A

PROJECT REPORT

ON

"RIDE SHARE"

SUBMITTED BY

HARSH PANDYA

VIRAJ BHAKTA

ABSTRACT

Ride Share is also known as car sharing, carpooling or lift sharing. Ride Share is an android based application that help sharing car journeys so that more than one person can travel in same car. By having more than one vehicle , Ride sharing can help in reducing the travel costs of every person such as fuel costs , tolls , stress of driving , and parking spaces.

The purpose of the application is to provide a platform where people will able to find car journeys that are suitable for them. Ride Share is an application that is more environmentally friendly and cheap way of travelling during high fuel prices and high pollution periods. Also by using these application users can help reduce amount of co_2 emission due to cars.

Our application provides user to post a ride for there journeys and others users will able to see the posted rides. Users will be able to communicate with other users through the chat box provided in the application. User will be able to edit and delete there individual posted rides.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

In today's world everyone is migrating from one place to another for many reasons. Many people uses private car transportation that makes more traffic on roads which generates air pollution. With the increase of environmental concerns and congestion of roads, Ride Sharing can be useful as it is environment friendly and cheap way of travelling. Ride Sharing is a sharing of car when two or more persons share a ride in one of there personal vehicle.

Ride Sharing is a solution to this problem by filling the vacant and unused seats in private vehicles. When passengers are from same area then communication is not a big issues but when area is not same then one cannot aware of a passenger for same destination. Ride Sharing is an application in finding people and their journey agendas and make an informed decision about you want to travel alone or travel with other peoples that helps in saving journey costs.

Ride Sharing can be advantageous for many people like it is economical i.e. travel expenses are shared among the riders and also it can help in reducing the pollution since we have less car on the road. It also help in reducing parking requirements resulting in more efficient land use and also helps in minimizing cost of building and maintain the infrastructure.

Our application aims to make a system which is user friendly and provides an opportunity to share cars with other peoples. We focused on making a system which would help users to post their rides ,edit and delete their own posted rides. And others users can see rides that will be useful for them for travelling in same direction. Both riders and drivers can communicate with each other through chat box and discuss with each other about their common meeting point in chat box and can set prices with their each other through chat box.

Features/Functionalities:

- User Sign-In/Register
- Update personal Information
- Post Rides to Destination
- Get(Book) Rides to Specific Destination
- Search Rides
- Communication between riders and drivers through chat box.

PROBLEM DESCRIPTION

Many people uses private car transportation that can many adverse effects on environment. Ride Sharing is a solution but problems like security and trust comes. Can this problem be solved? Solution to this problem is an android based Ride Sharing Application that provide user safe and secure way to involve in car sharing.

Literature Survey

There are several definitions of carpooling. Car-pooling is a collaborative use of car by at least two people for certain rides with primary purpose of gaining profit. A motor vehicle has a variable capacity ranging from 5 to 9 seats including driver and travel cost may be shared among the carpoolers.

Overview:

We have picked this topic because we believe that lesser traffic in our city and most likely all other cities in the world is need of hour. It has a general appeal to it because it would combine software to people's daily lives. This topic is what we thought as real world application but with basic notion of helping the society itself. The most basic reason why we wanted this project is because it will run in real time and connect people to other people who travel to the same route.

OVERVIEW OF THE TECHNOLOGIES USED

Basic Terminology

This section gives brief introduction of all the terminology used in this report.

Activity –

Activity is one of four core android components. It provides the user with a screen with which users can interact. Applications generally are made up of multiple activities. At any specific time, at max there can be only one active activity.

Fragment –

Fragment is a piece of an activity which enable more modular activity design. Fragments are used to build dynamic and multi-pane user interface. A fragment can be used in multiple activities. Fragments can be added or removed from an activity while the activity is running to give it dynamic behavior. Each fragment has its own layout and life cycle callbacks.

Service –

A service is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed.

Shared Preferences –

In android there are couple of ways to persist data. Android frameworks provides SQLite and shared preferences. Shared preferences allow the developer to save and retrieve data in form of key, value pair.

Recycler View –

Recycler View is a widget that acts a container for displaying large data sets that can be scrolled very efficiently by maintaining a limited number of views. It is a more efficient and flexible version of List View widget.

Adapters –

The adapter is a component that stands between the data model we want to show in our applications UI. The user interface components of the recycler view renders the information present in the adapter.

Navigation Drawer: Navigation Drawer is the sliding menu that appears on the android screen with a hamburger menu icon in the Action Bar. The construction of it requires placing multiple views inside the navigation portion of the Drawer Layout.

Action Bar:

The action bar is an important design element, usually at the top of each screen in an app, that provides a consistent familiar look between Android apps. It is used to provide better user interaction and experience by supporting easy navigation through tabs and drop-down lists.

Model:

The model is responsible for managing the data of the application. It responds to the request from the view and it also responds to instructions from the controller to manipulate the data.

Firebase Realtime database :

Firebase Realtime database is a cloud hosted database that supports multiple platforms Android, iOS and Web. All the data is stored in JSON format and any changes in data, reflects immediately by performing a sync across all the platforms & devices.

