PARK IT

Abstract: Present day's car parking has become a major issue in urban area with limited parking facilities. It is very difficult and frustrating to find a parking space in most metropolitan areas, especially during the rush hours. To solve this problem, we have proposed an application that provides an easy way for reservation of parking spots. Our overall goal is to develop an automated system to count the empty parking spaces in a parking lot, given only an image of the lot and the maximum capacity of the lot as input.

Keywords: Android application, Smart Parking System.

1. INTRODUCTION

Searching for street parking in crowded urban areas creates many problems and frustration for drivers. It has been shown that over 40% of the total traffic volume in urban areas is composed of vehicles cruising for parking.

With the rapid proliferation of vehicle availability and usage in recent years, finding a vacant parking space is becoming more and more difficult and time consuming. Parking problems are becoming ubiquitous and ever growing at an alarming rate in every major city. The use of Android technology combined with the recent advances in Machine Learning could be the key to solve emerging parking problems.

The main idea behind the Automated Car Parking System is to help user find nearest vacant parking spots. The user can pre-book a slot in the area he desires. This will help reduce the load on the administrator as his physical work reduces drastically. If a vacant parking spot is available, the user is the provided with the directions to the spot. Thus the application proposed makes the user relieved as it reduces the time required for manually searching and waiting for empty slots to park the vehicle. The system also provides an additional feature of cancelling the bookings.

2. PROPOSED SYSTEM

2.0 Technology Stack

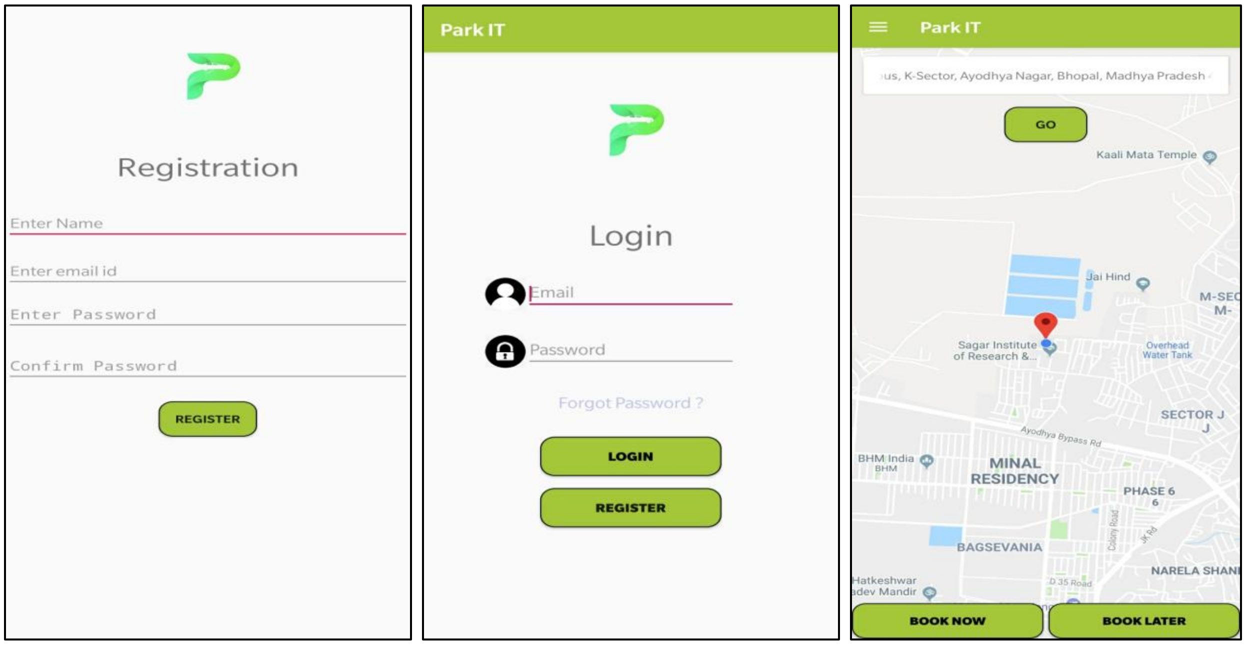
Server side : Django Framework, Tensorflow & Keras, Google Distance Matrix API

Client side : Android, Firebase Authentication, Firebase Real time database, Google maps API, Google Places API, Google direction API , Google son parser, volley, Instamojo for payments.

