A Project Report

On

RAIL HAILING ANDROID APP

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology

Under The Supervision of
DR. P. Muthusamy
Associate Professor
Department of Computer Science and Engineering

Submitted By

19SCSE1010879- Yash Agarwal

19SCSE1010880- Kartikey Chandra

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA

DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"Rail Hailing Application"** in partial fulfillment of the requirements for the award of the **Bachelor Of Technology** submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of "**DR. P. Muthusamy, Associate Professor**, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of

Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Yash Agarwal(19SCSE1010879)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

DR. P.Muthusamy

Associate Professor

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Yash Agarwal(19SCSE1010879) & Kartikey Chandra(19SCSE1010880) has been held on _____ and his work is recommended for the award of Bachelor Of Technology.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: November, 2013

Place: Greater Noida

ABSTRACT

This software aims at automating the process of interaction between cab drivers and passengers and hence, increasing the efficiency of both. The application provides quick and accurate results based on the corresponding input. The application helps keep track of all the data associated with the trips a user chooses to proceed with. It also provides reliability and accountability so that a user can choose exactly which other user he/she would like to proceed the interaction with. So in this application we link the Arogya Setu Server which helps to detect whether the rider has COVID-19 symptoms or not. It helps the driver to get away from the COVID-19 infected passengers.

INTRODUCTION

The application provides the user (rider) with a dashboard display where the user chooses a pickup location for the trip, the destination for the trip and the desired ride type. The application finds a user (driver) who matches the input criteria by the rider and fare is generated and the fare details along with a possible driver's details is sent back to the rider. The driver is sent a trip request which sends the pickup, destination and contact details of the rider to the driver. The application then asks the driver for a response to the request and a suitable message is generated for the rider based on the driver's response. The driver's approval provides the driver with the contact details of the rider and vice versa. If the driver chooses to deny the trip request received, the ride booking process is halted and restarted to look for more drivers and a suitable message is generated for the rider. If the rider chooses to cancel the trip mid-process, the request is taken back and suitable cancellation messages are displayed according to the stage the process has reached.

LITERATURE SURVEY

Distributed Databases

A distributed database system allows applications to access data from local and remote databases. Distributed databases use a client/server architecture to process information requests.

Advantages of using this application:

•Management of distributed data with different levels of transparency like network transparency, fragmentation transparency, replication transparency, etc.

- Increase reliability and availability/
- Easier expansion
- Protection of valuable data and information.
- Improved performance.
- Less cost.

•Systems can be modified, added and removed from the distributed database without affecting other modules.

- Reliable transactions.
- Distributed query processing can improve performance.

LITERATURE SURVEY

Most of the Indian tourists prefer travelling to their preferred destination via road. It is due to the fact that a road trip allows you to enjoy the scenic beauty at your own pace. Moreover, the comfort of travelling in a vehicle with your family and friends is unparalleled. You can opt for online cab booking that will make your journey hassle-free. In fact, cab booking is something which has become a part of our everyday life. You can also opt for intercity car rentals in order to visit other cities. The main problem arises in March 2020. Earlier everything was normal, no one was worried about coming in contact with anyone. Then after March 2020, corona started spreading all over the country. Everyone was afraid to even go near anyone. So after the lockdown, people's jobs started. In such a situation, those who do not have their own vehicle, they use online cab booking to reach their destination. So because of this cab drivers as well as riders are worried about getting infected with COVID-19. So we made this application so that everyone is safe whether it is a rider or a driver.

Project Design

Flow of Android Ride Hailing app:
Let’s see all the screens that you need to design for your application.

Splash Screen: You need to keep your app title as “Welcome to App” and then provide a button for the user to get started.

Sign In Screen: You need to build a screen where users can sign in from google.
You need to design a bottom navigation bar with five options: Home, Search, Add Item, Notification, Profile.

Home: Here, you can find all your lost items and get an option to send notifications to other users.

Search: You can search here if someone has your lost items with them.

Add Item: You can either add the lost item or the found item—someone who recognizes your lost item and notifies the owner.

Notification: Here, all the notifications related to the items are seen.

Profile: Here, you can see your profile and log out from the application.

Android Ride Hailing App Project Prerequisites

To build the whole project, you need to have some knowledge about the concepts in Android. We can list out some of the important stuff which you need to know before you get started.

