DBMS INNOVATIVE ASSIGNMENT 2023

COURSE CODE AND SUBJECT - 2CS402: Database management systems
MADE BY:
ISHA KHAKHAR 21BCE090
TOPIC: Applications of data analytics in vehicle parking management system

INTRODUCTION

A new technology that offers solutions to the parking sector is referred to as a parking management system. Any parking management system's fundamental concept is self-evident: it's a system that aids individuals, businesses, and organizations in managing their parking places. With so many variables in play, such as traffic and space availability, managing parking lots is not an easy chore for businesses and organizations. It takes a lot of time, involves labor, and is ineffective. Utilizing a parking management system can assist a business lower its administrative costs associated with parking and lessen the negative effects of its parking lot on the neighborhood.

Educational institutions, local governments, workplaces, commercial establishments, and corporate organizations all use parking software.

SYSTEM SPECIFICATIONS:

Processor: Intel(R) Core (TM) i5-6200U CPU @ 2.30GHz 2.40 GHz

Installed RAM: 8.00 GB (7.88 GB usable)

System type: 64-bit operating system, x64-based processor

Software used: Python (3.9.13) and MySQL

EXISTING WORK: -

Since 1999, Mercy Corps has been carrying out programme activities in Indonesia in response to the country's economic problems. MC has been operating in Indonesia from its headquarters in Jakarta with a number of representative offices all around the nation. MC Indonesia currently has branch offices in

West Sumatra's province of Padang, Ambon's province of Maluku, East Java's province of Bojonegoro, West Java's province of Bandung, Lampung City's province of Lampung, etc.

In order to support programme operations in the aforementioned areas and to improve its operational competitiveness relative to other

organizations, MC has been concentrating on internal and/or external strategies. The Country Strategy for FY 2015 No. 5 of "Increase Mercy Corps staff and organizational capacity to provide strong leadership and programme management skills" also made reference to the endeavor to concentrate on internal strategy.

PROBLEM STATEMENTS:

1) **Inefficient use of existing parking capacity:**

Parking spaces may be oversupplied and used inefficiently due to local zoning laws, building standards, and other development practices.

2) **Parking spaces that are an inconvenience:**

Parking places that cause a nuisance to the locals and businesses. Residents may have trouble locating parking close to their homes, and businesses may have trouble keeping customers.

3) **out of town parking:**

A large percentage of automobiles parked in residential areas are from outside the community.

4) **Low parking turnover rate:**

When cars are left in the same spot for at least four hours (on average), this can happen.

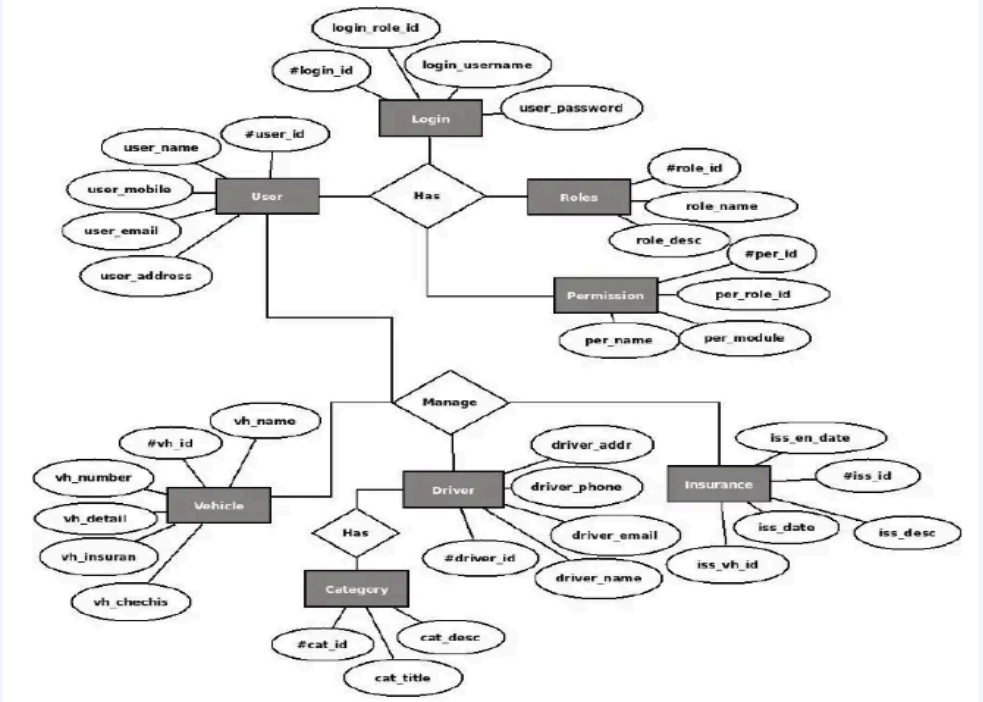
5) **loading and unloading zones:**

Commercial vehicles will obstruct traffic lanes if there isn't enough parking for them to load or unload.

6) Lack of sufficient parking at event site:

Special events may affect traffic flow and necessitate crowd control. Every occasion may result in different transportation problems.

ER – DIAGRAM :



RESULTS:

INPUT:

```
import mysql.connector
import time
from datetime import date

global conn,cursor
```

```
conn =
mysql.connector.connect(host='localhost',user='root',password='Muskann@12')

cursor = conn.cursor()

cursor.execute("CREATE DATABASE parking_sy")
cursor.execute("Use parking_sy")

cursor.execute("""CREATE TABLE Login(
               id int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
               name char(20),
               pwd char(20)
)""")

cursor.execute("""CREATE TABLE parking_space(
               id int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
               type_id int(11),
               status char(20)
)""")

cursor.execute("""CREATE TABLE parking_type(
               id int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
               name char(20),
               price float(7,2)
)""")

cursor.execute("""CREATE TABLE transaction(
               id int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
               vehicle_id char(20),
               parking_id int(11),
               entry_date date,
               exit_date date,
               amount float(10,2)
)""")

sql = "INSERT INTO Login (id, name, pwd) VALUES (%s, %s, %s)"
val = [
    (100,'Tina', 'tina123'),
    (200, 'riya', 'riya123'),
    (300, 'harsh', 'harsh123')
]
cursor.executemany(sql,val)
conn.commit()
print(cursor.rowcount,"was inserted")

sql1='INSERT INTO parking_space (id,type_id,status) VALUES (%s,%s,%s)'
val1=[
    (1,1,'open'),(2,2,'open'),
```

```
    (3,3,'open'),(4,4,'open'),
    (5,5,'open'),(6,1,'open'),
    (7,1,'open'),(8,1,'open'),
    (9,1,'open'),(10,1,'open'),
    (11,1,'open'),(12,1,'open'),
    (13,2,'open'),(14,2,'open'),
    (15,2,'open'),(16,2,'open'),
    (17,2,'open'),(18,2,'open'),
    (19,2,'open'),(20,3,'open'),
    (21,3,'open'),(22,3,'open'),
    (23,3,'open'),(24,3,'open'),
    (25,4,'open'),(26,4,'open'),
    (27,5,'open'),(28,5,'open'),
]
cursor.executemany(sql1,val1)
conn.commit()
print(cursor.rowcount,"was inserted")

sql2='INSERT INTO parking_type(id,name,price) VALUES(%s,%s,%s)'
val2=[
    (1,'Two Wheelar',30.00),
    (2,'car',50.00),
    (3,'bus',250.00),
    (4,'Truck',350.00),
    (5,'trolly',450.00)
]
cursor.executemany(sql2,val2)
conn.commit()
print(cursor.rowcount,"was inserted")

sql3='INSERT INTO
transaction(id,vehicle_id,parking_id,entry_date,exit_date,amount) VALUES
(%s,%s,%s,%s,%s,%s)'
val3=[
    (1,'dl14cb-1007',1,'2021-02-08','2021-03-08',30.00)
]
cursor.executemany(sql3,val3)
conn.commit()
print(cursor.rowcount,"was inserted")

#def clear():
#    for i in range(65):
#        print()

def introduction():
    msg = '''
           PARKING MANAGEMENT SYSTEM
           - An Introduction
```

```
           Parking is one of the big issues in metro cities, day by day basis
parking system are
           coming up with new technologies to solve this issue.

           This project is also trying to solve this simple but very useful
information for parking
           management. The whole database is stored in MySQL table ParkingSystem
that stores their
           parking slot information as well as how long a vehicle is pakrked in
the parking area and
           how much he/she needs to pay for that.

           Besides all these features it also tracks the total money collected
during the period of time
           with its extensive searching system.

           The whole project is divided into four major parts ie addition of
data, modification, searching
           and reporting. All these parts are further divided into menus for easy
navigation. '''

    for x in msg:
        print(x, end ='')
        time.sleep(0.002)

    wait = input('Press any key to continue....')

def made_by():
    msg = '''
            Parking Management System made by  :  Khushi Soni  (21bce123)
                                                  Muskan Bhan  (21bce159)
                                                  Isha Khakhar (21bce090)
            Sem IV
            Nirma University

            Thanks for evaluating my project.
            \n\n\n
    '''

    for x in msg:
        print(x, end='')
        time.sleep(0.002)

    wait = input('Press any key to continue.....')


def display_parking_type_records():
    cursor.execute('select * from parking_type;')
```

```
    records=cursor.fetchall()
    for row in records:
        print(row)

def login():
    while True:
        #clear()
        uname=input('Enter your name:')
        upass=input('Enter your password:')
        cursor.execute('select * from login where name="{}" and
pwd="{}"'.format(uname,upass))
        cursor.fetchall()
        rows=cursor.rowcount
        if rows!=1:
            print('Invalid Login details....Try again')
        else:
            print('You are eligible for operating this system.....')
            print('\n\n\n')
            print('Press any key to continue.....')
            break

def add_parking_type():
    #clear()
    name=input('Enter Parking Type(1. Two Wheeler 2. Car 3. Bus 4. Truck 5.
Trolly):')
    price=input('Enter Parking Price per day:')
    sql='insert into parking_type(name,price)
values("{}",{});'.format(name,price)
    cursor.execute(sql)
    print('\n\n New Offence Type added...')
    no=[]
    no.append(cursor.fetchone())
    print('New Parking Type ID is:() \n\n\n'.format(no[0]))
    wait=input('\n\n\nPress any key to continue.....')

def add_parking_slot_record():
    #clear()
    parking_type_id=input('Enter Parking Type(1. Two wheeler 2. Car 3. Bus 4.
Truck 5. Trolly):')
    status=input('Enter current status(Open/Full):')
    sql='insert into parking_space(type_id,status) values
("{}","{}");'.format(parking_type_id,status)
    cursor.execute(sql)
    print('\n\n New Parking Space Record added...')

    cursor.execute('select max(id) from parking_space;')
    no=[]
    no.append(cursor.fetchone())
```

```
    print('Your parking ID is: {} \n\n\n'.format(no[0]))
    display_parking_type_records()
    wait=input('\n\n\nPress any key to continue...')

def modify_parking_type_record():
    #clear()
    print('Modify Parking Type Screen')
    print('1. Parking Type Name \n')
    print('2. Parking Price \n')
    choice=int(input('Enter your choice:'))
    field=''
    if choice==1:
        field='name'
    if choice==2:
        field='price'

    park_id=input('Enter Parking Type ID:')
    value=input('Enter new values:')
    sql='update parking_type set '+field+'='+value+' where id='+park_id+';'
    cursor.execute(sql)
    print('Record updated succesfully....')
    display_parking_type_records()
    wait=input('\n\n\nPress any key to continue...')

def modify_parking_space_record():
    #clear()
    print('Modify Parking Space Record')
    print('1. Parking Type ID(1-Two Wheeler, 2:Car 3.Bus etc):')
    print('2. Status')
    choice=int(input('Enter your choice:'))
    field=''
    if choice==1:
        field='type_id'
    elif choice==2:
        field='status'
    print('\n\n\n')
    crime_id=input('Enter Parking space ID:')
    value=input('Enter new values:')
    sql=f'update parking_space set {field} = {value} where id = {crime_id};'
    cursor.execute(sql)
    print('Record updated succesfully....')
    wait=input('\n\n\nPress any key to continue...')

def add_new_vehicle():
    #clear()
    print('Vehicle Login Screen')
    print('-'*100)
    vehicle_id=input('Enter Vehicle Number:')
```

