Python Project

Parking Management System

Under the Guidance of

Dr Dhanpratap Singh

Done by

Name	Roll No	Reg.No
1.Abhishek kumar	45	11916395
2.Ritesh Chaubey	24	12001720

Section : K19QK

Submitted to:

Department Of Computer Science Engineering. SUB:INT213.

Dr Dhanpratap Singh. YEAR:2020

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to **Lovely Professional University** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my parents & member of **Dr Dhanpratap Singh** for their kind co-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to industry persons for giving me such attention and time.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

Date:31/10/2020

Abhishek kumar

Ritesh Chaubey

Content Table

LOVELY PROFESSIONAL UNIVERSITY

Introduction:

Parking management system for managing the records of the incoming and outgoing vehicles in an parking house

It's an easy for Admin to retrieve the data if the vehicle has been visited through number he can get that data .

Now days in many public places such as malls, multiplex system, hospitals, offices, market areas there is a crucial problem of vehicle parking. The vehicle parking area has many lanes/slots for car parking. So to park a vehicle one has to look for all the lanes. Moreover this involves a lot of manual labour and investment. Instead of vehicle caught in towing the vehicle can park on safe and security with low cost.

Parking control system has been generated in such a way that it is filled with many secure devices such as, parking control gates, toll gates, time and attendance machine, car counting system etc. These features are hereby very necessary nowadays to secure your car and also to evaluate the fee structure for every vehicles entry and exit

The objective of this project is to build a Vehicle Parking management system that enables the time management and control of vehicles using number plate recognition. The system that will track the entry and exit of cars, maintain a listing of cars within the parking lot, and determine if the parking lot is full or not. It will determine the cost of per vehicle according to their time consumption.

Objectives:

We can park our vehicle in our own slot by paying.

- Because of that there is no towing problems.
- And our vehicle has been parked as a secure condition.
- There is no risk for vehicle owner for parking the car.
- In case of any damages and problem of vehicle that will claim by parking management.
- As the world is facing many threads daily, robberies are done easily with no track to trace, bomb blasts occur with the use of vehicle, so if a proper system is adopted each and every record can be saved and anyone can be track easily therefore mainly is to make a better and fast software, most important user-friendly
- Maintain records in short time of period.
- Determines the parking area is full or not.
- Enhances the visitor's experience.

Scopes:

In the modern age. Many people have vehicles. Vehicle is now a basic need. Every place is under the process of urbanization. There are many corporate offices and shopping centers etc. There are many recreational places where people used to go for refreshment. So, all these places need a parking space where people can park their vehicles safely and easily. Every parking area needs a system that records the detail of vehicles to give the facility. These systems might be computerized or non-computerized. With the help of computerized system we can deliver a good service to customer who wants to park their vehicle into the any organization's premises.

Vehicle parking management system is an automatic system which delivers data processing in very high speed in systematic manner. Parking is a growing need of the time. Development of this system is very useful in this area of field. We can sell this system to any organization. By using our system they can maintain records very easily. Our system covers the every area of parking management. In coming future there will be excessive need of Vehicle parking management system.

Definition Of Problem:

- Now a days in parking like valet parking they maintain just with the tokens and they have records the vehicle details in books so that during some critical situations like police enquiry of terrorist car or vehicle roberrer that case it is difficult to find the details of particular vehicle but in this case is easy to find in 1 to 2 seconds.

- By parking the vehicle in public place the vehicle can be claimed by towing person but in this case there is no towing problems and no need to give fine for anything we can park our vehicle with securely.

Design the Parking Management System Using Python:

Features:

- Login for the users which are basically admin.
- They can add the vehicle details in the system.
- Admin can also add search vehicle details by the parking number in the system.
- Besides, the admin can also view and update all the vehicle details.

Software Requirements:

- Python IDLE(3.8.7)
- Mozilla Firefox & chrome browser
- Web Server.

Design Approach:

The project should have the following features for its successful functioning.

1. Technology.
2. Feasibility.
3. Efficiency.
4. Optimum Cost

The project will deliver a cost efficient system which would be feasible and at the same time will not comprise with the technology as far as possible.

How is it implemented and works?:

MODULE DESCRIPTION

The project will have the following modules:

1. User profile module:- This will have information of all users using the parking lot.

2. Device Management Module:-This will take care of the various technological devices like web camera, electronic gates, cctv cameras wireless devices and so forth.

3. Time management module:-This module will keep a track of time for which cars are parked in the three tiers of the parking area.

4. Geographical module:-This will be used to find area in the city which is best suited to build a parking lot as large as the one proposed.

