Case Study

Parking Management System

Professor
Dr. Pramod Yelmewad

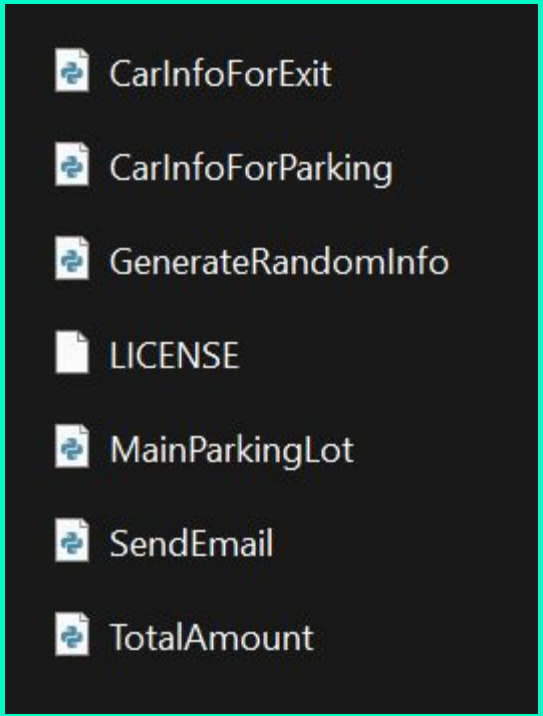
KARTHIK AVINASH
21BCS052
DBMS
CSE-Sem-4A

INDEX

Instructions

A zipped file was attached with this case study. We need to:

1. Extract the zip file.
2. Run the code. If you get python-specific errors, install respective packages.
3. For SQL specific errors, analyse the code, check what is required and do the needful.
4. The working code should allows you to allot a parking and generate a parking bill at the time of exit.



AIM

1. Working Model without errors.
2. Allot a parking lot.
3. Get Ticket (gmail).
4. Generate a parking bill.
5. Handle unexpected cases.

(takeaway with concepts)

(IDE: VSCODE)

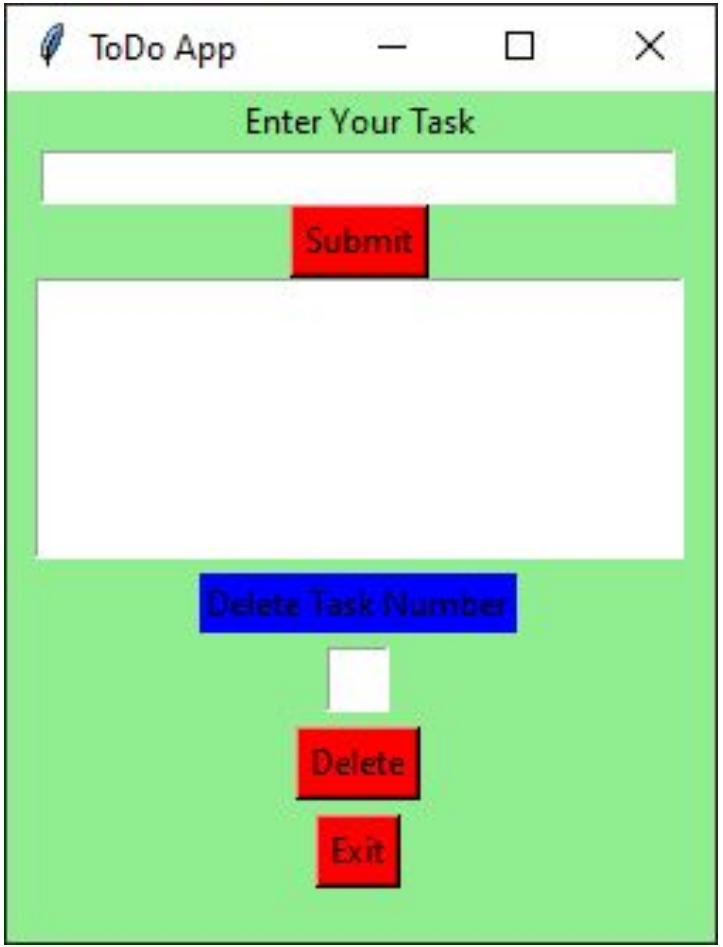
Objectives

1. Resolve all the errors produced (assignment)
2. See logical aspects of some operations.
3. Go through all the aspects that help when this application when used in real-time, with huge dataset.

Install “Tkinter” Library

1. Python, pip
2. Install Tkinter

   pip install tk



Next: Steps to solve errors...