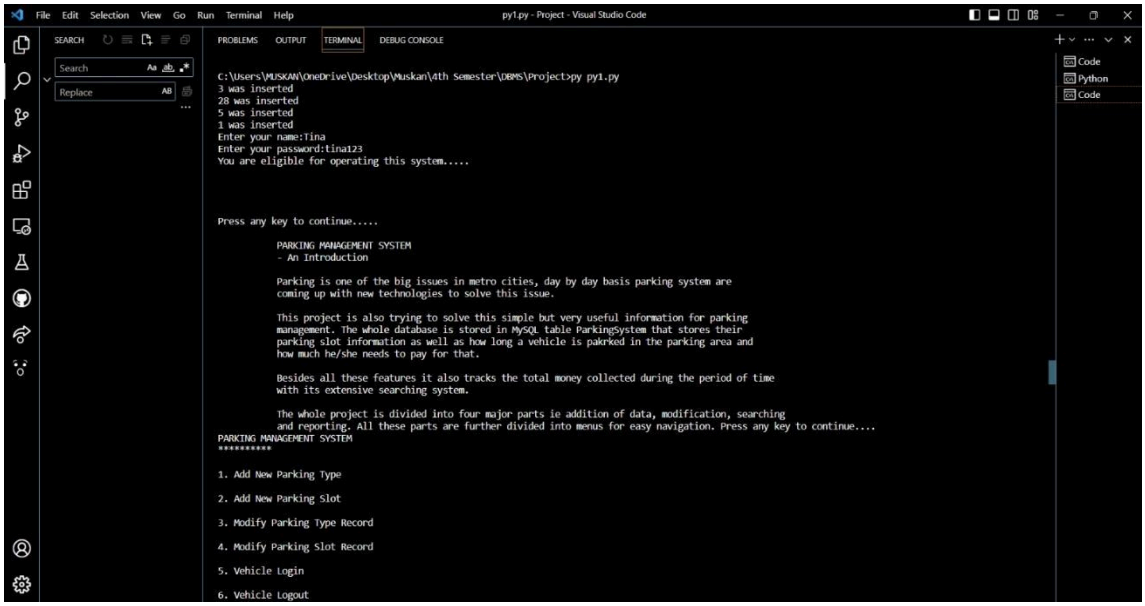
```
    parking_id=input('Enter parking ID:')
    entry_date=date.today()
    sql='insert into transaction(vehicle_id,parking_id,entry_date) values
("{}","{}","{}");'.format(vehicle_id,parking_id,entry_date)
    cursor.execute(sql)
    cursor.execute('update parking_space set status="full" where
id={}'.format(parking_id))
    print('\n\n\n Record added succesfully......')
    wait=input('\n\n\nPress any key to continue...')

def remove_vehicle():
    #clear()
    print('Vehicle Logout Screen')
    print('-'*100)
    vehicle_id=input('Enter vehicle No:')
    exit_date=date.today()
    sql='select parking_id,price,entry_date from transaction tr,parking_space
ps,parking_type pt \
        where tr.parking_id=ps.id and ps.type_id=pt.id and \
        vehicle_id="'+vehicle_id+'" and exit_date is NULL;'

    cursor.execute(sql)
    record=[]
    record.append(cursor.fetchone())
    print(record)
    days=(exit_date-record[2]).days
    if days==0:
        days=days+1
    amount=record[1]*days
    #clear()
    print('Logout details')
    print('-'*100)
    print('Parking ID: {}'.format(record[0]))
    print('Vehicle ID: {}'.format(vehicle_id))
    print('Parking Date: {}'.format(record[2]))
    print('Current Date: {}'.format(exit_date))
    print('Amount Payable: {}'.format(amount))
    wait=input('press any key to continue....')
    sql1='update transaction set exit_date="{}",amount={} where vehicle_id="{}"\
        and exit_date is NULL;'.format(exit_date,amount,vehicle_id)
    sql2='update parking_space set status="open" where id={}'.format(record[0])
    cursor.execute(sql1)
    cursor.execute(sql2)
    wait=input('Vehicle Out from our system succesfully....\n Press any key to
continue..')

def search_menu():
    #clear()
```

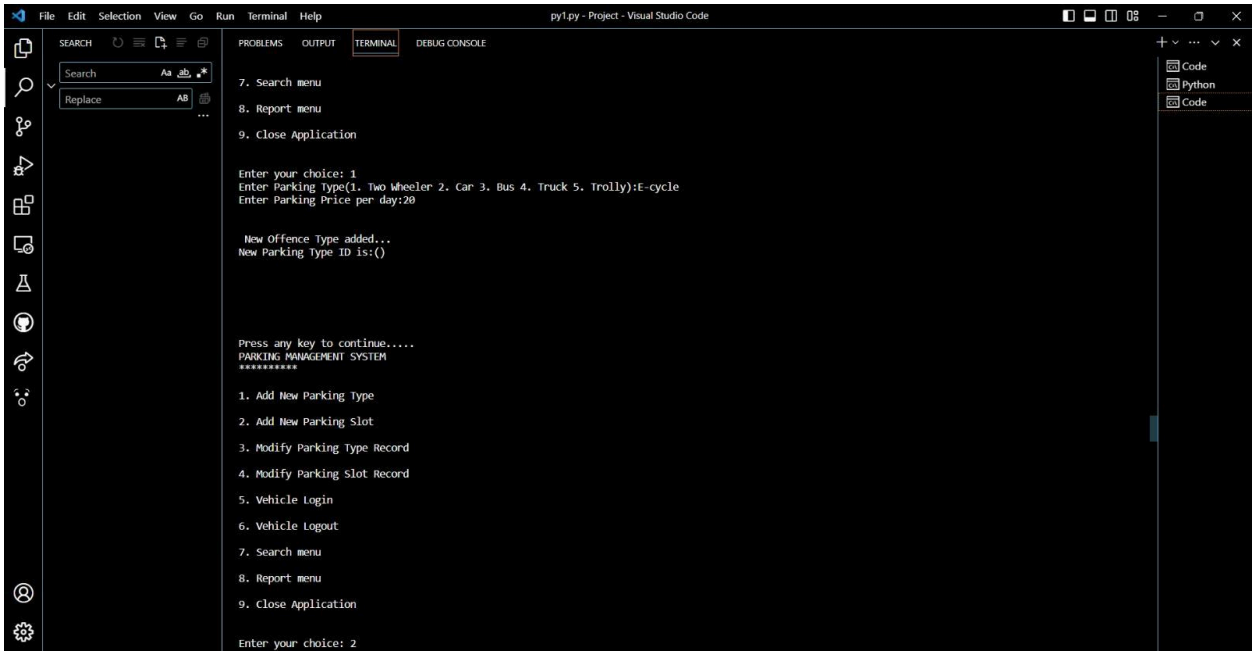
```
    print('Search Parking Menu')
    print('1. Parking ID\n')
    print('2. Vehicle Parking \n')
    print('3. Free Space\n')
    choice=int(input('Enter your choice:'))
    field=''
    if choice==1:
        field='id'
    if choice==2:
        field='vehicle No'
    if choice==3:
        field='status'
    value=input('Enter value to search:')
    if choice==1 or choice==3:
        sql='select ps.id,name,price,status \
            from parking_space ps, parking_type pt where ps.id=pt.id and
ps.id={}'.format(value)
    else:
        sql='select id,vehicle_id,parking_id,entry_date from transaction where
exit_date is NULL;'

    cursor.execute(sql)
    results=cursor.fetchall()
    records=cursor.rowcount
    for row in results:
        print(row)
    if records<1:
        print('Record not found \n\n\n')
    wait=input('\n\n\nPress any key to continue...')


def parking_status(status):
    #clear()
    print('Parking Status: ', status)
    print('-'*100)
    sql = "select * from parking_space where status = '{}'".format(status)
    cursor.execute(sql)
    records = cursor.fetchall()
    for row in records:
        print(row)

def vehicle_status_report():
    #clear()
    print('Vehicle Status - Currently Parked')
    print('_'*10)
    sql= 'select * from transaction where exit_date is NULL; '
    cursor.execute(sql)
    records= cursor.fetchall()
    for row in records:
```