PROJECT DESCRIPTION

The following section contains the user requirements for the ridesharing application. The application is a meeting point for carpoolers, both drivers and passengers. Users can share and find rides. The application will be divided into two main parts. The access of the application is only granted to authorized users.

a) Users

The users of the application are travelers who want to go from one place to another or users that are driving a trip and want to find passengers. Users can act as both passengers and drivers while using an application. Any user of the application can act as:

- A driver is any person that owns a car and wants to go from one place to another and publishes his ride on the application in order to find passengers to share the ride with.
- A passenger is any person that doesn't own a car and wants to join a driver in a trip he posted.

Functional Requirements:

1. **Registration**:

   A new user should register him to log in to the application. The registration screen helps user with registration process. Any user can register with a username, email address and password.

2. **Login:**

   After the registration process is completed, registered user can login through login screen with his email address and password.

3. **Session**:

   The session of user is maintained until he logs out of the application. This way the user does not have to login every time he loads pages.

   Once the user is login in he will redirected to home page. The dashboard is a navigation drawer that is used so the user can navigate from one activity to other.

4. **Post Rides**: A user will be able to post a rides.

5. **View Rides**: A user will be able to view all rides other users.

6. **Chat Box**: Rider and Driver Communicate with each others in chat box.

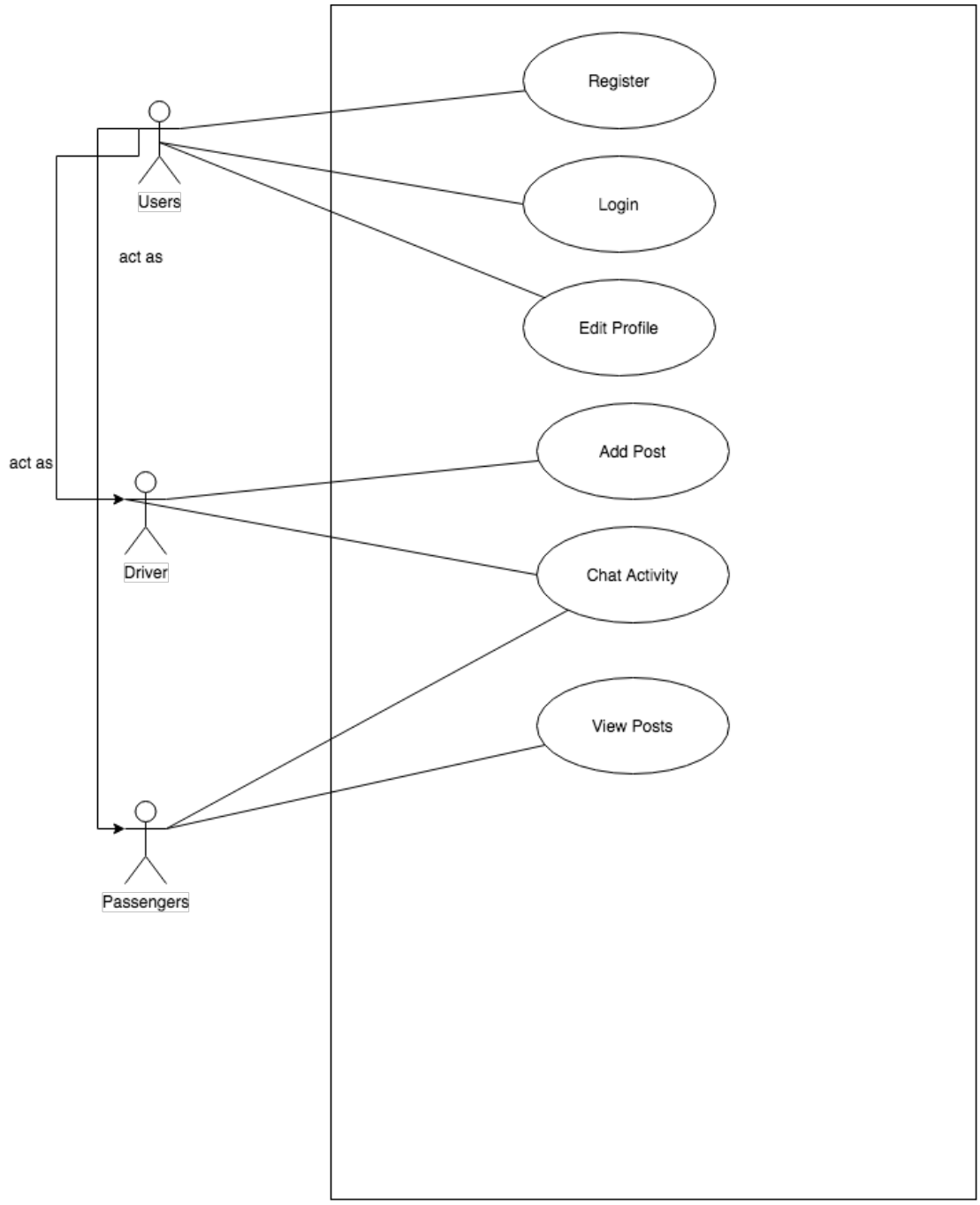
7. **Search Rides**: User will be able to search specific rides by source or destination.

