

CSE3999 - Technical Answers for Real World Problems (TARP)

Project Report

Parking Space Finder

By

18BCE1250

Mohit Kumar Singhal

18BCE1263

Anshu Kumar Pathak

B.Tech. Computer Science and Engineering

Submitted to

Dr. Asnath Phamila Y

School of Computer Science and Engineering



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

June 2021

DECLARATION

I hereby declare that the report titled “**College Admittance Predictor**” submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of **Dr. Asnath Phamila Y**, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

Mohit Singhal

Signature of the Candidate

MOHIT KUMAR SINGHAL

Reg. No. 18BCE1250

Anshu Pathak

Signature of the Candidate

ANSHU KUMAR PATHAK

Reg. No. 18BCE1263

CERTIFICATE

Certified that this project report entitled “**College Admittance Predictor**” is a bonafide work of **MOHIT KUMAR SINGHAL (18BCE1250)**, **ANSHU KUMAR PATHAK (18BCE1263)** and they carried out the Project work under my supervision and guidance for CSE3999 - Technical Answers for Real World Problems (TARP).

Dr. Asnath Phamila Y

SCOPE, VIT Chennai

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide **Dr. Asnath Phamila Y**, School of Computer Science and Engineering for his/her consistent encouragement and valuable guidance offered to us throughout the course of the project work.

We are extremely grateful to **Dr. R. Jagadeesh Kannan**, Dean and **Dr. S. Geetha**, Associate Dean, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for extending the facilities of the School towards our project and for their unstinting support.

We express our thanks to **Dr. Justus S**, Head of the Department, B.Tech. Computer Science and Engineering for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the courses.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Mohit Singhal

Signature of the Candidate

MOHIT KUMAR SINGHAL

Reg. No. 18BCE1250

ABSTRACT

Nowadays we have many big malls and offices in our world and for the growing number of private vehicles owned by individuals there are parking spaces provided for them to park their vehicles. In some cases, the parking areas are so big that the customers/users find it quite difficult to find a parking space. And not just that it also sometimes becomes troublesome to find their car back and reach the nearest exit when the users intend to leave that place.

We have taken upon us to solve this problem to provide the user with a route to an empty parking lot which he/she can reach in the shortest amount of time i.e. which is nearest to the entry of the parking area. The user will also be able to find the exact location to his/her vehicle and will be given the best route to reach the exit when he/she intends to leave.

We will use a map for the parking lot blueprint, and the nodes of the map will be used for finding the available space.

CONTENTS

	Declaration	2
	Certificate	3
	Acknowledgement	4
	Abstract	5
1	Introduction	7
2	Literature Survey	8
3	Requirements Specification	10
4	System Design	11
5	Implementation of System	12
6	Results & Discussion	16
7	Conclusion and Future Work	19
8	References	20
	Appendix	21

1. INTRODUCTION

1.1) PROBLEM STATEMENT

Nowadays we have big malls and offices all over the country and the same require quite a large area of parking space for the growing number of private vehicles owned by individuals so that they may have the facility to park their vehicles. In some cases, the parking areas are so big that the customers/users find it quite difficult to find an empty parking space. And not just that, it also sometimes becomes troublesome to find their car back and reach the nearest exit when the users intend to leave that place.

1.2) MOTIVATION

In the age of large constructions covering the cities, like malls, offices, residential apartments, there is a large possibility of these places becoming crowded and lose their beauty. As the places become more crowded, there is a need for some form of management to get the working back on track. We would like to solve one such problem that could regulate the parking facilities that are available in such places. There is an issue of the growing number of private vehicles owned by individuals there are parking spaces provided for them to park their vehicles. In some cases, the parking areas are so big that the customers/users find it quite difficult to find a parking space. And not just that it also some-times becomes troublesome to find their car back and reach the nearest exit when the users intend to leave that place. We have taken upon us to solve this problem to provide the user with a route to an empty parking lot which he/she can reach in the shortest amount of time i.e. which is nearest to the entry of the parking area. The user will also be able to find the exact location to his/her vehicle and will be given the best route to reach the exit when he/she in-tends to leave.