1. Android Studio and its tools
2. Android Notifications
3. Android Activities and its lifecycle
4. Android Fragments
5. Android Recycler View or List View
6. XML Layout designing
7. Object-Oriented Programming
8. Java or Kotlin Programming
9. Firebase Authentication(Google sign in) and Realtime Database
10. Libraries like Picasso and Circular Image Library.

Description of the Ride Hailing App project:

Now let’s see what all files you have in our project. Knowing these files will help you gather more understanding of what we did to build the project.

1.Manifest: The project’s activities, services, or receivers declarations are present here. Even the app permissions are defined in this file.

2.Main Activity: This is the first activity for your Find My Thing Application.

3. Gradle File: All the libraries you use are declared here. We mention the dependencies of such libraries here.

4.Firebase: It is used to set up the backend for Find My Thing Application. You will use it to sign in from google and also to store your data.

5.Resources: It has several sections like layouts, styles, colors, and fonts. It helps us to keep images, design layouts and define colors and styles.

6. Libraries: There are libraries like Picasso and Circular Image Library, which are used to display images.

MANIFEST

Every app project must have an AndroidManifest.xml file (with precisely that name) at the root of the project source set. The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.

Among many other things, the manifest file is required to declare the following:

- The app's package name, which usually matches your code's namespace. The Android build tools use this to determine the location of code entities when building your project. When packaging the app, the build tools replace this value with the application ID from the Gradle build files, which is used as the unique app identifier on the system and on Google Play.
- Syntax:-

```
<?xml version="1.0" encoding="utf-8"?>
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.myapp"
        android:versionCode="1"
        android:versionName="1.0" >
      ...
    </manifest>
```

MainActivity

- Most apps contain multiple screens, which means they comprise multiple activities. Typically, one activity in an app is specified as the main activity, which is the first screen to appear when the user launches the app. Each activity can then start another activity in order to perform different actions. For example, the main activity in a simple e-mail app may provide the screen that shows an e-mail inbox. From there, the main activity might launch other activities that provide screens for tasks like writing e-mails and opening individual e-mails.

- Although activities work together to form a cohesive user experience in an app, each activity is only loosely bound to the other activities; there are usually minimal dependencies among the activities in an app. In fact, activities often start up activities belonging to other apps. For example, a browser app might launch the Share activity of a social-media app.

Gradle Files

The Android build system compiles app resources and source code, and packages them into APKs or Android App Bundles that you can test, deploy, sign, and distribute. Android Studio uses Gradle, an advanced build toolkit, to automate and manage the build process, while allowing you to define flexible custom build configurations. Each build configuration can define its own set of code and resources, while reusing the parts common to all versions of your app. The Android plugin for Gradle works with the build toolkit to provide processes and configurable settings that are specific to building and testing Android applications.

Gradle and the Android plugin run independent of Android Studio. This means that you can build your Android apps from within Android Studio, the command line on your machine, or on machines where Android Studio is not installed (such as continuous integration servers). If you are not using Android Studio, you can learn how to build and run your app from the command line. The output of the build is the same whether you are building a project from the command line, on a remote machine, or using Android Studio.

THE BUILD PROCESS OF GRADLES

The build process for a typical Android app module, as shown in figure 1, follows these general steps:

1. The compilers convert your source code into DEX (Dalvik Executable) files, which include the bytecode that runs on Android devices, and everything else into compiled resources.
2. The packager combines the DEX files and compiled resources into an APK or AAB, depending on the chosen build target. Before your app can be installed onto an Android device or distributed to a store, such as Google Play, the APK or AAB must be signed.
3. The packager signs your APK or AAB using either the debug or release keystore:

- If you are building a debug version of your app, that is, an app you intend only for testing and profiling, the packager signs your app with the debug keystore. Android Studio automatically configures new projects with a debug keystore.
- If you are building a release version of your app that you intend to release externally, the packager signs your app with the release keystore that you need to configure. To create a release keystore, read about signing your app in Android Studio.

4. Before generating your final APK, the packager uses the zipalign tool to optimize your app to use less memory when running on a device.
5. At the end of the build process, you have either a debug or release APK or AAB of your app that you can use to deploy, test, or release to external users.

Firebase

- Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google’s infrastructure.

- Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

Key Features of Firebase

1. Authentication

- It supports authentication using passwords, phone numbers, Google, Facebook, Twitter, and more. The Firebase Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app.

2. Realtime database

- Data is synced across all clients in realtime and remains available even when an app goes offline.