WORKING

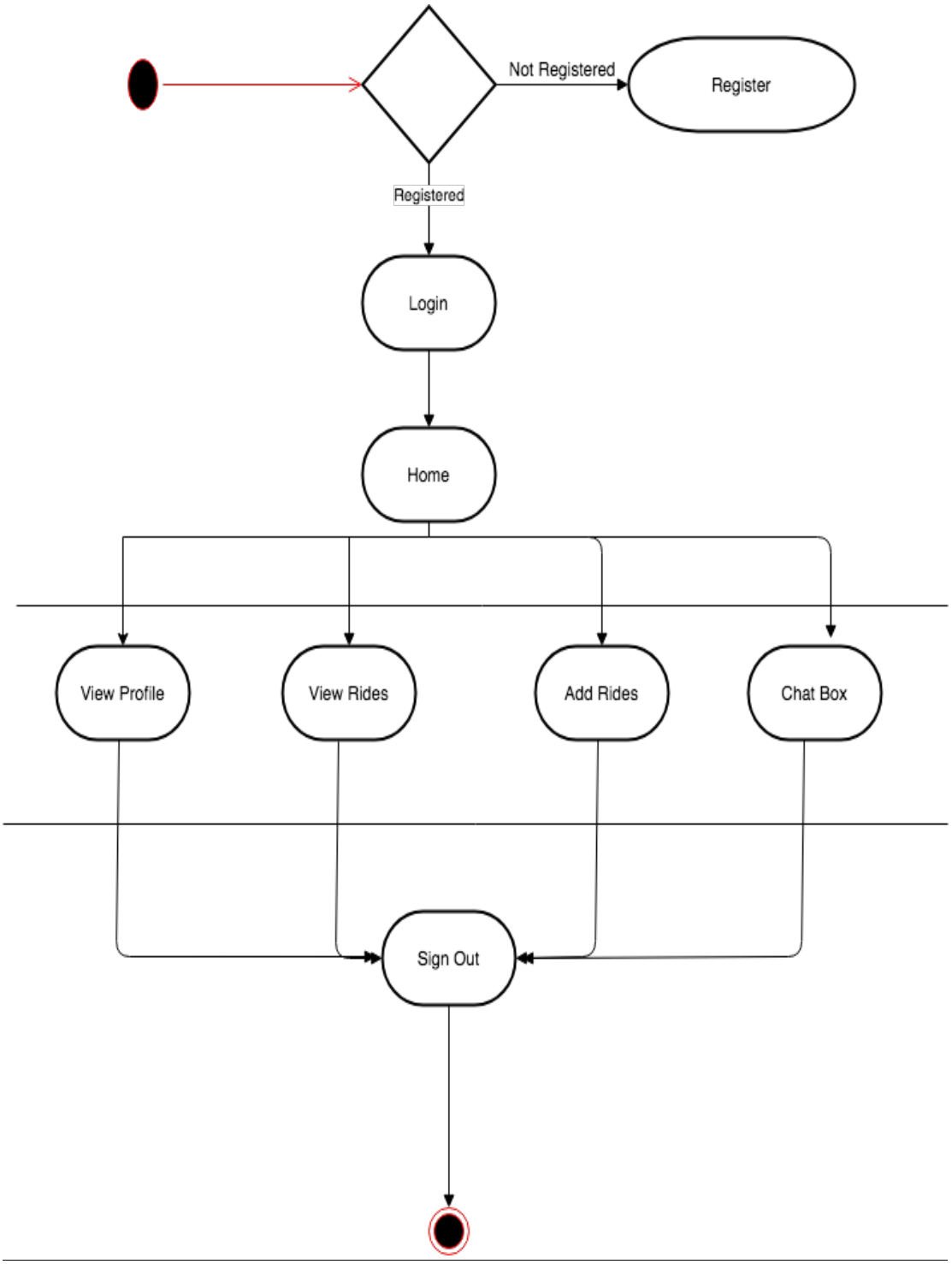
When the application starts with a page that displays two buttons i.e. First, Register Button and Second, Login Button. When the user clicks on register button, user moves to another activity that display page with three Edit Text i.e. First, Name , Second, Enter Email Address and Third, Password. When an improper email is entered it display error message invalid email and password should be of at least length of 6. Then after successful registration, user can login in to the application by clicking on login button that display two fields. When email address and password are entered the user is redirected to home page where the user will be able to see all the rides of various users. Application contains navigation drawer that have four fragments i.e. First, Home , Second, Profile , Third, Users and Fourth, Chat. When user clicks on home fragment it will display all rides(posts) that user will be posting on application. Each rides is displayed in form of recycler views. A button has been associated with each posts that can navigate you to chat with the user if you are willing to go on that trip. User cannot message to his own posts. At the top of application we have menu that displays three items i.e. 1) Search 2) Add Post 3) Log Out. When the user clicks on search it displays search result by entering source or destination where user wants to go by filtering out with others posts. When user clicks on add post it display new page on which user have to enter their source , destination , amount and click upload button so details of the user will be posted on Home Fragment. All rides displayed on main page have a pop menu where current user will be able to edit or delete his own post but with other posts user will be able to navigate to chat box of the other users. With profile fragments, current user will be able to see his own posts. Users fragments will have all users that are registered with app.

