highly efficient.

#### Limitations of Selected Model

* **Highly rigid:** The V Model is highly rigid. It is formed for the ease of development of highly complicated software.
* **Limited flexibility:** The V model is not very flexible. It isn’t elementary and takes time for new developers to get used to it.
* **Risky:** The management of the total environment is volatile and risky. It is not suitable for use to build object-oriented software due to its uncertainties in the design.

### Design

#### Methodology of the Proposed System

The design of a project is important for the structure of the application by using UML (Unified Modelling 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.

#### Entity Relationship Diagram

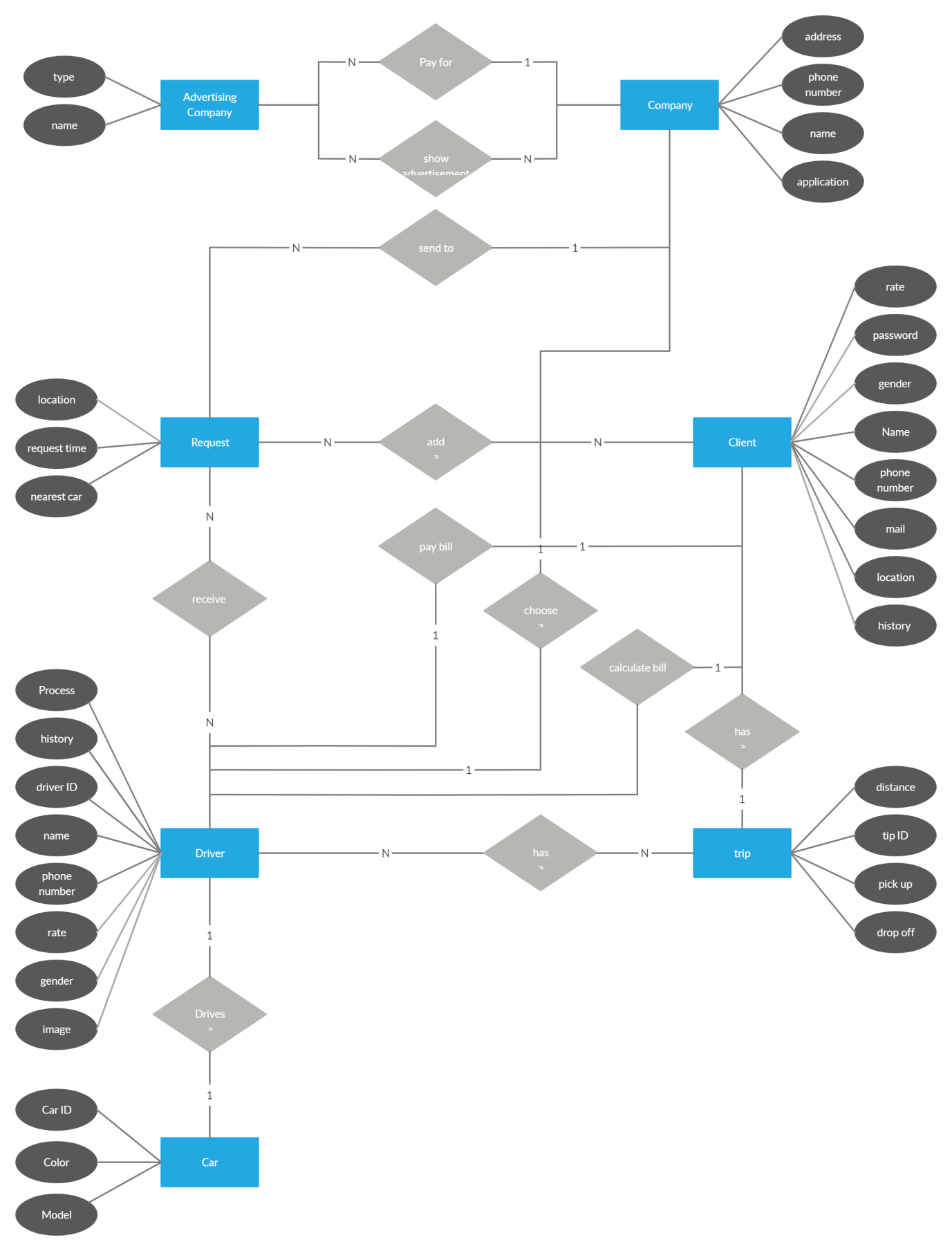


Figure 4.2: ER Diagram

#### UML Diagrams

##### Use Case Diagram of the System

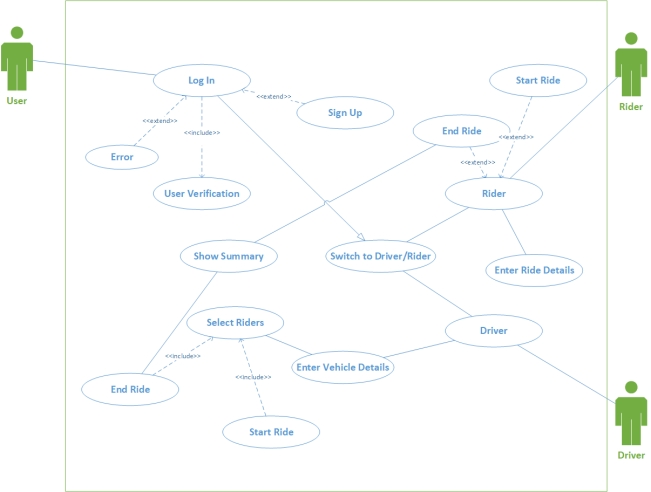


Figure 4.3: Use Case Diagram

##### Class Diagram of the System

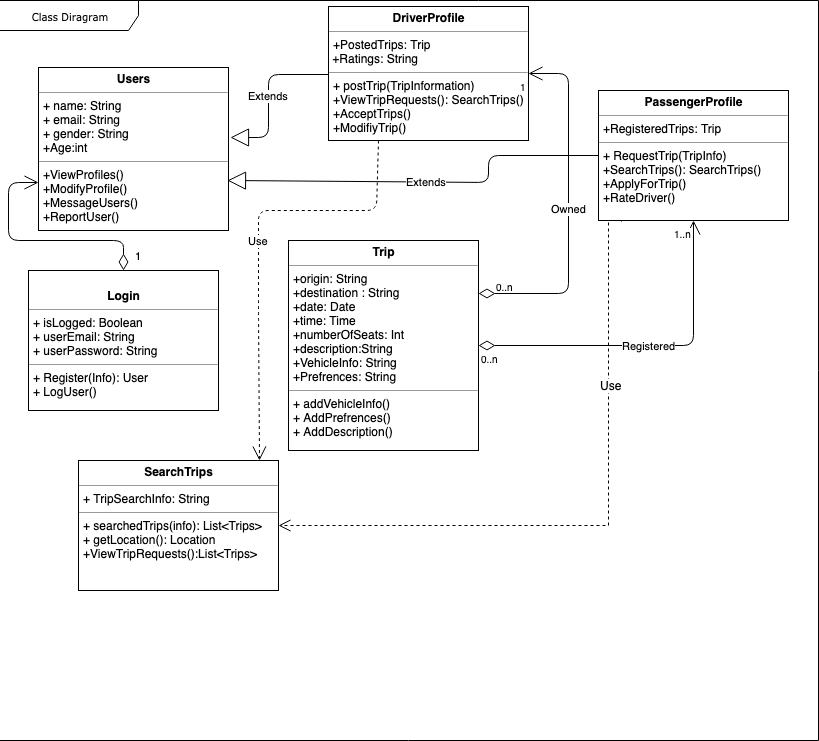


Figure 4.4: Class Diagram

##### Activity Diagram of the System

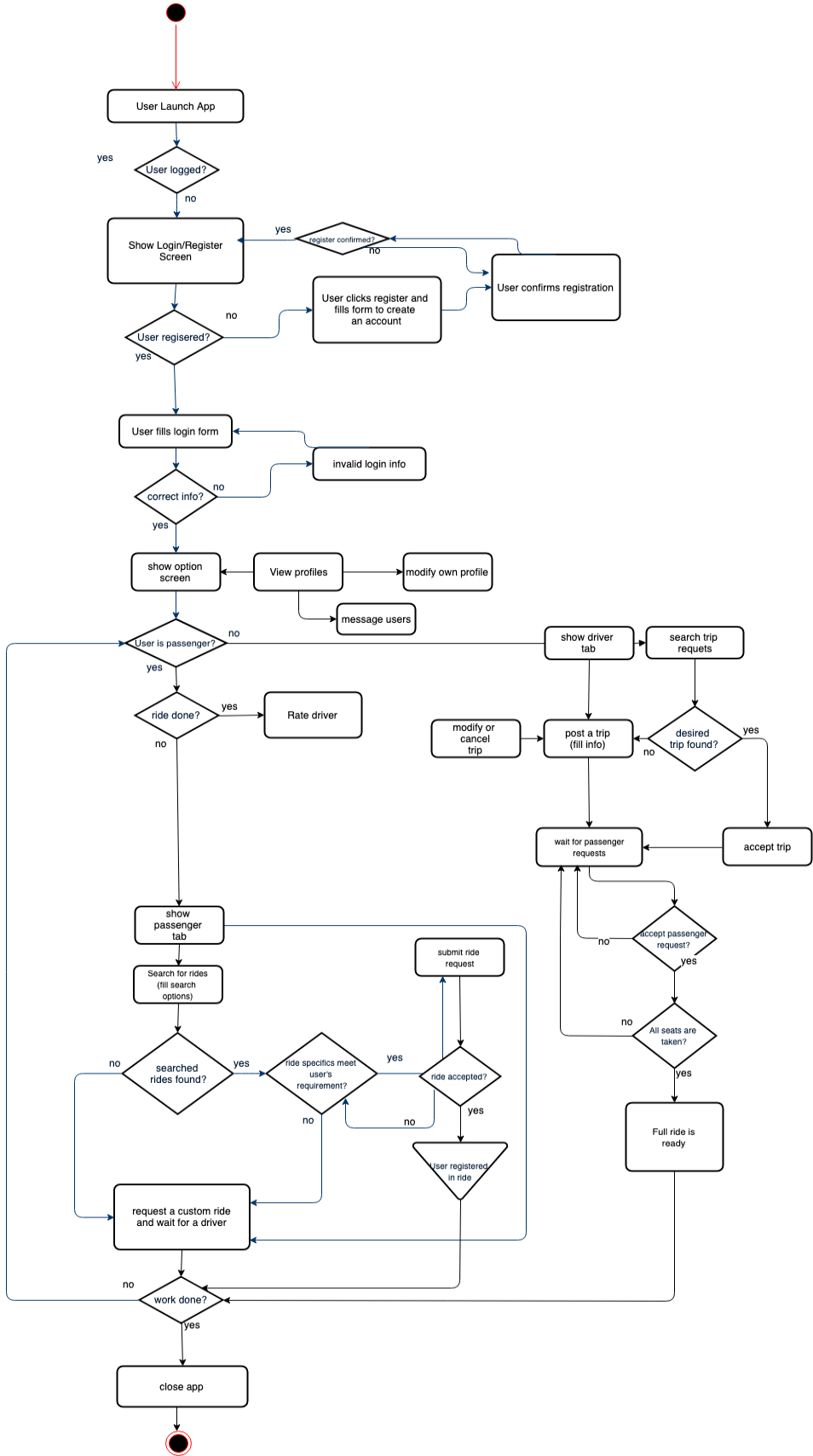


Figure 4.5: Activity Diagram

##### Sequence Diagram of the System

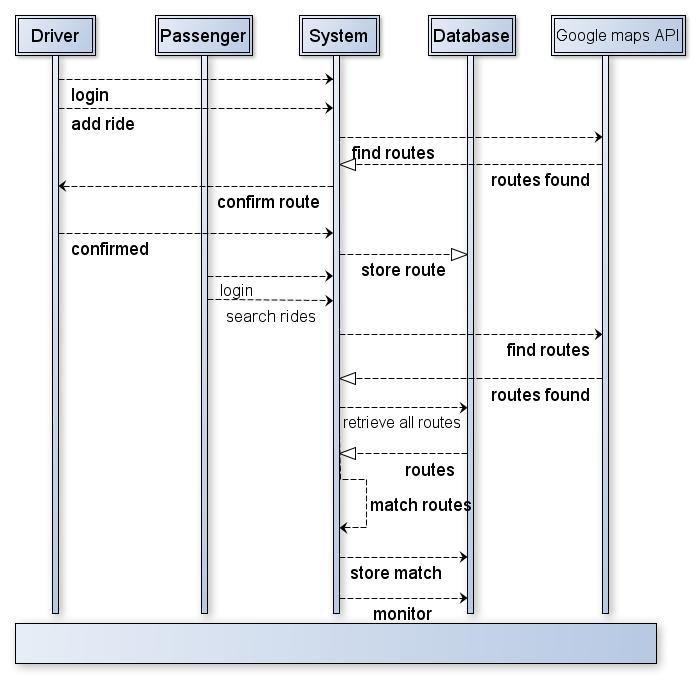


Figure 4.6: Sequence Diagram

### Chapter Summary

UML class diagrams and project specifications are useful when modeling data. By accurately modeling attributes and associations of class entities, we can easily map these class diagram specifications to better understand them. Furthermore it’s hard to map all the project’s features before doing all the planning of the features because some changes may occur during the project's development some features will be added depending on the situation and availability but these specifications and modeling are the core of the application and it will be made with these specifications as the core design.

