1. INTRODUCTION

1.1 Overview

The Parking Navigator is a set of two Android application that helps users quickly find and easily pay for parking spots. First app is for the users who have already registered themselves and the second app is for the administrator present at the parking lot helping people who have not registered on the app but still want to park.

1.2 Scope

This project requires location, camera, internet, etc. permissions from the user. One would be able to check the availability in a parking lot remotely which would save him time and indicate the user how busy a certain mall or a place is. In addition, the app would help the user navigate inside a parking lot and thus reducing any human assistance.

The goal is to create system efficient enough to utilize available parking space and keep track data and report to user accordingly. The online system, contributing to the digital age, to keep away printed parking tickets.

1.3 Purpose

This project would help solve parking issues that have been arising due to the lack of planning from the government side while passing building plans for cooperate offices and authorizing even if there is no provision for parking of fifty to hundred vehicles.

Sometimes it happens that we go to find a parking in somewhere and once we get there, we find that there is no available space for parking. This would waste a person’s time and he would have to go find parking somewhere else. Our application would provide a user with availability of parking space at a particular parking lot. Providing this information would help the user make an informed decision.

In addition to booking a parking space, our app would also provide navigation to the same. This feature would reduce the time required to find the parking slot. And it also helps in efficient record keeping for the past entries which used to be difficult in previous cases.

2. FEASIBILITY ANALYSIS

2.1 Technical Feasibility

It is feasible to use this application in Android version 6.0 and above. Also, if the above-mentioned requirements are present then there would be no issues. According to sources from Wikipedia and Android Studio, the version requirements cover 86 per cent of the android users as of March 2018.

2.2 Time Schedule Feasibility

The project is feasible in the given timeframe for the implementation. The application assigns slot and produces receipt for billing purposes in matter of seconds. The only place where a user might have to wait is when he would be waiting for OTP and after slot booking.

The part which consumed most of the scheduling time of project development was the OCR detection which included Firebase ML tool called Google Vision for which we had to have the thorough knowledge.

2.3 Operational Feasibility

With this new system we would be able help people know where parking spaces are available in different parking slots also the mode of payment would become much easier with the system calculating the time and charging based on the rental price formerly decided.

Camera focus, which was and is one of our main concerns while operating and testing the application. Since there are range of android devices to run our application, the hardware feasibility is to be taken into consideration. For example, high-end, costly phones like Samsung Note 9 has scored excellent on our camera focus test while OCR detection while some low-end, budget devices like Mi Note 3 has scored moderately.

2.4 Implementation Feasibility

Application requires android environment to run with fast internet for firebase access. Since we are using a free SMS service and depend on carrier network for interaction between user and operator, there might be some lags in the process.

2.5 Economic Feasibility

The cost to complete this project would depend on the number of users and the cost that would have to paid to the message services and the firebase for analysis and cloud functions. In addition, for the payment API upfront money needs to be submitted and thus implementing payment interface is not feasible now.

3. SOFTWARE REQUIREMENTS AND HARDWARE REQUIREMENTS

- An android smart phone with android version minimum 7.0 i.e. the Nougat version.
- Maximum android versions 9.0 i.e. Pie
- A Firebase Account
- Android Studio
- A working computer with minimum 8Gb ram
- High Speed Internet

4. PROJECT PLANNING AND SCHEDULING

4.1 Technology Selection

As we decided to choose a topic that was related to everyday task it seemed natural to choose the one thing that would be common among all public and which people would have with them all the time. Thus, we decided to implement this project in Android.

4.2 Database Selection

Due to the real time updating needed by the system and the availability of data at all time we decided to use Firebase as our database as it provides real time updates to data is already hosted on cloud which would make it available all the time.

4.3 Developed Optical Character Reader (OCR)

We decided for the betterment and the need to reduce human error we needed and optical character reader that would read the vehicle's license plate and identify the number there which would be used to book and end a parking slot rental. This is one of the most important modules of our project. This would help maintain data integrity and automate the process.

4.4 Developed Operator Module

First, we developed the operator's app which is the app installed on the device of the operator present at the parking lot. Here we first integrated the OCR for the car number detection that would be used to book a spot. The application would then check if the user with that car number is registered or not. If he is not, then operator needs to enter basic information but if user is already registered then information would be preloaded.
During the exit of a vehicle operator would search for the car number in occupied list and end the rental time which would lead to receipt generation.

4.5 Developed User Module

Once the app is installed in user's device user needs to sign up by entering required information including adding car number using OCR. Once the user reaches the parking lot operators would scan users number plate and a spot would be assigned to the user which would be sent to the user using a SMS. In the SMS there would be a link that clicking upon which user would be directed toward the user app and where he would find the navigation needed to reach the parking spot assigned to the user.

All the above-mentioned tasks have been performed and are in finishing phase. Only the payment interface remains which would be developed if this app is to be implemented as using such API would cost money which is feasible now.