5. Security Module:- This will ensure that unauthorised vehicles are not parked. The system will inform drivers if they are available at the parking spot and prevent unauthorised vehicles from entering the parking lot. The unauthorised vehicles will be tracked by a camera which will record the corresponding vehicles license plates. The images will be transmitted to a recording device.

6. Design Module:-Modelling and design.

7. Cost Management Module:-This module will aim to prepare a cost efficient system and at sane time will not compromise

with the use of technology. It will have the financial budget of the project.

Design Requirements

Basic input and output:-

- **Number of vehicles entering/exiting the parking lot**
- **Number of violators**.

Scenarios:-

1. **Entrance with a valid Id number**. The number will be provided to the user who visits for the first time by the machine .The number would have to be provided at the entrance for entry. If the user gives a wrong id num he will not allowed park the vehicle in the parking area. In case the user forgets his id no he has provide the wrong id to the machine at the gate five times and his id will be changed. Failing to remember the id for the second time will lead to prohibition of the user from entering the area in future.
2. **Entrance speed regulated by speed bumps**. This is done to ensure single vehicle entry.
3. **Entry/exit**. On entry and exit of a vehicle the automobile counter will be triggered which will link to the comp software to keep the system updated with no of parking spots available.

Security arrangements and rule violation. If two vehicles enter at a time with a speed of more than 10 km/hr a snapshot of the violators will be taken at the main gate.

4. **Separate gates for entry and exit.** If a person tries to leave from the entry gate his snapshot will also be taken through a cctv camera at the main gate.

Main Programme Of This Project:

:Parking Management System:

Description:

This repository gives an overview of how we can design a basic parking lot in Python. It creates parking lot with given number of slots. The cars follow Greedy approach while being parked in the slots.

ParkingLot.py script defines the following functions -

create_parking_lot n - Given n number of slots, create a parking lot

park car_regno car_color - Parks a vehicle with given registration number and color in the nearest empty slot possible. If there are no more empty slots available, it shows a message "Sorry, parking lot is full".

status - Prints the slot number, registration number and **color** of the parked vehicles.

leave x - Removes vehicle from slot number x

There are few query functions to retrieve slot number from registration number of car, get registration numbers of cars with particular **color** etc.

ParkingLot.py can be run through shell or through file containing test cases. An example file run_test_case.txt has been provided in repo.

I have followed this approach while designing this. **test_parking_lot.py** uses **unittest** module of python. Here 6 test cases are written in order to test **each functionality** mentioned in ParkingLot.py

Vehicle.py is a separate class where we can define the type of vehicles that can be parked. As of now, it only contains class Car.

Main python coding of parking management:

```
import
Vehicl
e
        import argparse
        import sys

        if sys.version_info[0] == 2:
              input = raw_input

        class ParkingLot:
              def __init__(self):
                    self.capacity = 0
                    self.slotid = 0
                    self.numOfOccupiedSlots = 0

              def createParkingLot(self,capacity):
                    self.slots = [-1] * capacity
                    self.capacity = capacity
                    return self.capacity

              def getEmptySlot(self):
                    for i in range(len(self.slots)):
                          if self.slots[i] == -1:
                                return i

              def park(self,regno,color):
                    if self.numOfOccupiedSlots < self.capacity:
                          slotid = self.getEmptySlot()
```

```
                self.slots[slotid] =
Vehicle.Car(regno,color)
                self.slotid = self.slotid+1
                self.numOfOccupiedSlots =
self.numOfOccupiedSlots + 1
                return slotid+1
          else:
                return -1

     def leave(self,slotid):

          if self.numOfOccupiedSlots > 0 and
self.slots[slotid-1] != -1:
                self.slots[slotid-1] = -1
                self.numOfOccupiedSlots =
self.numOfOccupiedSlots - 1
                return True
          else:
                return False

     def status(self):
          print("Slot No.\tRegistration No.\tColour")
          for i in range(len(self.slots)):
                if self.slots[i] != -1:
                     print(str(i+1) + "\t\t"
+str(self.slots[i].regno) + "\t\t" +
str(self.slots[i].color))
                else:
                     continue

     def getRegNoFromColor(self,color):

          regnos = []
          for i in self.slots:

                if i == -1:
                     continue
                if i.color == color:
                     regnos.append(i.regno)
          return regnos

     def getSlotNoFromRegNo(self,regno):

          for i in range(len(self.slots)):
                if self.slots[i].regno == regno:
                     return i+1
```

```
                    else:
                        continue
            return -1


    def getSlotNoFromColor(self,color):

        slotnos = []

        for i in range(len(self.slots)):

            if self.slots[i] == -1:
                continue
            if self.slots[i].color == color:
                slotnos.append(str(i+1))
        return slotnos

    def show(self,line):
        if line.startswith('create_parking_lot'):
            n = int(line.split(' ')[1])
            res = self.createParkingLot(n)
            print('Created a parking lot with
'+str(res)+' slots')

        elif line.startswith('park'):
            regno = line.split(' ')[1]
            color = line.split(' ')[2]
            res = self.park(regno,color)
            if res == -1:
                print("Sorry, parking lot is full")
            else:
                print('Allocated slot number:
'+str(res))

        elif line.startswith('leave'):
            leave_slotid = int(line.split(' ')[1])
            status = self.leave(leave_slotid)
            if status:
                print('Slot number
'+str(leave_slotid)+' is free')

        elif line.startswith('status'):
            self.status()
```

```
            elif
line.startswith('registration_numbers_for_cars_with_colour')
:
                  color = line.split(' ')[1]
                  regnos = self.getRegNoFromColor(color)
                  print(', '.join(regnos))

            elif
line.startswith('slot_numbers_for_cars_with_colour'):
                  color = line.split(' ')[1]
                  slotnos = self.getSlotNoFromColor(color)
                  print(', '.join(slotnos))

            elif
line.startswith('slot_number_for_registration_number'):
                  regno = line.split(' ')[1]
                  slotno = self.getSlotNoFromRegNo(regno)
                  if slotno == -1:
                        print("Not found")
                  else:
                        print(slotno)
            elif line.startswith('exit'):
                  exit(0)

def main():

      parkinglot = ParkingLot()
      parser = argparse.ArgumentParser()
      parser.add_argument('-f', action="store",
required=False, dest='src_file', help="Input File")
      args = parser.parse_args()

      if args.src_file:
            with open(args.src_file) as f:
                  for line in f:
                        line = line.rstrip('\n')
                        parkinglot.show(line)
      else:
                  while True:
                        line = input("$ ")
                        parkinglot.show(line)

if __name__ == '__main__':
      main()
```