1.3) PROBLEM IDENTIFICATION

Upon having conversations with our fellow mates, families and friends, we have jot down the demographics, motives, behaviors, pain-points of the users, discovering the following problems:

1. The parking facilities in most places are too behind in using technology to ease their process and the ease and satisfaction of their customers.
2. Many universities do not have a proper management strategy to maintain their parking spaces, this causes the vehicle owners to cause traffic in roads, and we end up finding vehicles all around us.
3. In large parking spaces there is a problem to locate a proper parking space, and we end up losing a lot of time in finding a proper parking spot in the lot.
4. In large parking spaces, we also tend to forget the location of where we have parked our vehicle, and the searching becomes tedious.
5. There are just a very few cities and parking facilities that allow people to view the location of parking facilities on a map.
6. The vehicle owners often park their car in a manner such that it becomes very hard to use that road to travel by.

2. LITERATURE SURVEY

In today's world parking lots have become redundant and needs lot of manpower to handle and maintain it. These parking lots are not user friendly and do not provide data regarding availability of free spaces. Many researchers have contributed to this issue and formalized with various methods to better optimize the parking lot to serve the needs.

Some of the literature survey on the work done in the past:

2.1 INTELLIGENT CAR PARKING LOCATOR SERVICE:

It allows mobile devices (mobile phones, laptops, personal digital assistants–PDAs) to communicate to each other and to a number of servers through geographically intermittent high-speed connections. This service allows registered users to locate available parking spaces throughout the campus and reserve a space that best suits them when approaching/entering the campus area. Details of how service content is tailored to specific devices and how the duration of the user's stay affects the charging and billing for utilization of the service have been outlined. That particular device's profile will specify its capability to only handle text information during its communications with the Info Stations. As such the Info Station will provide only the requisite text information.

2.2 AN ANDROID/IOS APPLICATION FOR CAR PARKING SYSTEM

The system gives a visual display to the user regarding the current parking scenario. The system reduces work of manual parking process by converting the entire parking process to automation. The system makes it easy for the user to book or reserve a space on the smartphone. Thus smartphone acts as a park finder. This ultimately reduces the time that every driver spends for searching a parking space which then reduces the fuel consumption, traffic volume, and environmental pollution by increasing the efficiency of transportation.

2.3 ANDROID APPLICATION FOR VEHICLE PARKING SYSTEM:

Its application is based on the client server architecture. The client is provided with an interactive Android based user interface for the process of pre-booking of parking slot.

- a) The user needs to install the application on his Android based device.
- b) Initially, the user has to register his details with the application for the first time.
- c) Once the user registers, he can use his email id and phone number to login in future.
- d) The client is provided with multiple parking locations. Client has to select one of the locations provided where he desires to park the vehicle.
- e) After selecting the location, options for the vehicle type is provided i.e. 2-wheeler or 4-wheeler alongside the rate chart for parking charges is prompted.
- f) Based on the type of vehicle selected availability of the empty slots will be displayed along with the total slots reserved for that vehicle type.
- g) In case the slot is available, the client can proceed further with the reservation process or else he can go back to change the location/vehicle type or else can terminate the entire process.
- h) On successful reservation, a confirmation page with user details is shown which is editable.

2.4 HARDWARE APPLICATION

It is an application that is consisted of an ultrasonic ranger and a GPS receiver to determine the distance from the vehicle to roadside. It is proposed that the units could be placed on vehicles such as buses, taxis or private cars to continually gather data as they travel along their routes. It leverages the users' heuristic searching and converts users to stakeholders. On the one hand, it requires much lower number of sensor units compared with the fixed sensing solutions thanks to users' participation; on the other hand, it cuts down the time and fuel consumed in searching thanks to the dissemination of parking availability information. As soon as the initial occupancy map is built up in the crowd-sourcing manner, it can substantially bring down the searching effort. As for next steps the algorithms for on-street parking need to be considered.