Modeling Data is a very important step of the project realization and by turning our idea into a design concept we are going to make the implementation more easy and organized. After designing our project we will turn concept into real functions which would be the implementation to an android application.

# 

## DATABASE

### Database Introduction

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

### Selected Database

The Firebase Real-time Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in real-time.

Cloud Firestore enables you to store, sync and query app data at global scale. Data is synced across all clients in real-time, and remains available when your app goes offline.

#### Reasons for Selection of the Database

* Firebase manages all data real-time in the database. So, the exchange of data to and fro from the database is easy, quick and secure. This is the one of many reasons to choose firebase.
* Firebase has a special authentication services which will help us to keep away the unidentified users and our application secure. Firebase Authentication is super secure as it has developed by none other than Google’s sign-in team.
* ML Kit helps develop machine learning features for all platforms & devices. Leverage the power of the Google Cloud platform to improve the accuracy level of the ML features.

#### Benefits of the Selected Database

* Create Application without backend server
* No need extra money spent for backend server
* Sync real time data in the application
* Quick display data in the application
* Faster than any backend web services
* No SQL database so it is more faster
* You can provide any social networking login with very few lines code
* Push notification
* Dynamic Linking
* Auto Backup and many more…

#### Limitations of the Selected Database

* Limited querying capabilities
* Limited data migration
* Platform-dependence
* Platform-dependence
* Less support for iOS
* Integrate with micro services

### Database Queries

### Database Tables

### Database Schema Diagram

### Chapter Summary

The nature to this application require to have to real-time database so firebase came into play. Firebase has vast amount of features that helps in real-time data handling. The nature of this project requires to get as much help as we can get for the dataset due to its unique nature.

# 

## DEVELOPMENT AND IMPLEMENTATION

### Development of the Computer Program

Development of the computer program is the 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.

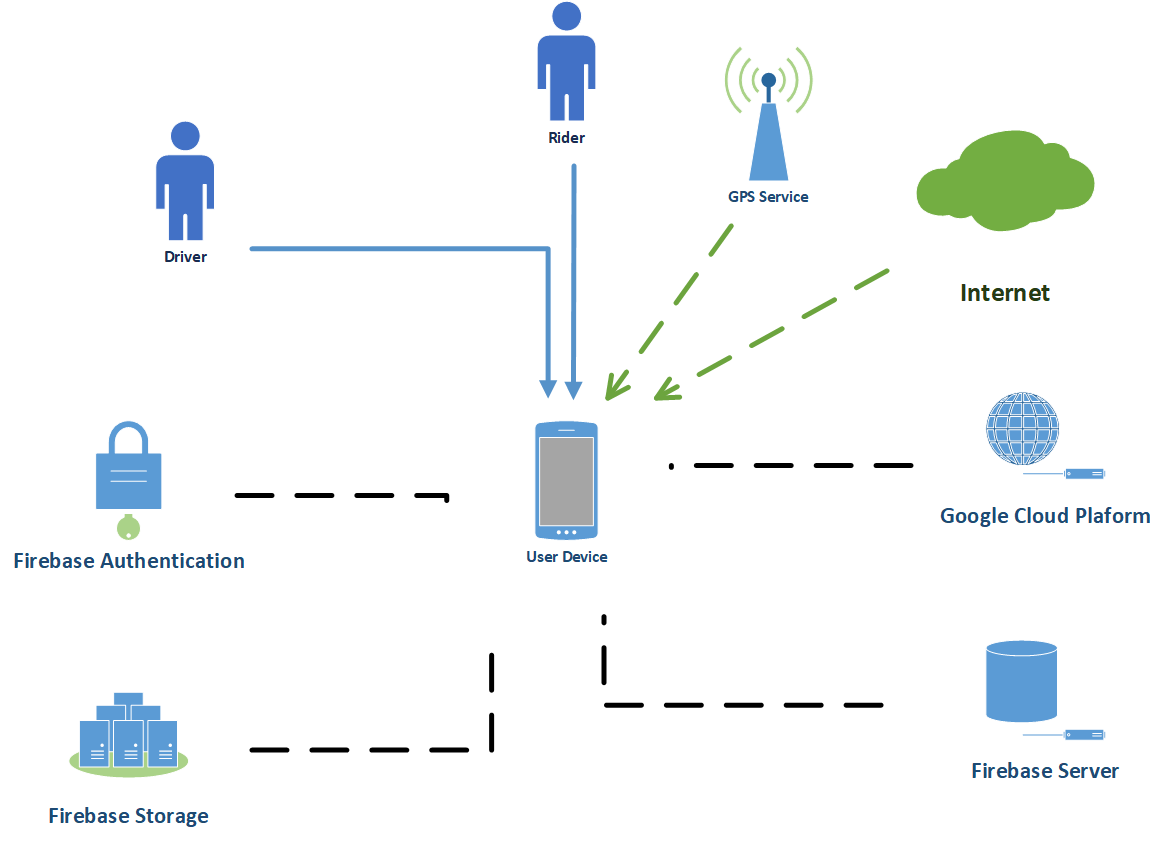


Table 6. 1: Development of the computer program

### Implementation Strategy

* Increased service range to cater for the needs of great amount of customers.
* High level of user convenience.
* Cost-saving through innovation.
* Growth through acquisition.

### Tools Selection

#### Hardware

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

* 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 Infinix Note 9

#### Software

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

### Coding

**public class** CreateAccount {  
 **private** String name;  
 **private** String id;  
 **private** String email;  
 **private** String number;  
 **private** String password;  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
  
 **public** String getId() {  
 **return** id;  
 }  
  
 **public void** setId(String id) {  
 **this**.id = id;  
 }  
  
 **public** String getEmail() {  
 **return** email;  
 }  
  
 **public void** setEmail(String email) {  
 **this**.email = email;  
 }  
  
 **public** String getNumber() {  
 **return** number;  
 }  
  
 **public void** setNumber(String number) {  
 **this**.number = number;  
 }  
  
 **public** String getPassword() {  
 **return** password;  
 }  
  
 **public void** setPassword(String password) {  
 **this**.password = password;  
 }  
  
  
  
 **public** CreateAccount() {  
 }  
  
 **public** CreateAccount(String name, String id, String email, String number, String password) {  
 **this**.name = name;  
 **this**.id = id;  
 **this**.email = email;  
 **this**.number = number;  
 **this**.password = password;  
 }  
  
  
}

**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Switch;  
**import** android.widget.Toast;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
  
**public class** Driver **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_driver);  
 getSupportActionBar().hide();  
  
 String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 **final** DatabaseReference ref= FirebaseDatabase.getInstance().getReference(**"Driver\_Vehicle\_Details"**).child(uid);  
 Button buttonM = (Button) findViewById(R.id.Next);  
  
 **final** Switch s1 =(Switch)findViewById(R.id.switch1);  
  
  
  
  
 buttonM.setOnClickListener(**new** View.OnClickListener(){  
 **public void** onClick(View v){  
 EditText t = (EditText) findViewById(R.id.number2);  
 String number = t.getText().toString();  
 **if**(number.isEmpty()){  
 Toast.makeText(Driver.**this**,**"Enter Vehicle Number"**,Toast.LENGTH\_LONG).show();  
 }**else** {  
 ref.child(**"Vehicle\_Number"**).setValue(number);  
 Intent intentM = **new** Intent(Driver.**this**, DriverMapsActivity.**class**);  
 intentM.putExtra(**"switch"**, s1.isChecked());  
 startActivity(intentM);}  
 }  
  
 });  
 }  
}

**public class** Driver\_Rides\_Summary **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_driver\_\_rides\_\_summary);  
 getSupportActionBar().hide();  
 }  
}

**public class** Driver\_Selected\_Rides **extends** AppCompatActivity {  
  
 RecyclerView recyclerView;  
 RecyclerView.LayoutManager layoutManager;  
  
 @Override  
 **protected void** onCreate(**final** Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_driver\_\_selected\_\_rides);  
 getSupportActionBar().hide();  
  
 Button button = (Button) findViewById(R.id.end);  
 **final** FirebaseDatabase database = FirebaseDatabase.getInstance();  
 **final** DatabaseReference myRef = database.getReference(**"MYRides"**);  
 **final** String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 **final** DatabaseReference myRef1 = myRef.child(uid);  
  
  
 recyclerView= (RecyclerView) findViewById(R.id.selected);  
 recyclerView.setHasFixedSize(**true**);  
 layoutManager = **new** LinearLayoutManager(**this**);  
 recyclerView.setLayoutManager(layoutManager);  
  
 **final** FirebaseRecyclerAdapter<SelectedRides, SelectedViewHolder> adapter1 = **new** FirebaseRecyclerAdapter<SelectedRides, SelectedViewHolder>(SelectedRides.**class**,R.layout.user\_data,SelectedViewHolder.**class**,myRef1) {  
 @Override  
 **protected void** populateViewHolder(SelectedViewHolder selectedViewHolder, SelectedRides selectedRides, **int** i) {  
  
 **final** SelectedRides clickItem = selectedRides;  
 selectedViewHolder.Rider\_Name.setText(selectedRides.getRider\_Name());  
 selectedViewHolder.Rider\_Number.setText(selectedRides.getRider\_Number());  
 selectedViewHolder.Rider\_Pickup.setText(**"Pickup Point :"** + selectedRides.getRider\_Pickup());  
 selectedViewHolder.Rider\_Destination.setText(**"Destination :"** + selectedRides.getRider\_Destination());  
  
 }  
  
  
 };  
  
 recyclerView.setAdapter(adapter1);  
  
 button.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **final** DatabaseReference re= FirebaseDatabase.getInstance().getReference(**"Driver\_Vehicle\_Details"**);  
 myRef.child(uid).removeValue();  
  
 re.child(uid).removeValue();  
  
 Intent i = **new** Intent(Driver\_Selected\_Rides.**this**,Driver\_Rides\_Summary.**class**);  
 startActivity(i);  
 }  
 });  
  
 Button b = (Button) findViewById(R.id.Pickup);  
 b.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent i = **new** Intent(Driver\_Selected\_Rides.**this**,Pickup.**class**);  
  
 startActivity(i);  
 }  
 });  
  
 }  
  
  
}

**public class** DriverMapsActivity **extends** FragmentActivity **implements** OnMapReadyCallback, GoogleApiClient.ConnectionCallbacks,com.google.android.gms.location.LocationListener {  
  
 **private** GoogleMap mMap;  
 **private** FusedLocationProviderClient mFusedLocationProviderClient;  
 **private** PlacesClient placesClient;  
 **private** List<AutocompletePrediction> predictionList;  
 **private** GoogleApiClient Api;  
  
  
 **private** Location mLastKnownLocation;  
 **private** LocationCallback locationCallback;  
  
 **private** MaterialSearchBar materialSearchBar;  
 **private** View mapView;  
 **private** Button btnFind;  
 **private** RippleBackground rippleBg;  
  