2.1 Making a parking request

First and foremost, a user needs to register him on the "Park It" mobile application so that he can use the parking services available. It is a onetime process and takes just few seconds. The registration process is done using Firebase Authentication. Once the user fills his credentials, an API request is fired; if the API request is successful then the user is successfully registered; else the user

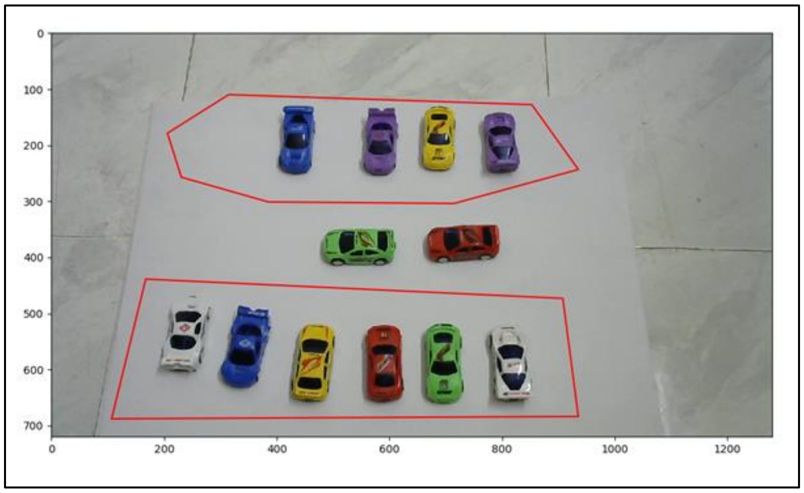
will be prompted with an error message asking him to try again. Data of the users who have successfully registered on our system is stored in the firebase real-time database. (JSON format)
Once the user gets registered, he/she can login to the application using the login page. By chance if the user forgets his/her password, he/she can recover it in few seconds.
Once the user selects the location nearby which he needs to park his vehicle(in case of pre-booking) or his own location and clicks on the Book Now button, a GET request containing the latitude,



longitude of the user's location and a key unique to the user(required when the user arrives at the parking area) gets fired.

2.2 Identifying vacant spaces in parking areas

The parking area owner will install the cameras (which are often present) in his parking lot and keep a note of the maximum capacity of the parking lot. It is not necessary that the parking lot have demarcated cells. The parking owner will be guided with the initial steps of installation using a manual.
He has an additional facility to specify the region of interest in the images captured by the installed cameras so that the redundant regions are neglected. To specify the region of interest is as simple as using a Paint tool, about which will be elucidated in the manual.
Following is the depiction of a road-side view where the cars on the road are of no interest to us.



The images from the camera will be sent to the server (running on cloud). The server uses a Mask-RCNN algorithm for object detection to detect and localize cars in the image. The parking lot will be declared to contain vacant space only if the number of cars detected would be less than the maximum capacity of the lot.

2.3 Backend Django Server

The Server built using the Django Web Framework responds to following three API calls:

i) Get Nearby Parking Lots API

ii) Booking Cancellation API

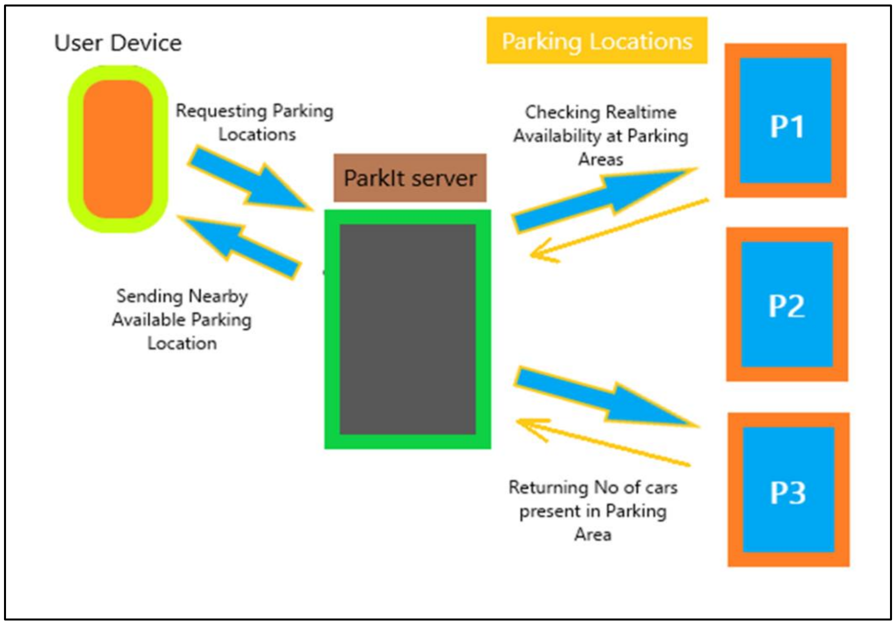
iii) User Verification API

Whenever a user requests for a nearby parking spot, the Android application fires Get Nearby Parking Areas API. The Django server receives this request containing the user's location. Using Google Distance matrix API, nearby parking lots are searched iteratively (within a radius of 200m in first iteration, 300m in the next iteration and so on till 500m). Then a request is made to existing parking areas to get the parking lot images for inference. Using our inference engine we calculate the number of cars present in the particular parking lot. If a vacant spot is available in the nearest found parking lots, we do need iterate further. The database maintains records consisting of maximum capacity of parking lots and records consisting of customers who would have pre-booked and would not have arrived at the parking lot at that time.

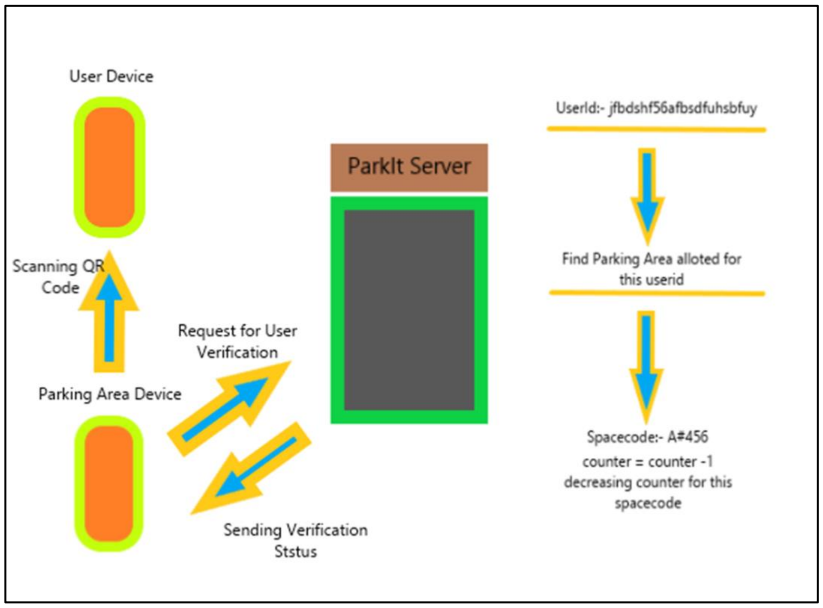
Using these findings, we calculate the total number of parking spots available for that particular parking spot.