The solution could be employing dual ranger sensors positioned at each end of the car to measure the differentials. For example, if the vehicle moves away from one lane to another, then the difference of the two sensors' reading remains constant if we adjust the lateral time difference.

2.5 RESEARCH GAP

Time and Cost are the two most important factors in human life. More and more difficulties are increasing and affecting all the human life physically as well as mentally. One of which is car parking problem rapidly arising. Nowadays parking problem is faced due to parking space falling short of the current requirements in the country as the total number of motor vehicles exceeds the total number of heads per family. In Indian cities, the parked cars claim a lot of space which leads to congestion and traffic problems. Thus fundamentally parking is a problem of space.

With the growing culture of automobile dependency in INDIAN cities, the demand for parking spaces is also increased. This is especially because the infrastructural growth of our cities is unable to keep up with the growing demand for spaces to park. The other aspects of urban life have begun to spill over in form of congestion, fuel loss, dispersed land use and low air quality due to the scarcity in parking spaces. It is, therefore, strongly desired to provide an effective strategy to address these concerns.

Although all the works mentioned above, has tried to solve the issue of parking space, there are many gaps which our project will try to fulfil.

We have taken upon us to solve this problem to provide the user with a route to an empty parking lot which he/she can reach in the shortest amount of time i.e. which is nearest to the entry of the parking area. The user will also be able to find the exact location to

his/her vehicle and will be given the best route to reach the exit when he/she intends to leave.

If two cars arrive at the same time, then the car with lexicographically smaller car number will be given priority for parking the car.

The Car user can also see the parking lot where the car is parked.

Our Project also uses the best efficient algorithm using heap data structure which gives us result in least amount of time.

