LIST OF FIGURES

Figure No.	Name of Figures	Page No.
1.5	Android architecture	4
1.6	Android components	5
2.1.1	Flow of the Activities in Application	7
2.2	The design of Application	9
4.1.1	Home Activity	14
4.1.2	Admin Home Activity	14
4.1.3	Login Activity	14
4.1.4	Register Activity	14
4.1.5	Available Slots Activity	15
4.1.6	Parked Vehicles Activity	15
4.1.7	Parked History Activity	15
4.1.8	Add Vehicle Activity	15
4.1.9	Remove Vehicle Alert Dialog	16
4.1.10	Vehicle Exit Activity	16
4.1.11	Add Vehicle Alert Dialog	16
4.1.12	Forgot Password Alert Dialog	16

Table No.	Name of Tables	Page No.
5.0	Application Test	17-18

CHAPTER-1

INTRODUCTION

With the increase in use of private vehicle in recent years, the problem of car parking has raised in busy and big cities of the world. In crowded cities of the world, mostly a person must spend a lot of time in finding the vacant parking lot.

As an important component of traffic system, parking management system is playing an important role and affecting people's daily life.

By detecting and processing the information from parking lots, smart parking systems allows driverto obtain real-time parking information and alleviates parking contentions.

A Smart Parking Management system is a parking solution which is embedded into parking spots to detect whether parking bays are free or occupied through real-time data collection.It will not only help in generating bill for parking of car but will also keep record of all the vehicles parked at any parking spot with specific date and time.

1.1 INTRODUCTION TO MOBILE APPLICATION DEVELOPMENT

Mobile application development is the process to making software for Smartphone and digital assistants, most commonly for Android and iOS.

The software can be preinstalled on the device, downloaded from a mobile app store or accessed through a mobile web browser. The programming and markup languages used for this kind of software development include Java, Swift, C# and HTML5.

Mobile app development is rapidly growing. From retail, telecommunications and e-commerce to insurance, healthcare and government, organizations across industries must meet user expectations for real-time, convenient ways to conduct transaction and access information. Today, mobile devices- and the mobile applications that unlock their value-are the most popular way for people and business to connect to the internet. To stay relevant, responsive and successful, organizations need to develop the mobile applications that their customers, partners and employees'demand.

1.2 WHAT IS MOBILE APP?

A mobile application or mobile app is a computer program or software application designedto run on a mobile device such as a phone, tablet, or watch. Apps were originally intended for productivity assistance such as email, calendar, and contact databases, but the public demand for apps caused rapid expansion into other areas such as mobile games, factory automation, GPS and location-based services, order-tracking, and ticket purchases, so that there are now millions of apps available.

Apps are generally downloaded from application distribution platforms which are operated by the owner of the mobile operating system, such as the App Store (iOS) or Google Play Store. Mobile applications often stand in contrast to desktop applications which are designed to run on desktop computers, and web applications which run in mobile web browsers rather than directly on the mobile device.

Mobile App has many advantages like within a short app we can communicate a lot of information to the client/customers and even it is an ease of access to client/customer for services update or sale/purchase activity.

1.3 WHAT IS MOBILE OS?

A mobile operating system is an operating system for mobile phones, tablets, smart watches, 2-in-1 PCs, smart speakers, or other mobile devices. While computers such as typical laptops are ‘mobile’, the operating systems used on them are generally not considered mobile ones, as they were originally designed for desktop computers that historically did not have or need specific mobile features. This distinction is becoming blurred in some newer operating systems that are hybrid made for both uses.

A mobile OS is responsible for identifying and defining mobile device features and functions, including keypads, application synchronization, email, thumbwheel and text messaging. A mobile OS is similar to a standard OS (like Windows, Linux, and Mac) but is relatively simple and light and primarily manages the wireless variations of local and broadband connections, mobilemultimedia and various input methods.

1.4 INTRODUCTION TO ANDROID STUDIO

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of Intellij's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices.
- Apply Changes to push code and resource changes to your running app without restartingyour app.
- Code templates and GitHub integration to help you build common app features and importsample code.
- Extensive testing tools and frameworks.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- C++ and NDK support.
- Built-in support for Google Cloud Platform, making it easy to integrate Google CloudMessaging and App Engine.

Android Studio provides a unified environment where we can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto.

1.5 ANDROID ARCHITECTURE

Android architecture contains different number of components to support any android device needs. Android software contains an open-source Linux Kernel having collection of number of C/C++ libraries which are exposed through an application framework services. Among all the components Linux Kernel provides main functionality of operating system functions to smartphones and Dalvik Virtual Machine (DVM) provide platform for running an android application.