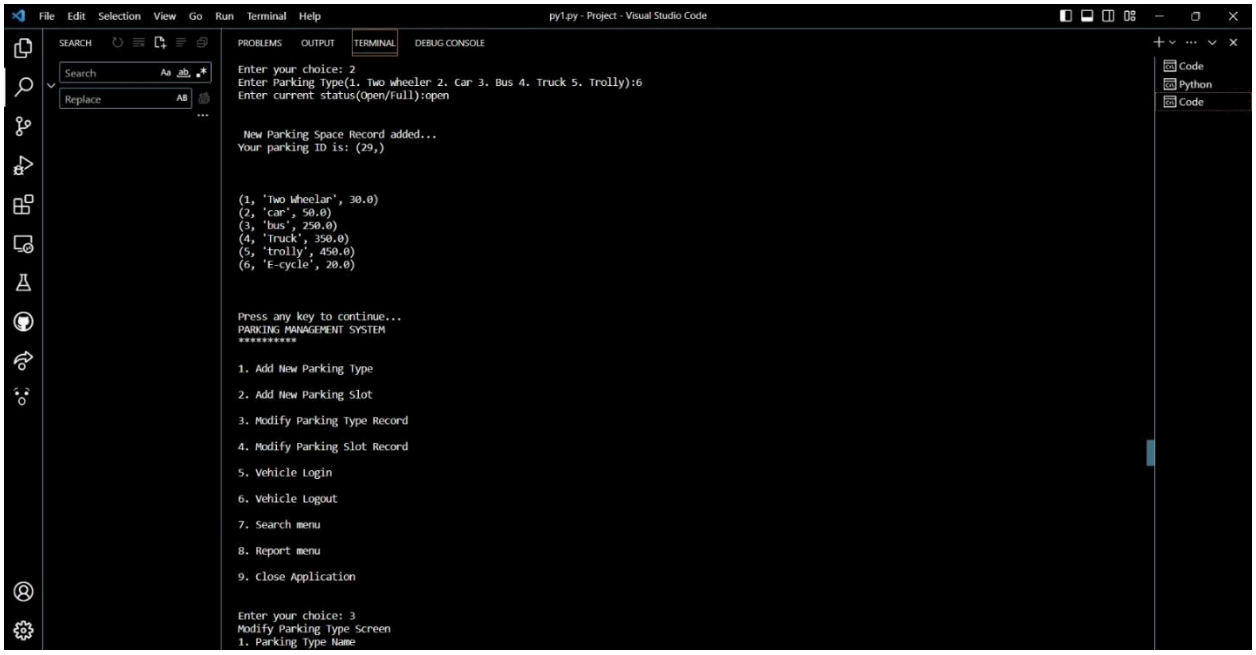
```
        print(row)


def money_collected():
    #clear()
    start_date = input('Enter start date(yyyy-mm-dd): ')
    end_date = input('Enter end date (yyyy-mm-dd): ')
    sql = "select sum(amount) from transaction where entry_date = '{}' and
exit_date = '{}'".format(start_date,end_date)
    cursor.execute(sql)
    result = cursor.fetchone()
    #clear()
    print('Total money collected from {} to {}'.format(start_date,end_date))
    print('*'*10)
    print(result[0])
    wait = input('\n\n\nPress any key to continue.....')

def report_menu():
     while True:
        #clear()
        print('PARKING REPORTS')
        print('-'*100)
        print('1. Parking Types \n')
        print('2. Free Space \n')
        print('3. Occupied Space \n')
        print('4. Vehicle Status \n')
        print('5. Money Collected \n')
        print('6. Exit \n')

        choice = int(input('Enter your choice: '))
        field = ''

        if choice==1:
           display_parking_type_records()
        if choice==2:
            parking_status("open")
        if choice==3:
            parking_status("full")
        if choice==4:
          vehicle_status_report()
        if choice==5:
            money_collected()
        if choice==6:
            break

def main_menu():
    #clear()
    login()
    #clear()
```