3. REQUIREMENTS SPECIFICATION

3.1 Hardware Requirements

None

3.2 Software Requirements

System OS: Windows/Linux/Android/MacOS

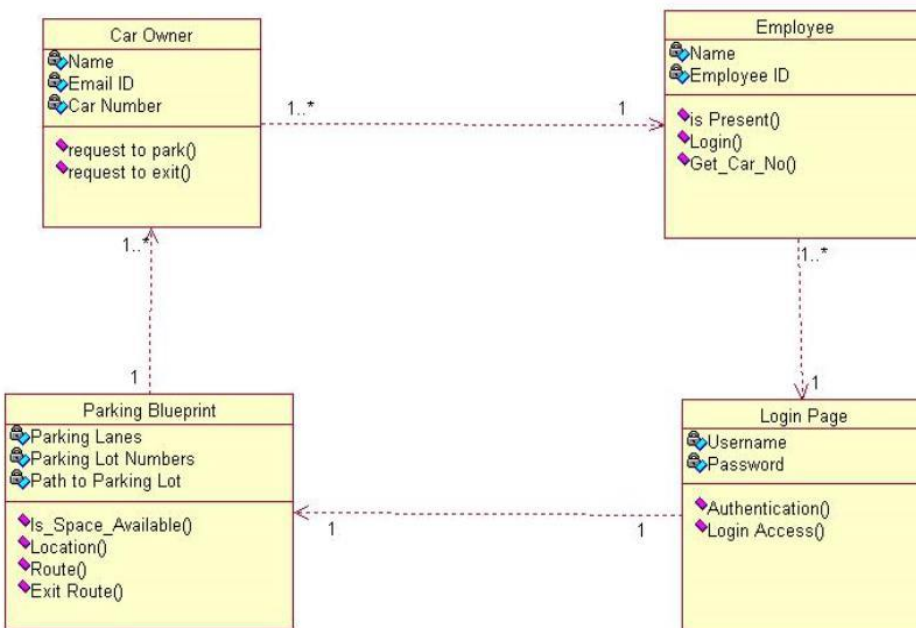
Browser: Chrome/ Safari/ Edge

Flask

Python 3.8

4. SYSTEM DESIGN

- A computer/smartphone to display the location and the route.
- A login portal to verify the authentication of the employee using the system.
- A software that first tells the customer whether there is any empty parking space or not in the parking area.
- If there is a parking space available, the software should be able to tell the location of the nearest empty parking space with the route to it.
- A web link of this location and the route should be provided to the customers so that they can see it in their own smart devices.
- The software should be able to provide the parking location of the parked car again whenever the user wishes to leave.
- The route to the closest exit should finally be also provided.