5. SYSTEM DESIGN

5.1 Use Case

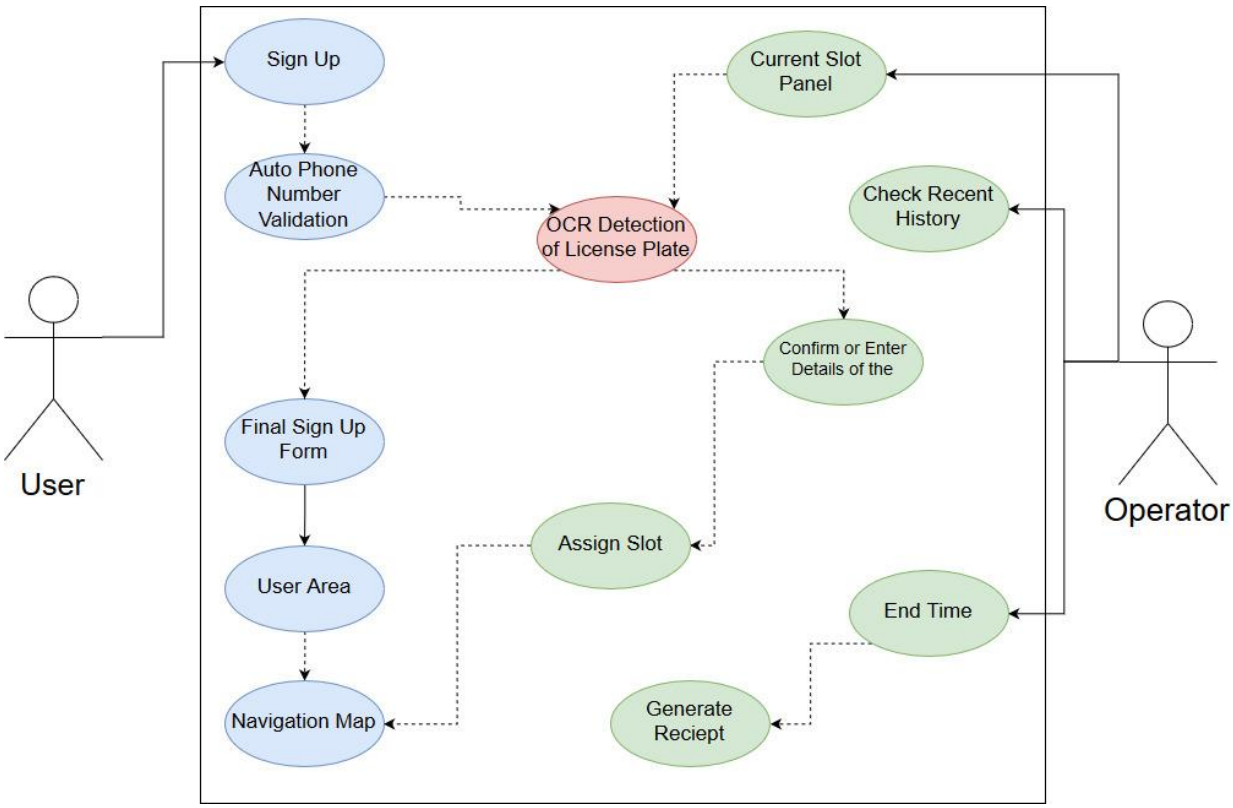


Figure 5.1.1: Use Case Diagram

5.2 Activity Diagram

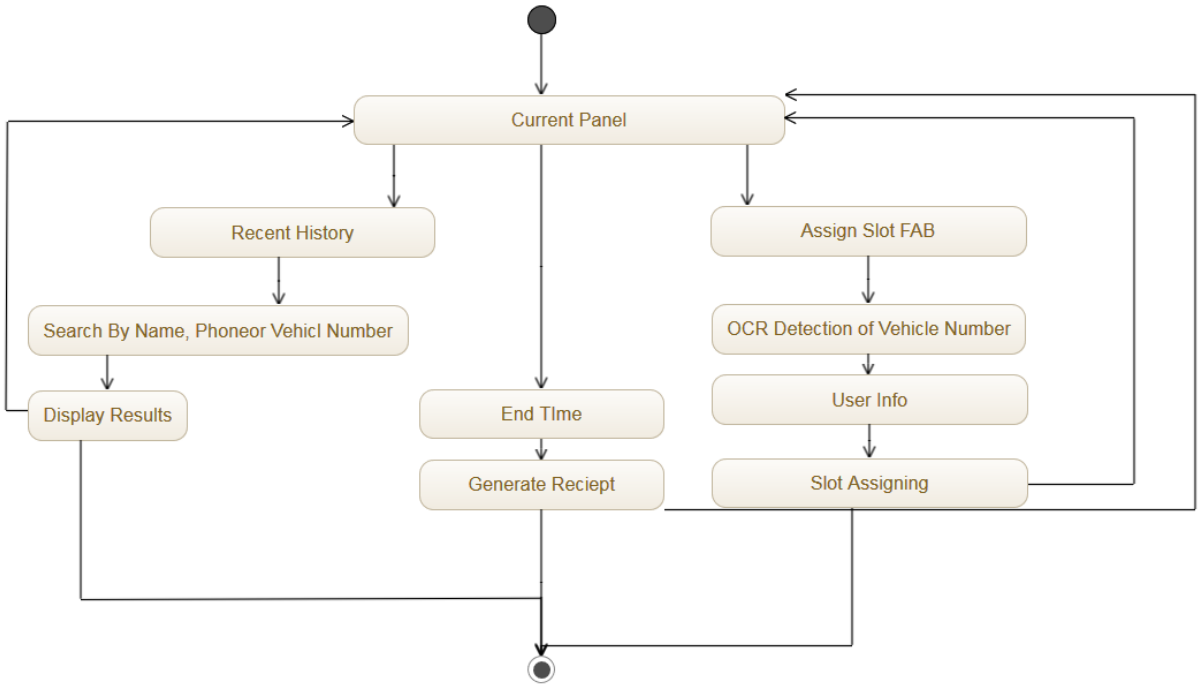


Figure 5.2.1: Activity Diagram Operator

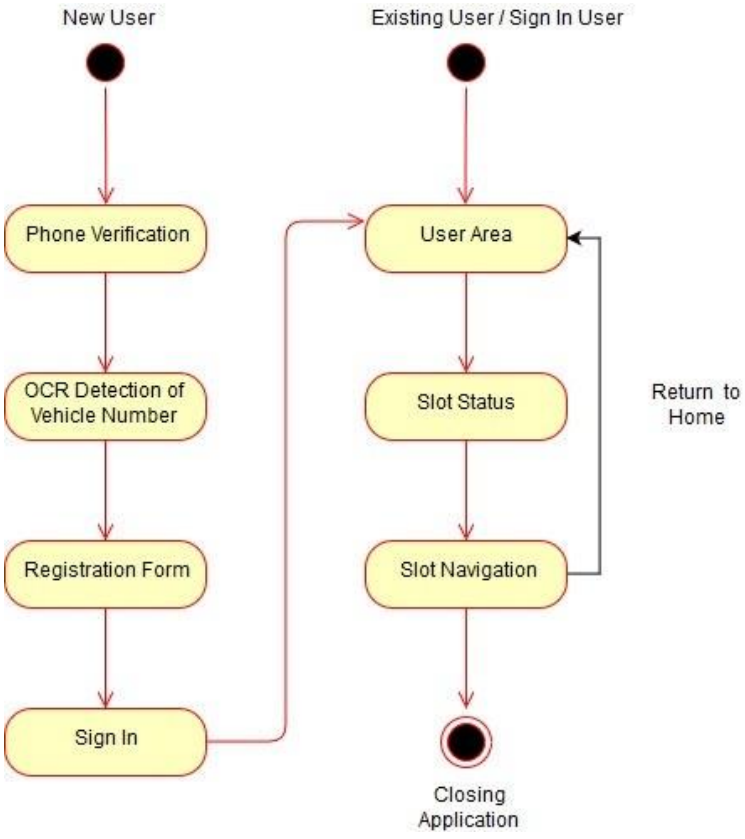


Figure 5.2.2: Activity Diagram User

5.3 Data Flow Diagram

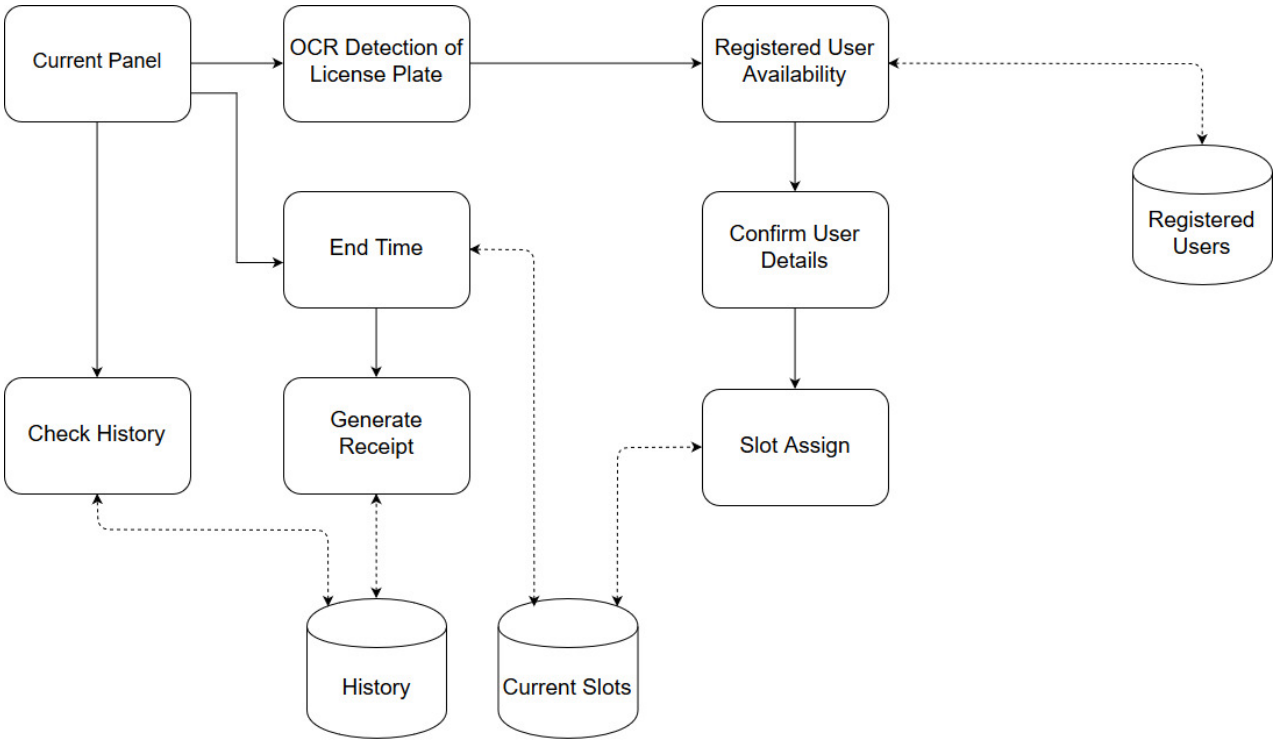


Figure 5.3.1: Data Flow Operator

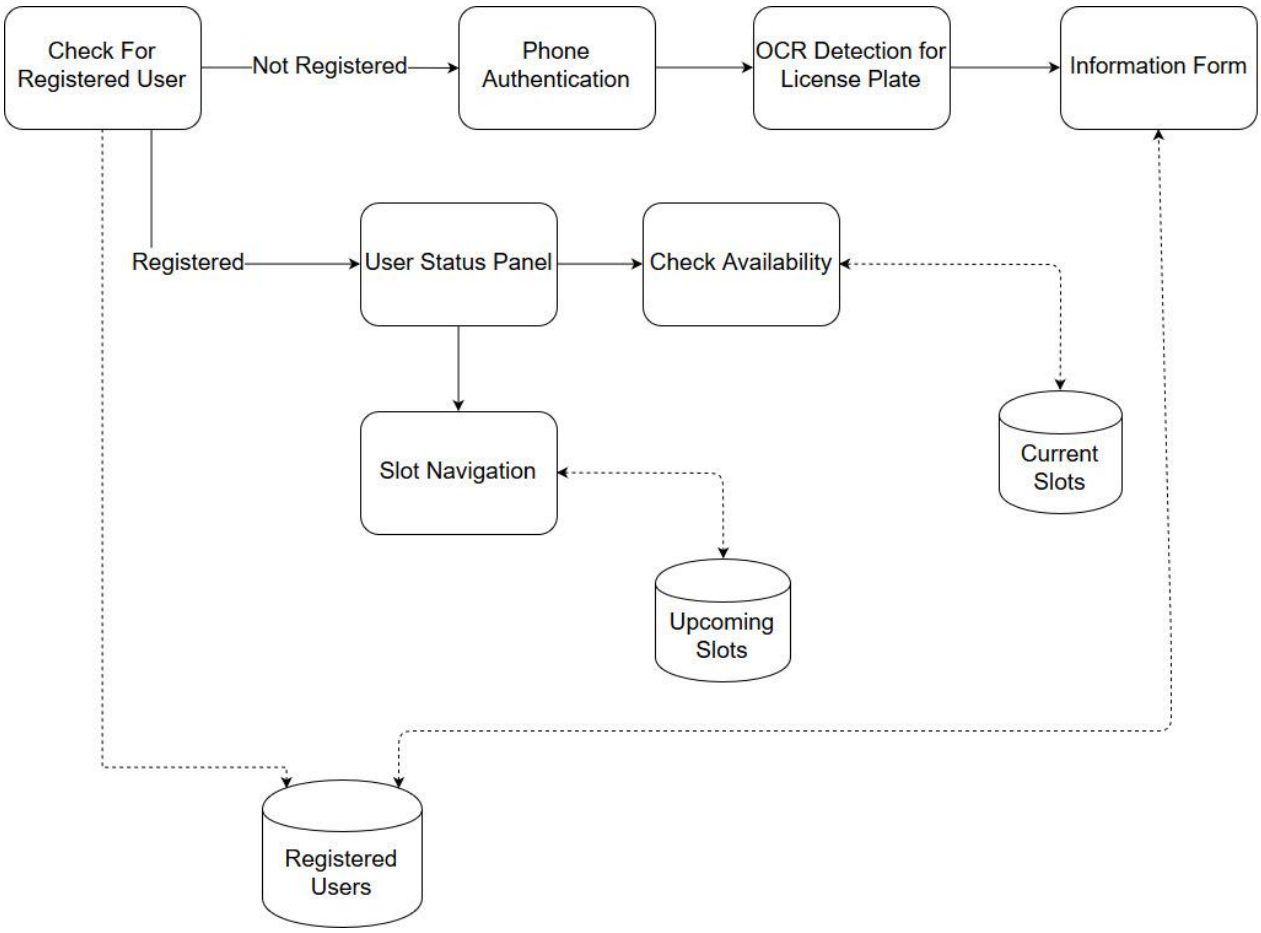


Figure 5.3.2: Data Flow User

5.4 Data Dictionary

Table 5.4.1: Current Slots

Column Name	Data Type	Description
Date	String	Date of Parking for billing purpose
Phone	String	Users phone number for SMS purpose
SlotNo	String	Slot Assigned to the user
StartTime	String	Current time stored in milliseconds for charging purpose
Status	String	Shows if slot is occupied or not
Username	String	Name to be recognized by the application chosen by user at time of sign up
VehicleNo	String	Car number scanned with help of OCR during sign up.