- Firebase Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

4. Test lab

- The application is tested on virtual and physical devices located in Google’s data centers.

5. Notifications

- Notifications can be sent with firebase with no additional coding.

- Users can get started with firebase for free; more details can be found on the official website.

Resources

- Resources are the additional files and static content that your code uses, such as bitmaps, layout definitions, user interface strings, animation instructions, and more.

- You should always externalize app resources such as images and strings from your code, so that you can maintain them independently. You should also provide alternative resources for specific device configurations, by grouping them in specially-named resource directories. At runtime, Android uses the appropriate resource based on the current configuration. For example, you might want to provide a different UI layout depending on the screen size or different strings depending on the language setting.

- Once you externalize your app resources, you can access them using resource IDs that are generated in your project's R class. This document shows you how to group your resources in your Android project and provide alternative resources for specific device configurations, and then access them from your app code or other XML files.

Library

- An Android library is structurally the same as an Android app module. It can include everything needed to build an app, including source code, resource files, and an Android manifest. However, instead of compiling into an APK that runs on a device, an Android library compiles into an Android Archive (AAR) file that you can use as a dependency for an Android app module. Unlike JAR files, AAR files offer the following functionality for Android applications:

- AAR files can contain Android resources and a manifest file, which allows you to bundle in shared resources like layouts and drawables in addition to Java classes and methods.
- AAR files can contain C/C++ libraries for use by the app module's C/C++ code.
- A library module is useful in the following situations:

- When you're building multiple apps that use some of the same components, such as activities, services, or UI layouts.
- When you're building an app that exists in multiple APK variations, such as a free and paid version and you need the same core components in both.
- In either case, simply move the files you want to reuse into a library module then add the library as a dependency for each app module. This page teaches you how to do both.

SOURCE CODES

- LOGIN ACTIVITY

```
package com.dataflair.foundandlost.Activities;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import com.dataflair.foundandlost.MainActivity;
import com.dataflair.foundandlost.R;
import com.google.android.gms.auth.api.signin.GoogleSignIn;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInClient;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.common.SignInButton;
import com.google.android.gms.common.api.ApiException;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.GoogleAuthProvider;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.annotations.NotNull;
```

```
import java.util.HashMap;

public class LoginActivity extends AppCompatActivity {
    GoogleSignInClient mSignInClient;
    FirebaseAuth firebaseAuth;
    ProgressDialog progressBar;
    SignInButton signInButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        createRequest();
        //Firebase Authentication
        firebaseAuth = FirebaseAuth.getInstance();
        //Progress bar
        progressBar = new ProgressDialog(this);
        progressBar.setTitle("Please Wait...");
        progressBar.setMessage("We are setting Everything for you...");
        progressBar.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        signInButton = (SignInButton) findViewById(R.id.GoogleSignInBtn);
```

```
//Google signin options to use gmail login
      GoogleSignInOptions signInOptions = new GoogleSignInOptions
            .Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken("284817088166-2jrga6sv4g0sgjna5hcpd1ic7jpju83s.apps.googleusercontent.com")
            .requestEmail()
            .build();
      mSignInClient = GoogleSignIn.getClient(getApplicationContext(), signInOptions);
      //implementing onclick listener to accesss all available gmail accounts
      signInButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            Intent intent = mSignInClient.getSignInIntent();
            startActivityForResult(intent, 100);
         }
      });
}
private void createRequest() {
   // Configure Google Sign In
   GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
         .requestIdToken(getString(R.string.default_web_client_id))
         .requestEmail()
         .build();
   GoogleSignInClient mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
}
```

```
@Override
  protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data)
    { super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 100) {
       Task<GoogleSignInAccount> googleSignInAccountTask = GoogleSignIn
             .getSignedInAccountFromIntent(data);
       if (googleSignInAccountTask.isSuccessful()) {
          progressBar.show();
          try {
             GoogleSignInAccount googleSignInAccount = googleSignInAccountTask.getResult(ApiException.class);
             if (googleSignInAccount != null) {
AuthCredential authCredential = GoogleAuthProvider
       .getCredential(googleSignInAccount.getIdToken(), null);
firebaseAuth.signInWithCredential(authCredential).addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
   @Override
   public void onComplete(@NonNull Task<AuthResult> task) {
      if (task.isSuccessful()) {
         FirebaseDatabase database = FirebaseDatabase.getInstance();
         DatabaseReference myRef = database.getReference().child("users");
         //Hashmap to store the userdetails and setting it to fireabse
         HashMap<String, Object> user_details = new HashMap<>();
```

```
//Accessing the user details from gmail
                    String id = googleSignInAccount.getId().toString();
                    String name = googleSignInAccount.getDisplayName().toString();
                    String mail = googleSignInAccount.getEmail().toString();
                    String pic = googleSignInAccount.getPhotoUrl().toString();
                    user_details.put("id", id);
                    user_details.put("name", name);
                    user_details.put("mail", mail);
                    user_details.put("profilepic", pic);
//updating the user details in firebase
                    myRef.child(id).updateChildren(user_details).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull @NotNull Task<Void> task) {
                            if (task.isSuccessful()) {
                                progressBar.cancel();
                                //navigating to the main activity after user successfully registers
                                Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                                //Clears older activities and tasks
                               intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                                startActivity(intent);
                            }}}}}}}
} catch (ApiException e) {
                e.printStackTrace();
            }}}}}
```