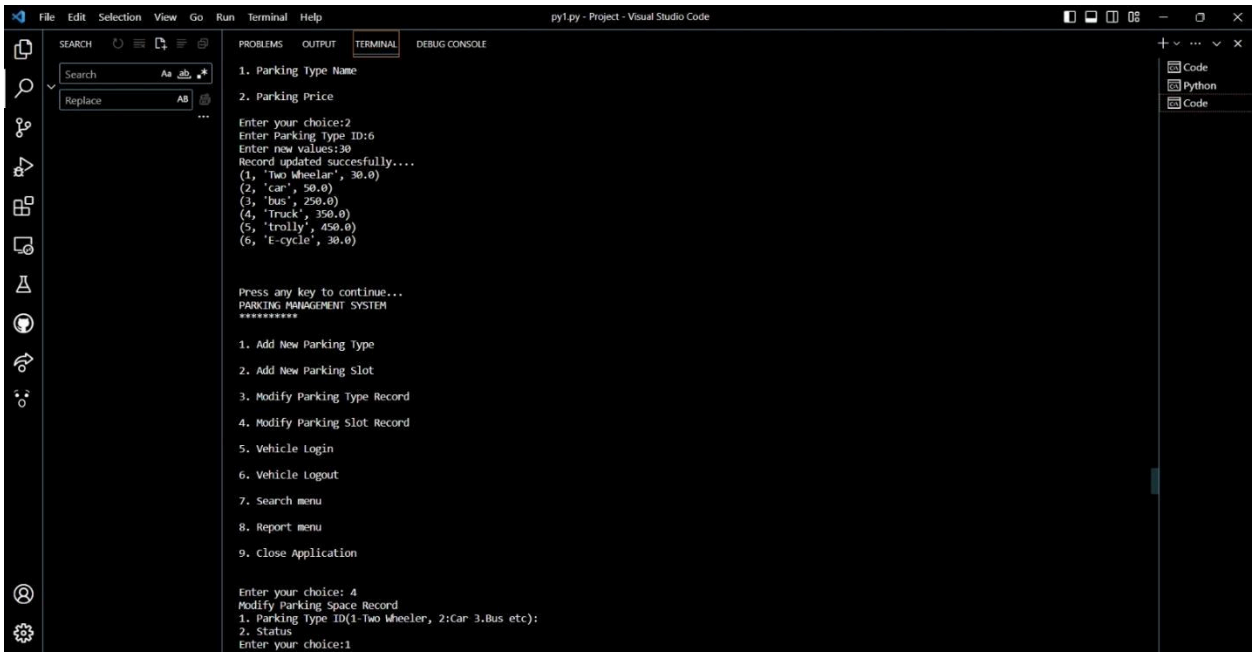
```
    introduction()

    while True:
        #clear()
        print('PARKING MANAGEMENT SYSTEM ')
        print('*'*10)
        print('\n1. Add New Parking Type ')
        print('\n2. Add New Parking Slot')
        print('\n3. Modify Parking Type Record')
        print('\n4. Modify Parking Slot Record')
        print('\n5. Vehicle Login')
        print('\n6. Vehicle Logout')
        print('\n7. Search menu')
        print('\n8. Report menu')
        print('\n9. Close Application')
        print('\n')

        choice = int(input('Enter your choice: '))

        if choice==1:
            add_parking_type()
        if choice==2:
            add_parking_slot_record()
        if choice==3:
            modify_parking_type_record()
        if choice==4:
            modify_parking_space_record()
        if choice==5:
            add_new_vehicle()
        if choice==6:
            remove_vehicle()
        if choice==7:
            search_menu()
        if choice==8:
            report_menu()
        if choice==9:
            break
    made_by()

if __name__ == "__main__":
    main_menu()
```

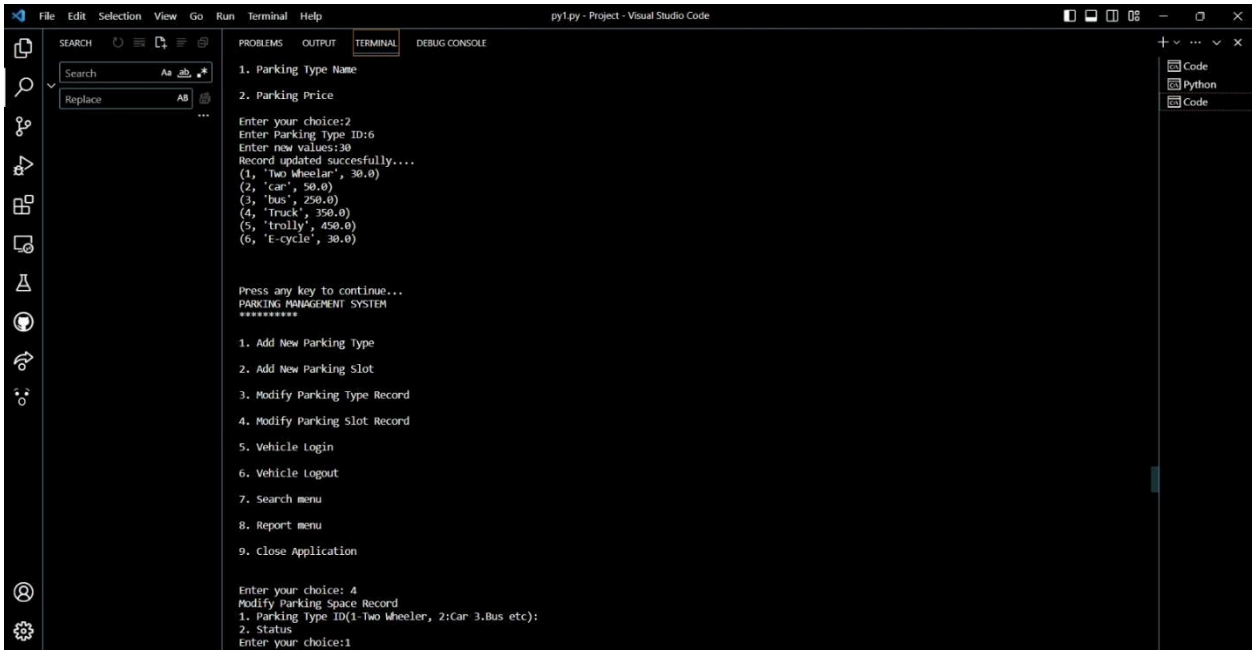
OUTPUT :