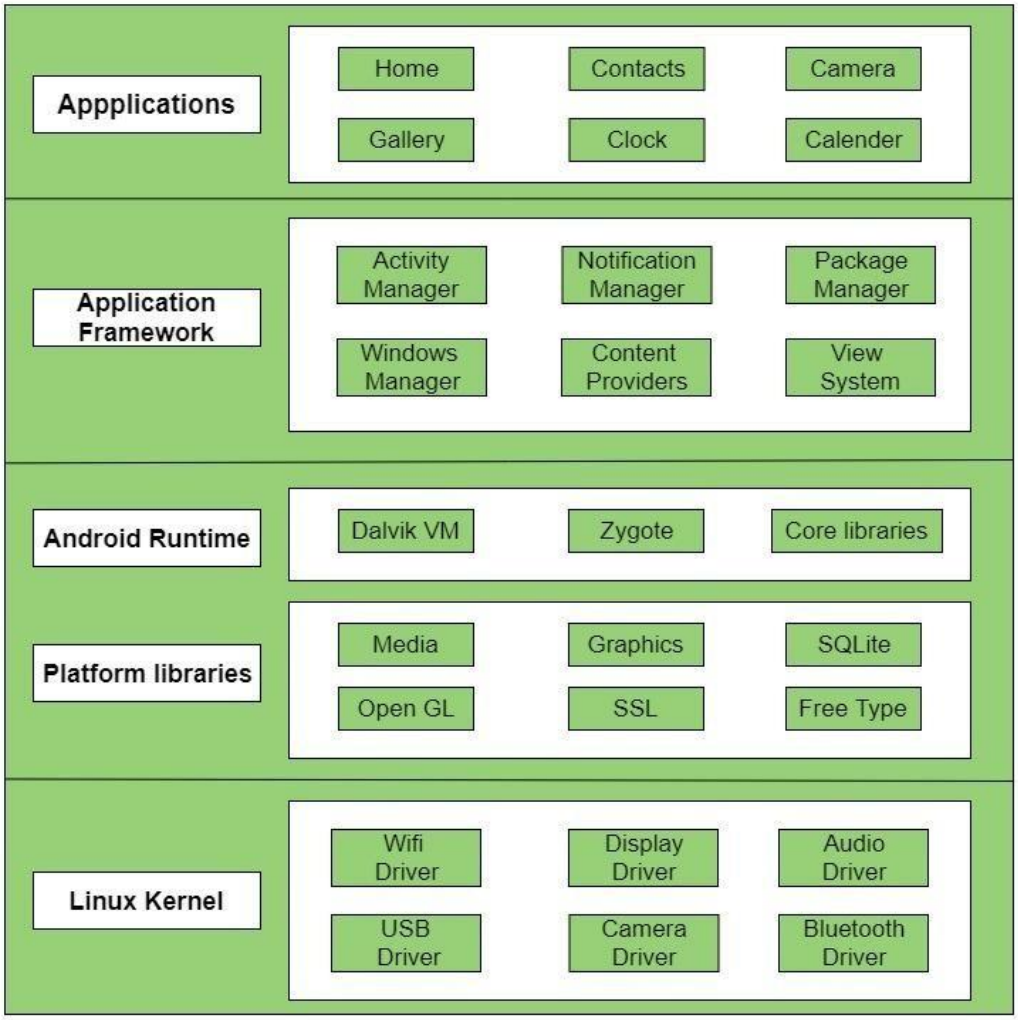

fig 1.5: Android Architecture

1.6 ANDROID APPLICATION COMPONENTS

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact.

Components & Description
Activities They dictate the UI and handle the user interaction to the smart phone screen.
Services They handle background processing associated with an application.
Broadcast Receivers They handle communication between Android OS and applications.
Content Providers They handle data and database management issues.

Fragments Represents a portion of user interface in an Activity.
Views UI elements that are drawn on-screen including buttons, lists forms etc.
Layouts View hierarchies that control screen format and appearance of the views.
Intents Messages wiring components together.
Resources External elements, such as strings, constants and drawable pictures.
Manifest Configuration file for the application.

Fig 1.6 Android components

1.7 PROBLEM STATEMENT

With the increasing number of vehicles and the decreasing efficiency of modern busyparking slots, major problems which we people are facing is:

1. Valuable time wasted from inconvenient and inefficient parking lots.
2. More fuel consumed while driving around parking lots, leading to CO2 emissions.
3. Potential accidents caused by abundance of moving vehicles in disorganizedparking lots.

Therefore, there is a need to develop an affordable system which solve the problem andobtain the information about the parking lot on real time.

1.8 OBJECTIVES

- To develop an intelligent, user friendly automated car parking system which reduces the manpower and traffic congestion.
- To improve the performance and satisfy the need of free parking lot.
- To reduces the time wastage in finding the vacant parking lot.

1.9 PROJECT APPLICATION

- This project can be implemented in shopping malls, public parking areas to monitor parking.
- Automation and increased efficiency and transparency of parking charges.
- General public can check the slots available for parking.