Step-1: Run MainParkingLot.py

```
PS C:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement> python -u "c:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement\
MainParkingLot.py"
Traceback (most recent call last):
  File "c:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement\MainParkingLot.py", line 3, in <module>
    import CarInfoForParking
  File "c:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement\CarInfoForParking.py", line 4, in <module>
    import GenerateRandomInfo
  File "c:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement\GenerateRandomInfo.py", line 6, in <module>
    mydb = mysql.connector.Connect(
           ^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\site-packages\mysql\connector\pooling.py", line 294, in connect
    return MySQLConnection(*args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\site-packages\mysql\connector\connection.py", line 167, in __init__
    self.connect(**kwargs)
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\site-packages\mysql\connector\abstracts.py", line 1178, in connect
    self._open_connection()
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\site-packages\mysql\connector\connection.py", line 573, in _open_connection
    self._do_auth(
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\site-packages\mysql\connector\connection.py", line 312, in _do_auth
    self._auth_switch_request(username, password)
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\site-packages\mysql\connector\connection.py", line 369, in _auth_switch_request

    raise get_exception(packet)
mysql.connector.errors.ProgrammingError: 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
PS C:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement>
```

Step Taken:

1. Go through all the 6 ".py" files, and give valid credentials to the MySql connector.
2. We find that in "CarInfoForExit", "CarInfoForParking", "GenerateRandomInfo" files. (Also create a database called "ParkingLot")

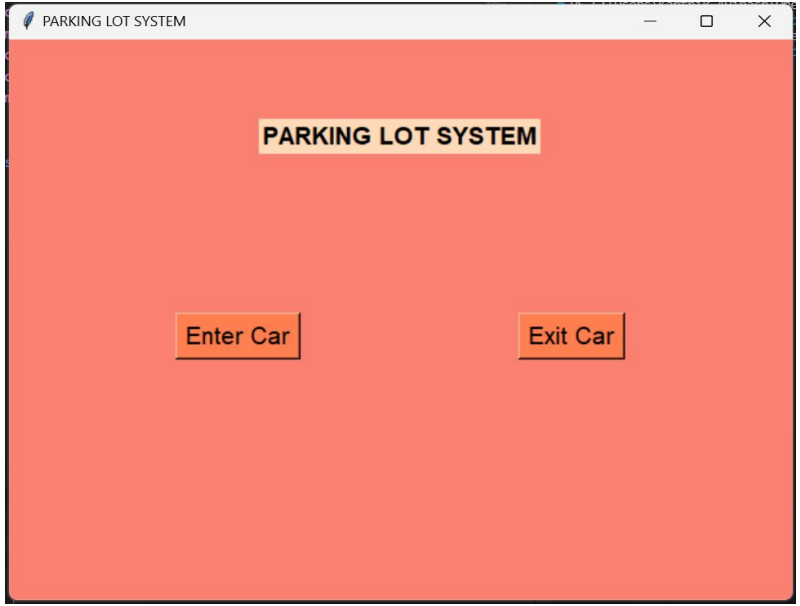
```
mydb = mysql.connector.Connect(
    host='localhost',
    user='vscode',
    password='1234',
    database='ParkingLot'
)
```

Note:

a. mysql.connector is not an ODBC (Open Database Connectivity) driver.
b. mysql.connector is a Python library that provides a pure-Python MySQL client to interact with MySQL databases.
c. It is designed to be lightweight, efficient, and easy to use, and it allows you to execute SQL queries and manage transactions, among other things.

Step-2: Run MainParkingLot.py again

Step-3: Press “Enter Car” in tkinter window.




```
PS C:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement> python -u "c:\Users\K
arthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement\MainParkingLot.py"
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Program Files\Python311\Lib\tkinter\__init__.py", line 1948, in __call__
    return self.func(*args)
           ^^^^^^^^^^^^^^^^
  File "c:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement\MainParkingLot.py
", line 54, in CheckSpot
    if GenerateRandomInfo.getSpotNumDB() is None:
       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement\GenerateRandomInf
o.py", line 79, in getSpotNumDB
    mycursor.execute(sql)
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\site-packages\mysql\conn
ector\cursor.py", line 617, in execute
    self._handle_result(self._connection.cmd_query(stmt))
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\site-packages\mysql\conn
ector\connection.py", line 1046, in cmd_query
    result = self._handle_result(self._send_cmd(ServerCmd.QUERY, query))
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\site-packages\mysql\conn
ector\connection.py", line 824, in _handle_result
    raise get_exception(packet)
mysql.connector.errors.ProgrammingError: 1146 (42S02): Table 'parkinglot.parkingspot' does
n't exist
```

Step Taken:

1. Locate error:

```python
def getSpotNumDB():
    mycursor = mydb.cursor()

    sql = "SELECT SrNum FROM ParkingLot.ParkingSpot WHERE Spot = 'Null'"

    mycursor.execute(sql)
    li = []
    for i in mycursor:
        li.append(list(i)[0])
```