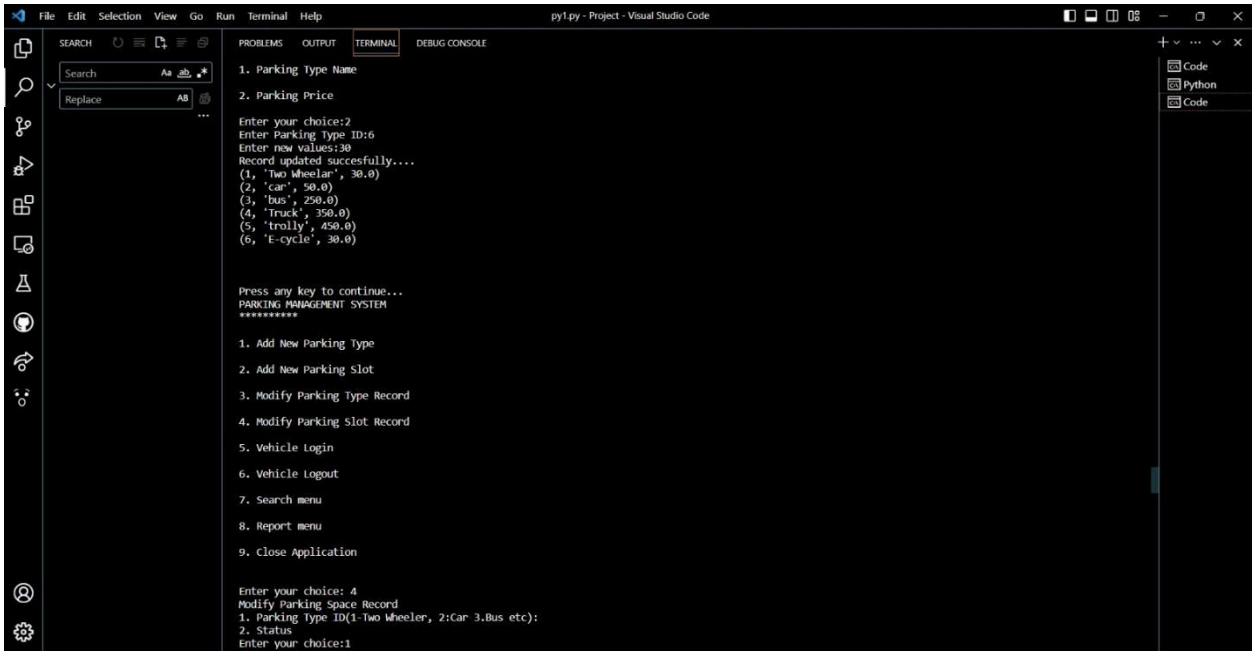

File Edit Selection View Go Run Terminal Help
py1.py - Project - Visual Studio Code
SEARCH
Search
Replace
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
C:\Users\MUSKAN\OneDrive\Desktop\Muskan\4th Semester\DBMS\Project>py py1.py
3 was inserted
28 was inserted
5 was inserted
1 was inserted
Enter your name:Tina
Enter your password:tina123
You are eligible for operating this system.....
Press any key to continue.....
PARKING MANAGEMENT SYSTEM
- An Introduction
Parking is one of the big issues in metro cities, day by day basis parking system are coming up with new technologies to solve this issue.
This project is also trying to solve this simple but very useful information for parking management. The whole database is stored in MySQL table ParkingSystem that stores their parking slot information as well as how long a vehicle is pakrked in the parking area and how much he/she needs to pay for that.
Besides all these features it also tracks the total money collected during the period of time with its extensive searching system.
The whole project is divided into four major parts ie addition of data, modification, searching and reporting. All these parts are further divided into menus for easy navigation. Press any key to continue....
PARKING MANAGEMENT SYSTEM
****
1. Add New Parking Type
2. Add New Parking Slot
3. Modify Parking Type Record
4. Modify Parking Slot Record
5. Vehicle Login
6. Vehicle Logout
Code
Python
Code

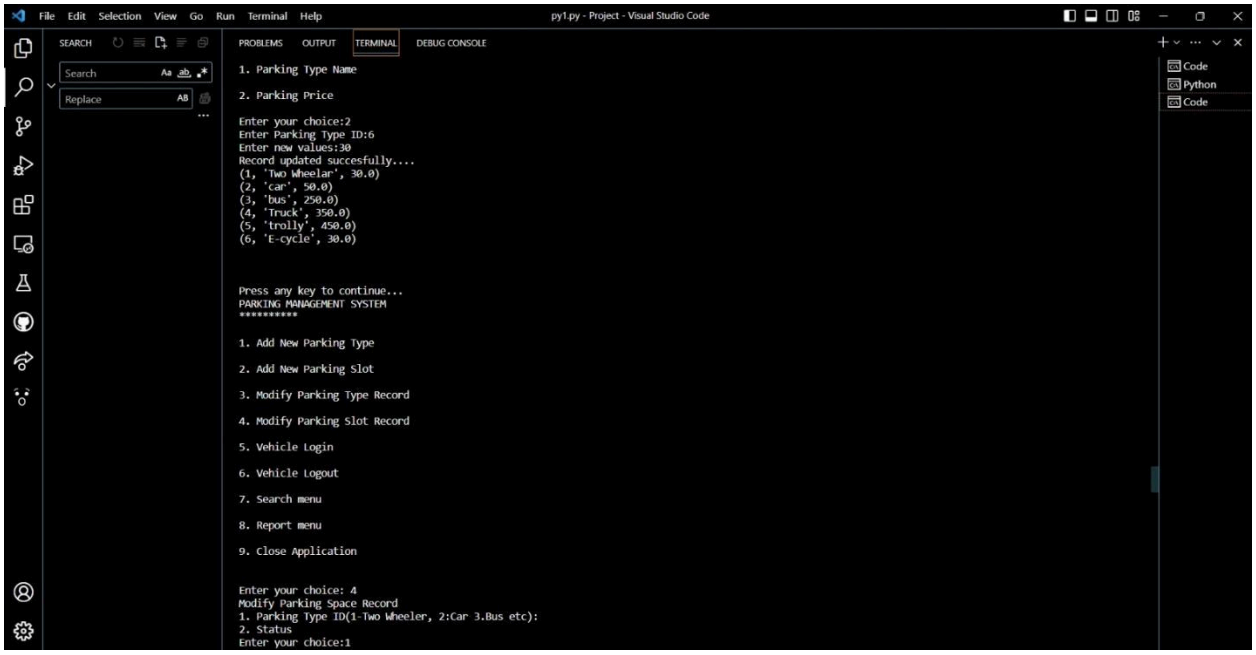

File Edit Selection View Go Run Terminal Help
py1.py - Project - Visual Studio Code
SEARCH
Search
Replace
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
7. Search menu
8. Report menu
9. Close Application
Enter your choice: 1
Enter Parking Type(1. Two Wheeler 2. Car 3. Bus 4. Truck 5. Trolly):E-cycle
Enter Parking Price per day:20
New Offence Type added...
New Parking Type ID is:()
Press any key to continue.....
PARKING MANAGEMENT SYSTEM
****
1. Add New Parking Type
2. Add New Parking Slot
3. Modify Parking Type Record
4. Modify Parking Slot Record
5. Vehicle Login
6. Vehicle Logout
7. Search menu
8. Report menu
9. Close Application
Enter your choice: 2
Code
Python
Code