Table 5.4.2: History

Column Name	Data Type	Description
Amount	String	Amount charged for parking for a particular time
Date	String	Date at which parking happened
EndTime	String	To keep track when a car left the building
PhoneNo	String	Storing contact information of the parker
SlotNo	String	Keep track where the parker parked.
StartTime	String	Keep track when the parker entered the parking lot.
Username	String	Parkers username
VehicleNo	String	Car number of the vehicle parked

Table 5.4.1:Registered User

Column Name	Data Type	Description
PhoneNo	String	Users registered mobile number
Name	String	Parkers username
VehicleNo	String	Car number of the vehicle parked

Table 5.4.4: Upcoming Slots

Column Name	Data Type	Description
PhoneNo	String	Users registered mobile number
SlotNo	String	Keep track where the parker parked.

5.5 Firebase JSON

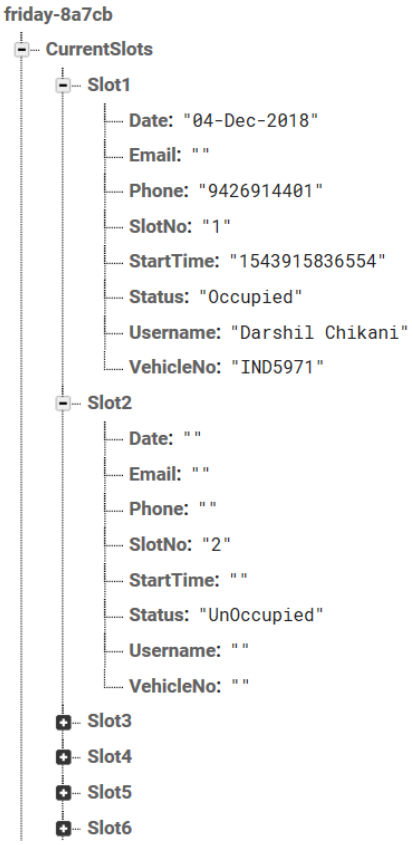
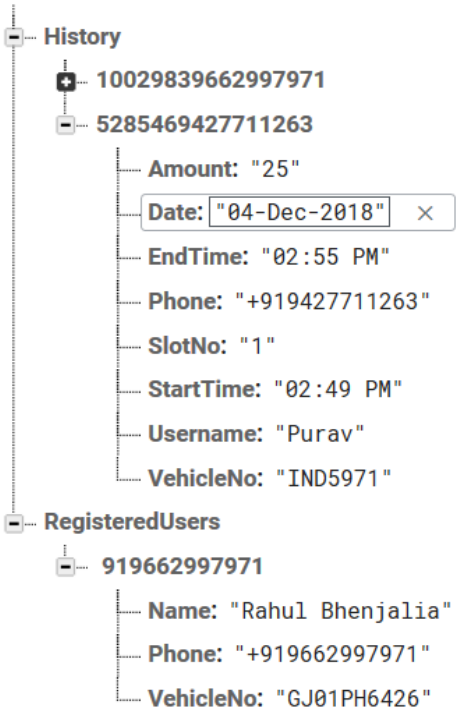