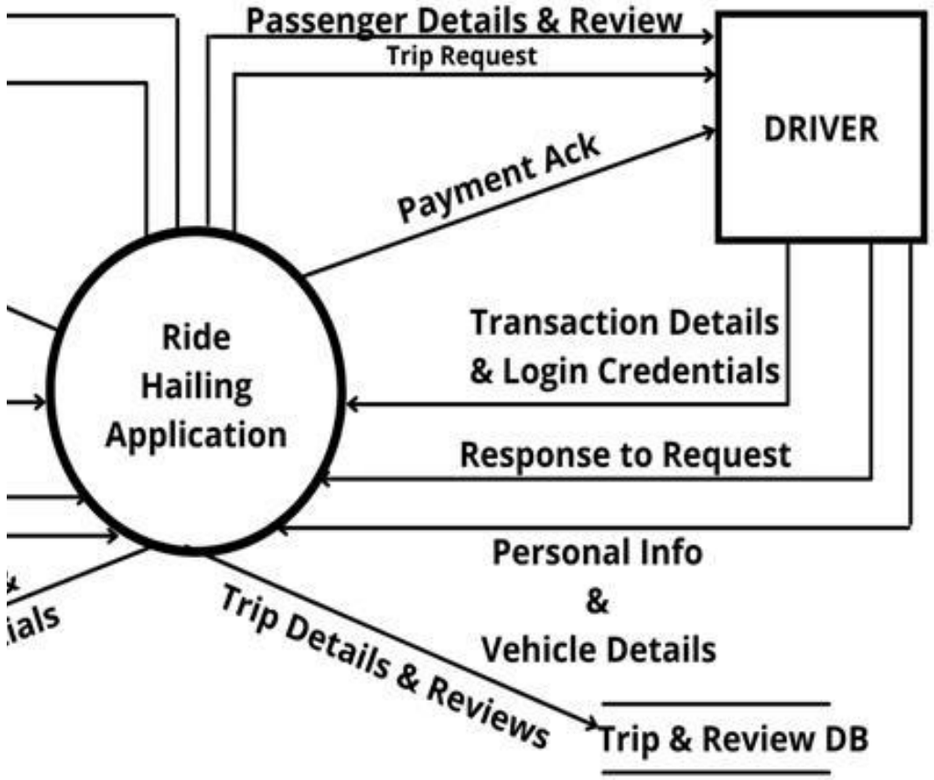
- **Starting Activity**

```
package com.dataflair.foundandlost.Activities;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import com.dataflair.foundandlost.R;

public class StartingActivity extends AppCompatActivity {

    Button getStartedBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_starting);

        //Button To navigate to next Activity
        getStartedBtn = (Button) findViewById(R.id.GetStartedBtn);
        //OnClick Implementation to navigate Between activities
        getStartedBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                startActivity(intent);
            }
        });
    }
}
```