CHAPTER-2

SYSTEM DESIGN

2.1 APPLICATION COMPONENTS OF PROJECT

Application components are Core Building Blocks of an Android Application. It is an entry Point for System or Users from which they can enter in App.The project consists of the following components:

1. Activities:

An activity represents a single screen with a user interface, in-short Activity performs actions on the screen. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched, which is done in the manifest file. The project has 9 activities, starting from “Home Activity”, which is the first screen visible to the users once they open the app, The clicks on the views will lead to trigger of respective activities, the flow is represented below:

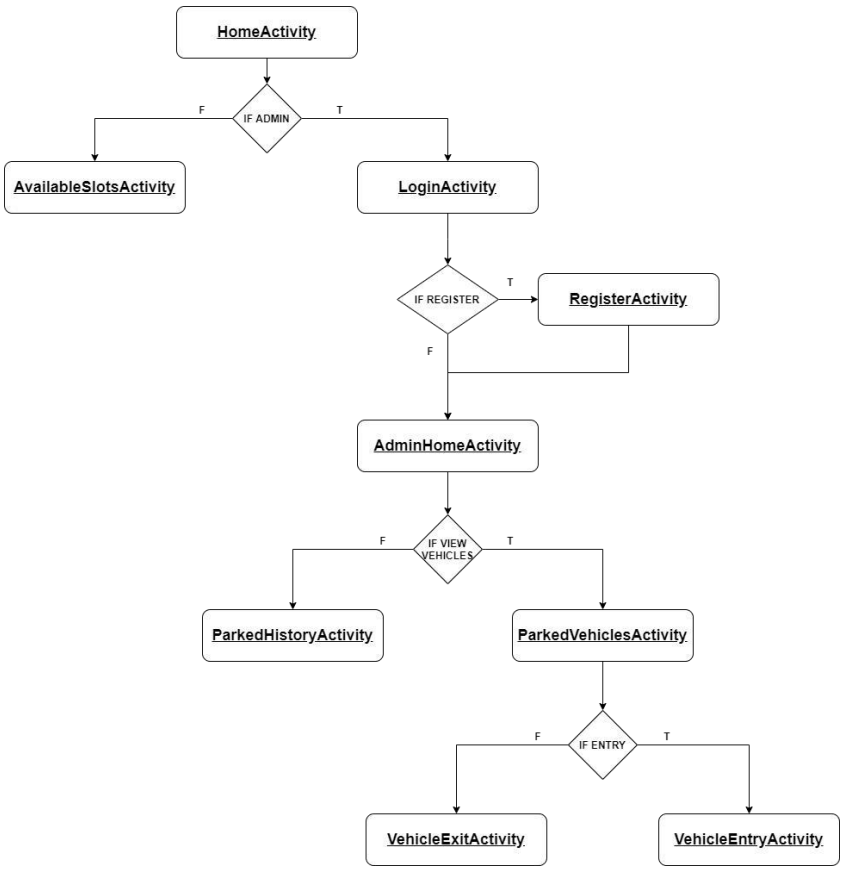

Figure 2.1.1: Flow of the Activities in Application

2. Views:

View is the basic building block of UI (User Interface) in android. View refers to the android.view.View class, which is the super class for all the GUI components. The app contains Text View, Image View, Button, Edit Text, List View, Progress Bar, Card View which helps in achieving the flow of Activities and design the layoutresponsively. Text Views are used to display the field labels and the details of the Admin. Edit Texts are used to take the details during login, register and vehicle entry. Image Views are used to display the icon of the app and List Views are used to display the slots for the users with details of number of available slots, area, name and display the admin list of parked vehicles and list of vehicles visited.

3. Layouts:

Android Layout is used to define the user interface that holds the UI controls or widgets that will appear on the screen of an android application or activity screen. The app is designed using Constraint Layout, Linear Layout, Swipe Refresh layout. Swipe Refresh layout is used in AvailableSlotsActivity for users, ParkedVehicleActivity and ParkedHistoryActivity for admin to retrieve the latest information from the database.

4. Intents, Resources:

The app communicates or transfers the control and information from one Activity to other Activity with the help of intents. The ID's, String values, Colors, Styles, XML file for every Activity, Drawable such asicons are stored in resource folder.