Figure 5.5.1: JSON Snapshot



5 SNAPSHOT WITH BRIEF DESCRIPTION

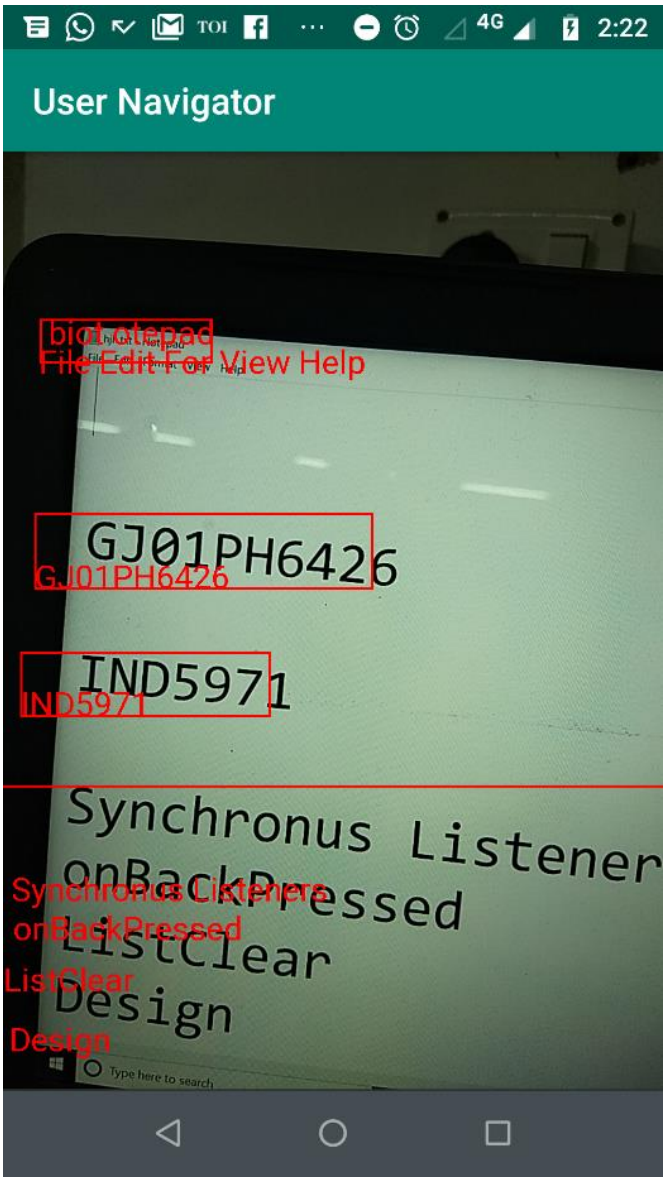


Figure 1.1: OCR for Registering Car

The user after installing the app would be asked to register the vehicle, he wants with the app for doing that OCR activity be opened and the user would choose his car number from all the data recognized by the user.

Figure 6.2: Inserting Personal Information

After detecting the car number application would require the user to enter his personal information, so that further process can be performed using that personal information.

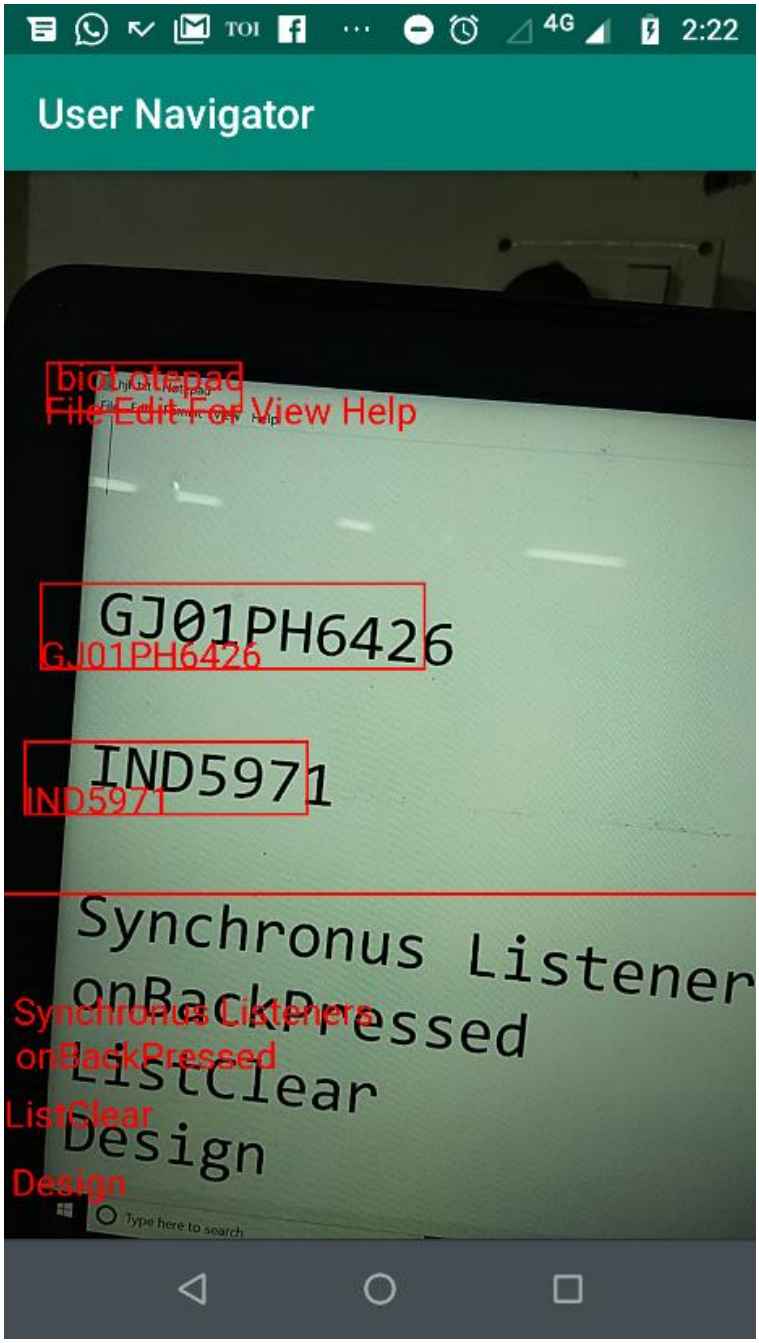


Figure 6.3: Operator Scanning the arrived vehicle

On arrival of a car operator scans the car number if the car number is registered then the information is preloaded else it needs to be asked and entered by the operator.

Figure 6.4: Registered User Arrives

Information is preloaded with no editing permissions for the operator thus data integrity is maintained. On clicking get slot an unoccupied slot would assign to the user.