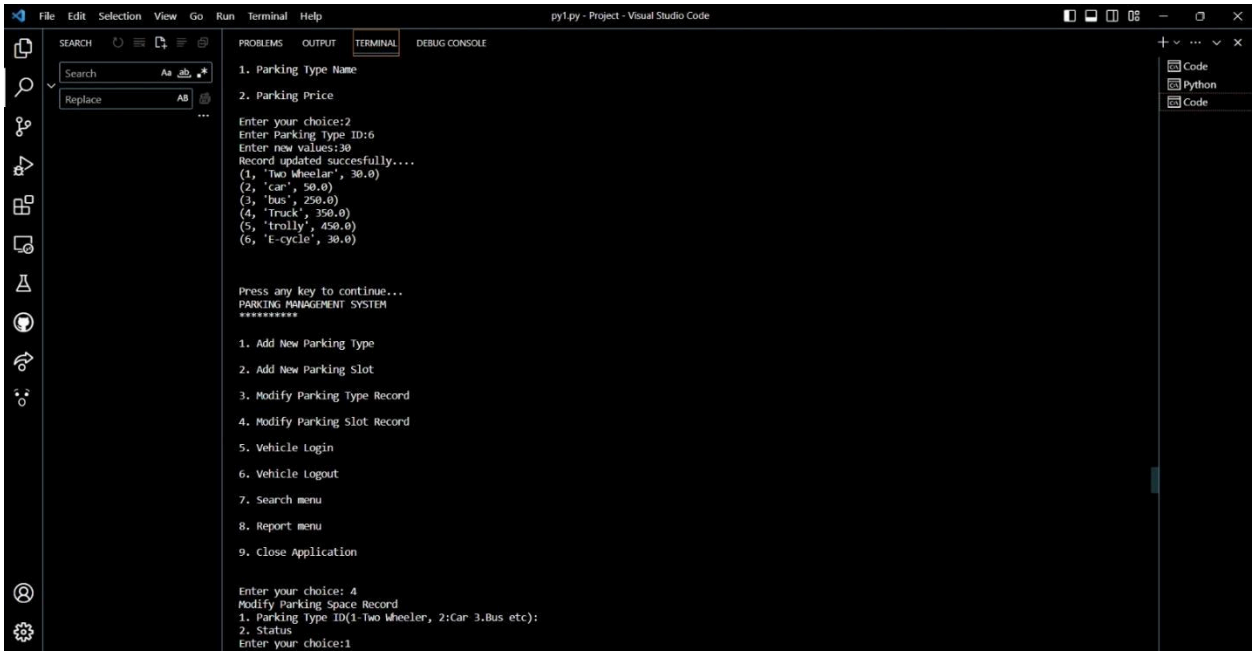

File Edit Selection View Go Run Terminal Help
py1.py - Project - Visual Studio Code
SEARCH
Search
Replace
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Enter your choice: 2
Enter Parking Type(1. Two Wheeler 2. Car 3. Bus 4. Truck 5. Trolly):6
Enter current status(Open/Full):open
New Parking Space Record added...
Your parking ID is: (29,)
(1, 'Two Wheelar', 30.0)
(2, 'car', 50.0)
(3, 'bus', 250.0)
(4, 'Truck', 350.0)
(5, 'trolly', 450.0)
(6, 'E-cycle', 20.0)
Press any key to continue...
PARKING MANAGEMENT SYSTEM
****
1. Add New Parking Type
2. Add New Parking Slot
3. Modify Parking Type Record
4. Modify Parking Slot Record
5. Vehicle Login
6. Vehicle Logout
7. Search menu
8. Report menu
9. Close Application
Enter your choice: 3
Modify Parking Type Screen
1. Parking Type Name
Code
Python
Code

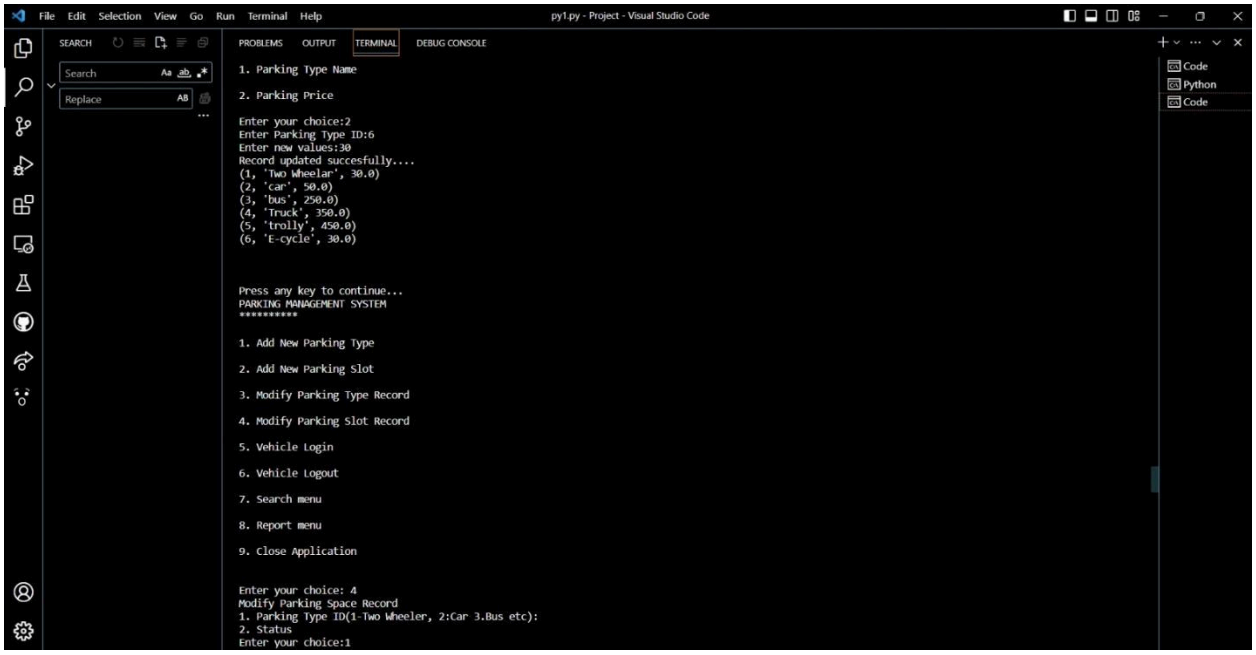

File Edit Selection View Go Run Terminal Help
py1.py - Project - Visual Studio Code
SEARCH
Search
Replace
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1. Parking Type Name
2. Parking Price
Enter your choice:2
Enter Parking Type ID:6
Enter new values:30
Record updated succesfully....
(1, 'Two Wheelar', 30.0)
(2, 'car', 50.0)
(3, 'bus', 250.0)
(4, 'Truck', 350.0)
(5, 'trolly', 450.0)
(6, 'E-cycle', 30.0)
Press any key to continue...
PARKING MANAGEMENT SYSTEM
****
1. Add New Parking Type
2. Add New Parking Slot
3. Modify Parking Type Record
4. Modify Parking Slot Record
5. Vehicle Login
6. Vehicle Logout
7. Search menu
8. Report menu
9. Close Application
Enter your choice: 4
Modify Parking Space Record
1. Parking Type ID(1-Two Wheeler, 2:Car 3.Bus etc):
2. Status
Enter your choice:1
Code
Python
Code