 **private final float** DEFAULT\_ZOOM = 15;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_driver\_maps);  
  
 *// materialSearchBar = findViewById(R.id.searchBar);* btnFind = findViewById(R.id.btn\_find);  
 rippleBg = findViewById(R.id.ripple\_bg);  
  
 SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);  
 mapFragment.getMapAsync(**this**);  
 mapView = mapFragment.getView();  
  
 mFusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(DriverMapsActivity.**this**);  
 Places.initialize(DriverMapsActivity.**this**, getString(R.string.map\_api));  
 placesClient = Places.createClient(**this**);  
 **final** AutocompleteSessionToken token = AutocompleteSessionToken.newInstance();  
  
*//// materialSearchBar.setOnSearchActionListener(new MaterialSearchBar.OnSearchActionListener() {  
//// @Override  
//// public void onSearchStateChanged(boolean enabled) {  
////  
//// }  
////  
//// @Override  
//// public void onSearchConfirmed(CharSequence text) {  
//// startSearch(text.toString(), true, null, true);  
//// }  
////  
//// @Override  
//// public void onButtonClicked(int buttonCode) {  
//// if (buttonCode == MaterialSearchBar.BUTTON\_NAVIGATION) {  
//// //opening or closing a navigation drawer  
//// } else if (buttonCode == MaterialSearchBar.BUTTON\_BACK) {  
//// //materialSearchBar.disableSearch();  
//// }  
//// }  
//// });  
//  
// materialSearchBar.addTextChangeListener(new TextWatcher() {  
// @Override  
// public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
//  
// }  
//  
// @Override  
// public void onTextChanged(CharSequence s, int start, int before, int count) {  
// FindAutocompletePredictionsRequest predictionsRequest = FindAutocompletePredictionsRequest.builder()  
// .setTypeFilter(TypeFilter.ADDRESS)  
// .setSessionToken(token)  
// .setQuery(s.toString())  
// .build();  
// placesClient.findAutocompletePredictions(predictionsRequest).addOnCompleteListener(new OnCompleteListener<FindAutocompletePredictionsResponse>() {  
// @Override  
// public void onComplete(@NonNull Task<FindAutocompletePredictionsResponse> task) {  
// if (task.isSuccessful()) {  
// FindAutocompletePredictionsResponse predictionsResponse = task.getResult();  
// if (predictionsResponse != null) {  
// predictionList = predictionsResponse.getAutocompletePredictions();  
// List<String> suggestionsList = new ArrayList<>();  
// for (int i = 0; i < predictionList.size(); i++) {  
// AutocompletePrediction prediction = predictionList.get(i);  
// suggestionsList.add(prediction.getFullText(null).toString());  
// }  
// materialSearchBar.updateLastSuggestions(suggestionsList);  
// if (!materialSearchBar.isSuggestionsVisible()) {  
// materialSearchBar.showSuggestionsList();  
// }  
// }  
// } else {  
// Log.i("mytag", "prediction fetching task unsuccessful");  
// }  
// }  
// });  
// }  
//  
// @Override  
// public void afterTextChanged(Editable s) {  
//  
// }  
// });  
//  
// materialSearchBar.setSuggestionsClickListener(new SuggestionsAdapter.OnItemViewClickListener() {  
// @Override  
// public void OnItemClickListener(int position, View v) {  
// if (position >= predictionList.size()) {  
// return;  
// }  
// AutocompletePrediction selectedPrediction = predictionList.get(position);  
// String suggestion = materialSearchBar.getLastSuggestions().get(position).toString();  
// materialSearchBar.setText(suggestion);  
//  
// new Handler().postDelayed(new Runnable() {  
// @Override  
// public void run() {  
// materialSearchBar.clearSuggestions();  
// }  
// }, 1000);  
// InputMethodManager imm = (InputMethodManager) getSystemService(INPUT\_METHOD\_SERVICE);  
// if (imm != null)  
// imm.hideSoftInputFromWindow(materialSearchBar.getWindowToken(), InputMethodManager.HIDE\_IMPLICIT\_ONLY);  
// final String placeId = selectedPrediction.getPlaceId();  
// List<Place.Field> placeFields = Arrays.asList(Place.Field.LAT\_LNG);  
//  
// FetchPlaceRequest fetchPlaceRequest = FetchPlaceRequest.builder(placeId, placeFields).build();  
// placesClient.fetchPlace(fetchPlaceRequest).addOnSuccessListener(new OnSuccessListener<FetchPlaceResponse>() {  
// @Override  
// public void onSuccess(FetchPlaceResponse fetchPlaceResponse) {  
// Place place = fetchPlaceResponse.getPlace();  
// Log.i("mytag", "Place found: " + place.getName());  
// LatLng latLngOfPlace = place.getLatLng();  
// if (latLngOfPlace != null) {  
// mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLngOfPlace, DEFAULT\_ZOOM));  
// }  
// }  
// }).addOnFailureListener(new OnFailureListener() {  
// @Override  
// public void onFailure(@NonNull Exception e) {  
// if (e instanceof ApiException) {  
// ApiException apiException = (ApiException) e;  
// apiException.printStackTrace();  
// int statusCode = apiException.getStatusCode();  
// Log.i("mytag", "place not found: " + e.getMessage());  
// Log.i("mytag", "status code: " + statusCode);  
// }  
// }  
// });  
// }  
//  
// @Override  
// public void OnItemDeleteListener(int position, View v) {  
//  
// }  
// });* btnFind.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
*//  
// rippleBg.startRippleAnimation();  
//  
//  
// switch (v.getId())  
// {  
// case R.id.btn\_find:  
//  
// MaterialSearchBar addressField = (MaterialSearchBar) findViewById(R.id.searchBar);  
// String address = addressField.getText().toString();  
//  
// List<Address> addressList = null;  
// MarkerOptions userMarkerOptions = new MarkerOptions();  
//  
// if (!TextUtils.isEmpty(address)) {  
// Geocoder geocoder = new Geocoder(DriverMapsActivity.this);  
// try {  
// addressList = geocoder.getFromLocationName(address, 6);  
// if (addressList != null) {  
// for (int i = 0; i <= addressList.size(); i++) {  
// Address userAddress = addressList.get(i);  
// LatLng latLng = new LatLng(userAddress.getLatitude(), userAddress.getLongitude());  
//  
// userMarkerOptions.position(latLng);  
// userMarkerOptions.title(address);  
// userMarkerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE\_GREEN));  
//  
// mMap.addMarker(userMarkerOptions);  
// mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));  
// mMap.animateCamera(CameraUpdateFactory.zoomTo(10));  
// }  
// }  
//  
// else  
// {  
// Toast.makeText(DriverMapsActivity.this,"not found",Toast.LENGTH\_SHORT).show();  
// }  
// } catch (IOException e) {  
// e.printStackTrace();  
// }  
//  
// }  
// else  
// {  
// Toast.makeText(DriverMapsActivity.this,"Please write any location anme",Toast.LENGTH\_SHORT).show();  
// }  
// break;  
// }* Intent intent = **new** Intent(DriverMapsActivity.**this**,Driver\_Rides\_Selection.**class**);  
 startActivity(intent);  
  
  
 }  
 });  
 }  
  
 @SuppressLint(**"MissingPermission"**)  
 @Override  
 **public void** onMapReady(GoogleMap googleMap) {  
 mMap = googleMap;  
 *// mMap.setMyLocationEnabled(true);  
 // mMap.getUiSettings().setMyLocationButtonEnabled(true);  
  
 /\* if (mapView != null && mapView.findViewById(Integer.parseInt("1")) != null) {  
 View locationButton = ((View) mapView.findViewById(Integer.parseInt("1")).getParent()).findViewById(Integer.parseInt("2"));  
 RelativeLayout.LayoutParams layoutParams = (RelativeLayout.LayoutParams) locationButton.getLayoutParams();  
 layoutParams.addRule(RelativeLayout.ALIGN\_PARENT\_TOP, 0);  
 layoutParams.addRule(RelativeLayout.ALIGN\_PARENT\_BOTTOM, RelativeLayout.TRUE);  
 layoutParams.setMargins(0, 0, 40, 180);  
 }\*/  
  
 //check if gps is enabled or not and then request user to enable it* LocationRequest locationRequest = LocationRequest.create();  
 locationRequest.setInterval(10000);  
 locationRequest.setFastestInterval(5000);  
 locationRequest.setPriority(LocationRequest.PRIORITY\_HIGH\_ACCURACY);  
  
 LocationSettingsRequest.Builder builder = **new** LocationSettingsRequest.Builder().addLocationRequest(locationRequest);  
  
 SettingsClient settingsClient = LocationServices.getSettingsClient(DriverMapsActivity.**this**);  
 Task<LocationSettingsResponse> task = settingsClient.checkLocationSettings(builder.build());  
  
 task.addOnSuccessListener(DriverMapsActivity.**this**, **new** OnSuccessListener<LocationSettingsResponse>() {  
 @Override  
 **public void** onSuccess(LocationSettingsResponse locationSettingsResponse) {  
 getDeviceLocation();  
 buildGoogleApiClient();  
 }  
  
  
 });  
  
 task.addOnFailureListener(DriverMapsActivity.**this**, **new** OnFailureListener() {  
 @Override  
 **public void** onFailure(@NonNull Exception e) {  
 **if** (e **instanceof** ResolvableApiException) {  
 ResolvableApiException resolvable = (ResolvableApiException) e;  
 **try** {  
 resolvable.startResolutionForResult(DriverMapsActivity.**this**, 51);  
 } **catch** (IntentSender.SendIntentException e1) {  
 e1.printStackTrace();  
 }  
 }  
 }  
 });  
  
 mMap.setOnMyLocationButtonClickListener(**new** GoogleMap.OnMyLocationButtonClickListener() {  
 @Override  
 **public boolean** onMyLocationButtonClick() {  
 **if** (materialSearchBar.isSuggestionsVisible())  
 materialSearchBar.clearSuggestions();  
 *//if (materialSearchBar.isSearchEnabled())  
 // materialSearchBar.disableSearch();* **return false**;  
 }  
 });  
 }  
  
 @Override  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  
 **super**.onActivityResult(requestCode, resultCode, data);  
 **if** (requestCode == 51) {  
 **if** (resultCode == RESULT\_OK) {  
 getDeviceLocation();  
 }  
 }  
 }  
  
 @SuppressLint(**"MissingPermission"**)  
 **private void** getDeviceLocation() {  
 mFusedLocationProviderClient.getLastLocation()  
 .addOnCompleteListener(**new** OnCompleteListener<Location>() {  
 @Override  
 **public void** onComplete(@NonNull Task<Location> task) {  
 **if** (task.isSuccessful()) {  
 mLastKnownLocation = task.getResult();  
 **if** (mLastKnownLocation != **null**) {  
 mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(**new** LatLng(mLastKnownLocation.getLatitude(), mLastKnownLocation.getLongitude()), DEFAULT\_ZOOM));  
 } **else** {  
 **final** LocationRequest locationRequest = LocationRequest.create();  
 locationRequest.setInterval(10000);  
 locationRequest.setFastestInterval(5000);  
 locationRequest.setPriority(LocationRequest.PRIORITY\_HIGH\_ACCURACY);  
 locationCallback = **new** LocationCallback() {  
 @Override  
 **public void** onLocationResult(LocationResult locationResult) {  
 **super**.onLocationResult(locationResult);  
 **if** (locationResult == **null**) {  
 **return**;  
 }  
 mLastKnownLocation = locationResult.getLastLocation();  
 mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(**new** LatLng(mLastKnownLocation.getLatitude(), mLastKnownLocation.getLongitude()), DEFAULT\_ZOOM));  
 mFusedLocationProviderClient.removeLocationUpdates(locationCallback);  
 }  
 };  
 mFusedLocationProviderClient.requestLocationUpdates(locationRequest, locationCallback, **null**);  
  
 }  
 } **else** {  
 Toast.makeText(DriverMapsActivity.**this**, **"unable to get last location"**, Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 }  
  