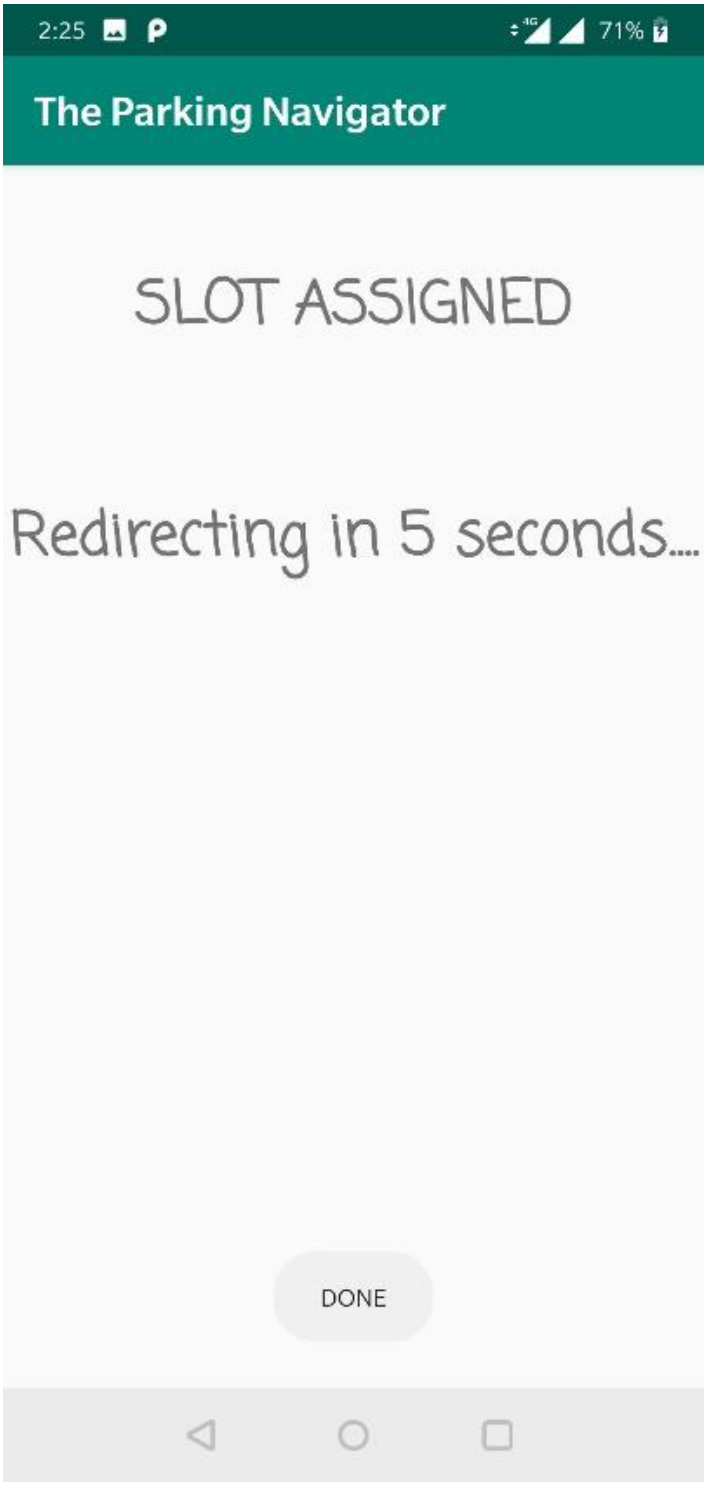


Figure 6.5: After Slot assignment on operator side

Figure 6.6: After Slot Assignment on Operator Side

Wait till slot is assigned. After 5 seconds slot would be assigned, and a SMS would be sent to the registered mobile number corresponding to that car number with slot number and a link to navigate to the spot

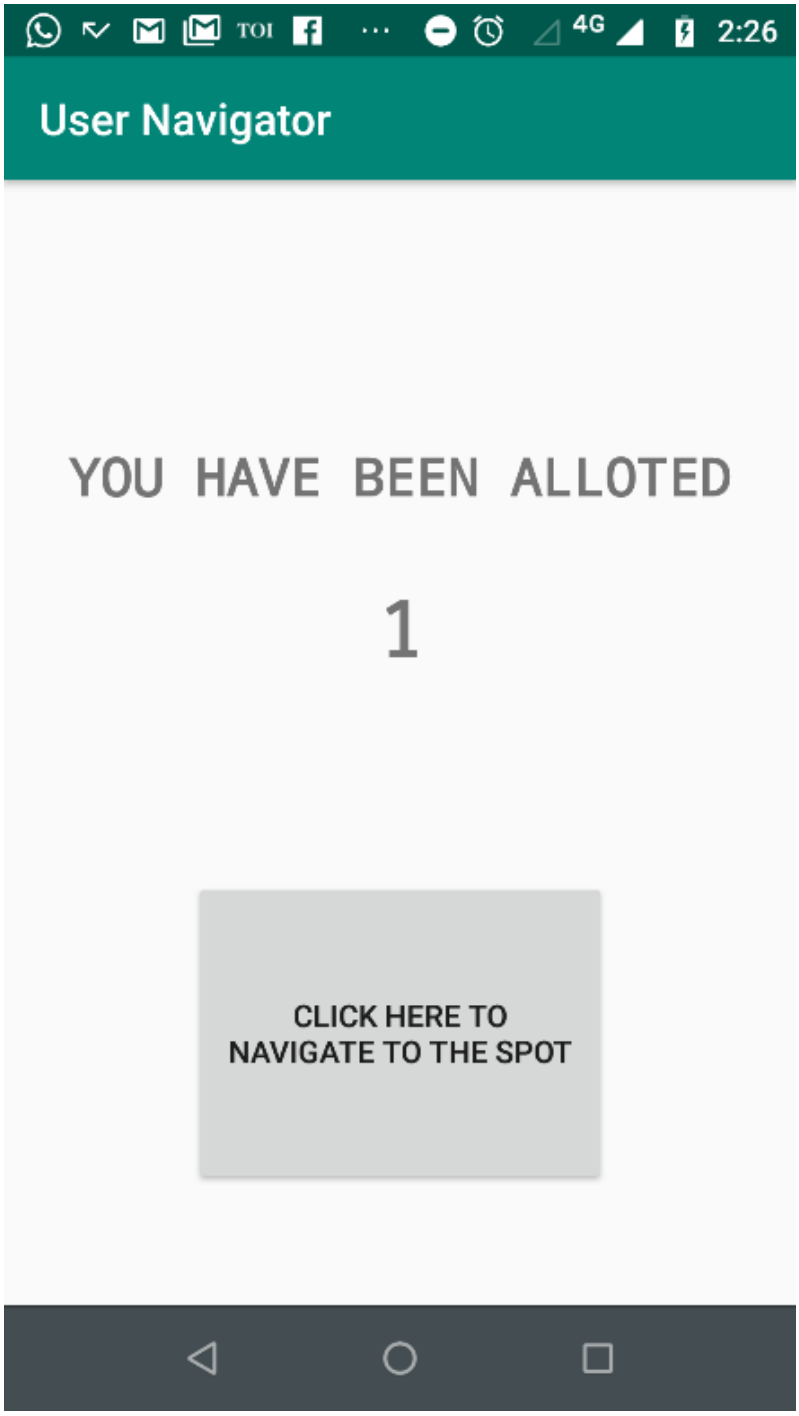


Figure 6.7: After Slot Assignment on User Side

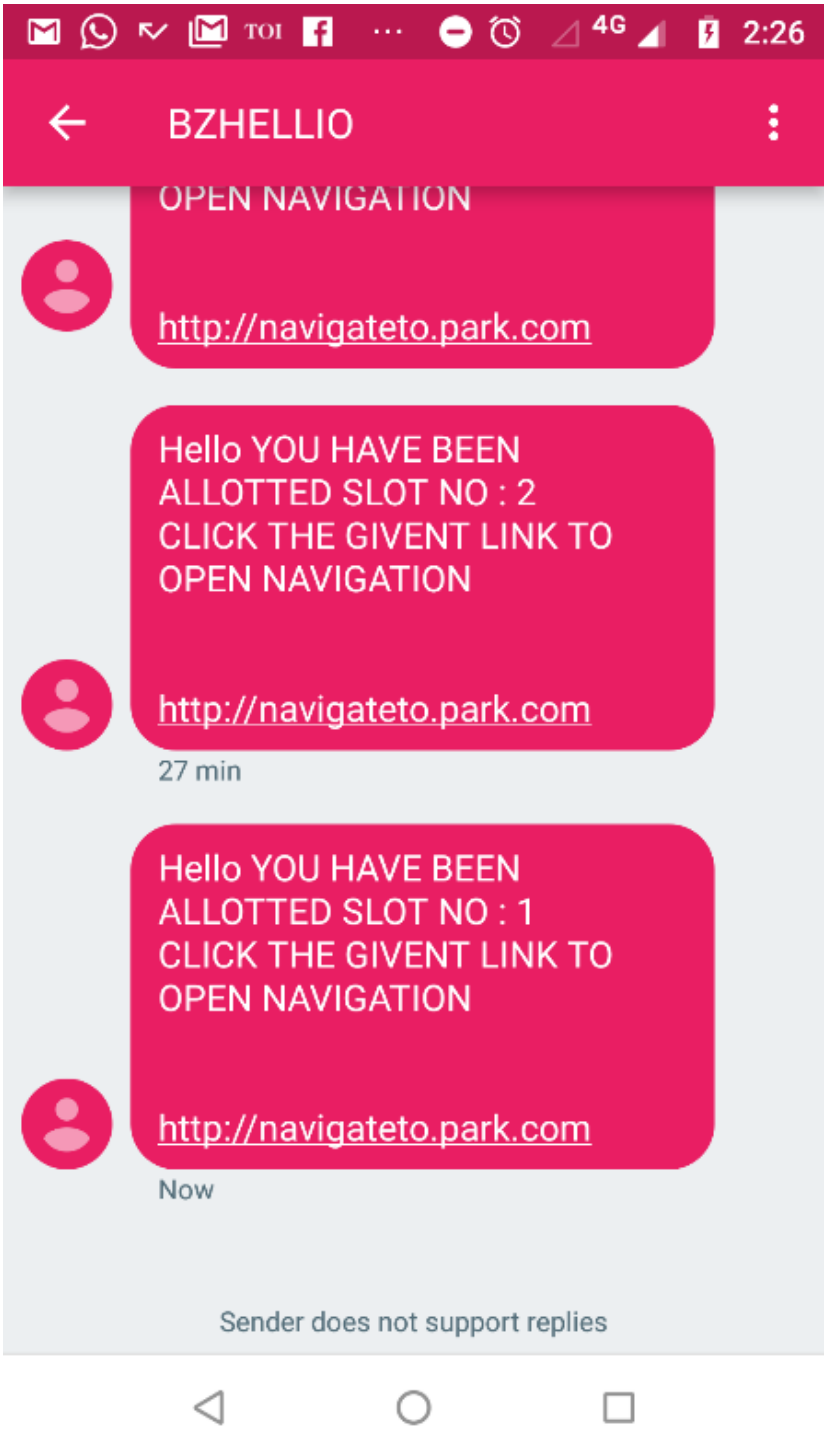


Figure 6.8: After Slot Assignment on User Side

Here on the user side first an SMS would be sent which would contain a link to open the app from where user can navigate to his assigned parking slot. On that screen users assigned slot number would be displayed.