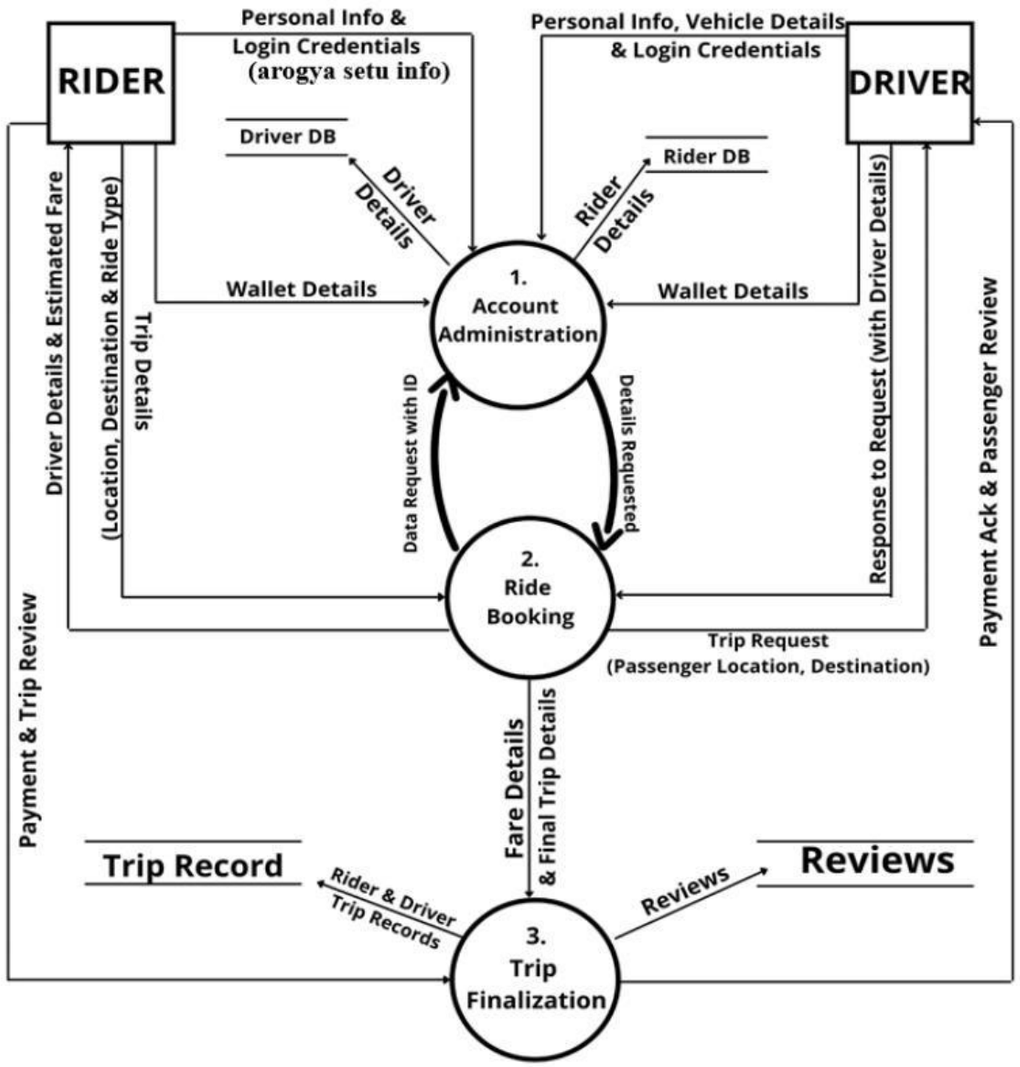
List of Figures

Data Flow Diagram

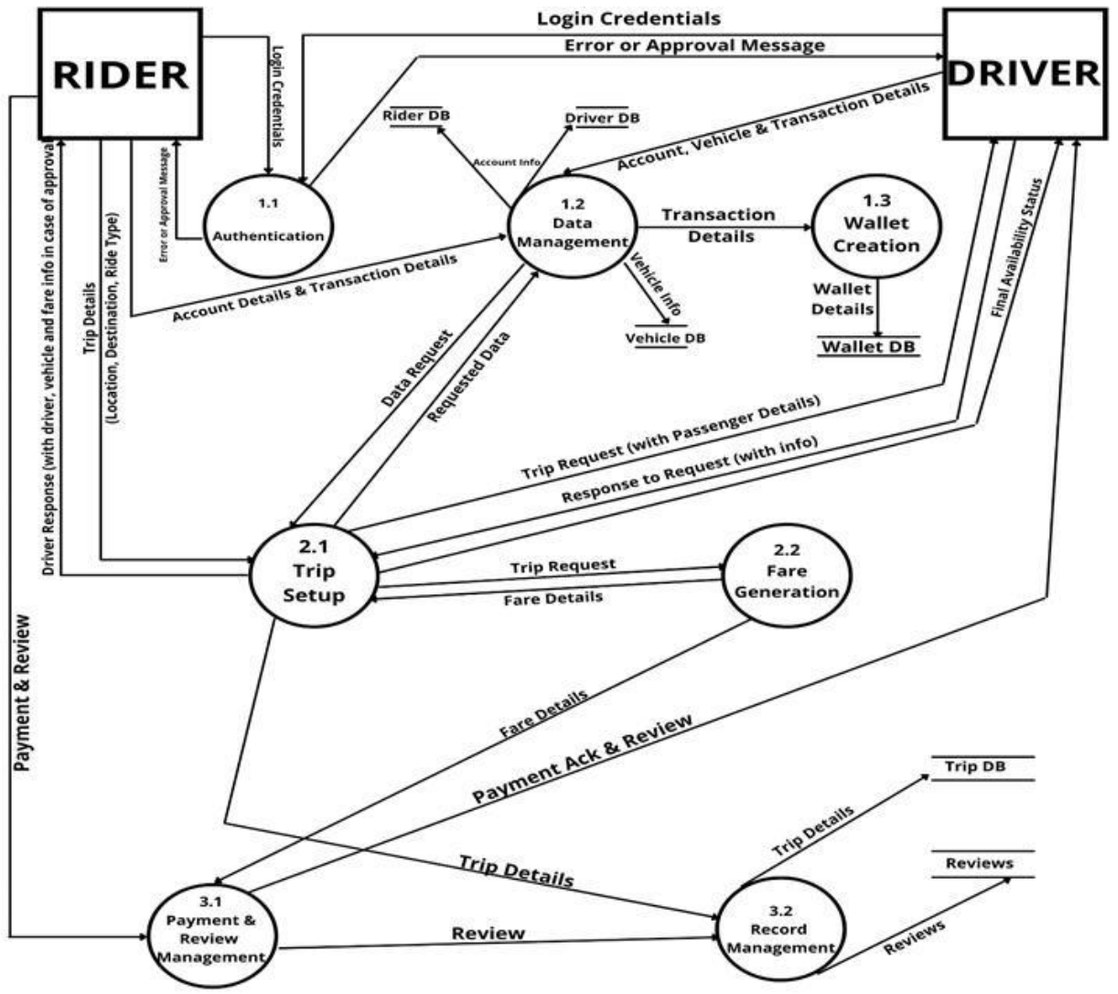
LEVEL - 0



LEVEL - 1


RIDER
DRIVER
Personal Info &
Login Credentials
(arogya setu info)
Personal Info, Vehicle Details
& Login Credentials
Driver DB
Rider DB
Driver Details
Rider Details
Wallet Details
Wallet Details
1.
Account
Administration
Driver Details & Estimated Fare
(Location, Destination & Ride Type)
Trip Details
Data Request with ID
Details Requested
Response to Request (with Driver Details)
Payment Ack & Passenger Review
2.
Ride
Booking
Trip Request
(Passenger Location, Destination)
Payment & Trip Review
Fare Details
& Final Trip Details
Trip Record
Rider & Driver
Trip Records
Reviews
Reviews
3.
Trip
Finalization

LEVEL - 2


RIDER
DRIVER
Login Credentials
Error or Approval Message
Login Credentials
Error or Approval Message
Account, Vehicle & Transaction Details
Rider DB
Driver DB
Account Info
1.1
Authentication
1.2
Data
Management
Transaction
Details
1.3
Wallet
Creation
Vehicle Info
Vehicle DB
Wallet
Details
Wallet DB
Account Details & Transaction Details
Data Request
Requested Data
Final Availability Status
Trip Request (with Passenger Details)
Response to Request (with info)
Response (with driver, vehicle and fare info in case of approval)
Driver Response
Trip Details
(Location, Destination, Ride Type)
2.1
Trip
Setup
Trip Request
Fare Details
2.2
Fare
Generation
Payment & Review
Fare Details
Payment Ack & Review
Trip DB
Trip Details
Reviews
Trip Details
3.1
Payment &
Review
Management
Review
3.2
Record
Management
Reviews

REFERENCES

- Google Firebase Console: https://console.firebase.google.com/?pli=1
- JavaScript: https://www.tutorialspoint.com/java/index.htm
- Wiki: https://en.wikipedia.org/wiki/Main_Page
- Stack Overflow: https://stackoverflow.com/

THANK YOU