No of parking spots available = (Max Capacity)-(No of cars detected)-(No of bookings made)

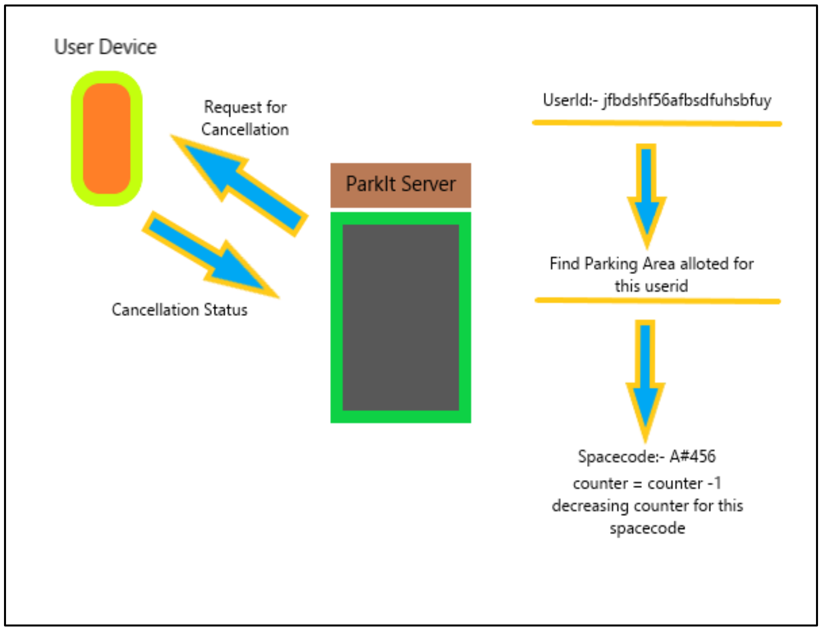
If a vacancy is available in a parking lot, then location of this parking lot is send back to mobile application.


User Device
Parking Locations
Requesting Parking Locations
Checking Realtime Availability at Parking Areas
ParkIt server
P1
P2
P3
Sending Nearby Available Parking Location
Returning No of cars present in Parking Area

When a user arrives at a parking spot, parking spot guard employed to verify user bookings scans the user's id using his device. Upon scanning, the counter stored in the database which holds the number of customers who have pre-booked but not parked the vehicle is decreased. User Verification API is fired from the parking lot guard's device and our backend verifies if a booking associated with a particular user-id exists.


User Device
Scanning QR Code
Parking Area Device
Request for User Verification
Sending Verification Ststus
ParkIt Server
UserId:- jfbdshf56afbsdfuhsbfuy
Find Parking Area alloted for this userid
Spacecode:- A#456
counter = counter -1
decreasing counter for this spacecode

When user cancels the booking, Booking Cancellation API gets fired from the mobile application and the server gets user-id from the request and cancels the booking made for that parking lot. It also decreases the counter that signifies the total user pre-bookings.