 **protected synchronized void** buildGoogleApiClient() {  
 Api= **new** GoogleApiClient.Builder(**this**)  
 .addConnectionCallbacks(**this**).addApi(LocationServices.API).build();  
 Api.connect();  
 }  
 @Override  
 **public void** onLocationChanged(Location location) {  
*// mLastKnownLocation = location;  
// LatLng latLng = new LatLng(location.getLatitude(),location.getLongitude());  
// mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));  
// mMap.animateCamera(CameraUpdateFactory.zoomTo(11));  
//  
// final String Uid = FirebaseAuth.getInstance().getCurrentUser().getUid();  
// DatabaseReference ref = FirebaseDatabase.getInstance().getReference("RidesAvailable");  
//// // DatabaseReference data = FirebaseDatabase.getInstance().getReference("User");  
//////  
// DatabaseReference zonesRef = FirebaseDatabase.getInstance().getReference("User");  
// DatabaseReference zone1Ref = zonesRef.child(Uid);  
// DatabaseReference name = zone1Ref.child("name");  
//  
// name.addValueEventListener(new ValueEventListener() {  
// @Override  
// public void onDataChange(DataSnapshot snapshot) {  
// DatabaseReference ref = FirebaseDatabase.getInstance().getReference("RidesAvailable");  
// // System.out.println(snapshot.getValue()); //prints "Do you have data? You'll love Firebase."  
// //DataSnapshot value = new DataSnapshot().getValue();  
// // Log.i("SAmi", snapshot.getValue(String.class));  
// ref.child(Uid).child("Name").setValue(snapshot.getValue(String.class));  
// }  
// @Override  
// public void onCancelled(DatabaseError databaseError) {  
// }  
// });  
//  
//  
// // name.setValue("Sami");  
// // ref.child(Uid).child("g").setValue();  
//  
// GeoFire geoFire = new GeoFire(ref);  
// // geoFire.setLocation(Uid,new GeoLocation(location.getLatitude(),location.getLongitude()));  
// geoFire.setLocation(Uid,new GeoLocation(location.getLatitude(), location.getLongitude()), new GeoFire.CompletionListener() {  
// @Override  
// public void onComplete(String key, DatabaseError error) {  
// if (error!=null)  
// {  
// Toast.makeText(DriverMapsActivity.this,"Can't go Active",Toast.LENGTH\_SHORT).show();  
// }  
// Toast.makeText(DriverMapsActivity.this,"You are Active",Toast.LENGTH\_SHORT).show();  
// }  
// });* }  
  
 @Override  
 **public void** onConnected(@Nullable Bundle bundle) {  
 LocationRequest locationRequest = **new** LocationRequest();  
 locationRequest.setInterval(10000);  
 locationRequest.setFastestInterval(5000);  
 locationRequest.setPriority(LocationRequest.PRIORITY\_HIGH\_ACCURACY);  
  
 LocationServices.FusedLocationApi.requestLocationUpdates(Api,locationRequest,**this**);  
  
 *// mFusedLocationProviderClient.requestLocationUpdates(locationRequest, locationCallback, null);* }  
  
 @Override  
 **public void** onConnectionSuspended(**int** i) {  
*// final String Uid = FirebaseAuth.getInstance().getCurrentUser().getUid();  
// DatabaseReference ref = FirebaseDatabase.getInstance().getReference("RidesAvailable");  
//// ref.child(Uid).removeValue();  
// GeoFire geoFire = new GeoFire(ref);  
// geoFire.removeLocation(Uid, new GeoFire.CompletionListener() {  
// @Override  
// public void onComplete(String key, DatabaseError error) {  
// if (error!=null)  
// {  
// Toast.makeText(DriverMapsActivity.this,"REHHH",Toast.LENGTH\_SHORT).show();  
// }  
// Toast.makeText(DriverMapsActivity.this,"Request Removed",Toast.LENGTH\_SHORT).show();  
// }  
//  
// });  
//* }  
  
 @Override  
 **protected void** onStop() {  
 **super**.onStop();  
*// final String Uid = FirebaseAuth.getInstance().getCurrentUser().getUid();  
// DatabaseReference ref = FirebaseDatabase.getInstance().getReference("RidesAvailable");  
//  
//// ref.addValueEventListener(new ValueEventListener() {  
//// @Override  
//// public void onDataChange(DataSnapshot snapshot) {  
//// DatabaseReference ref = FirebaseDatabase.getInstance().getReference("RidesAvailable");  
//// // System.out.println(snapshot.getValue()); //prints "Do you have data? You'll love Firebase."  
//// //DataSnapshot value = new DataSnapshot().getValue();  
//// // Log.i("SAmi", snapshot.getValue(String.class));  
//// ref.child(Uid).child("Name").removeValue();  
//// }  
//// @Override  
//// public void onCancelled(DatabaseError databaseError) {  
//// }  
//// });  
//  
//  
//  
// GeoFire geoFire = new GeoFire(ref);  
// geoFire.removeLocation(Uid, new GeoFire.CompletionListener() {  
// @Override  
// public void onComplete(String key, DatabaseError error) {  
// if (error!=null)  
// {  
// Toast.makeText(DriverMapsActivity.this,"REHHH",Toast.LENGTH\_SHORT).show();  
// }  
// Toast.makeText(DriverMapsActivity.this,"Request Removed",Toast.LENGTH\_SHORT).show();  
// }  
//  
// });* }  
  
  
}

**public class** ForgotPassword **extends** AppCompatActivity {  
 **private** EditText inputEmail;  
  
 **private** Button btnReset;  
  
 **private** FirebaseAuth auth;  
  
 **private** ProgressBar progressBar;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_forgot\_password);  
 getSupportActionBar().hide();  
 inputEmail = (EditText) findViewById(R.id.EtEmailReset);  
  
 btnReset = (Button) findViewById(R.id.reset);  
  
 progressBar = (ProgressBar) findViewById(R.id.progressBar);  
  
 auth = FirebaseAuth.getInstance();  
 btnReset.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
  
 String email = inputEmail.getText().toString().trim();  
  
 **if** (TextUtils.isEmpty(email)) {  
 Toast.makeText(getApplication(), **"Enter your registered email id"**, Toast.LENGTH\_SHORT).show();  
 **return**;  
 }  
  
  
  
 auth.sendPasswordResetEmail(email)  
  
 .addOnCompleteListener(**new** OnCompleteListener<Void>() {  
 @Override  
 **public void** onComplete(@NonNull Task<Void> task) {  
 **if** (task.isSuccessful()) {  
 Toast.makeText(ForgotPassword.**this**, **"We have sent you instructions to reset your password!"**, Toast.LENGTH\_SHORT).show();  
 } **else** {  
 Toast.makeText(ForgotPassword.**this**, **"Failed to send reset email!"**, Toast.LENGTH\_SHORT).show();  
 }  
  
 progressBar.setVisibility(View.GONE);  
 }  
 });  
 }  
 });  
 }  
}

**public class** LogIn **extends** AppCompatActivity {  
  
  
 **private** FirebaseDatabase database;  
 **private** DatabaseReference ref;  
 **private** FirebaseAuth mAuth;  
  
  
  
 **private** FirebaseAuth.AuthStateListener mAutL = **new** FirebaseAuth.AuthStateListener() {  
 @Override  
 **public void** onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {  
  
 }  
  
  
 };  
 **private static final** String TAG = **"LogIn"**;  
  
 **private** EditText email;  
 **private** EditText pass;  
 **private** Button login;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_log\_in);  
 getSupportActionBar().hide();  
 email = (EditText) findViewById(R.id.EtEmail);  
 pass = (EditText) findViewById(R.id.EtPass);  
 login = (Button) findViewById(R.id.LogIn);  
  
 *// Forgot Password* TextView textViewF = (TextView) findViewById(R.id.ForgotPasswrod);  
  
  
  
 textViewF.setOnClickListener(**new** View.OnClickListener(){  
 **public void** onClick(View v){  
 Intent intentF = **new** Intent(LogIn.**this**, ForgotPassword.**class**);  
 startActivity(intentF);  
 Toast.makeText(LogIn.**this**,**"Forgot Password"**, Toast.LENGTH\_LONG).show();  
 }  
  
 });  
  
  
 mAuth = FirebaseAuth.getInstance();  
 mAutL = **new** FirebaseAuth.AuthStateListener() {  
 @Override  
 **public void** onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {  
  
 }  
  
  
 *//Create New Account* };  
 *// create Account Intent* TextView signup = (TextView) findViewById(R.id.CreateAccount);  
 signup.setOnClickListener(**new** View.OnClickListener() {  
 **public void** onClick(View v) {  
 Intent intentS = **new** Intent(LogIn.**this**, SignUp.**class**);  
 startActivity(intentS);  
 }  
 });  
 database = FirebaseDatabase.getInstance();  
 ref = database.getReference(**"message"**);  
  
 ref.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(@NonNull DataSnapshot dataSnapshot) {  
  
 String value = dataSnapshot.getValue(String.**class**);  
  
 }  
  
 @Override  
 **public void** onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
  
 mAutL = **new** FirebaseAuth.AuthStateListener() {  
 @Override  
 **public void** onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {  
 FirebaseUser user = firebaseAuth.getCurrentUser();  
 **if** (user != **null**) {  
 *//user is signed in* Log.d(TAG, **"User Signed IN"**);  
 } **else** {  
 *//user is signed out* Log.d(TAG, **"User Signed out"**);  
 }  
 }  
 };  
 }  
  
 @Override  
 **protected void** onStart() {  
 **super**.onStart();  
 mAuth.addAuthStateListener(mAutL);  
  
 }  
  
 @Override  
 **protected void** onStop() {  
 **super**.onStop();  
 **if** (mAutL != **null**) {  
 mAuth.removeAuthStateListener(mAutL);  
 }  
  
 login.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 String emailt = email.getText().toString();  
 String passt = pass.getText().toString();  
  
 **if** (!emailt.equals(**""**) && !pass.equals(**""**)) {  
 mAuth.signInWithEmailAndPassword(emailt, passt)  
 .addOnCompleteListener(LogIn.**this**, **new** OnCompleteListener<AuthResult>() {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task) {  
  
 **if** (!task.isSuccessful()) {  
 Toast.makeText(LogIn.**this**, **"Wrong Email or Password"**, Toast.LENGTH\_LONG).show();  
 *//ref.setValue("NOOOOOOO");* } **else** {  
 *// Toast.makeText(LogIn.this, "Welcome", Toast.LENGTH\_LONG).show();  
 //ref.setValue("Hello");* checkIfEmailVerified();  
 }  
 }  
 });  
  
 }  
 }  
 });  
  
  
 }  
  
 **private void** checkIfEmailVerified() {  
 FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();  
  
 **if** (user.isEmailVerified())  
 {  
 *// user is verified, so you can finish this activity or send user to activity which you want.* finish();  
 Toast.makeText(LogIn.**this**, **"Successfully logged in"**, Toast.LENGTH\_SHORT).show();  
 Intent intent = **new** Intent(LogIn.**this**,SignIn.**class**);  
  
 startActivity(intent);  
  
 }  
 **else** {  
 *// email is not verified, so just prompt the message to the user and restart this activity.  
 // NOTE: don't forget to log out the user.* Toast.makeText(LogIn.**this**, **"Verify your Email First"**, Toast.LENGTH\_SHORT).show();  
 FirebaseAuth.getInstance().signOut();  
  
 *//restart this activity* }  
  
  
  
 }  
  
}  
  
  
  
 *//Forgot Password  
  
  
  
 /\* // Log In Button  
 Button buttonL = (Button) findViewById(R.id.LogIn);  
  
 buttonL.setOnClickListener(new View.OnClickListener(){  
 public void onClick(View v){  
 Intent intentL = new Intent(LogIn.this, SignIn.class);  
 startActivity(intentL);  
 }  
  
 });  
  
  
  
 }  
}  
\*/*

**public class** MainActivity **extends** AppCompatActivity {  
 **private static final int** Request\_Code=101;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 getSupportActionBar().hide();  
  
  
  