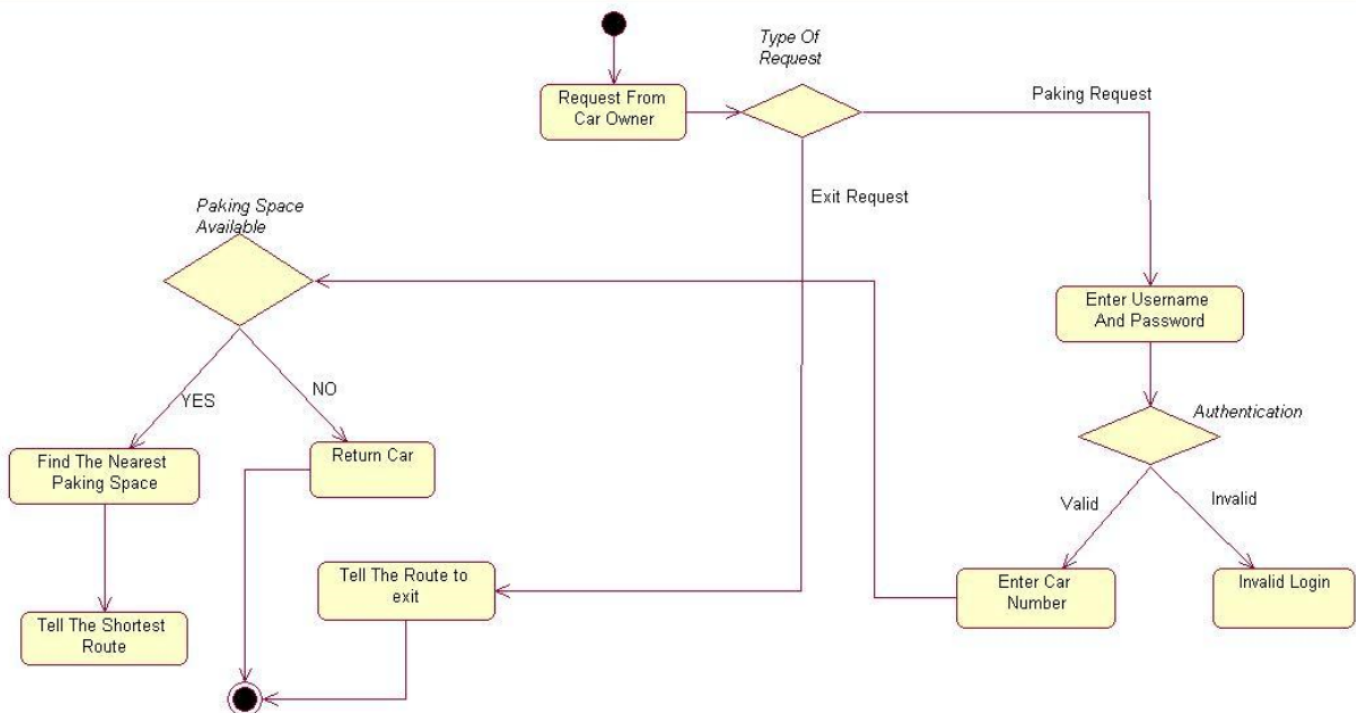


5. IMPLEMENTATION OF SYSTEM

5.1 Algorithm Used:

- We will be using a parking lot blueprint which is kind a like the figure shown above. The lanes will be numbered as A, B, C... and the parking lots in each lane will be represented in the following fashion A1,A2,A3,A4..... B1,B2,B3,B4..... C1,C2,C3,C4..... and so on.
- The parking lot numbers will be in sorted order. We will use a min heap for storing parking lots as Nodes. The parking lot Number will act as the data for each node in the mean heap. When a Car arrives in entry gate, we will search for the node in mean heap which is nearest to the entry gate.
- We will also maintain a dictionary in which key will act as car number and value as parking lot number.
- When Car arrives, we will get node from heap, delete it, apply hipification and append key to the dictionary. The map will show the path to driver to park the car. While exiting, we will get the parking lot number from the dictionary .
- We will insert into the heap and the repeat the process again. We would be implementing the algorithm in a website which is de-signed using HTML, CSS, JavaScript, and the actual work is done in Python which has been stitched together using some Flask libraries.

5.2 ACTIVITY DIAGRAM



5.3 MODULES INVOLVED:

5.3.1)Login Page: Login Page consisting of username and password



Parking Space Finder

The login page has a dark gray background. At the top, the word "Welcome" is written in white. Below it, a message states: "This is a website to help people find their way to an empty lot in your parking area. Please enter your details to login". There are two input fields: "Username" and "Password". Below these fields is a light gray button labeled "LOGIN". At the bottom, there are three social media icons: Facebook, Twitter, and Email.

5.3.2)Choice for Entry and Exit: A web Page to give user choice for entry and exit for the car.



Parking Space Finder

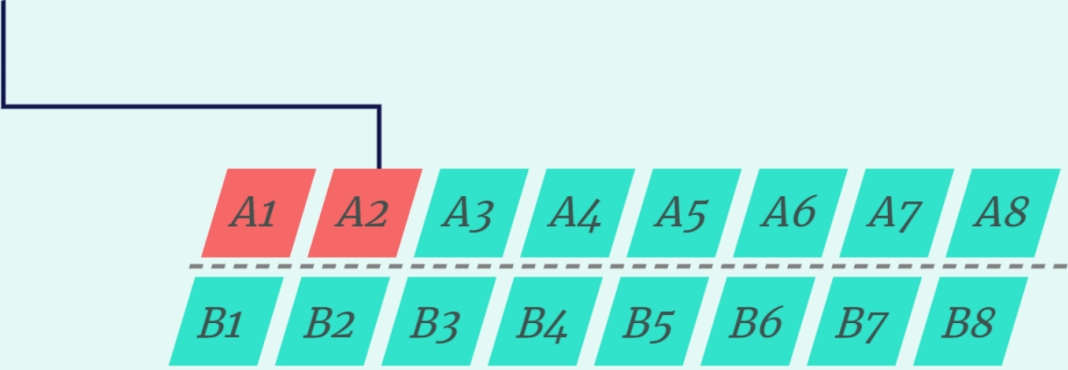
The post-login page has a dark gray background. At the top, the word "Welcome" is written in white. Below it, a message states: "You have successfully logged in. Choose one of the below options for further details". There are three light gray buttons stacked vertically: "Find a Parking Space", "Find Your Car", and "Find the Exit". At the bottom, there are three social media icons: Facebook, Twitter, and Email.

5.3.3) Designing the Parking Blueprint: The Blueprint consisting of lanes. Implemented by HTML, CSS, JavaScript.