2.2 USER INTERFACE DESIGN

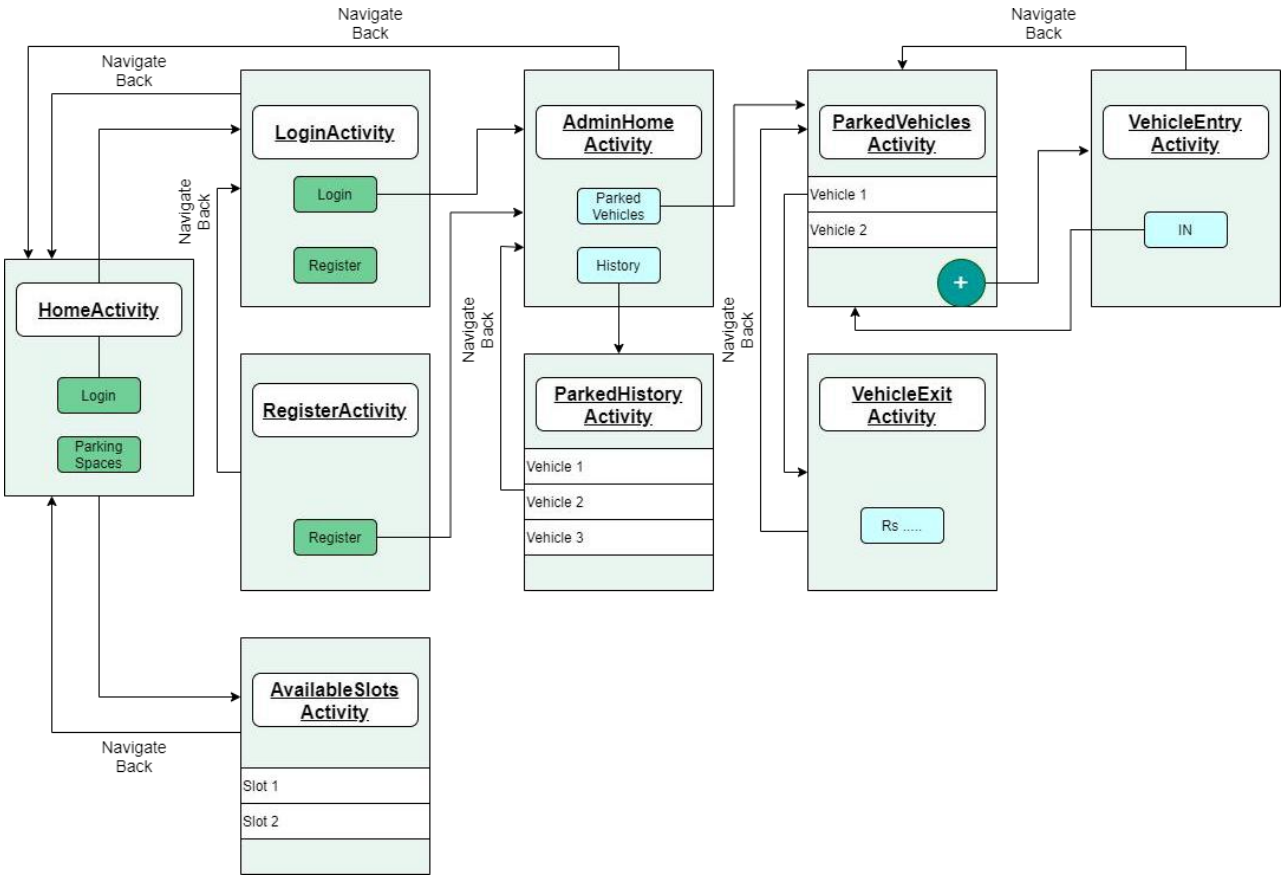

Figure 1.2: The design of Application

The user interface (UI) for an Android app is built as a hierarchy of layouts and widgets. The layouts are View Group objects, containers that control how their child views are positioned onthe screen. Widgets are View objects, UI components such as buttons and text boxes.

The Application is designed using the components of android such as, Layouts, Buttons, Text Views, Edit Texts, Card Layouts etc. The above figure gives a brief design of the application and its flow. The app is designed using Constraint Layout which is a View Group containing many Views such as Edit Texts, Buttons etc. The Constraint Layout gives the option of constraining the views with respect to the parent i.e.,View Group or other Views in the layout. The List View is designed using a Recycler View, whichis been loaded with data with the help of adapters which acts as a bridge between the View and the underlying data. These are used in AvailableSlotsActivity, ParkedVehicleActivity,ParkedHistoryActivity.

CHAPTER-3

IMPLEMENTATION

3.1 EXPLANATION OF THE MODULES WITH JAVA AND XML CODE

1. Adding data to database:

The main operation carried out throughout the app is adding the data to firebase. Starts by adding the admin details to the database after they register in RegisterActivity and then adding the vehicle number, entry time and exit time to the database for the respective slots in the VehicleEntryActivity and VehicleExitActivity, respectively.

```
public <T extends Settable> void addDataToFirestore(@NonNull T object,
@NonNull String collectionPath, @Nullable String documentPath, @NonNull fi-
nal OnGetDataListener listener) {
    DocumentReference documentReference;
    CollectionReference collectionReference = firebaseFirestore.collec-
tion(collectionPath);
    if (documentPath == null) {
        documentReference = collectionReference.document();
    } else {
        documentReference = collectionReference.document(documentPath);
    }
    object.setId(documentReference.getId());
    documentReference.set(object)
            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void unused) {
                    listener.onSuccess(documentReference);
                    makeSuccessToast("Data added successfully");
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    listener.onFailure("Adding data failed");
                    makeFailureToast("Data could not be added success-
fully");
                }
            });
}
```

2. Retrieving the data from database:

The data to be displayed in AdminHomeActivity, ParkedVehiclesActivity, ParkedHistoryActivity, VehicleExitActivity and AvailableSlotsActivty is carried out by this function, it fetches the data from the firebase, which would be added before by using the previous method.

```
public <T> void trackDocument(Class<T> className, String documentPath, On-
SnapshotListener listener) {
    DocumentReference documentReference = toDocumentReference(docu-
mentPath);
    documentReference
            .addSnapshotListener(new EventListener<DocumentSnapshot>() {
                @Override
                public void onEvent(DocumentSnapshot snapshot, Fire-
baseFirestoreException e) {
                    if (e != null) {
                        Logger.w("Listen failed: " + e);
                        return;
                    }

                    if (snapshot != null && snapshot.exists()) {
                        Logger.d(snapshot);
                        T object = snapshot.toObject(className);
                        if (object != null) {
                            listener.onSuccess(object);
                        }
                    } else {
                        Logger.d("Current data: null");
                        listener.onFailure("Current data: null");
                    }
                }
            });
}
```

3. Validating the Vehicle Number:

The vehicle number is first read through a masked Edit Text, which gives the feature of formatting the text when entered and give the restriction on the length etc.