Output:

LOVELY PROFESSIONAL UNIVERSITY

Car coding:

```
class Vehicle:

        def __init__(self,regno,color):

                self.color =  color

                self.regno = regno
```

```
class Car(Vehicle):


        def __init__(self,regno,color):

                Vehicle.__init__(self,regno,color)


        def getType(self):

                return "Car"
```

Test Parking Lot Coding:

```
import
unittest
          from ParkingLot import ParkingLot
          class TestParkingLot(unittest.TestCase):

                def test_create_parking_lot(self):
                      parkingLot = ParkingLot()
                      res = parkingLot.createParkingLot(6)
                      self.assertEqual(6,res)

                def test_park(self):
                      parkingLot = ParkingLot()
                      res = parkingLot.createParkingLot(6)
                      res = parkingLot.park("KA-01-HH-1234","White")
                      self.assertNotEqual(-1,res)

                def test_leave(self):
                      parkingLot = ParkingLot()
                      res = parkingLot.createParkingLot(6)
                      res = parkingLot.park("KA-01-HH-1234","White")
                      res = parkingLot.leave(1)
                      self.assertEqual(True,res)

                def test_getRegNoFromColor(self):
                      parkingLot = ParkingLot()
                      res = parkingLot.createParkingLot(6)
                      res = parkingLot.park("KA-01-HH-1234","White")
                      res = parkingLot.park("KA-01-HH-9999","White")
                      regnos = parkingLot.getRegNoFromColor("White")
                      self.assertIn("KA-01-HH-1234",regnos)
                      self.assertIn("KA-01-HH-9999",regnos)
```

```
        def test_getSlotNoFromRegNo(self):
                parkingLot = ParkingLot()
                res = parkingLot.createParkingLot(6)
                res = parkingLot.park("KA-01-HH-1234","White")
                res = parkingLot.park("KA-01-HH-9999","White")
                slotno = parkingLot.getSlotNoFromRegNo("KA-01-HH-9999")
                self.assertEqual(2,slotno)

        def test_getSlotNoFromColor(self):
                parkingLot = ParkingLot()
                res = parkingLot.createParkingLot(6)
                res = parkingLot.park("KA-01-HH-1234","White")
                res = parkingLot.park("KA-01-HH-9999","White")
                slotnos = parkingLot.getSlotNoFromColor("White")
                self.assertIn("1",slotnos)
                self.assertIn("2",slotnos)

if __name__ == '__main__':
        unittest.main()
```

Set Up:

- **To create my own Parking Lot –**

1. Clone the repository

2. Run python ParkingLot.py to run without input test case file. This opens a shell where you can write your commands like -