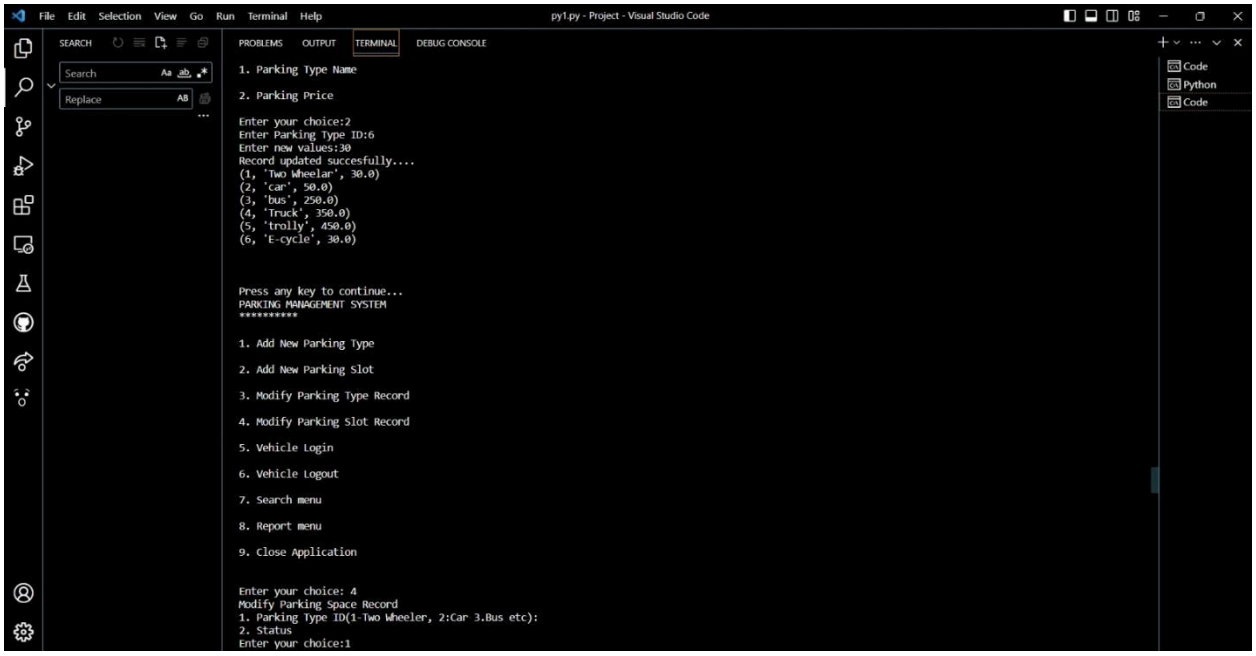

File Edit Selection View Go Run Terminal Help
py1.py - Project - Visual Studio Code
SEARCH
Search
Replace
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Code
Python
Code
1. Parking Type Name
2. Parking Price
Enter your choice:2
Enter Parking Type ID:6
Enter new values:30
Record updated succesfully....
(1, 'Two Wheelar', 30.0)
(2, 'car', 50.0)
(3, 'bus', 250.0)
(4, 'Truck', 350.0)
(5, 'trolly', 450.0)
(6, 'E-cycle', 30.0)
Press any key to continue...
PARKING MANAGEMENT SYSTEM
****
1. Add New Parking Type
2. Add New Parking Slot
3. Modify Parking Type Record
4. Modify Parking Slot Record
5. Vehicle Login
6. Vehicle Logout
7. Search menu
8. Report menu
9. Close Application
Enter your choice: 4
Modify Parking Space Record
1. Parking Type ID(1-Two Wheeler, 2:Car 3.Bus etc):
2. Status
Enter your choice:1


File Edit Selection View Go Run Terminal Help
py1.py - Project - Visual Studio Code
SEARCH
Search
Replace
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Code
Python
Code
1. Parking Type Name
2. Parking Price
Enter your choice:2
Enter Parking Type ID:6
Enter new values:30
Record updated succesfully....
(1, 'Two Wheelar', 30.0)
(2, 'car', 50.0)
(3, 'bus', 250.0)
(4, 'Truck', 350.0)
(5, 'trolly', 450.0)
(6, 'E-cycle', 30.0)
Press any key to continue...
PARKING MANAGEMENT SYSTEM
****
1. Add New Parking Type
2. Add New Parking Slot
3. Modify Parking Type Record
4. Modify Parking Slot Record
5. Vehicle Login
6. Vehicle Logout
7. Search menu
8. Report menu
9. Close Application
Enter your choice: 4
Modify Parking Space Record
1. Parking Type ID(1-Two Wheeler, 2:Car 3.Bus etc):
2. Status
Enter your choice:1


File Edit Selection View Go Run Terminal Help
py1.py - Project - Visual Studio Code
SEARCH
Search
Replace
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Code
Python
Code
1. Parking Type Name
2. Parking Price
Enter your choice:2
Enter Parking Type ID:6
Enter new values:30
Record updated succesfully....
(1, 'Two Wheelar', 30.0)
(2, 'car', 50.0)
(3, 'bus', 250.0)
(4, 'Truck', 350.0)
(5, 'trolly', 450.0)
(6, 'E-cycle', 30.0)
Press any key to continue...
PARKING MANAGEMENT SYSTEM
****
1. Add New Parking Type
2. Add New Parking Slot
3. Modify Parking Type Record
4. Modify Parking Slot Record
5. Vehicle Login
6. Vehicle Logout
7. Search menu
8. Report menu
9. Close Application
Enter your choice: 4
Modify Parking Space Record
1. Parking Type ID(1-Two Wheeler, 2:Car 3.Bus etc):
2. Status
Enter your choice:1


File Edit Selection View Go Run Terminal Help
py1.py - Project - Visual Studio Code
SEARCH
Search
Replace
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Code
Python
Code
1. Parking Type Name
2. Parking Price
Enter your choice:2
Enter Parking Type ID:6
Enter new values:30
Record updated succesfully....
(1, 'Two Wheelar', 30.0)
(2, 'car', 50.0)
(3, 'bus', 250.0)
(4, 'Truck', 350.0)
(5, 'trolly', 450.0)
(6, 'E-cycle', 30.0)
Press any key to continue...
PARKING MANAGEMENT SYSTEM
****
1. Add New Parking Type
2. Add New Parking Slot
3. Modify Parking Type Record
4. Modify Parking Slot Record
5. Vehicle Login
6. Vehicle Logout
7. Search menu
8. Report menu
9. Close Application
Enter your choice: 4
Modify Parking Space Record
1. Parking Type ID(1-Two Wheeler, 2:Car 3.Bus etc):
2. Status
Enter your choice:1





LEARNING OUTCOME:

We learnt about the connectivity of python and SQL.

We learnt about the how automobiles got connected to technology and their existing works and projects, distribution about them. We got to

know the problems about the parking and found out the solution in best way possible using mysql and python.