After reading the vehicle number, it is validated using regular expression methods available in java. We have considered most recommended and common format of a vehicle i.e., 2 letters, 2 digits followed by 2 letters and 4 digits. Each of this has its meaning where the first 2 letters represent the state at which the vehicle has registered (Ex: KA – KARNATAKA), the next 2 digits indicate sequential number of a district(Ex : 03 – BANGALORE EAST), the next 2 letters shows the ongoing series of an RTO, and the last 4 numbers are unique to each plate.

XML

```
<com.santalu.maskara.widget.MaskEditText
    android:id="@+id/vehicle_edit_text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="144dp"
    android:layout_marginRight="20dp"
    android:background="@drawable/bg_vehicle_input"
    android:drawableStart="@drawable/ic_car"
    android:drawableLeft="@drawable/ic_car"
    android:drawablePadding="12dp"
    android:ems="10"
    android:hint="@string/aa_00_bb_1111"
    android:inputType="textCapCharacters"
    android:paddingStart="12dp"
    android:paddingLeft="5dp"
    android:paddingTop="10dp"
    android:paddingEnd="12dp"
    android:paddingRight="5dp"
    android:paddingBottom="10dp"
    android:textAllCaps="true"
    android:textColor="@color/black"
    android:textColorHint="@android:color/darker_gray"
    android:typeface="monospace"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:mask="__-__-__-____"
    app:maskCharacter="_"
    app:maskStyle="normal"
    tools:ignore="MissingClass" />
```

JAVA

The regex used is (**"^[A-Z]{2}[-][0-9]{2}[-][A-Z]{2}[-][0-9]{4}$"**) which is matched with the vehicle number using matches() method, which returns true if the number matches with the regex else returns false.

After this the available space value will be reduced by 1 and updated to the firebase for that building, and the entry time and vehicle number will get stored in a new collection named “parked-vehicles” and during exit of vehicle the exit time will get stored, the available space will be increased by 1 and updated, The bill will be calculated based on the minute difference between exit and entry time, respectively.

```
if (!vehicleNumber.equals("AA-00-BB-1111")) {
    Logger.d(vehicleNumber);
    if ((vehicleNumber.matches("^[A-Z]{2}[-][0-9]{2}[-][A-Z]{2}[-][0-
9]{4}$"))) {
        maskEditText.setText("");
        Logger.d(vehicleNumber);
        new AlertDialog.Builder(this)
                .setTitle("Insert entry")
                .setMessage("Are you sure you want to insert " +
vehicleNumber + "?")
                .setPositiveButton(android.R.string.yes, (dialog, which) ->
{
                    in.setVisibility(View.INVISIBLE);
                    wait.setVisibility(View.VISIBLE);
                    ParkedVehicle parkedVehicle = new ParkedVehicle(null,
vehicleNumber, Timestamp.now());
                    String collectionReference = Constants.PARKING_SLOTS +
"/" + userId + "/" + Constants.PARKED_VEHICLES;
                    fbHelper.addDataToFirestore(parkedVehicle,
collectionReference, null,
                            new OnGetDataListener() {
                                @Override
                                public void onSuccess(DocumentReference
dataSnapshotValue) {

Toast.makeText(VehicleEntryActivity.this, "added successfully",
Toast.LENGTH_LONG).show();
                                    int updatedAvailableSpace =
availableSpace - 1;
                                    Map<String, Object> map = new
HashMap<>();
                                    map.put("availableSpace",
updatedAvailableSpace);

fbHelper.updateField(parkingSlotDocumentStr, map, new OnSnapshotListener()
{
                                        @Override
                                        public <T> void onSuccess(T object)
{
                                            makeToast(object.toString());
                                            finish();
                                        }
                                        @Override
                                        public void onFailure(String
errorMessage) {
                                            makeToast(errorMessage);
                                        }
                                    });
                                }
                                @Override
                                public void onFailure(String str) {
                                    Logger.e("Error adding document " +
str);
                                }
                            });
                })
                .setNegativeButton(android.R.string.no, null)
                .show();
    } else {
        Logger.e("wrong");
        makeToast("wrong format");
```