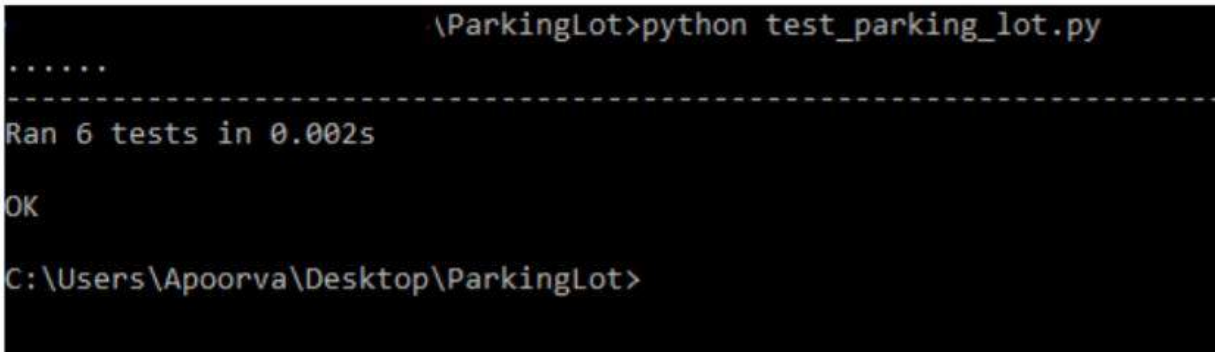
```
                        \ParkingLot>python ParkingLot.py
$ create_parking_lot 6
Created a parking lot with 6 slots
$ park KA-01-HH-1234 White
Allocated slot number: 1
$ park KA-01-HH-9999 White
Allocated slot number: 2
$ park KA-01-BB-0001 Black
Allocated slot number: 3
$ status
Slot No.        Registration No.        Colour
1               KA-01-HH-1234           White
2               KA-01-HH-9999           White
3               KA-01-BB-0001           Black
$ leave 2
Slot number 2 is free
$ status
Slot No.        Registration No.        Colour
1               KA-01-HH-1234           White
3               KA-01-BB-0001           Black
$ registration_numbers_for_cars_with_colour White
KA-01-HH-1234
$ slot_numbers_for_cars_with_colour White
1
$ slot_number_for_registration_number KA-01-HH-1234
1
```

- **To run with a file execute `python ParkingLot.py -f run_test_case.txt. You can modify the test cases**

```
                    ParkingLot>python ParkingLot.py -f run_test_case.txt
reated a parking lot with 6 slots
llocated slot number: 1
llocated slot number: 2
llocated slot number: 3
llocated slot number: 4
llocated slot number: 5
llocated slot number: 6
lot number 4 is free
lot No.        Registration No.        Colour
               KA-01-HH-1234           White
               KA-01-HH-9999           White
               KA-01-BB-0001           Black
               KA-01-HH-2701           Blue
               KA-01-HH-3141           Black
llocated slot number: 4
orry, parking lot is full
A-01-HH-1234, KA-01-HH-9999, KA-01-P-333
, 2, 4

ot found
```

- You can also run the test cases separately as python **test_parking_lot.py.** This runs the 6 test cases written in file. This is very useful when you want to create your own function and test it simultaneously.



Advantages of Parking Management System:

- The main important benefit of parking management system is its advanced technology. It follows latest technologies and concepts to assure profitable outcome.
- For parking authorities and vehicle owner, parking management system resources are flexible enough to operate and manage.
- The design and implementation of parking management system is very easy to supervise and manage. This system can be easily handled by the staff members because of its well organized structure.
-

Disadvantages of Parking Management System:

There is a greater construction cost per space (but this may be offset by the chance for lesser land costs per space and the system manufacturers say that the operation and maintenance cost will be lower as compared to a conventional ramped parking

* Use of redundant systems will result in a greater cost.

* It may be a bit confusing for unfamiliar users.
* It is not recommended for high peak hour volume facilities.
* There may be a fear of breakdown (How do I get my car out?).
* There is an uncertain building department review and approval process.
* It requires a maintenance contract with the
supplier.

Application:

- **Overhead radars/lidars.**
- **Ground sensors.**
- **Naval defense systems and aviation.**
- **Quality fitness trackers.**
- **vehicle detection accuracy.**

Conclusion:

It was a wonderful learning experience for me while working on this project. This project took me through the various phases of project development and gave me real insight into the world of software engineering. The joy of working and the thrill involved while tackling the various problems and challenges gave me a feel of the developers' industry.

It was due to this project I came to know how professional software is designed.

THANK YOU

Python Project

Parking Management System

Under the Guidance of

Dr Dhanpratap Singh

Done by

Name	Roll No	Reg.No
1.Abhishek kumar	45	11916395
2.Ritesh Chaubey	24	12001720

Section : K19QK

Submitted to:

Department Of Computer Science Engineering. SUB:INT213.

Dr Dhanpratap Singh. YEAR:2020

ACKNOWLEDGEMENT