  
 Button button = (Button) findViewById(R.id.button);  
 **if** (ActivityCompat.checkSelfPermission(**this**, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {  
 ActivityCompat.requestPermissions(**this**, **new** String[]  
 {Manifest.permission.ACCESS\_FINE\_LOCATION}, Request\_Code);  
 }  
  
  
 button.setOnClickListener(**new** View.OnClickListener(){  
 **public void** onClick(View v){  
 Intent intent = **new** Intent(MainActivity.**this**, LogIn.**class**);  
 startActivity(intent);  
 *// Toast.makeText(MainActivity.this,"Welcome", Toast.LENGTH\_LONG).show();* }  
  
 });  
  
  
 }  
  
}

**public class** Pickup **extends** AppCompatActivity **implements** OnMapReadyCallback {  
 GoogleMap gMap;  
 Location mlocation;  
 RecyclerView recyclerView;  
 RecyclerView.LayoutManager layoutManager;  
 **public** TextView ss;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_pickup);  
 getSupportActionBar().hide();  
  
  
  
 SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);  
 mapFragment.getMapAsync(**this**);  
 *// mapView = mapFragment.getView()* }  
  
 @Override  
 **public void** onMapReady(GoogleMap googleMap) {  
 gMap = googleMap;  
  
 *// String id = FirebaseAuth.getInstance().getCurrentUser().getUid();* **final** FirebaseDatabase database = FirebaseDatabase.getInstance();  
 **final** DatabaseReference myRef = database.getReference(**"MYRides"**);  
 **final** String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 **final** DatabaseReference myRef1 = myRef.child(uid);  
  
 myRef.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 **for**(DataSnapshot snapshot: dataSnapshot.getChildren()) {  
 **for** (DataSnapshot snapshot1 : snapshot.getChildren()) {  
  
  
  
 String n = snapshot1.child(**"Rider\_Pickup"**).getValue(String.**class**);  
  
 *//n = "Select Pickup point";* **if** (n == **null**) {  
 Toast.makeText(Pickup.**this**, **"No Pickup found"**, Toast.LENGTH\_LONG).show();  
  
 }  
 **else** {  
 **if** (n.equals(**"G9 Markaz"**)) {  
 LatLng g9 = **new** LatLng(33.690036, 73.030187);  
 gMap.addMarker(**new** MarkerOptions().position(g9).title(**"G9 Markaz"**));  
 gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(g9, 10));  
  
  
 } **else** {  
 **if** (n.equals(**"Air University"**)) {  
 LatLng Air = **new** LatLng(33.713818, 73.026399);  
 gMap.addMarker(**new** MarkerOptions().position(Air).title(**"Sami"**));  
 gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(Air, 10));  
  
  
 } **else** {**if**(n.equals(**"F8 Markaz"**)){  
 LatLng f8 = **new** LatLng(33.712382, 73.036899);  
 *// sydney.showInfoWindow();* gMap.addMarker(**new** MarkerOptions().position(f8).title(**"F8 Markaz"**));  
 gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(f8, 10));  
 }**else** {  
 **if**(n.equals(**"Sadar"**)){  
 LatLng sadar = **new** LatLng(33.5914237, 73.0535122);  
 gMap.addMarker(**new** MarkerOptions().position(sadar).title(**"Sadar"**));  
 gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(sadar, 10));  
 }  
 }  
  
  
 }  
 }  
  
 }  
  
 }}}  
*// // for (snapshot.)  
// // List <String> post = snapshot.getValue();  
//  
//  
//  
// }  
//  
//* @Override  
 **public void** onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
  
*// recyclerView= (RecyclerView) findViewById(R.id.sss);  
// recyclerView.setHasFixedSize(true);  
// layoutManager = new LinearLayoutManager(this);  
// recyclerView.setLayoutManager(layoutManager);  
//  
//  
// final FirebaseRecyclerAdapter<SelectedRides, SelectedViewHolder> adapter1 = new FirebaseRecyclerAdapter<SelectedRides, SelectedViewHolder>(SelectedRides.class,R.layout.user\_data,SelectedViewHolder.class,myRef1) {  
// @Override  
// protected void populateViewHolder(SelectedViewHolder selectedViewHolder, SelectedRides selectedRides, int i) {  
//  
//  
//  
//  
// selectedViewHolder.Rider\_Name.setText(selectedRides.getRider\_Name());  
// selectedViewHolder.Rider\_Number.setText(selectedRides.getRider\_Number());  
// selectedViewHolder.Rider\_Pickup.setText("Pickup Point :" + selectedRides.getRider\_Pickup());  
// selectedViewHolder.Rider\_Destination.setText("Destination :" + selectedRides.getRider\_Destination());  
//  
// final DatabaseReference selected = FirebaseDatabase.getInstance().getReference("Rides");  
// selected.child(uid).child("Rider\_Name").setValue(selectedRides.getRider\_Pickup());  
// // final TextView s = (TextView)findViewById(R.id.textView15);  
// String ss= selectedRides.getRider\_Pickup();  
//  
// // s.setText(ss);  
//// arrayOfSongs= new ArrayList<>();  
//// arrayOfSongs.add(ss);  
//// final DatabaseReference myRef = database.getReference("ss");  
//// myRef.child(uid).push().setValue(arrayOfSongs);  
////  
//  
//  
// }  
// };  
// Intent intent= getIntent();  
//  
//  
// recyclerView.setAdapter(adapter1);  
// if (ss==null){  
// Toast.makeText(Pickup.this,"No Pickup found",Toast.LENGTH\_LONG).show();  
//  
// }else  
//  
//  
// if(ss.equals("G9 Markaz")) {  
// LatLng g9 = new LatLng(33.690036, 73.030187);  
// gMap.addMarker(new MarkerOptions().position(g9).title("G9 Markaz"));  
// gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(g9, 15));  
//  
// LatLng Air = new LatLng(33.713818, 73.026399);  
// gMap.addMarker(new MarkerOptions().position(Air).title("Air University"));  
// gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(Air, 15));  
// }else  
// if(ss.equals("G9 Markaz") || ss.equals("Select Pickup Point") ) {  
// LatLng g9 = new LatLng(33.690036, 73.030187);  
// gMap.addMarker(new MarkerOptions().position(g9).title("G9 Markaz"));  
// gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(g9, 15));  
//  
//  
// }  
// else  
// if(ss.equals("Air University") ) {  
// LatLng Air = new LatLng(33.713818, 73.026399);  
// gMap.addMarker(new MarkerOptions().position(Air).title("Air University"));  
// gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(Air, 15));  
//  
//  
// }  
// else {  
//  
//  
// }  
// }  
  
 //String n= null ;  
  
  
  
// if(ss.equals(null)){  
// Toast.makeText(Pickup.this,"Saaaaaaa",Toast.LENGTH\_LONG).show();  
// }  
// else  
  
  
  
  
  
  
  
// gMap.clear();  
//// if(text.equals("F8 Markaz")) {  
// LatLng sydney = new LatLng(33.712382, 73.036899);  
// // sydney.showInfoWindow();  
// gMap.addMarker(new MarkerOptions().position(sydney).title("F8 Markaz"));  
// gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(sydney, 10));  
//  
// LatLng sydney2 = new LatLng(33.5914237, 73.0535122);  
// gMap.addMarker(new MarkerOptions().position(sydney2).title("Sadar"));  
// gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(sydney2, 10));  
//  
  
// LatLng sydney3 = new LatLng(33.712382, 73.036899);  
// gMap.addMarker(new MarkerOptions().position(sydney3).title("G8 Markaz"));  
// gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(sydney3, 10));* Button b = (Button)findViewById(R.id.back);  
 b.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
  
 Intent intent = **new** Intent(Pickup.**this**,Driver\_Selected\_Rides.**class**);  
 startActivity(intent);  
  
 }  
 });  
}}

**public class** ReadAgreement **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_read\_agreement);  
 }  
}

**public class** Rider\_Maps **extends** FragmentActivity **implements** OnMapReadyCallback, GoogleApiClient.ConnectionCallbacks,com.google.android.gms.location.LocationListener {  
 Location mlocation;  
 FusedLocationProviderClient fusedLocationProviderClient;  
 **private** View mapView;  
 *// Search View Code* GoogleMap map;  
 **private** GoogleApiClient Api;  
 *// SearchView searchView;* SupportMapFragment mapFragment;  
 **private** Button request;  
  
 Spinner spinner;  
 **private static final int** Request\_Code=101;  
  
  
  
  
  
 *// Create a new Places client instance.  
 //PlacesClient placesClient = Places.createClient(this);* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_rider\_\_maps);  
 *// spinner = (Spinner)findViewById(R.id.spinner1);  
  
  
 // Search View Code  
  
  
  
  
 // searchView = (SearchView)findViewById(R.id.location);* mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);  
*// searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
// @Override  
// public boolean onQueryTextSubmit(String s) {  
// String location = searchView.getQuery().toString();  
//// searchView.setQuery(text,false);  
// searchView.clearFocus();  
// List<Address> addressList = null;  
// if (location!=null||!location.equals("")){  
// Geocoder geocoder = new Geocoder(Rider\_Maps.this);  
// try {  
// addressList=geocoder.getFromLocationName(location,1);  
// } catch (IOException e) {  
// e.printStackTrace();  
// }  
// Address address = addressList.get(0);  
// LatLng latLng = new LatLng(address.getLatitude(),address.getLongitude());  
//  
// map.addMarker(new MarkerOptions().position(latLng).title(location));  
// map.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,10));  
// }  
// return false;  
// }  
//  
// @Override  
// public boolean onQueryTextChange(String s) {  
// return false;  
// }  
// });  
 // mapFragment.getMapAsync(this);* mapView = mapFragment.getView();  
 fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(**this**);  
 GetlastLocation();  
  
 }  
  
  
  
 **private void** GetlastLocation() {  
 **if** (ActivityCompat.checkSelfPermission(**this**, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {  
 ActivityCompat.requestPermissions(**this**, **new** String[]  
 {Manifest.permission.ACCESS\_FINE\_LOCATION}, Request\_Code);  
 **return**;  
 }  
 Task<Location> task = fusedLocationProviderClient.getLastLocation();  
 task.addOnSuccessListener(**new** OnSuccessListener<Location>() {  
 @Override  
 **public void** onSuccess(Location location) {  
 **if** (location != **null**) {  
 mlocation = location;  
 Toast.makeText(getApplicationContext(), mlocation.getLatitude() + **""** + mlocation.getLongitude(),  
 Toast.LENGTH\_SHORT).show();  
 SupportMapFragment supportMapFragment=(SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map); supportMapFragment.getMapAsync(Rider\_Maps.**this**);  
 }  
 }  
 });  
  
 request = (Button) findViewById(R.id.search\_driver);  
 request.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **final** String Uid = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 DatabaseReference ref = FirebaseDatabase.getInstance().getReference(**"RidesAvailable"**);  
*// // DatabaseReference data = FirebaseDatabase.getInstance().getReference("User");  
////* DatabaseReference zonesRef = FirebaseDatabase.getInstance().getReference(**"User"**);  
 DatabaseReference zone1Ref = zonesRef.child(Uid);  
 DatabaseReference name = zone1Ref.child(**"name"**);  
 DatabaseReference number = zone1Ref.child(**"number"**);  
  
  
  
 number.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 DatabaseReference ref = FirebaseDatabase.getInstance().getReference(**"RidesAvailable"**);  
 ref.child(Uid).child(**"Number"**).setValue(dataSnapshot.getValue(String.**class**));  
  
 }  
  
 @Override  
 **public void** onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
  
 name.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(DataSnapshot snapshot) {  
 DatabaseReference ref = FirebaseDatabase.getInstance().getReference(**"RidesAvailable"**);  
 *// System.out.println(snapshot.getValue()); //prints "Do you have data? You'll love Firebase."  
 //DataSnapshot value = new DataSnapshot().getValue();  
 // Log.i("SAmi", snapshot.getValue(String.class));* ref.child(Uid).child(**"Name"**).setValue(snapshot.getValue(String.**class**));  
 *// final String text = spinner.getSelectedItem().toString();  
 // ref.child(Uid).child("PickUp\_Point").setValue(text);* ref.child(Uid).child(**"Key"**).setValue(Uid);  
  