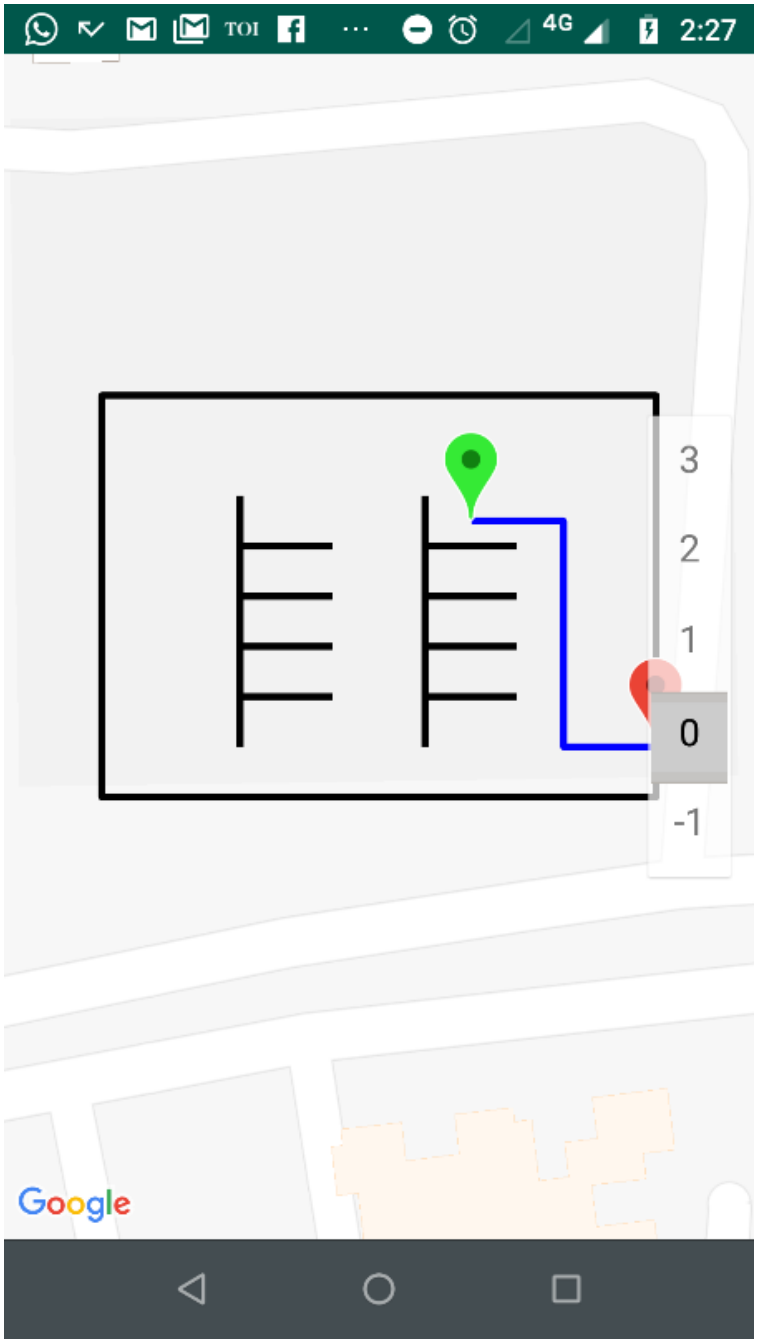


Figure 6.9: Navigating to Slot 1

Red mark indicates entrance to the building, blue line shows the path to follow and green mark represents destination.