Your Car number is: RJ25CAR4

Go to options Page

Your Parking location is: A2

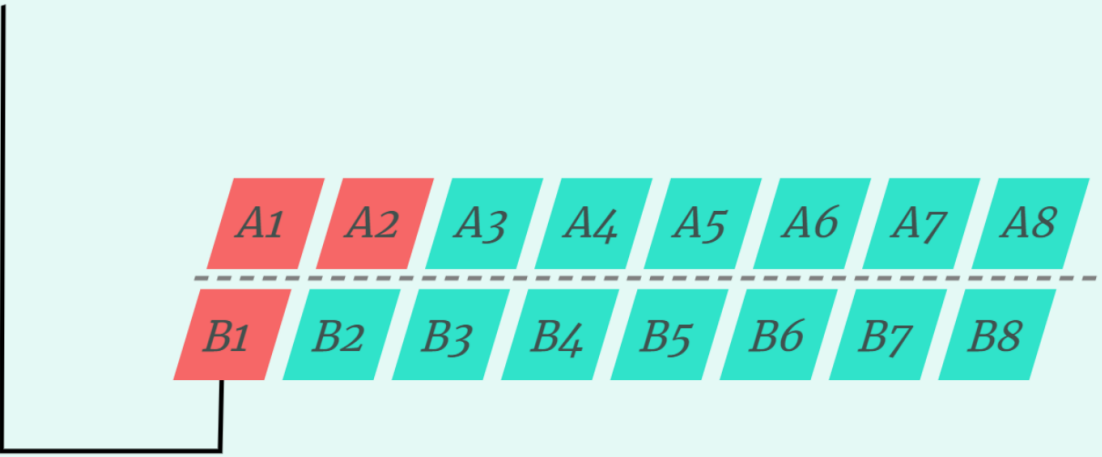


5.4.4) Implementation of Algorithm: Implement the algorithm for the shortest entry and exit of the car.

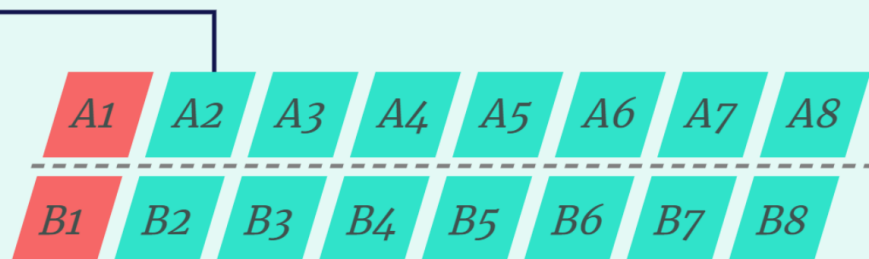
Your Car number is: RJ25CAR5

Go to options Page

Your Parking location is: B1



Follow the below path from A2 to exit



6. RESULTS AND DISCUSSION

6.1 DISCUSSIONS:

- To avoid the congestion of public transport and to gain comfort, we would like to propose our project to manage parking lots, make them more coordinated throughout the world, and to ease the experience of the users who use our parking spaces.
- This technology is cost efficient and more importantly, can be applied to almost any parking space the world, to transform it from a traditional parking space to one with technical features that could ease the user experience.
- We could also slowly expand the horizons of this project once we get settled with our current plan and the revenues start coming in. Though there are more and more companies emerging out to make new parking spaces with advanced technologies, we would instead like to upgrade an existing parking space with as much technology as it is required.
- We consider the experience of the customer to be of our highest priority and would strive to improve ourselves, our goals and even the plans to achieve them throughout the life time of our project.
- We have built a parking service that cares about the user, providing you more than just good service.

6.2 Results

The Following objectives were completed in our project.

For Finding a Parking Space: To do this we are going to represent the above parking lot in the form of a heap, a min heap to be precise. This data structure enables us to extract and delete efficiently. The data stored in the nodes of the heap would be the parking lot numbers. Each time we want to find a parking space, the root of the heap would be extracted.

For Finding the Location of any Vehicle: To achieve this task in constant time. We are storing the vehicles entering the parking lot in a dictionary with key acting as the vehicle number and value as the parking lot number allotted to it.