CHAPTER-4

SCREENSHOTS

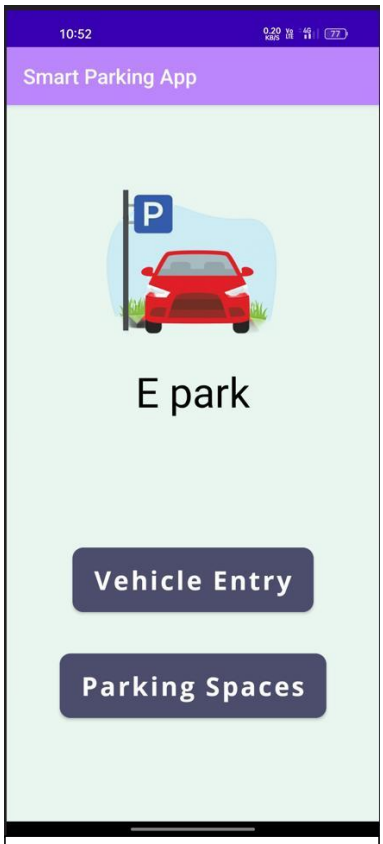

Fig. 4.1.1 Home Activity

Fig. 4.1.2 Admin Home

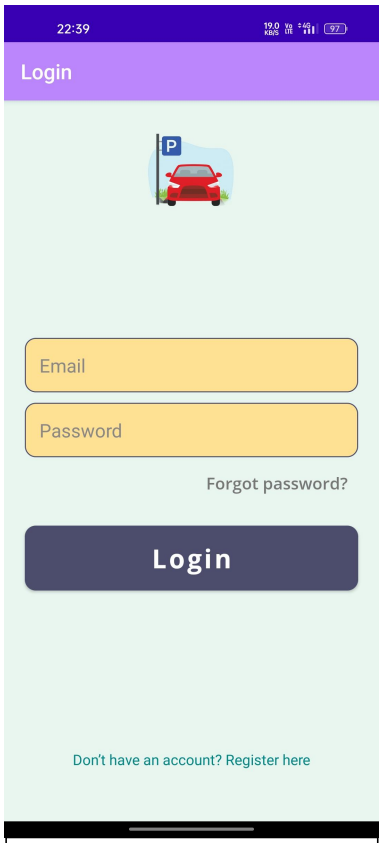

Fig. 4.1.3 Login Activity

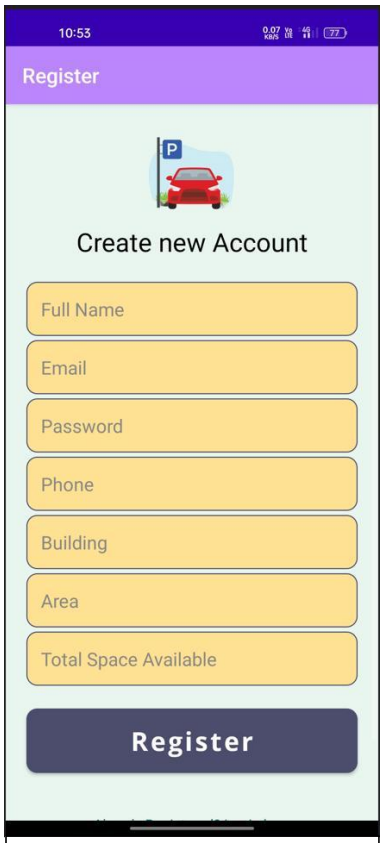

Fig. 4.1.4 Register Activity

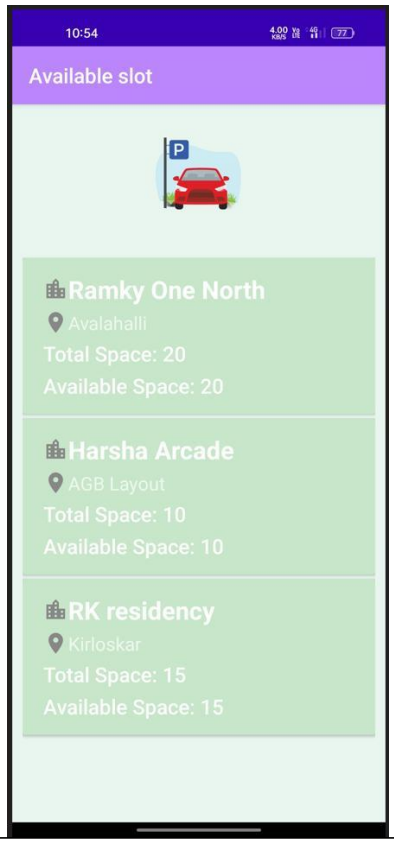

Fig. 4.1.5 Available Slots Activity

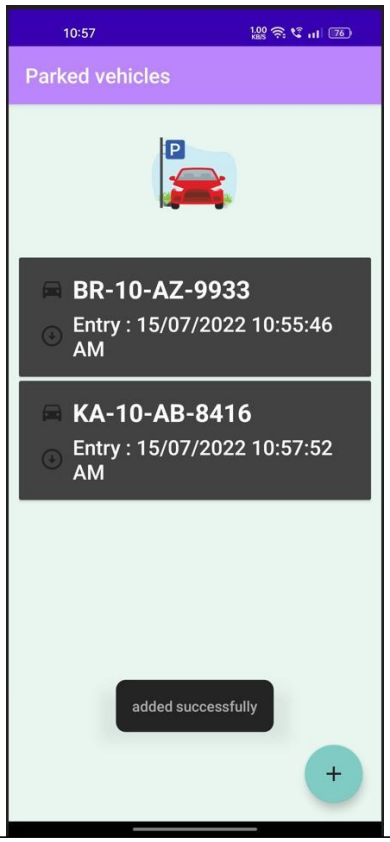

Fig. 4.1.6 Parked Vehicles Activity

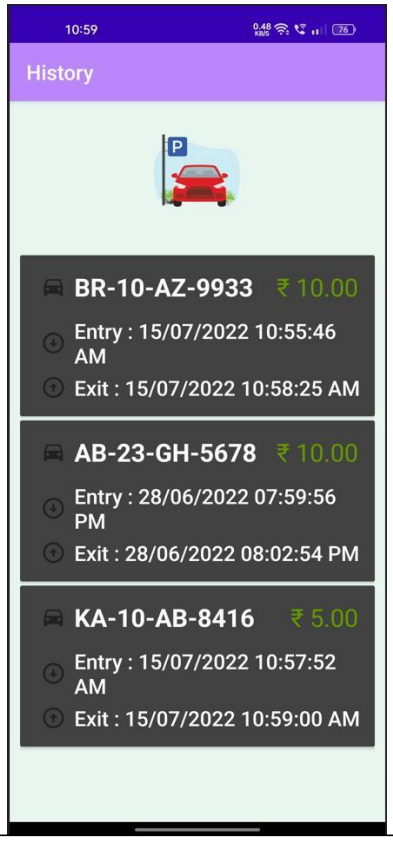

Fig. 4.1.7 Parked History Activity

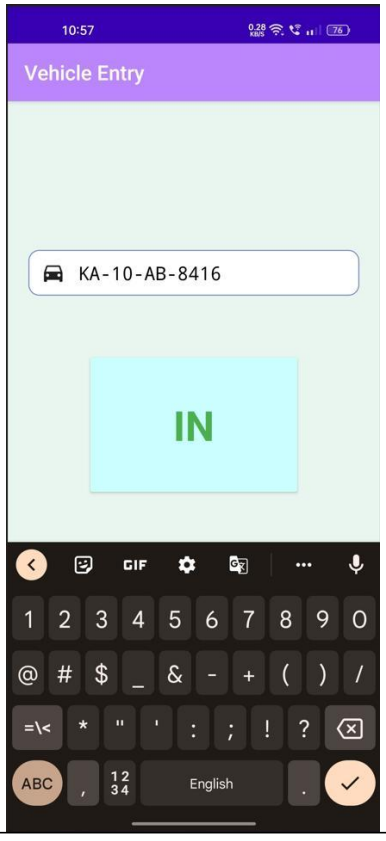

Fig. 4.1.8 Add Vehicle Activity

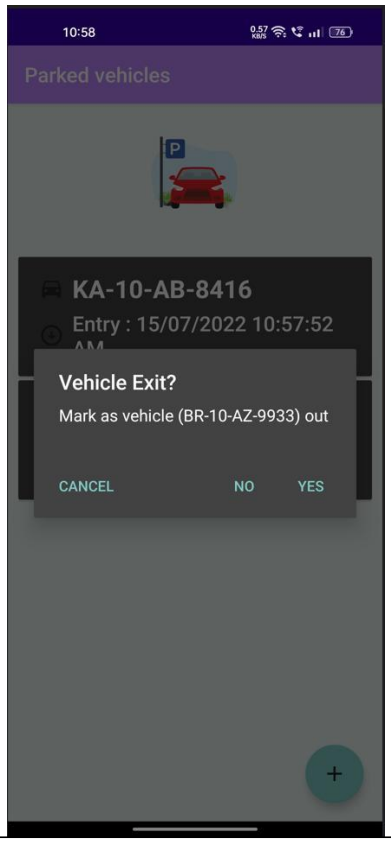

Fig. 4.1.9 Remove Vehicle Alert Dialog

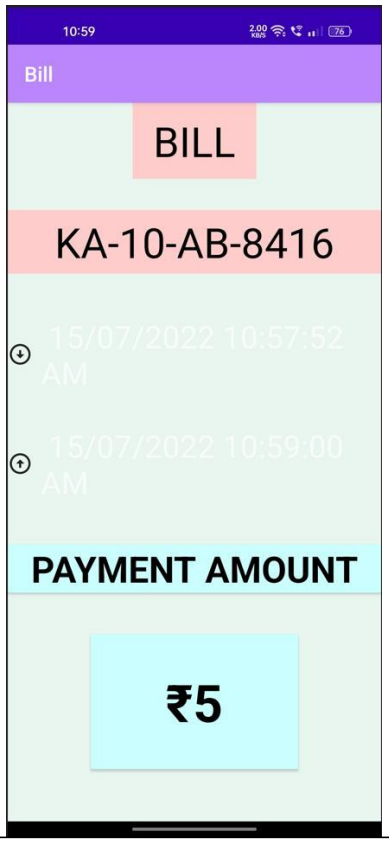

Fig. 4.1.10 Vehicle Exit Activity

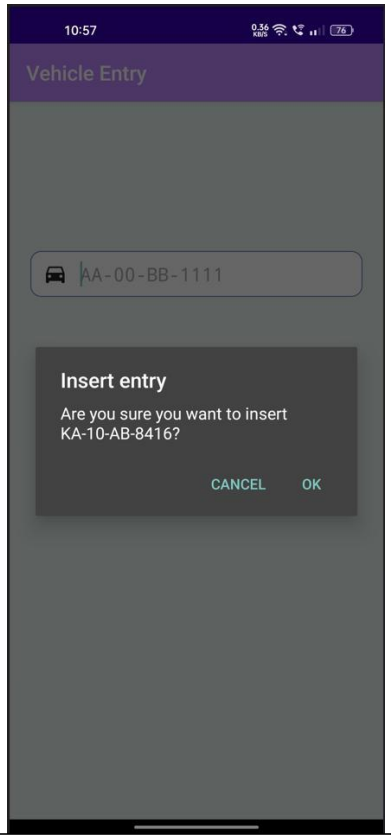

Fig. 4.1.11 Add Vehicle Alert Dialog

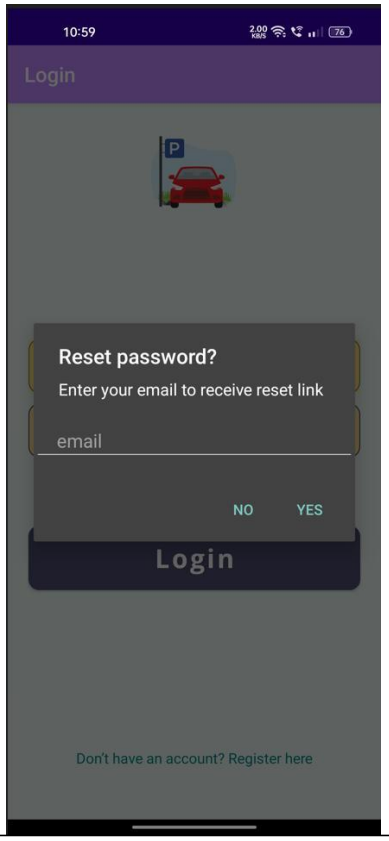

Fig. 4.1.12 Forgot Password Alert Dialog

CHAPTER-5

APPLICATION TESTING

Table 5.0Application Test

System Unit	Test Case	Result
Login Activity	Click on Login button	Authenticate user based on the input user credentials
Login Activity	Click on Register TextView	Navigates to Register Activity
Admin Login Activity	Click on the Parked Vehicles button	Navigates to Parked Vehicles Activity
Admin Login Activity	Click on the History button	Navigates to Parked History Activity
Admin Login Activity	Click on the Logout button	Signs out the currently authenticated user from the app
Home Activity	Click on the Login Page button.	Navigates to Login Activity