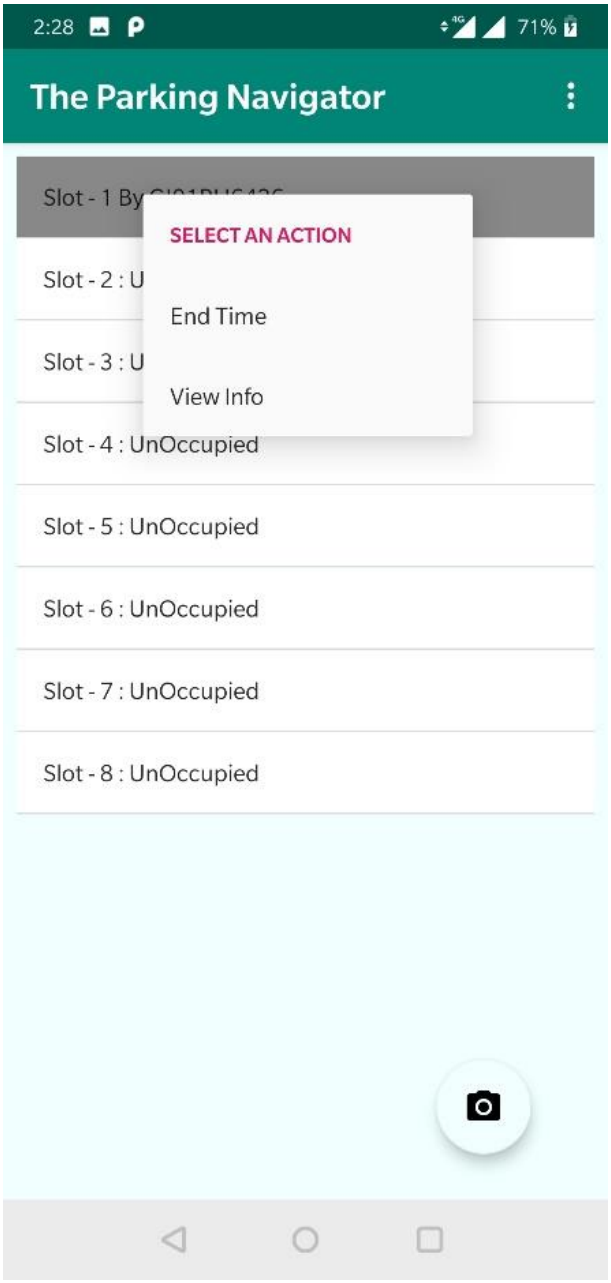


Figure 6.10: End time

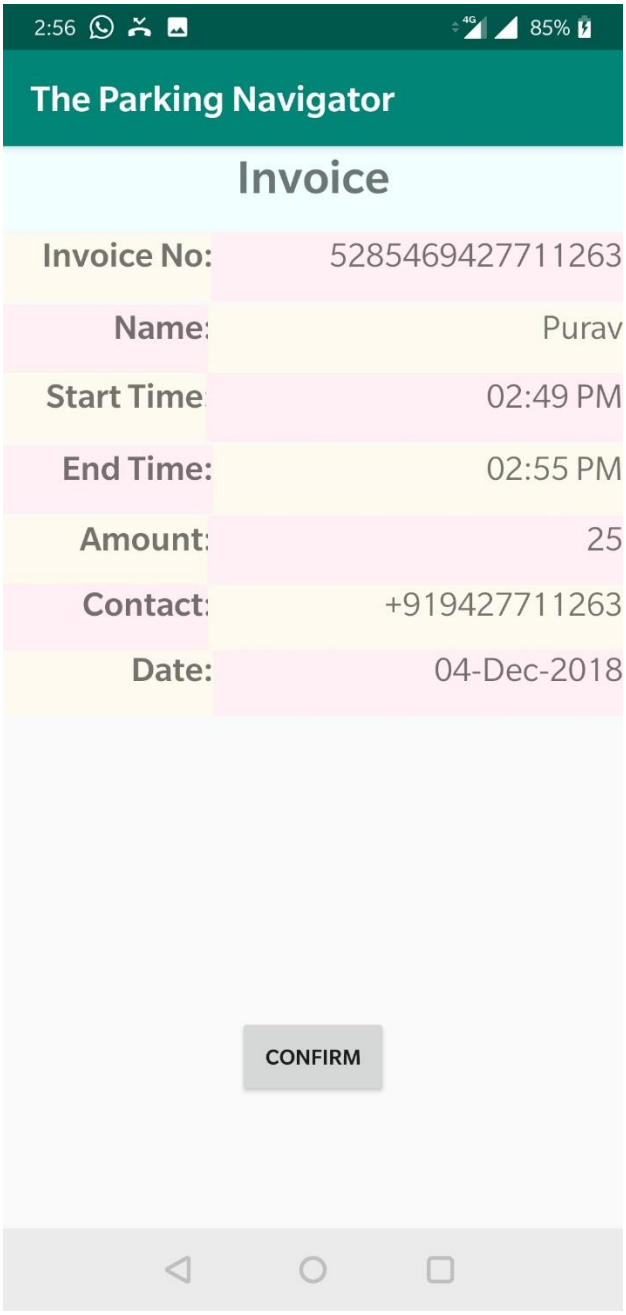


Figure 6.11: Receipt Generation

On ending time by the operator on exiting the parking lot a reciept is generated which is added to the history database for record keeping and provide bill to the customer.

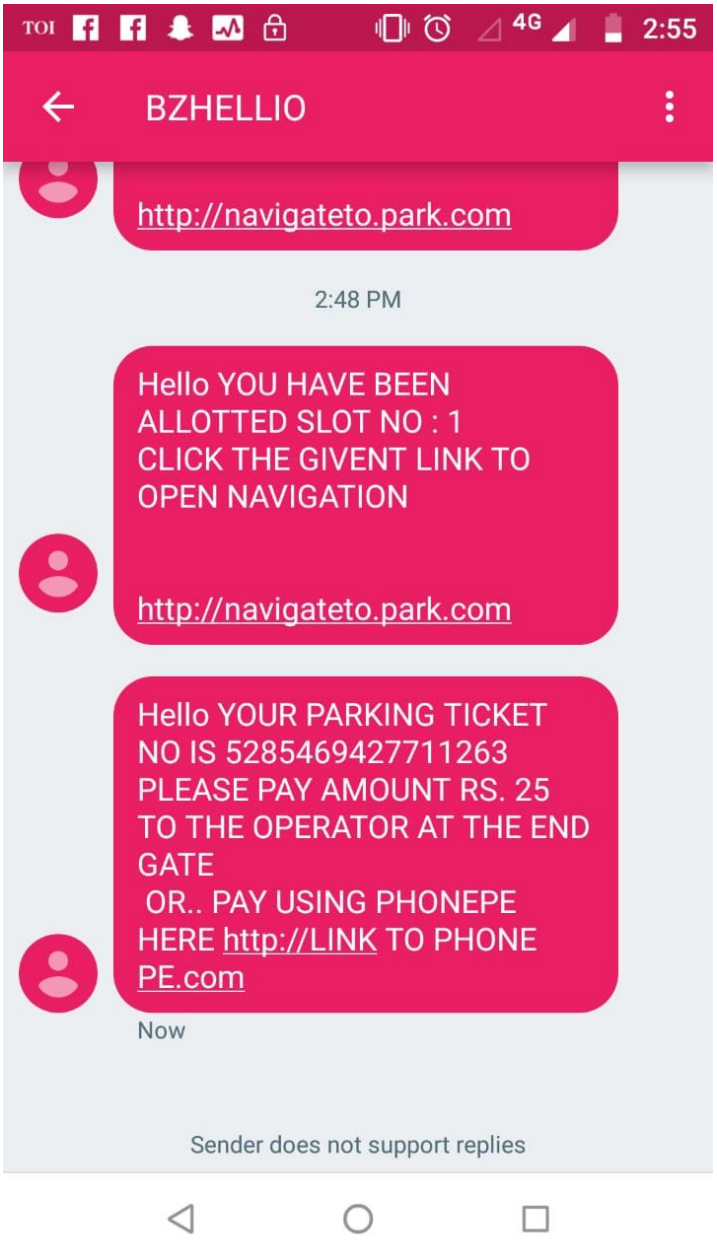


Figure 6.12: Receipt Confirmation

Customer will receive a confirmation message with amount to be paid and receipt number.