  
 }  
  
 @Override  
 **public void** onCancelled(DatabaseError databaseError) {  
 }  
  
  
 });  
 DatabaseReference des = FirebaseDatabase.getInstance().getReference(**"Destination"**);  
 DatabaseReference des1 = des.child(Uid);  
 *// DatabaseReference des2 = des1.child("Destination");* des1.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 String Des2 = dataSnapshot.child(**"Destination"**).getValue().toString();  
 String Des3 = dataSnapshot.child(**"Vehicle\_Type"**).getValue().toString();  
 String Des4 = dataSnapshot.child(**"PickUp\_Point"**).getValue().toString();  
 DatabaseReference ref = FirebaseDatabase.getInstance().getReference(**"RidesAvailable"**);  
 ref.child(Uid).child(**"Destination"**).setValue(Des2);  
 ref.child(Uid).child(**"Vehicle\_Type"**).setValue(Des3);  
 ref.child(Uid).child(**"PickUp\_Point"**).setValue(Des4);  
  
  
  
 }  
  
 @Override  
 **public void** onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 GeoFire geoFire = **new** GeoFire(ref);  
 *// geoFire.setLocation(Uid,new GeoLocation(location.getLatitude(),location.getLongitude()));* geoFire.setLocation(Uid,**new** GeoLocation(mlocation.getLatitude(), mlocation.getLongitude()), **new** GeoFire.CompletionListener() {  
 @Override  
 **public void** onComplete(String key, DatabaseError error) {  
 **if** (error!=**null**)  
 {  
 Toast.makeText(Rider\_Maps.**this**,**"Can't go Active"**,Toast.LENGTH\_SHORT).show();  
 }  
 Toast.makeText(Rider\_Maps.**this**,**"You are Active"**,Toast.LENGTH\_SHORT).show();  
 }  
 });  
  
*// String [] destination = Rider\_Maps.this.getResources().getStringArray(R.array.);  
// AlertDialog.Builder builder = new AlertDialog.Builder(Rider\_Maps.this);  
// builder.setTitle("Pick up your Destination");  
// builder.setSingleChoiceItems(R.arr)* Intent intent = **new** Intent(Rider\_Maps.**this**,rider\_waiting.**class**);  
 startActivity(intent);  
  
 }  
 });  
 }  
  
  
 @Override  
 **public void** onMapReady(GoogleMap googleMap) {  
 map = googleMap;  
 map.setMyLocationEnabled(**true**);  
 map.getUiSettings().setMyLocationButtonEnabled(**true**);  
  
 **if** (mapView != **null** && mapView.findViewById(Integer.parseInt(**"1"**)) != **null**) {  
 View locationButton = ((View) mapView.findViewById(Integer.parseInt(**"1"**)).getParent()).findViewById(Integer.parseInt(**"2"**));  
 RelativeLayout.LayoutParams layoutParams = (RelativeLayout.LayoutParams) locationButton.getLayoutParams();  
 layoutParams.addRule(RelativeLayout.ALIGN\_PARENT\_TOP, 0);  
 layoutParams.addRule(RelativeLayout.ALIGN\_PARENT\_BOTTOM, RelativeLayout.TRUE);  
 layoutParams.setMargins(0, 0, 40, 250);  
 }  
  
  
 *//map = googleMap;* LocationRequest locationRequest = LocationRequest.create();  
 locationRequest.setInterval(10000);  
 locationRequest.setFastestInterval(5000);  
 locationRequest.setPriority(LocationRequest.PRIORITY\_HIGH\_ACCURACY);  
  
 LatLng latLng = **new** LatLng(mlocation.getLatitude(),mlocation.getLongitude());  
*// String address = getAddress(this);* MarkerOptions markerOptions=**new** MarkerOptions().position(latLng).title(**"Current Location"**);  
 googleMap.animateCamera(CameraUpdateFactory.newLatLng(latLng));  
 googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,15));  
 googleMap.addMarker(markerOptions);  
 buildGoogleApiClient();  
  
 String Uid2 = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 DatabaseReference d= FirebaseDatabase.getInstance().getReference(**"Destination"**);  
 DatabaseReference d2 = d.child(Uid2);  
  
  
 d2.addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(@NonNull DataSnapshot dataSnapshot) {  
  
 String Des4 = dataSnapshot.child(**"PickUp\_Point"**).getValue().toString();  
  
 **if**(Des4.equals(**"Sadar"**)){  
 LatLng sadar = **new** LatLng(33.5914237, 73.0535122);  
 map.addMarker(**new** MarkerOptions().position(sadar).title(**"Sadar"**));  
 map.animateCamera(CameraUpdateFactory.newLatLngZoom(sadar, 15));  
  
 }**else if**(Des4.equals(**"F8 Markaz"**)){  
 LatLng f8 = **new** LatLng(33.712382, 73.036899);  
 *// sydney.showInfoWindow();* map.addMarker(**new** MarkerOptions().position(f8).title(**"F8 Markaz"**));  
 map.animateCamera(CameraUpdateFactory.newLatLngZoom(f8, 15));  
  
 }  
 **else if**(Des4.equals(**"G9 Markaz"**)) {  
 LatLng g9 = **new** LatLng(33.690036, 73.030187);  
 map.addMarker(**new** MarkerOptions().position(g9).title(**"G9 Markaz"**));  
 map.animateCamera(CameraUpdateFactory.newLatLngZoom(g9, 15));  
 }  
 **else** {  
 LatLng Air = **new** LatLng(33.713818, 73.026399);  
 map.addMarker(**new** MarkerOptions().position(Air).title(**"Air University"**));  
 map.animateCamera(CameraUpdateFactory.newLatLngZoom(Air, 15));  
  
 }  
  
  
  
 }  
  
 @Override  
 **public void** onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
  
  
  
  
  
  
 }  
  
 **private void** buildGoogleApiClient() {  
 Api= **new** GoogleApiClient.Builder(**this**)  
 .addConnectionCallbacks(**this**).addApi(LocationServices.API).build();  
 Api.connect();  
 }  
  
  
 *// Get Address from lat long* **public** String getAddress(Context ctx){  
 String fullAdd=**null**;  
 **try**{  
 Geocoder geocoder = **new** Geocoder(ctx, Locale.getDefault());  
 List<android.location.Address> addresses = geocoder.getFromLocation(mlocation.getLatitude(),mlocation.getLongitude(),1);  
 **if** (addresses.size()>0){  
 Address address = addresses.get(0);  
 fullAdd = address.getAddressLine(0);  
 }  
 }**catch** (IOException ex){  
 ex.printStackTrace();  
 }  
  
 **return** fullAdd;  
  
 }  
  
 @Override  
 **public void** onRequestPermissionsResult(**int** requestCode, @NonNull String[] permissions, @NonNull **int**[] grantResults) {  
 **switch** (requestCode){  
 **case** Request\_Code:  
 **if** (grantResults.length>0 && grantResults[0]== PackageManager.PERMISSION\_GRANTED){  
 GetlastLocation();  
 }  
 **break**;  
 }  
 }  
  
  
  
 @Override  
 **public void** onConnected(@Nullable Bundle bundle) {  
 LocationRequest locationRequest = **new** LocationRequest();  
 locationRequest.setInterval(10000);  
 locationRequest.setFastestInterval(5000);  
 locationRequest.setPriority(LocationRequest.PRIORITY\_HIGH\_ACCURACY);  
  
 LocationServices.FusedLocationApi.requestLocationUpdates(Api,locationRequest,**this**);  
  
  
 }  
  
 @Override  
 **public void** onConnectionSuspended(**int** i) {  
  
 }  
  
 @Override  
 **public void** onLocationChanged(Location location) {  
  
  
 }  
}

**public class** Rider\_Select\_Ride\_Type **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_rider\_\_select\_\_ride\_\_type);  
 getSupportActionBar().hide();  
  
 **final** DatabaseReference ref = FirebaseDatabase.getInstance().getReference(**"Destination"**);  
 **final** FirebaseUser UId = FirebaseAuth.getInstance().getCurrentUser();  
 *// final String user = UId.getUid();* **final** Button button = (Button) findViewById(R.id.SelectType);  
  
 Spinner spinner = (Spinner)findViewById(R.id.spinner1);  
 **final** Spinner spinner2 = (Spinner)findViewById(R.id.spinner2);  
 String text = spinner.getSelectedItem().toString();  
 String text2 = spinner2.getSelectedItem().toString();  
*// if(text.equals("Select Destination")){  
// button.setVisibility(View.VISIBLE);  
// }else {  
// button.  
// button.setVisibility(View.VISIBLE);  
// }  
  
  
 // Toast.makeText(Rider\_Select\_Ride\_Type.this,text,Toast.LENGTH\_LONG).show();* **final** Switch s = (Switch) findViewById(R.id.UserTypeSwitch);  
  
  
*// String sa = "Select Destination";  
// if(text == sa){  
// button.setClickable(false);  
// }else {  
// button.setEnabled(true);  
//  
// }* button.setOnClickListener(**new** View.OnClickListener(){  
 **public void** onClick(View v){  
  
  
 Spinner spinner = (Spinner)findViewById(R.id.spinner1);  
 String text = spinner.getSelectedItem().toString();  
 String text2 = spinner2.getSelectedItem().toString();  
  
*// if(text.equals("Select Destination")){  
// button.setVisibility(View.INVISIBLE);  
// // Toast.makeText(Rider\_Select\_Ride\_Type.this,"SAmi",Toast.LENGTH\_LONG).show();  
// }else{  
// button.setVisibility(View.VISIBLE);  
 // Toast.makeText(Rider\_Select\_Ride\_Type.this,text,Toast.LENGTH\_LONG).show();* ref.child(UId.getUid()).child(**"Destination"**).setValue(text);  
 ref.child(UId.getUid()).child(**"PickUp\_Point"**).setValue(text2);  
  
 **if**(s.isChecked()){  
 ref.child(UId.getUid()).child(**"Vehicle\_Type"**).setValue(**"Car"**);  
  
 }**else**{  
 ref.child(UId.getUid()).child(**"Vehicle\_Type"**).setValue(**"Bike"**);  
 }  
  
  
 *//Toast.makeText(Rider\_Select\_Ride\_Type.this,text,Toast.LENGTH\_LONG).show();* Intent intent = **new** Intent(Rider\_Select\_Ride\_Type.**this**, Rider\_Maps.**class**);  
 startActivity(intent);  
  
 *// Toast.makeText(MainActivity.this,"Welcome", Toast.LENGTH\_LONG).show();* }*//}* });  
 }  
}

**public class** Rider\_Summary **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_rider\_\_summary);  
 getSupportActionBar().hide();  
 }  
}

**public class** rider\_waiting **extends** AppCompatActivity {  
  
  
 *// private static int TIME\_OUT = 10000;  
  
// RecyclerView recyclerView;  
// RecyclerView.LayoutManager layoutManager;* ImageView im;  
 TextView la,dname,dnumber,vnumber;  
  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_rider\_waiting);  
 getSupportActionBar().hide();  
  
 **final** FirebaseDatabase database = FirebaseDatabase.getInstance();  
 **final** DatabaseReference myRef = database.getReference(**"Connected\_Rides"**);  
  
 **final** String Uid = FirebaseAuth.getInstance().getCurrentUser().getUid();  
 *// DatabaseReference ref = myRef.child(Uid);  
  
// recyclerView= (RecyclerView) findViewById(R.id.selected);  
// recyclerView.setHasFixedSize(true);  
// layoutManager = new LinearLayoutManager(this);  
// recyclerView.setLayoutManager(layoutManager);* im = (ImageView) findViewById(R.id.wait);  
 la = (TextView) findViewById(R.id.label);  
 dname = (TextView) findViewById(R.id.dname);  
  
 dnumber = (TextView) findViewById(R.id.number2);  
*// dnumber.setOnClickListener(new View.OnClickListener() {  
// @Override  
// public void onClick(View v) {  
// Intent intent = new Intent(Intent.ACTION\_DIAL);  
// intent.setData(Uri.parse(dnumber.getText().toString()));  
// startActivity(intent);  
//  
// }  
// });* vnumber= (TextView)findViewById(R.id.Vehicle\_number);  
  
 myRef.child(Uid).addValueEventListener(**new** ValueEventListener() {  
 @Override  
 **public void** onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 String name = dataSnapshot.child(**"Driver\_Name"**).getValue(String.**class**);  
 **final** String number = dataSnapshot.child(**"Driver\_Number"**).getValue(String.**class**);  
 String v\_n =dataSnapshot.child(**"Vehicle\_Number"**).getValue(String.**class**);  
  