For Finding the route to exit: Now when a vehicle wants to exit the parking area, it's parking lot would be extracted from the dictionary and that key-value pair is deleted. Then this parking lot number is inserted back into the min-heap so that it could be used again.

6.3 FEASIBILITY OF PROJECT WORK:

Technical feasibility

The application has been entirely made using python language which efficiently works and is easily scalable in terms of capacity of users. The user of the system gets a well-designed interface with all necessary functionalities and easy to use application. The concept and the data structure used is the most efficient and feasible one of all possible solutions. The heap data structure performs insertion operation in $O(\log n)$ time which is very efficient in case of larger values of n as well. So, the application is well-to-do and technically efficient and feasible.

Economic feasibility

When the application will be deployed at any parking area, it will be a free of cost application for the users of the system. So, it is economically feasible.

Social feasibility

The application doesn't harm the society in any way rather gives a wonderful solution to day-to-day life problem of everyone. So, it is socially feasible.

Environmental feasibility

The application doesn't use any hardware or environment harming component rather it is a completely software based complete solution to the problem and can be accessed by any user on the click of his phone. So, it is environment friendly.

Political feasibility

The application has nothing related to politics rather it can be utilized at political gatherings where parking of cars of vvip's is a big issue.

Demographic feasibility

It can be used anywhere at zero cost and easily accessible as now, internet is everywhere. So, there is no demographic boundary to the application.

7. CONCLUSION AND FUTURE WORK

7.1 CONCLUSION:

To avoid the congestion of public transport and to gain comfort, we would like to propose our project to manage parking lots, make them more coordinated throughout the world, and to ease the experience of the users who use our parking spaces. This technology is cost efficient and more importantly, can be applied to almost any parking space the world, to transform it from a traditional parking space to one with technical features that could ease the user experience. We could also slowly expand the horizons of this project once we get settled with our current plan and the revenues start coming in. Though there are more and more companies emerging out to make new parking spaces with advanced technologies, we would instead like to upgrade an existing parking space with as much technology as it is required. We consider the experience of the customer to be of our highest priority and would strive to improve ourselves, our goals and even the plans to achieve them throughout the life time of our project. We have build a parking service that cares about the user, providing you more than just good service.

7.2 FUTURE WORK:

Our project as of now is just a basic prototype of the problem at hand. To implement it fully in the real world and in country such as ours is a bit challenging. But we can go towards the next level which might be to deploy an app which through which the user would even be able to reserve a parking lot. We can also use IOT to scan the number plates of the vehicles to verify whether they have parked at the assigned location or not and so on.

8. REFERENCES

- Prototype of Parking Finder Application for Intelligent Parking System Lana Benny , Prashant Kumar Soori, Department of Electrical and Electronics Engineering, Heriot-Watt University Dubai Campus, Dubai, U.A.E
- SMART PARKING: <https://www.happiestminds.com/whitepapers/smart-parking.pdf> Happiest Minds Technologies Private Ltd.