Home Activity	Click on the Parking Spaces button.	Navigates to Login Activity
Available Slots Activity	On opening this Activity	Displays a list of Parking Spaces that are available
Parking Vehicles Activity	On opening this Activity	View the currently parked vehicles in the Parking Space
Parking Vehicles Activity	Click on Floating Action Button	Navigates to Vehicles Entry Activity
Parked History Activity	On opening this Activity	View the previously parked vehicles in the Parking Space

Register Activity	click on Register	A user account will be created if the entered
Vehicles Entry	click on IN button	Adds a vehicle into the database if the entered
Parking Vehicles	Click on added Vehicle Card	Alert Dialog pops up asking to whether toremove vehicle from database. On successful removal of the vehicle, app
Vehicles Exit	On opening this	View the auto generated bill for the
ForgotPassword Alert	Click on Yes button	Sends a password reset email to the entered

CONCLUSION AND FUTURE ENHANCEMENTS

The parking management system app is effective in detecting the available parking slots through real-time data collection. In addition to that the app can be a part of a bigger revolution by reducing thetraffic in a city by informing the user about the parking spaces. The manual process of entering the vehicles in the vehicle register book is eliminated. Further enhancement is required to make the user view his or her nearby parking areas. Overall system can handle all the requests from the User to give a fresh digital look for current parking management system