*// if(name== null) {  
//  
// }else {* dname.setText( name);  
 dname.setCompoundDrawablesWithIntrinsicBounds(R.drawable.ic\_person\_white\_24dp, 0, 0, 0);  
 dnumber.setText(number);  
 dnumber.setCompoundDrawablesWithIntrinsicBounds(R.drawable.ic\_phone\_white\_24dp, 0, 0, 0);  
 vnumber.setText( v\_n);  
 *// vnumber.setCompoundDrawablesWithIntrinsicBounds(R.drawable.nu, 0, 0, 0);* la.setText(**"Driver Information"**);  
 im.setVisibility(View.INVISIBLE);  
 }  
  
  
  
  
  
  
 @Override  
 **public void** onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 Button b = (Button)findViewById(R.id.end);  
 b.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 DatabaseReference r = FirebaseDatabase.getInstance().getReference(**"Destination"**);  
 r.child(Uid).removeValue();  
 myRef.child(Uid).removeValue();  
*// la.setText("Waiting for Driver To Response");  
// im.setVisibility(View.VISIBLE);* Intent intent = **new** Intent(rider\_waiting.**this**,Rider\_Summary.**class**);  
 startActivity(intent);  
 }  
 });  
 }}  
  
  
  
  
  
  
  
  
  
  
  
*// new Handler().postDelayed(new Runnable() {  
// @Override  
// public void run() {  
// Intent i = new Intent(rider\_waiting.this, Rider\_Summary.class);  
// startActivity(i);  
// finish();  
// }  
// }, TIME\_OUT);  
// }*

**public class** SelectedRides {  
  
 **private** String Driver\_Name, Driver\_Number;  
 **private** String Rider\_Name, Rider\_Number;  
 **private** String Rider\_Destination, Rider\_Pickup;  
  
 **public** SelectedRides() {  
 }  
  
 **public** SelectedRides(String driver\_Name, String driver\_Number, String rider\_Name, String rider\_Number, String rider\_Destination, String rider\_Pickup) {  
 Driver\_Name = driver\_Name;  
 Driver\_Number = driver\_Number;  
 Rider\_Name = rider\_Name;  
 Rider\_Number = rider\_Number;  
 Rider\_Destination = rider\_Destination;  
 Rider\_Pickup = rider\_Pickup;  
 }  
  
 **public** String getDriver\_Name() {  
 **return** Driver\_Name;  
 }  
  
 **public void** setDriver\_Name(String driver\_Name) {  
 Driver\_Name = driver\_Name;  
 }  
  
 **public** String getDriver\_Number() {  
 **return** Driver\_Number;  
 }  
  
 **public void** setDriver\_Number(String driver\_Number) {  
 Driver\_Number = driver\_Number;  
 }  
  
 **public** String getRider\_Name() {  
 **return** Rider\_Name;  
 }  
  
 **public void** setRider\_Name(String rider\_Name) {  
 Rider\_Name = rider\_Name;  
 }  
  
 **public** String getRider\_Number() {  
 **return** Rider\_Number;  
 }  
  
 **public void** setRider\_Number(String rider\_Number) {  
 Rider\_Number = rider\_Number;  
 }  
  
 **public** String getRider\_Destination() {  
 **return** Rider\_Destination;  
 }  
  
 **public void** setRider\_Destination(String rider\_Destination) {  
 Rider\_Destination = rider\_Destination;  
 }  
  
 **public** String getRider\_Pickup() {  
 **return** Rider\_Pickup;  
 }  
  
 **public void** setRider\_Pickup(String rider\_Pickup) {  
 Rider\_Pickup = rider\_Pickup;  
 }  
}

**public class** SignIn **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_sign\_in);  
  
 getSupportActionBar().hide();  
  
  
  
  
 *// Switch Button Rider or Driver* Button buttonS = (Button) findViewById(R.id.Switch);  
  
 buttonS.setOnClickListener(**new** View.OnClickListener(){  
 Switch userTypeSwitch = (Switch) findViewById(R.id.UserTypeSwitch);  
 **public void** onClick(View v){  
 **if** (userTypeSwitch.isChecked()) {  
  
 Intent intentR = **new** Intent(SignIn.**this**, Rider\_Select\_Ride\_Type.**class**);  
 startActivity(intentR);  
 Toast.makeText(SignIn.**this**,**"Rider"**, Toast.LENGTH\_LONG).show();  
  
 }  
 **else** {  
 Intent intentD = **new** Intent(SignIn.**this**, Driver.**class**);  
 startActivity(intentD);  
 Toast.makeText(SignIn.**this**,**"Driver"**, Toast.LENGTH\_LONG).show();  
  
 }  
 }  
  
 });  
  
  
  
  
  
 }  
}

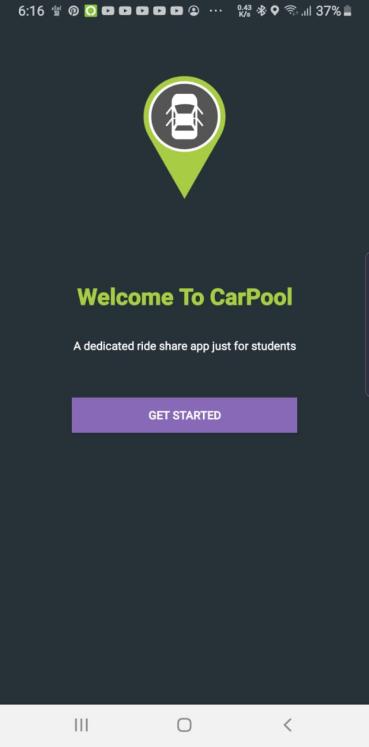
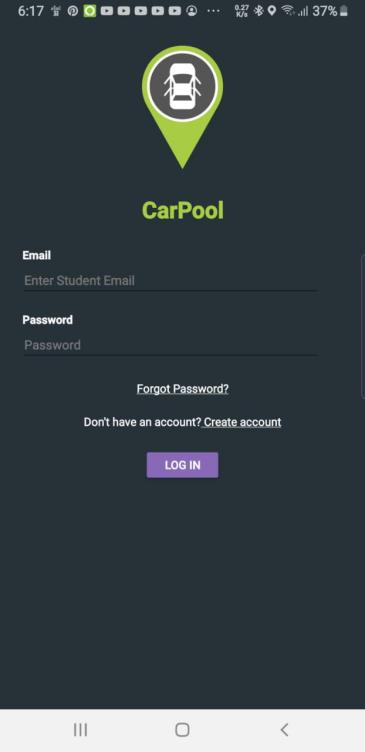
**public class** SignUp **extends** AppCompatActivity {  
  
 EditText emailId, passwd,name,id,number;  
 Button btnSignUp;  
  
 FirebaseAuth firebaseAuth;  
 **private** DatabaseReference ref;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_sign\_up);  
 getSupportActionBar().hide();  
  
 *//Fire Base COde* firebaseAuth = FirebaseAuth.getInstance();  
 emailId = (EditText) findViewById(R.id.EtEmailS);  
 passwd = (EditText) findViewById(R.id.EtPassS);  
 name = (EditText) findViewById(R.id.editText3);  
 id = (EditText) findViewById(R.id.editText5);  
 number = (EditText) findViewById(R.id.editText7);  
 btnSignUp = findViewById(R.id.Submit);  
 **final** CreateAccount account = **new** CreateAccount();  
 ref= FirebaseDatabase.getInstance().getReference().child(**"User"**);  
  
  
  
 btnSignUp.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 *//String names = name.getText().toString();  
 //int ids = Integer.parseInt(id.getText().toString().trim());  
 //long numbers = Long.parseLong(number.getText().toString().trim());* String emailID = emailId.getText().toString().trim();  
 String paswd = passwd.getText().toString();  
 String emailPattern = **"[0-9]+@[s]+[t]+[u]+[d]+[e]+[n]+[t][s]+\\.+[a]+[u]+\\.+[e]+[d]+[u]+\\.+[p]+[k]+"**;  
 account.setName(name.getText().toString().trim());  
 account.setId(id.getText().toString().trim());  
 account.setNumber(number.getText().toString().trim());  
 account.setEmail(emailID);  
 *// ref.push().setValue(account);  
 // Toast.makeText(SignUp.this,"SignUP Succesfull",Toast.LENGTH\_LONG).show();* **if** (emailID.isEmpty()) {  
 emailId.setError(**"Provide your Email first!"**);  
 emailId.requestFocus();  
 } **else if** (paswd.isEmpty()) {  
 passwd.setError(**"Set your password"**);  
 passwd.requestFocus();  
 } **else if** (emailID.isEmpty() && paswd.isEmpty()) {  
 Toast.makeText(SignUp.**this**, **"Fields Empty!"**, Toast.LENGTH\_SHORT).show();  
 } **else if** (!(emailID.isEmpty() && paswd.isEmpty()) && emailID.matches(emailPattern)) {  
 firebaseAuth.createUserWithEmailAndPassword(emailID, paswd).addOnCompleteListener(SignUp.**this**, **new** OnCompleteListener() {  
 @Override  
 **public void** onComplete(@NonNull Task task) {  
 FirebaseUser user = firebaseAuth.getCurrentUser();  
  
  
 **if** (!task.isSuccessful()) {  
  
 Toast.makeText(SignUp.**this**.getApplicationContext(),  
 **"SignUp unsuccessful: "** + task.getException().getMessage(),  
 Toast.LENGTH\_SHORT).show();  
  
 }  
 **else** {  
 sendVerificationEmail();  
  
 startActivity(**new** Intent(SignUp.**this**, LogIn.**class**));  
 Toast.makeText(SignUp.**this**, **"Verification Email Sent"**, Toast.LENGTH\_SHORT).show();  
 *// ref. .setValue(account);* ref.child(user.getUid()).setValue(account);  
  
 }  
 }  
 });  
 } **else** {  
 Toast.makeText(SignUp.**this**, **"Error: Enter University Email "**, Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
  
 *// Read Agreement Button* TextView textView = (TextView) findViewById(R.id.ReadAgreement);  
  
  
 textView.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent(SignUp.**this**,ReadAgreement.**class**);  
 startActivity(intent);  
 Toast.makeText(SignUp.**this**,**"Read Agreement"**, Toast.LENGTH\_LONG).show();  
 }  
 });  
  
 }  
  
 **private void** sendVerificationEmail() {  
 {  
 FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();  
  
 user.sendEmailVerification()  
 .addOnCompleteListener(**new** OnCompleteListener<Void>() {  
 @Override  
 **public void** onComplete(@NonNull Task<Void> task) {  
 **if** (task.isSuccessful()) {  
 *// email sent  
  
  
 // after email is sent just logout the user and finish this activity* FirebaseAuth.getInstance().signOut();  
 *// startActivity(new Intent(SignUp.this, SignIn.class));* finish();  
 }  
 **else** {  
 *// email not sent, so display message and restart the activity or do whatever you wish to do  
  
 //restart this activity* overridePendingTransition(0, 0);  
 finish();  
 overridePendingTransition(0, 0);  
 startActivity(getIntent());  
  