UML DIAGRAMS

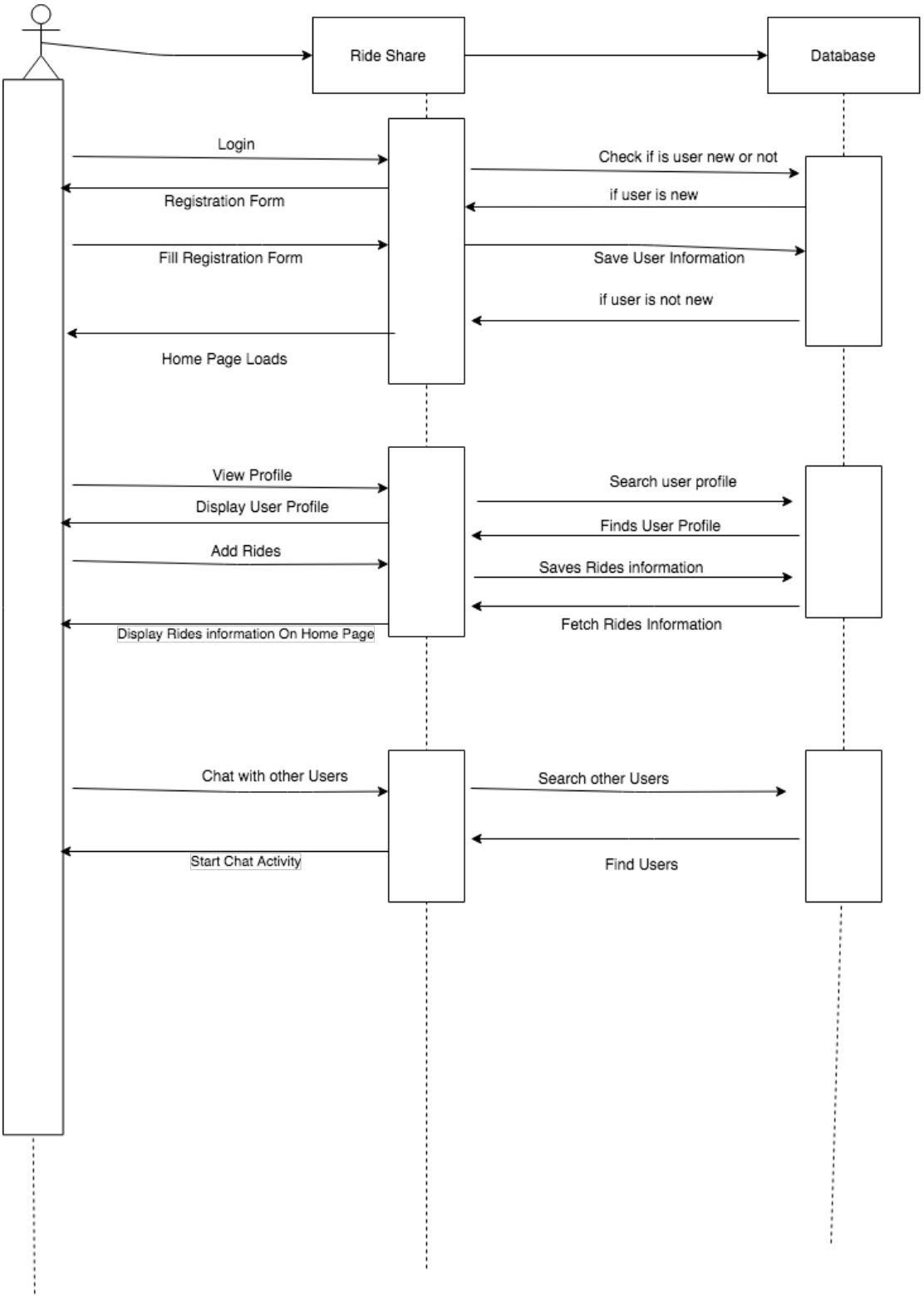
1) USE CASE DIAGRAM:



2) ACTIVITY DIAGRAM:



3) SEQUENCE DIAGRAM:



Code Snippet and Explanation:

1) RegisterActivity.java

```
//handle register btn click
mRegisterBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        //input email,password
        String email = mEmailEt.getText().toString().trim();
        String password = mPasswordEt.getText().toString().trim();

        //validate
        if(!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
            mEmailEt.setError("Invalid Email");
            mEmailEt.setFocusable(true);

        }
        else if (password.length()<6){
            //set error focuss to password edittext
            mPasswordEt.setError("Password length at least 6 characters");
            mPasswordEt.setFocusable(true);

        }
        else {
            registerUser(email,password);//register the user
        }

    }
```

Above Figure shows that when user clicks on register button a validation of email and password is done to make sure that email is in correct format and password must be of at least 6 characters. After the validation, user will be able to login in an application.

2) LoginActivity.java

```
private void showRecoverPasswordDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Recover Password");

    //set layout linear layout
    LinearLayout linearLayout = new LinearLayout( context: this);

    //views to set in dialog
    final EditText emailEt=new EditText( context: this);
    emailEt.setHint("Email");
    emailEt.setInputType(InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS);

    /* sets the min width of EditView to fit a text of n 'M' letters regardless of actual text extension and text size*/
    emailEt.setMinEms(16);

    linearLayout.addView(emailEt);
    linearLayout.setPadding( left: 10, top: 10, right: 10, bottom: 10);

    builder.setView(linearLayout);

    //buttons recover
    builder.setPositiveButton( text: "Recover", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            //input email
            String email = emailEt.getText().toString().trim();
            beginRecovery(email);

        }
    });
    builder.setNegativeButton( text: "Cancel", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            //dismiss dialog
            dialog.dismiss();
```

Above Figure shows us that a function is created for recovery of password. Alert Dialog is used where user will enter email address in Edit Text and two button are provided i.e. Recover and Cancel. When user clicks on recover the email is sent to user for resetting the password. On clicking the cancel button, Alert Dialog is dismiss.