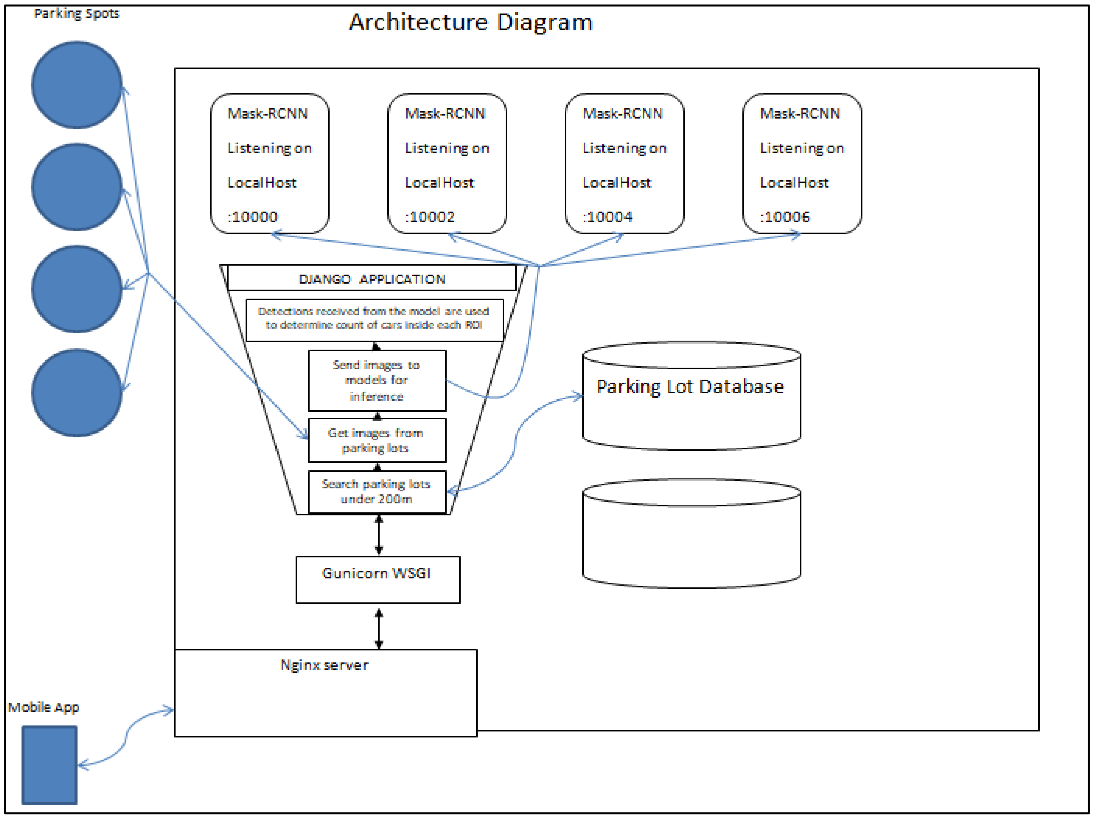

User Device
Request for Cancellation
Cancellation Status
ParkIt Server
UserId:- jfbdshf56afbsdfuhsbfuy
Find Parking Area alloted for this userid
Spacecode:- A#456
counter = counter -1
decreasing counter for this spacecode

2.4 Training the neural network

We used an implementation of Mask R-CNN on Python 3, Keras, and TensorFlow. The model generates bounding boxes and segmentation masks for each instance of an object in the image. It's based on Feature Pyramid Network (FPN) and a ResNet101 backbone.

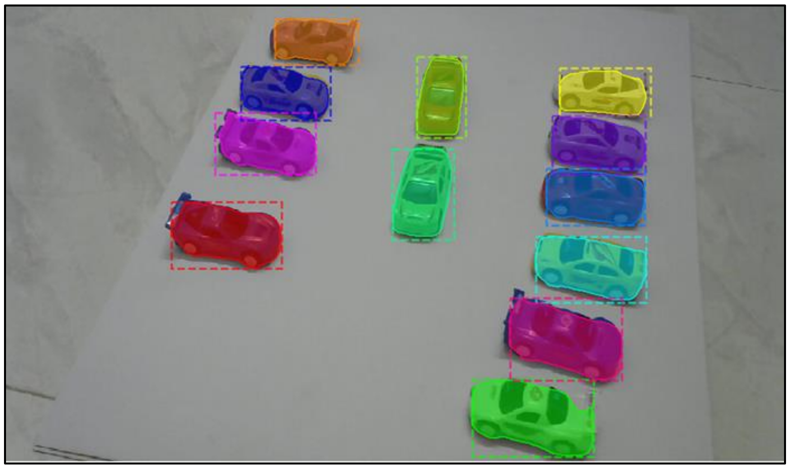
We used the PKLot dataset contains 12,417 images of parking lots and 695,899 images of parking spaces segmented from them. VGG Image Annotator was used to annotate images. Figure shows an annotated image of the PKLot dataset.