 }  
 }  
 });  
 }  
  
 }  
}

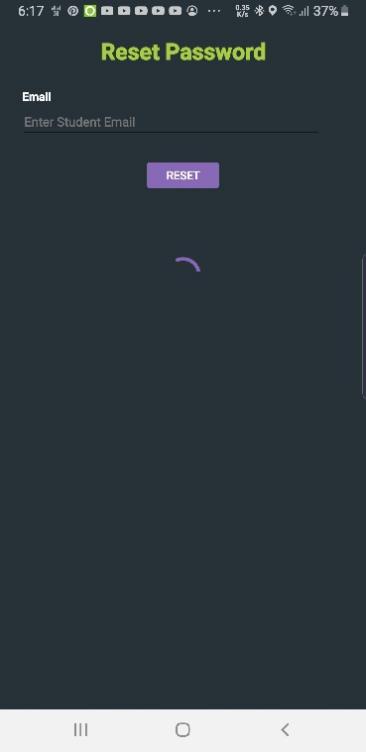
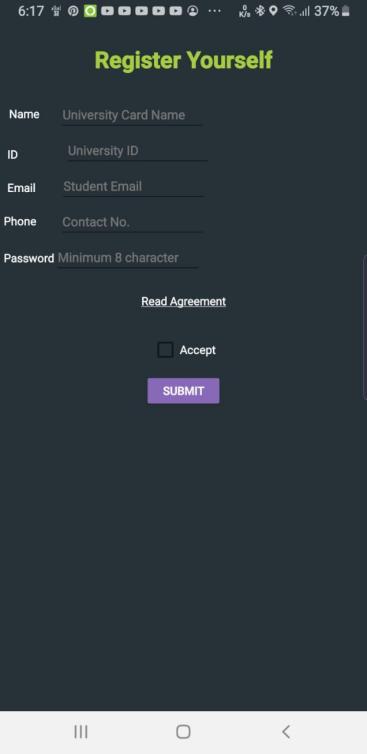
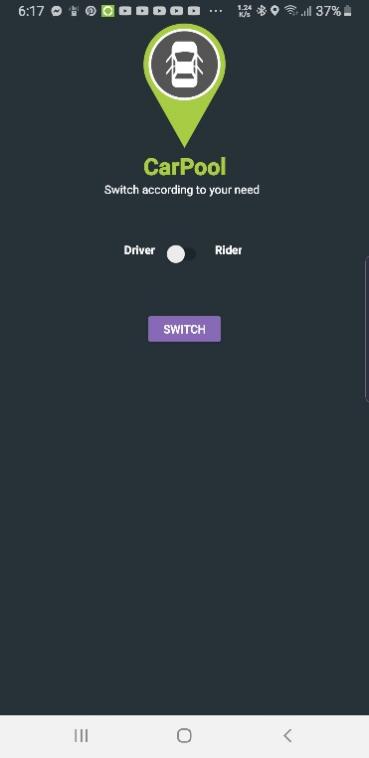
**public class** SignUp **extends** AppCompatActivity {  
  
 EditText emailId, passwd,name,id,number;  
 Button btnSignUp;  
  
 FirebaseAuth firebaseAuth;  
 **private** DatabaseReference ref;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_sign\_up);  
 getSupportActionBar().hide();  
  
 *//Fire Base COde* firebaseAuth = FirebaseAuth.getInstance();  
 emailId = (EditText) findViewById(R.id.EtEmailS);  
 passwd = (EditText) findViewById(R.id.EtPassS);  
 name = (EditText) findViewById(R.id.editText3);  
 id = (EditText) findViewById(R.id.editText5);  
 number = (EditText) findViewById(R.id.editText7);  
 btnSignUp = findViewById(R.id.Submit);  
 **final** CreateAccount account = **new** CreateAccount();  
 ref= FirebaseDatabase.getInstance().getReference().child(**"User"**);  
  
  
  
 btnSignUp.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 *//String names = name.getText().toString();  
 //int ids = Integer.parseInt(id.getText().toString().trim());  
 //long numbers = Long.parseLong(number.getText().toString().trim());* String emailID = emailId.getText().toString().trim();  
 String paswd = passwd.getText().toString();  
 String emailPattern = **"[0-9]+@[s]+[t]+[u]+[d]+[e]+[n]+[t][s]+\\.+[a]+[u]+\\.+[e]+[d]+[u]+\\.+[p]+[k]+"**;  
 account.setName(name.getText().toString().trim());  
 account.setId(id.getText().toString().trim());  
 account.setNumber(number.getText().toString().trim());  
 account.setEmail(emailID);  
 *// ref.push().setValue(account);  
 // Toast.makeText(SignUp.this,"SignUP Succesfull",Toast.LENGTH\_LONG).show();* **if** (emailID.isEmpty()) {  
 emailId.setError(**"Provide your Email first!"**);  
 emailId.requestFocus();  
 } **else if** (paswd.isEmpty()) {  
 passwd.setError(**"Set your password"**);  
 passwd.requestFocus();  
 } **else if** (emailID.isEmpty() && paswd.isEmpty()) {  
 Toast.makeText(SignUp.**this**, **"Fields Empty!"**, Toast.LENGTH\_SHORT).show();  
 } **else if** (!(emailID.isEmpty() && paswd.isEmpty()) && emailID.matches(emailPattern)) {  
 firebaseAuth.createUserWithEmailAndPassword(emailID, paswd).addOnCompleteListener(SignUp.**this**, **new** OnCompleteListener() {  
 @Override  
 **public void** onComplete(@NonNull Task task) {  
 FirebaseUser user = firebaseAuth.getCurrentUser();  
  
  
 **if** (!task.isSuccessful()) {  
  
 Toast.makeText(SignUp.**this**.getApplicationContext(),  
 **"SignUp unsuccessful: "** + task.getException().getMessage(),  
 Toast.LENGTH\_SHORT).show();  
  
 }  
 **else** {  
 sendVerificationEmail();  
  
 startActivity(**new** Intent(SignUp.**this**, LogIn.**class**));  
 Toast.makeText(SignUp.**this**, **"Verification Email Sent"**, Toast.LENGTH\_SHORT).show();  
 *// ref. .setValue(account);* ref.child(user.getUid()).setValue(account);  
  
 }  
 }  
 });  
 } **else** {  
 Toast.makeText(SignUp.**this**, **"Error: Enter University Email "**, Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
  
 *// Read Agreement Button* TextView textView = (TextView) findViewById(R.id.ReadAgreement);  
  
  
 textView.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Intent intent = **new** Intent(SignUp.**this**,ReadAgreement.**class**);  
 startActivity(intent);  
 Toast.makeText(SignUp.**this**,**"Read Agreement"**, Toast.LENGTH\_LONG).show();  
 }  
 });  
  
 }  
  
 **private void** sendVerificationEmail() {  
 {  
 FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();  
  
 user.sendEmailVerification()  
 .addOnCompleteListener(**new** OnCompleteListener<Void>() {  
 @Override  
 **public void** onComplete(@NonNull Task<Void> task) {  
 **if** (task.isSuccessful()) {  
 *// email sent  
  
  
 // after email is sent just logout the user and finish this activity* FirebaseAuth.getInstance().signOut();  
 *// startActivity(new Intent(SignUp.this, SignIn.class));* finish();  
 }  
 **else** {  
 *// email not sent, so display message and restart the activity or do whatever you wish to do  
  
 //restart this activity* overridePendingTransition(0, 0);  
 finish();  
 overridePendingTransition(0, 0);  
 startActivity(getIntent());  
  
 }  
 }  
 });  
 }  
  
 }  
}

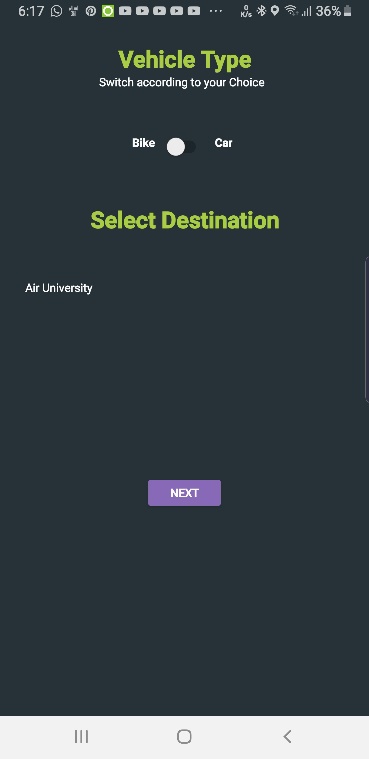
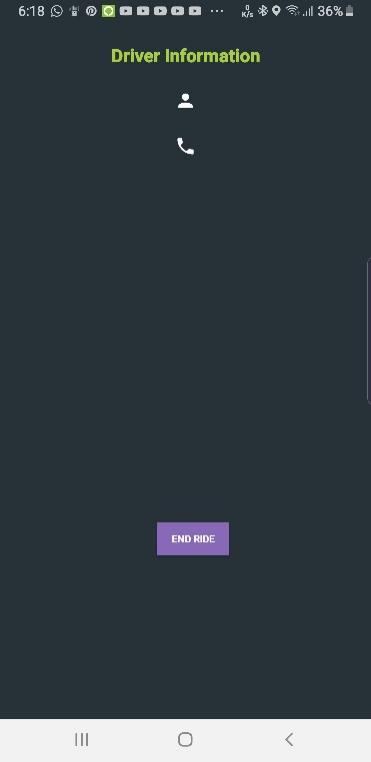
**public class** User {  
  
 **private** String Name, Number;  
 **private** String Destination, PickUp\_Point;  
 **private** String Key;  
  
 **public** User() {  
 }  
  
 **public** User(String name, String number, String destination, String pickUp\_Point, String key) {  
 Name = name;  
 Number = number;  
 Destination = destination;  
 PickUp\_Point = pickUp\_Point;  
 Key = key;  
 }  
  
 **public** String getName() {  
 **return** Name;  
 }  
  
 **public void** setName(String name) {  
 Name = name;  
 }  
  
 **public** String getNumber() {  
 **return** Number;  
 }  
  
 **public void** setNumber(String number) {  
 Number = number;  
 }  
  
 **public** String getDestination() {  
 **return** Destination;  
 }  
  
 **public void** setDestination(String destination) {  
 Destination = destination;  
 }  
  
 **public** String getPickUp\_Point() {  
 **return** PickUp\_Point;  
 }  
  
 **public void** setPickUp\_Point(String pickUp\_Point) {  
 PickUp\_Point = pickUp\_Point;  
 }  
  
 **public** String getKey() {  
 **return** Key;  
 }  
  
 **public void** setKey(String key) {  
 Key = key;  
 }  
}

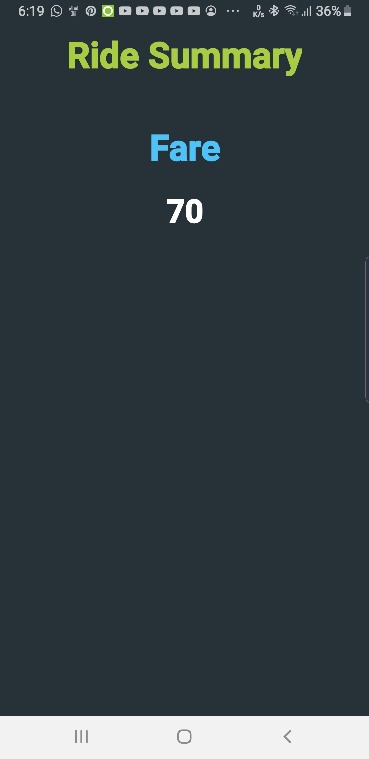
### User Interface

#### Interface Screenshots



### Chapter Summary

# 

## TESTING

### Introduction

Define as test objectives, quality goals, and entrance and exit criteria.

### Testing Methods

Table 7. 1: Testing Method

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |
| **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 |

### Software Evaluation

#### Testing Strategy

Table 7. 2: Testing Strategy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |
| **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 |

#### Test Plans

Table 7. 3: Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |

#### Test Cases

Table 7. 4: Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |

### Chapter Summary

# REFERENCES

|  |  |
| --- | --- |
| [1] | OwanaMarziaMoushi, Mostofa Kamal, Mahmuda Haque and Md. Shamim,“Design and Development of an Online Bus Monitoring System”, 10th International Conference on Electrical and Computer Engineering 20-22 December, 2018, Dhaka, Bangladesh |
| [2] | K Sujatha, K J Sruthi, P V Nageswara Rao and A Arjun Rao, “Design and Development of Android Mobile Bus Tracking System”, 2014 First International Conference on Networks & Soft Computing, |
| [3] | globalmissing.org, “Missing children’s statistics” |

## APPENDIX (Optional)