Online booking of parking slots in a parking space/area so that users can reserve the parking slot for his/her vehicle. Redesigning the user interface of the app. Creating a profile page for the admin of the parking space so that he/she can update not just his/her profile but also details regarding the parking space/area. Integrating the app with Google maps to easily locate or navigate to the parking area. A feature to view the nearby parking areas.

REFERENCES

[1]. M. Owayjan, B. Sleem, E. Saad and A. Maroun, "Parking management system using mobile application," 2017 Sensors Networks Smart and Emerging Technologies (SENSET),2017, pp. 1-4, doi: 10.1109/SENSET.2017.8125048.

[2]. C. Ng, S. Cheong, E. Hajimohammadhosseinmemar and W. Yap, "Mobile outdoor parking space detection application," 2017 IEEE 8th Control and System Graduate Research Colloquium (ICSGRC), 2017, pp. 81-86, doi: 10.1109/ICSGRC.2017.8070573.

[3]. B. K. Patil, A. Deshpande, S. Suryavanshi, R. Magdum and B. Manjunath, "Smart Parking System for Cars," 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE), 2018, pp. 1118-1121, doi: 10.1109/ICRIEECE44171.2018.9008662.

For Websites:

[1] https://www.seminarsonly.com/Engineering-Projects/Computer/smart-parking-system.php

[2] https://opengeekslab.com/blog/parking-app-development/

[3] https://projectworlds.in/android-projects-with-source-code/android-smart-parking-app