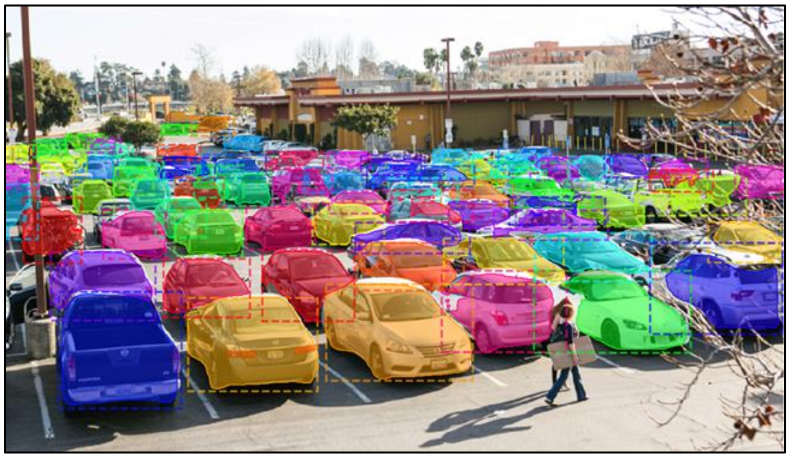
We used pre-trained weights for MS COCO and used those weights as a starting point to train our own variation of the network containing only 2 classes: car and background.


Parking Spots
Architecture Diagram
Mask-RCNN Listening on LocalHost :10000
Mask-RCNN Listening on LocalHost :10002
Mask-RCNN Listening on LocalHost :10004
Mask-RCNN Listening on LocalHost :10006
DJANGO APPLICATION
Detections received from the model are used to determine count of cars inside each ROI
Send images to models for inference
Get images from parking lots
Search parking lots under 200m
Parking Lot Database
Gunicorn WSGI
Nginx server
Mobile App

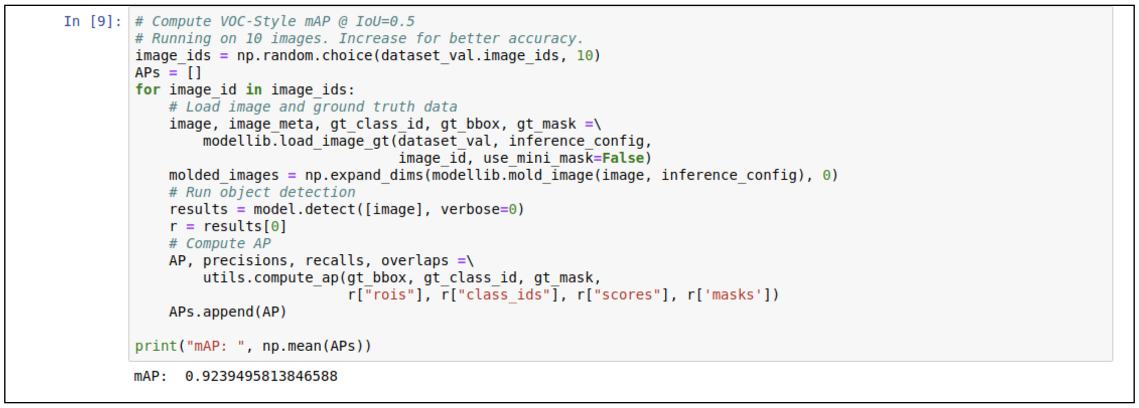
The Inference Server and the Django server communicate via Sockets. When the Inference server is started, the model gets created at multiple ports. All the threaded models are in wait state on a socket till the Django server sends the parking lot images to them. This was done so that multiple inferences can be carried out at the same time and thus multiple user requests could be handled simultaneously.

The model was trained using a GTX 1060 Ti GPU and 6 GB VRAM. **Following are the results of the network :**





Then a popular metric for measuring accuracy of object detectors like RCNN, mAP(mean Average Precision) was used to determine the accuracy of the trained model. Following is the Jupyter notebook