3) HomeFragment.java

```
private void loadPosts(){


    //path of all posts
    DatabaseReference ref = FirebaseDatabase.getInstance().getReference( path: "Posts");

//get all data from this ref
    ref.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        postList.clear();
        if(dataSnapshot.exists()) {

            for (DataSnapshot ds : dataSnapshot.getChildren()) {

                ModelPost modelPost = ds.getValue(ModelPost.class);

                postList.add(modelPost);


            }

            //adapter
            adapterPosts = new AdapterPosts(getActivity(),postList);
            //set Adapter to recyclerview
            recyclerView.setAdapter(adapterPosts);
        }

    }
```

Above Figure shows that a function loadposts() is defined for loading of posts which are stored in firebase database. Data Snapshot contains data from database location. Any time reading data from Database, you receive data as data snapshot. In Database, if any data is available then the data is added to Array List. Recycler view will be used to display the loaded data.

4) AddPostActivity.java

Below Figure shows that Hash Map has been used to insert the data in database. Hash Map is a type of Collection, that stores our data in a pair such that each element has a key associated with it. The pair of key and value is often known as Entry and these entries can have only unique keys. A Firebase reference represents a particular location in your Database and can be used for reading or writing data to that Database location.

```
HashMap<Object, String> hashMap = new HashMap<>();
//put post info

hashMap.put("uid", uid);
hashMap.put("uName", name);
hashMap.put("uEmail", email);
hashMap.put("uDp", dp);
hashMap.put("pId", timeStamp);
hashMap.put("pTitle", title);
hashMap.put("pDescr", description);
hashMap.put("pCurrencyEt", currency);
hashMap.put("pImage", "noImage");
hashMap.put("pTime", timeStamp);

//path to store post data
DatabaseReference ref = FirebaseDatabase.getInstance().getReference( path: "Posts");
//put data in this ref
ref.child(timeStamp).setValue(hashMap).addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        //added in database
        pd.dismiss();
        Toast.makeText( context: AddPostActivity.this, text: "Post Published",Toast.LENGTH_SHORT).show();

        //reset views
        titleEt.setText("");
        descriptionEt.setText("");
        currencyEt.setText("");
        imageIv.setImageURI(null);
        image_rui = null;
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        //failed adding post in database
        pd.dismiss();
        Toast.makeText( context: AddPostActivity.this, text: ""+e.getMessage()  ,Toast.LENGTH_SHORT).show();
```

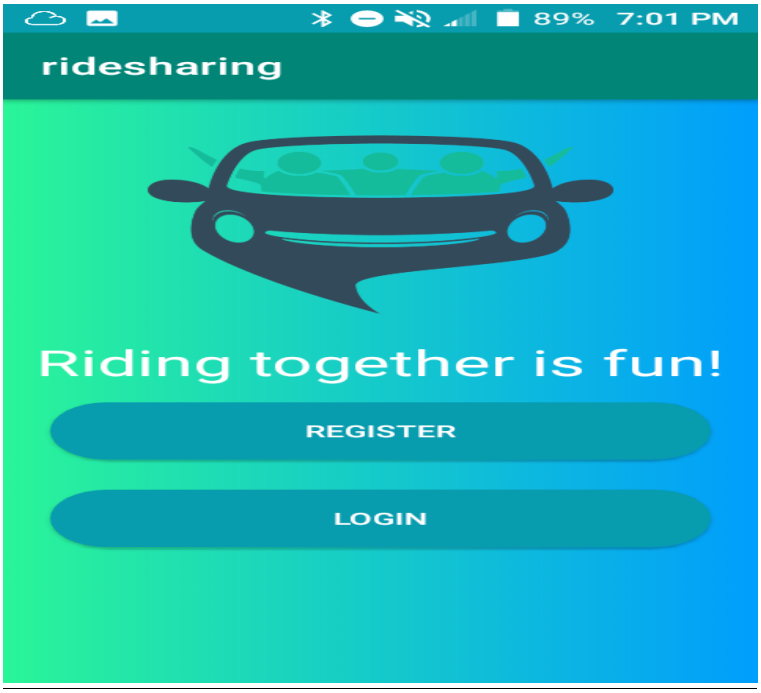
5) ChatActivity.java

```
private void sendMessage(final String message) {
    /*
    Chats node will be created that will contain all chats
    Whenever user sends message it will create new child in chats node and that child will contain
    following key values
    sender: UID of sender
    return: UID if receiver
    message : actual message

     */
    DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference();

    String timestamp = String.valueOf(System.currentTimeMillis());

    HashMap<String, Object> hashMap = new HashMap<>();
    hashMap.put("sender",myUid);
    hashMap.put("receiver",hisUid);
    hashMap.put("message",message);
    hashMap.put("timestamp",timestamp);
    hashMap.put("isSeen",false);
    databaseReference.child("Chats").push().setValue(hashMap);


    final DatabaseReference database = FirebaseDatabase.getInstance().getReference( path: "Users").child(myUid);
    database.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            ModelUser user = dataSnapshot.getValue(ModelUser.class);
            if(notify){
                senNotification(hisUid,user.getName(),message);
            }

            notify = false;

        }
```