Appendix(sample code)

```
from flask import Flask, render_template, url_for, request, redirect

app = Flask(__name__)

login_flag = 0
option_flag_1 = 0
option_flag_2 = 0

@app.route('/')
def welcome():
    global login_flag
    login_flag = 1
    return render_template('loginpage.html')

@app.route('/loginpage', methods=['POST', 'GET'])
def loginpage():
    global login_flag
    if login_flag and request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if username == "" and password == "":
            return render_template('options.html')
        else:
            return render_template('loginpage.html')
    else:
        return render_template('loginpage.html')

@app.route('/options', methods=['POST', 'GET'])
def options():
    global heap_size
    if login_flag:
        return render_template('options.html', name=heap_size)
    else:
        return render_template('loginpage.html')

@app.route('/choice1', methods=['POST', 'GET'])
def choice1():
    global option_flag_1
    option_flag_1 = 1
    return render_template('choice1.html')
```

```

@app.route('/givloc', methods=['POST', 'GET'])
def givloc():
    global arr
    global heap_size
    global option_flag_1
    global diction
    global curr_used
    #print(2)
    if option_flag_1 and request.method == 'POST':
        carnum = request.form['carnum']

        try:
            diction[carnum]
        except:
            minimum = extract_min()

            diction[carnum] = minimum
            curr_used[minimum[0]] = 1
            option_flag_1 = 0
            return render_template('givloc.html', name=[minimum[0], carnum, curr_used])
        else:
            return "<h1>Your Car is Already there</h1>"
    else:
        return render_template('options.html')

@app.route('/choice2', methods=['POST', 'GET'])
def choice2():
    return render_template('choice2.html')

@app.route('/givcar', methods=['POST', 'GET'])
def givcar():
    carnum = request.form['carnum']
    try:
        diction[carnum]
    except:
        return "<h1>Car not there in Parking Area</h1>"
    else:
        plotnum = diction[carnum][0]
        return render_template('givcar.html', name=plotnum)

@app.route('/choice3', methods=['POST', 'GET'])
def choice3():
    global option_flag_2
    option_flag_2 = 1
    return render_template('choice3.html')

```

```

@app.route('/givexit', methods=['POST', 'GET'])
def givexit():
    global arr
    global heap_size
    global option_flag_2
    global diction
    global curr_used
    if option_flag_2 and request.method == 'POST':
        carnum = request.form['carnum']
        try:
            diction[carnum]
        except:
            return "<h1>Car not there in Parking Area</h1>"
        else:
            plotnum = diction[carnum]
            curr_used[plotnum[0]] = 0
            del diction[carnum]
            insert(plotnum)
            option_flag_2 = 0
            return render_template('givexit.html', name=[plotnum[0], curr_used])
    else:
        return render_template('options.html')

def heapify(i):
    global heap_size
    global arr
    smallest = i
    l = 2 * i + 1
    r = 2 * i + 2
    if (l < heap_size and arr[l][1] < arr[smallest][1]):
        smallest = l
    if (r < heap_size and arr[r][1] < arr[smallest][1]):
        smallest = r
    if (smallest != i):
        swap = arr[i]
        arr[i] = arr[smallest]
        arr[smallest] = swap
        heapify(smallest)

def insert(key):
    global arr
    global heap_size
    heap_size += 1
    arr[heap_size - 1] = key
    heapify_bottom(heap_size - 1)

```

```

def heapify_bottom(i):
    global arr
    global heap_size
    parent = (i - 1) // 2
    if (parent >= 0):
        if (arr[parent][1] > arr[i][1]):
            swap = arr[parent]
            arr[parent] = arr[i]
            arr[i] = swap
            heapify_bottom(parent)

def extract_min():
    global arr
    global heap_size
    if heap_size==0:
        return(-1)
    minimum = arr[0]
    arr[0]= arr[heap_size-1]
    heap_size = heap_size-1
    heapify(0)
    return(minimum)

def buildHeap():
    global heap_size
    global arr
    startIdx = int((heap_size / 2)) - 1
    for i in range(startIdx, -1, -1):
        heapify(i)

if __name__ == "__main__":
    global heap_size
    global arr
    global diction
    global curr_used
    curr_used = {}
    diction = {}
    arr = []
    c = chr(65)
    for i in range(1, 7):
        for j in range(1, 9):
            s = c + str(j)
            if(i<=2):
                dist=i+j-1
            elif(i<=4):
                dist=i+j+1
            elif(i<=6):

```

```
        dist=i+j+5
    elif(i<=8):
        dist=i+j+7

    arr.append((s,dist))
    curr_used[s] = 0
    c = chr(65 + i)
    heap_size = len(arr)
    buildHeap()
    #print(arr)
    app.run(debug=True)
```