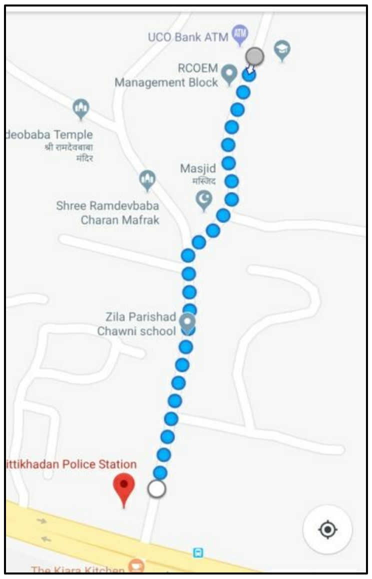


```
In [9]: # Compute VOC-Style mAP @ IoU=0.5
        # Running on 10 images. Increase for better accuracy.
        image_ids = np.random.choice(dataset_val.image_ids, 10)
        APs = []
        for image_id in image_ids:
            # Load image and ground truth data
            image, image_meta, gt_class_id, gt_bbox, gt_mask =\
                modellib.load_image_gt(dataset_val, inference_config,
                                       image_id, use_mini_mask=False)
            molded_images = np.expand_dims(modellib.mold_image(image, inference_config), 0)
            # Run object detection
            results = model.detect([image], verbose=0)
            r = results[0]
            # Compute AP
            AP, precisions, recalls, overlaps =\
                utils.compute_ap(gt_bbox, gt_class_id, gt_mask,
                                 r["rois"], r["class_ids"], r["scores"], r['masks'])
            APs.append(AP)

        print("mAP: ", np.mean(APs))

        mAP:  0.9239495813846588
```

code along with the acquired accuracy value:

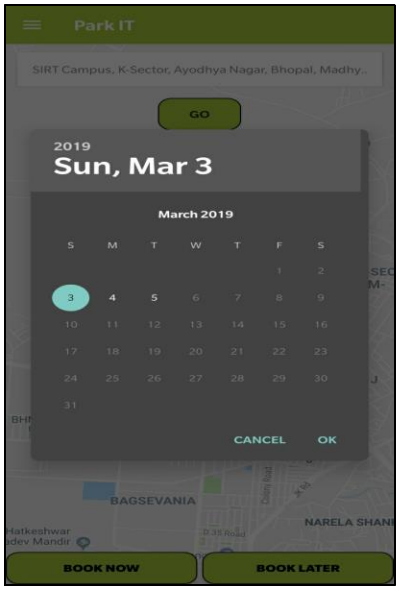
2.5 Directing the user to the parking location

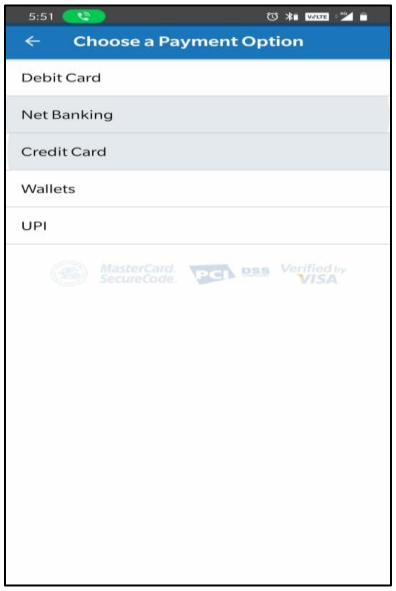
Once the query has been processed and a parking location is available, then the user receives the latitude and longitude of the parking location along with the code for verification purposes. The response comes in the form of JSON format which is parsed using the Google volley library and the useful information is stored in the database. Once the response is processed then the user will be navigated to the parking lot. This is done using the Google Direction API and the route is generated on the maps using the polylines (blue line indicating the route that the user should take in order to reach to the parking space).