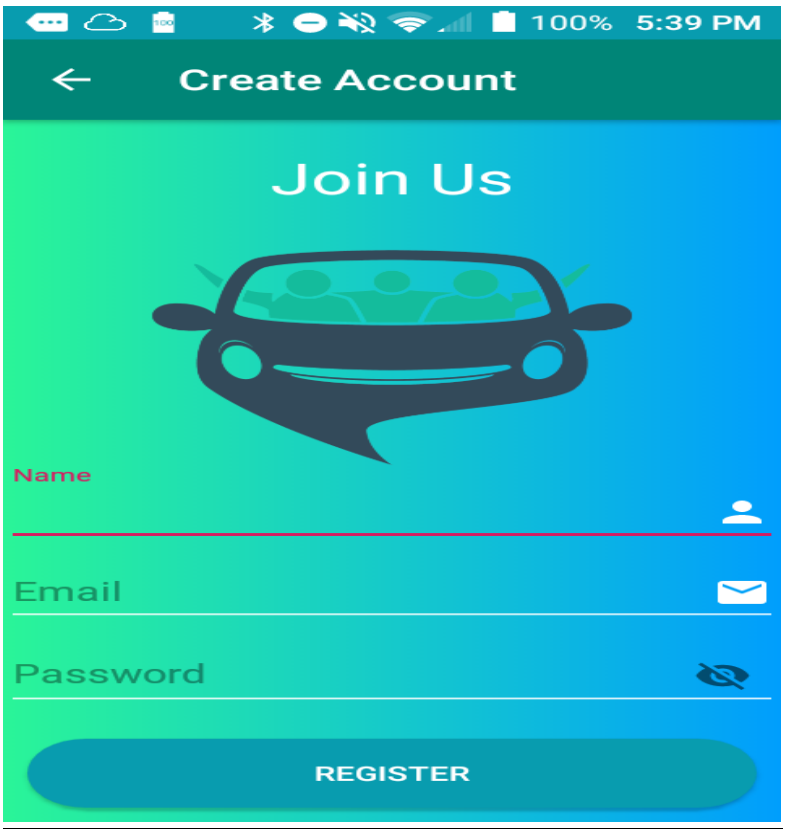
Above Figure shows that function of SendMessage() is used which takes message as input parameter. Chats node will be created that will contain all chats. When a user send Message it will create new child in chats node and child will contain key values like sender, receiver, message.

OUTPUTS

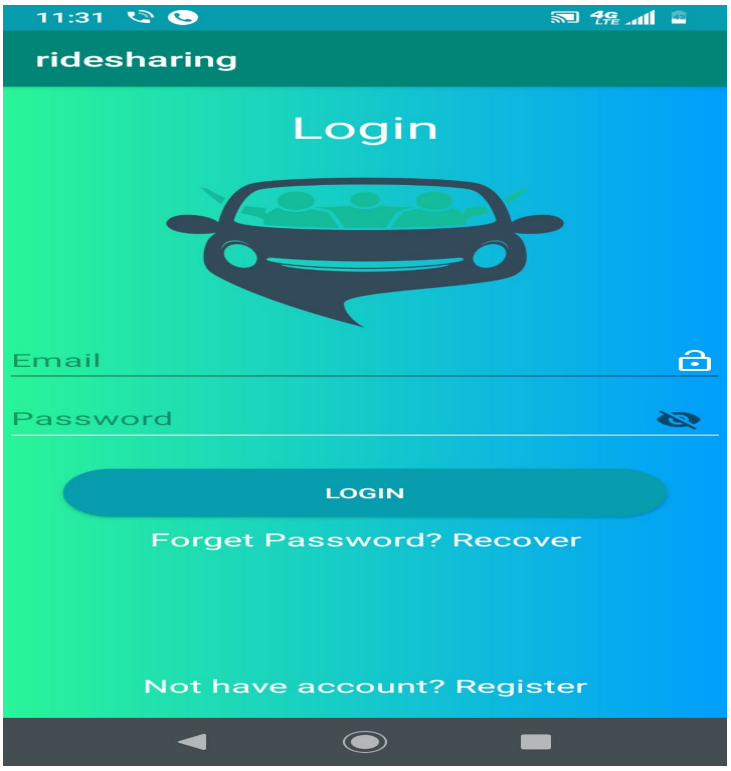
1) FRONT PAGE:



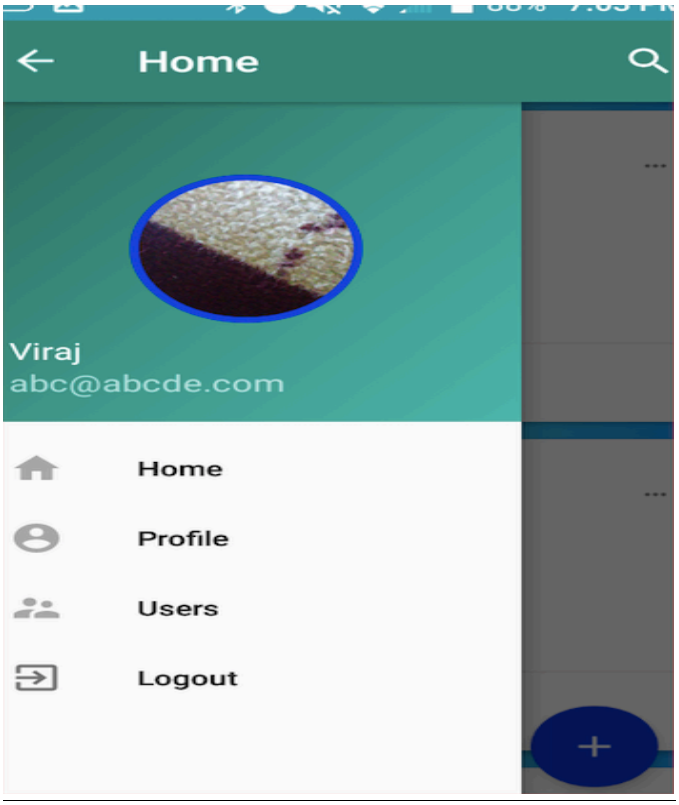
2)REGISTRATION



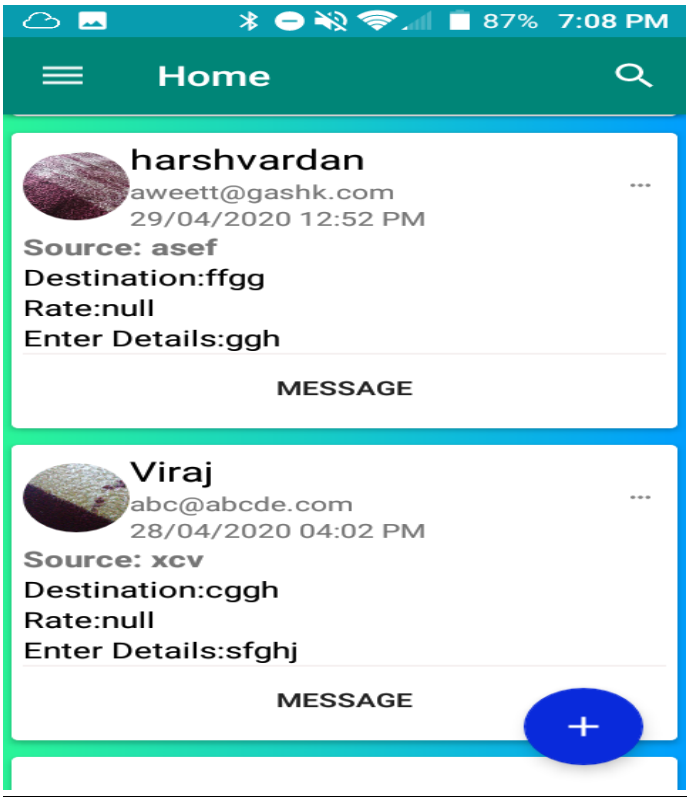
3)LOGIN:



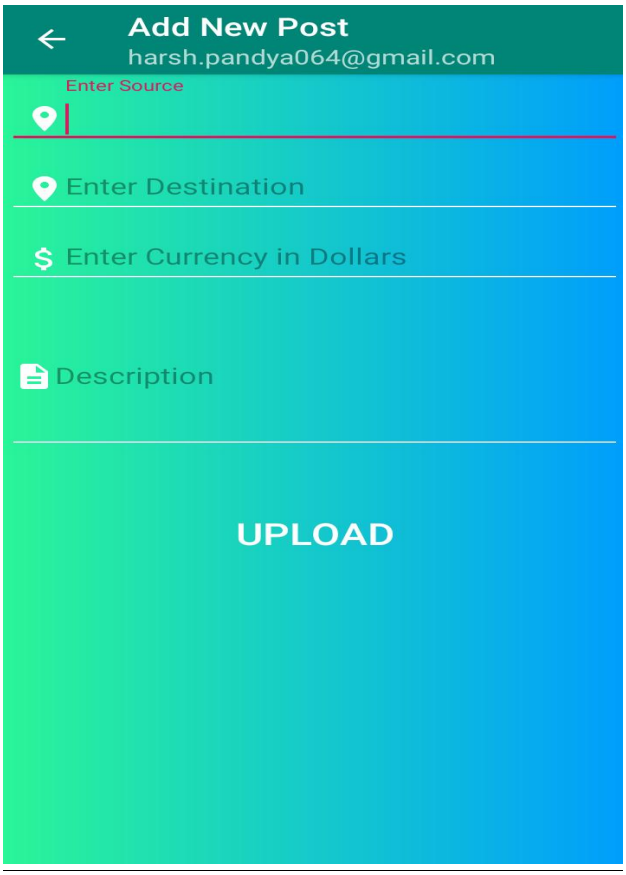
4) NAVIGATION DRAWER



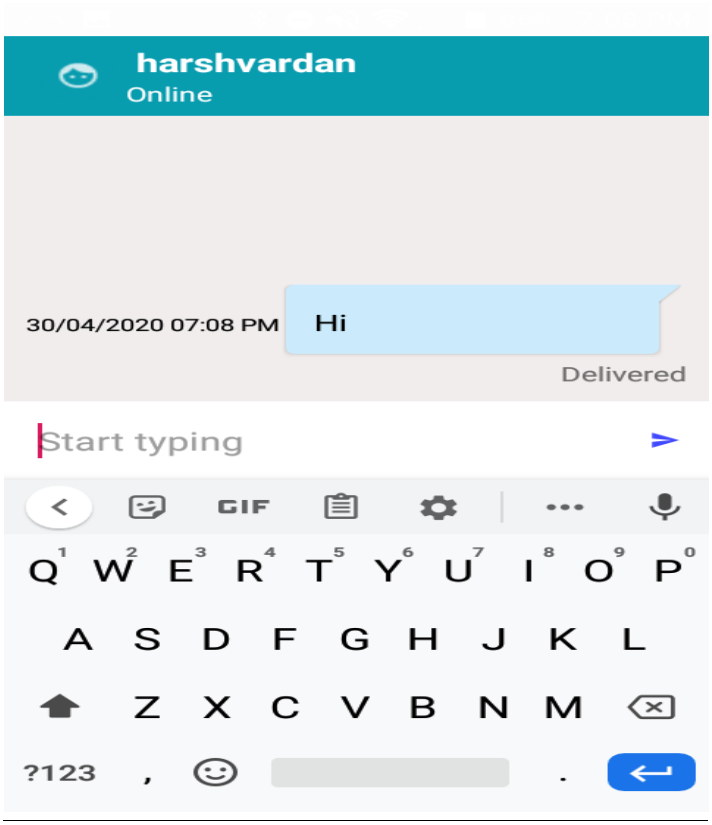
5) HOME



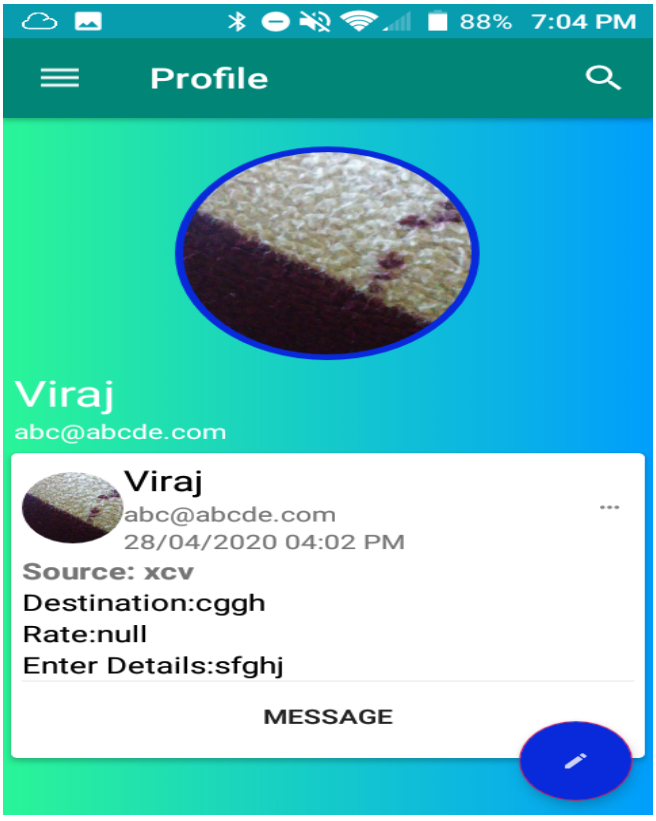
6) ADD POST



7) CHAT



8) PROFILE



CHALLENGES

- Finding backend database to manage data used by application.
- One of the major challenges we faced in creating this project is in writing Firebase Database Queries and to obtain information from database. Stack Overflow answered most of the questions.
- Loading data from Firebase Database in Recycler View was very difficult.
- Then we faced challenge in designing a Chat Box in which users (Driver and Riders) can communicate with each other. We don't know how we can matched driver id with rider id to display chat box. But we figure out at the end.

EVALUATION

	HARSH PANDYA (TEAM LEADER)	VIRAJ BHAKTA
USER INTERFACE	✔	
ICON	✔	
FIREBASE DATABASE	✔	✔
ADD POST	✔	✔
HOME PAGE	✔	✔
CHAT BOX		✔
USER LIST		✔
PROFILE	✔	

CONCLUSION AND FUTURE WORK

This section talks about the lessons that we have learned in the process of developing this project and how this application can be improved by adding additional features in the future.

Lessons Learnt

During the course of this application development I learnt to develop an android app development using Android Studio that has various functionalities related to authentication of users, storing of user data and retrieval of users data, chat activity. In terms of technical skills we learnt about Firebase Real time database that help us in fetching and storing of user data in database. And also learnt how user is authenticated by using Firebase Query. Furthermore, we learnt to use different types of layout in our application like card views, recycler views , various Image Assets for action bar ,etc.

Future Work

Due to this lack of time, many things can be improved in the present application.

- Better user interface with more attractive styles
- Adding more support for authentication systems can be an improvement.
- Also, allowing user to comment to an event could improve the coordination among users.
- Add maps Functionality
- Integrate with Payment gateway Service.
- Introduce reward points to increase number of customers.

REFRENCES

1. Android Developer. Fragment.
   https://developer.android.com/guide/components/fragments.html.
   Accessed in March 2020.

2. Android Developer. Services.
   http://developer.android.com/guide/components/services.html.
   Accessed in March 2020.

3. Android Developer. Shared Preferences.
   http://developer.android.com/reference/android/content/SharedPreferences.html.
   Accessed in April 2020.

4. Android Developer. Recycler View.
   https://developer.android.com/training/material/lists-cards.html.
   Accessed in April 2020.

5. Android Developer. Activity.
   https://developer.android.com/guide/components/activities.html.

   Accessed in March 2020.

6. Firebase Database https://firebase.google.com/docs/database .
   Accessed in April 2020.

7. Firebase data in Recycler Views
   https://www.android-examples.com/show-firebase-database-data-into-recyclerview/
   Accessed in April 2020.

8. Chat Activity
   https://www.scaledrone.com/blog/android-chat-tutorial/
   Accessed in April 2020.