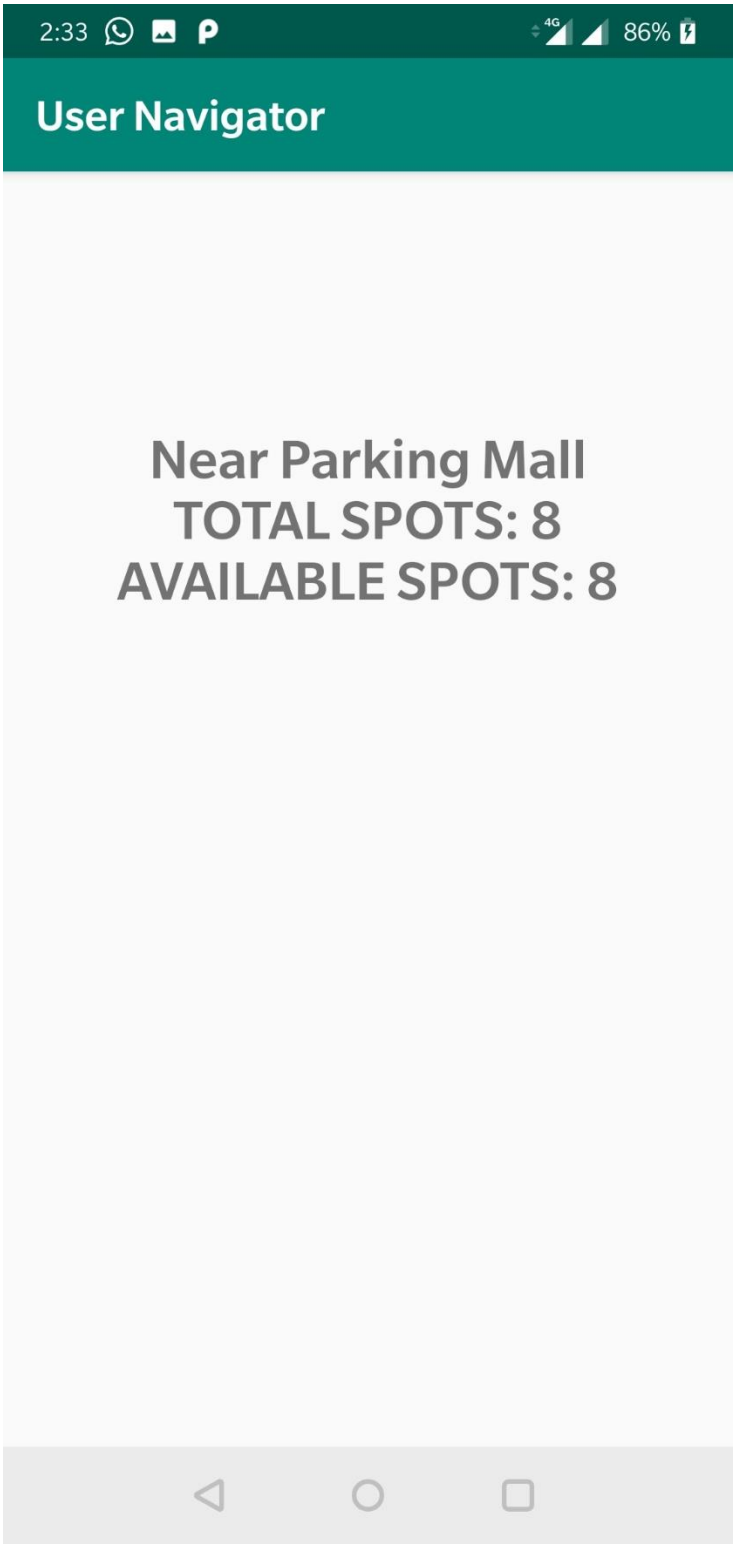


Figure 6.13: Receipt Confirmation

Customer can check availability of the parking spots in nearby areas

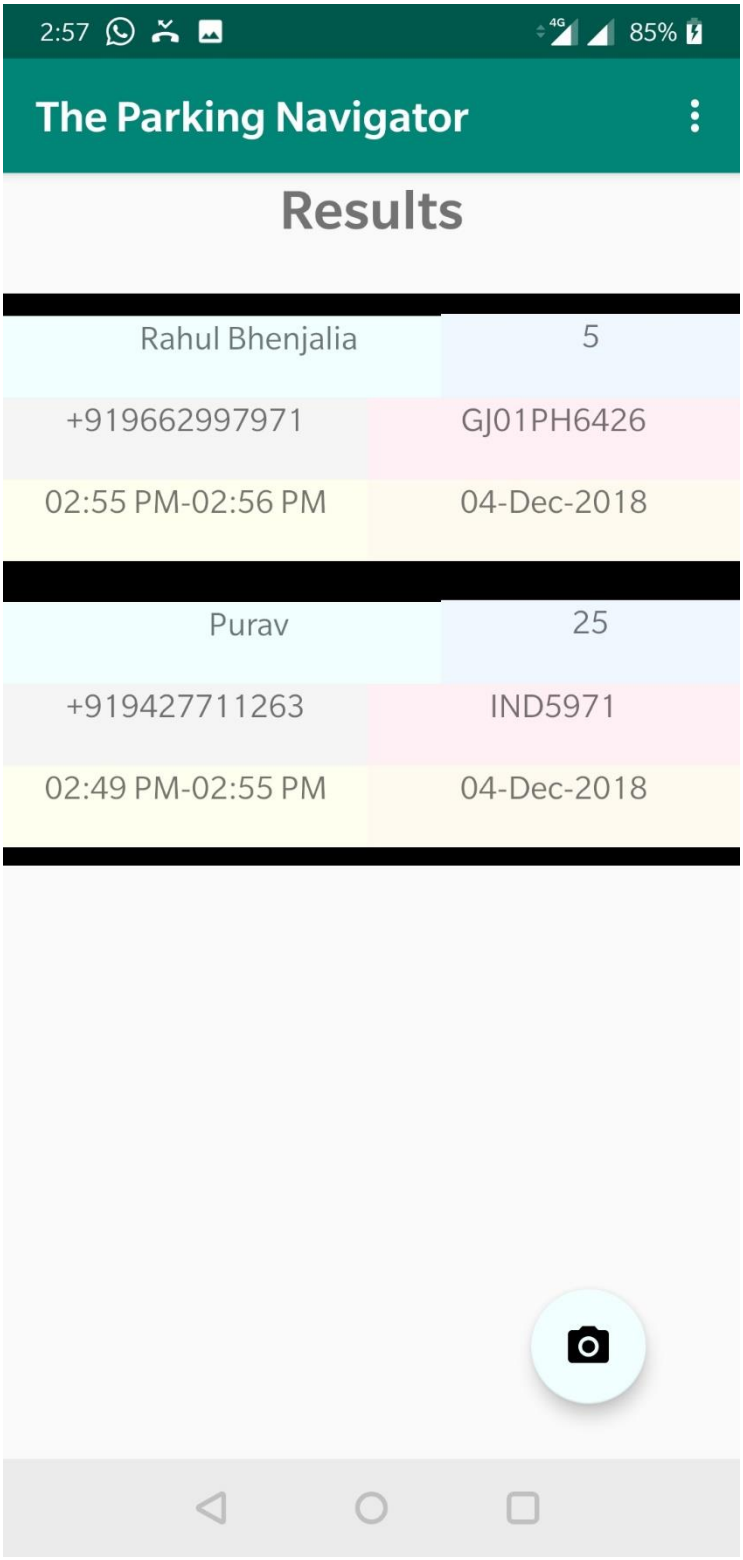


Figure 6.13: Check History

Operator can also check the recent history records.

6 FURTHER ENHANCEMENTS

The project has a broad-spectrum scope in future. The project can be implemented on world-wide in future. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. With the proposed software of Friday Firebase database ready and fully functional the client is now able to manage and hence run the entire work in a much better, accurate and error free manner. The following are the future scope for the project.

- Performing the same for more than one parking lot
- Allowing user to select a particular parking lot and display its details
- Implement payment interface
- Displaying parking lot with most available spots
- More efficient search method

7.1 Payments API Integration (PhonePe SDK)

PhonePe Android Merchant SDK makes integrating PhonePe on Android Apps quick and effortless. It supports collecting payment, viewing PhonePe accounts among many other features. PhonePe SDK provides functionalities for completing the payment inline without having the need to stepping out of merchant app for anything.

Merchants should ensure:

1. To collect the Test Environment(UAT) credentials.

2. Complete the integration and testing in the test environment.

3. Share a debuggable signed build for the sanity check with PhonePe team.

4. Collect Production credentials post UAT sign-off.

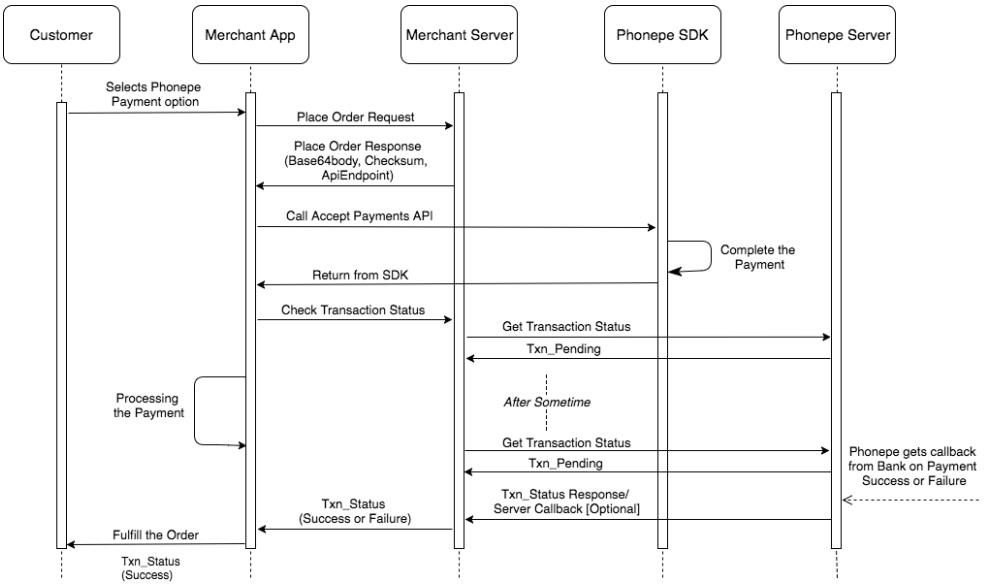


Figure 7.1.1: System Flow of PhonePe

```
buildscript {
    repositories {
        jcenter()
        maven {
            url 'https://maven.google.com/'
            name 'Google'
        }
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.0.0'
    }
}

allprojects {
    repositories {
        jcenter()

        maven {
            url "<url>"
            credentials {
                username "<username>"
                password "<password>"
            }
        }

    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Figure 7.1.1: Dependencies of PhonePe

7 CONCLUSION

To conclude, Project Parking Navigator works like a component which can access all the parking lot databases and picks and assign slots to users efficiently. It overcomes the many limitations incorporated in the current scenario of parking lot taxations.

8 REFERENCES AND BIBLIOGRAPHY

1. https://firebase.google.com/docs/ml-kit/
2. https://firebase.google.com/docs/database/android/
3. https://firebase.google.com/docs/auth/android/
4. https://firebase.google.com/docs/in-app-messaging/
5. https://github.com/firebase/quickstart-android
6. https://www.draw.io/
7. https://www.msg91.com/
8. https://github.com/PhonePe/MerchantAndroidThinSDKDemo
9. https://developer.phonepe.com/v1/reference#api-usage-android