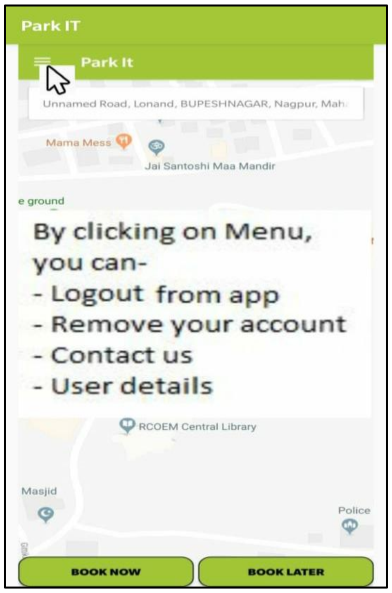

UCO Bank ATM
RCOEM Management Block
Ramdeobaba Temple
Masjid
Shree Ramdevbaba Charan Mafrak
Zila Parishad Chawni school
Gittikhadan Police Station
The Kiara Kitchen

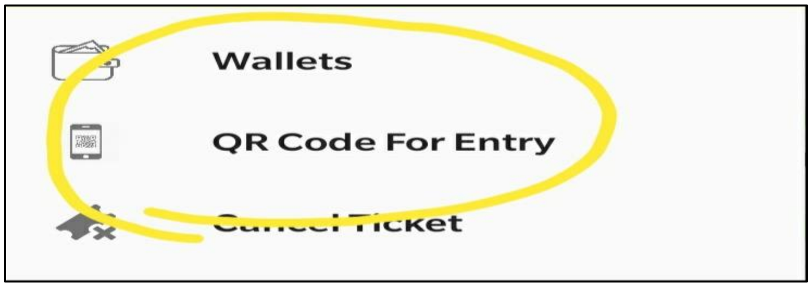
2.6 Additional Features

Additional features like Book Later, various payment options, Cancel Booking, Help Page, QR Code for entry are incorporated in the application.






By clicking on Menu, you can-
- Logout from app
- Remove your account
- Contact us
- User details



2.7 Conclusion

The proposed system reduce drive frustration and traffic by providing the nearest available parking lot. As smart parking system increases the service levels in the operation, there is a lot of scope for innovations. It helps the clients in finding out the availability of a parking spot and get the availability confirmed. It also makes the management easier at the administrator side.

Basically, smart parking system save time, money, space and help to simplify the often tedious task of parking.

2.8 My Experience

I had an amazing learning experience working on this real-world project. It was a "boys' team" of 5 and I was the only girl in it. They wanted people from all domains and I was the only one at that time in the class who had prior hands-on experience in Machine Learning because of which I was chosen to be a part of the team. I didn't see it just as an opportunity to learn new skills but also to show that I truly deserved being in the team. Before we started our project, we had to do some brainstorming with ideas to figure out the best problem statement. We had it clear in our minds that it isn't just going to be a term project to work upon but a higher goal of solving a real-life problem and creating an impact.

We had to face a lot of technical challenges. I started researching on the various approaches used for object detection. We needed a binary classifier that can label an individual space image as empty or occupied. It took me long to figure out the best approach that suits our use-case. Out of the many approaches present including SVM with Gaussian Color Model, kNN with Color Histogram, SVM with Color Histogram and Adaboost, I decided to choose CNN which was said to yield very encouraging results, with over a 99% overall accuracy and strong indications that it would generalize well to other parking lots.

I wrote the whole code for the training and testing of the Mask RCNN neural network model. After this, we wanted to scale our model so that multiple inferences could be carried out at the same time. I had to then modify the code so that it works on multiple processes.

Django was totally new for me! I learnt Django Web framework for the project in a week's time and I, along with one of my team-mates started writing the API. The Django server contains the whole logic of getting images from parking lots to sending images to Inference Server and getting the number of vacant spots back.

Another major challenge was to integrate all the separate units of code that were ready. The whole flow from the Android application to the Django Server to the Inference server and the Parking Lot local server back to the Django server had to be done flawlessly. Thorough understanding of the concepts and constant efforts of correcting the errors made us overcome this challenge too.

It was a great experience for me not only technically, but I learnt how to work in a team. Understanding each other's strengths and weaknesses, being empathetic helped me and my team a lot in distribution of tasks and responsibilities. The dynamism got us selected in the Top 1% for the Smart India Hackathon 2019 from all over India organized by MHRD's Innovation Cell, Government of India.