Relational Algebra:

$$\pi_{SrNum}\left(\sigma_{spot='Null'}(ParkingSpot)\right)$$

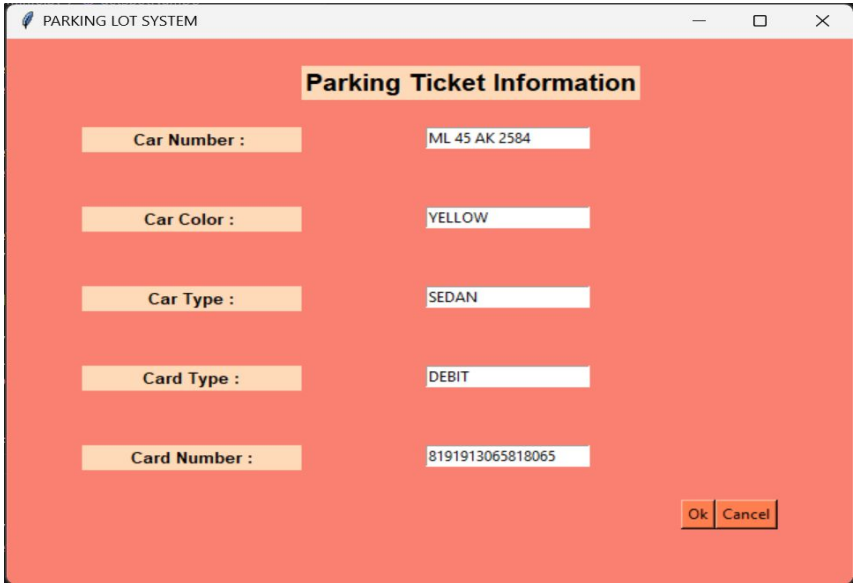
2. Create table ParkingSpot with attributes: “SrNum”, “Spot”. (check all files where such a sql statement is found and get a list of all its attributes). (We have only 2 attributes).

```sql
CREATE TABLE parkingSpot (
  SrNum varchar(200) NOT NULL,
  spot varchar(200),
  PRIMARY KEY (SrNum)
);
```

```sql
INSERT INTO parkingSpot (SrNum, spot)
VALUES
  ('PS001', "Null"),
  ('PS002', "Null"),
  ('PS003', "Null"),
  ('PS004', "Null"),
  ('PS005', "Null");
```

SrNum	spot
abc Filter...	abc Filter...
PS001	Null
PS002	Null
PS003	Null
PS004	Null
PS005	Null

Step-4: Run MainParkingLot.py again and press “enter car”.



Step-5: Press “OK”. We get error.

```
PS C:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManagement>
p\DEMO\ParkingManagement\MainParkingLot.py"
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Program Files\Python311\Lib\tkinter\__init__.py", line 19
    return self.func(*args)
           ^^^^^^^^^^^^^^^
  File "c:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManage
    self.storeInfoInDB()
  File "c:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\ParkingManage
DB
    myc.execute(sql, val)
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\si
 execute
    self._handle_result(self._connection.cmd_query(stmt))
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\si
6, in cmd_query
    result = self._handle_result(self._send_cmd(ServerCmd.QUERY, que
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Karthik Avinash\AppData\Roaming\Python\Python311\si
, in _handle_result
    raise get_exception(packet)
mysql.connector.errors.ProgrammingError: 1146 (42S02): Table 'parkin
```

Step Taken:

1. Locate error:

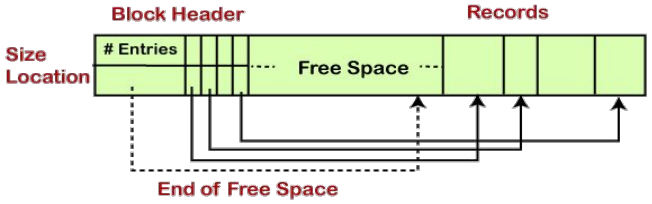
```python
# Store Information in Data base
def storeInfoInDB(self):
    myc = mydb.cursor()
    sql = """INSERT INTO ParkingLot.ParkingInfo
    (CarNumber, CarColor, CarType, CardType, CardNumber,
    ParkingTime, ParkingDate, SpotNum)
    VALUES(%s, %s, %s, %s, %s, %s, %s, %s)"""
```

2. Create table "ParkingInfo" with attributes as below(check all files where such a sql statement is found and get a list of all its attributes). (We have only 2 attributes). (Dynamically data inserted into table).

```sql
CREATE TABLE ParkingInfo (
  CarNumber varchar(250) Primary Key,
  CarColor VARCHAR(255),
  CarType VARCHAR(255) ,
  CardType varchar(250) ,
  CardNumber varchar(250) ,
  ParkingTime varchar(250),
  ParkingDate varchar(250),
  SpotNum varchar(250)
);
```

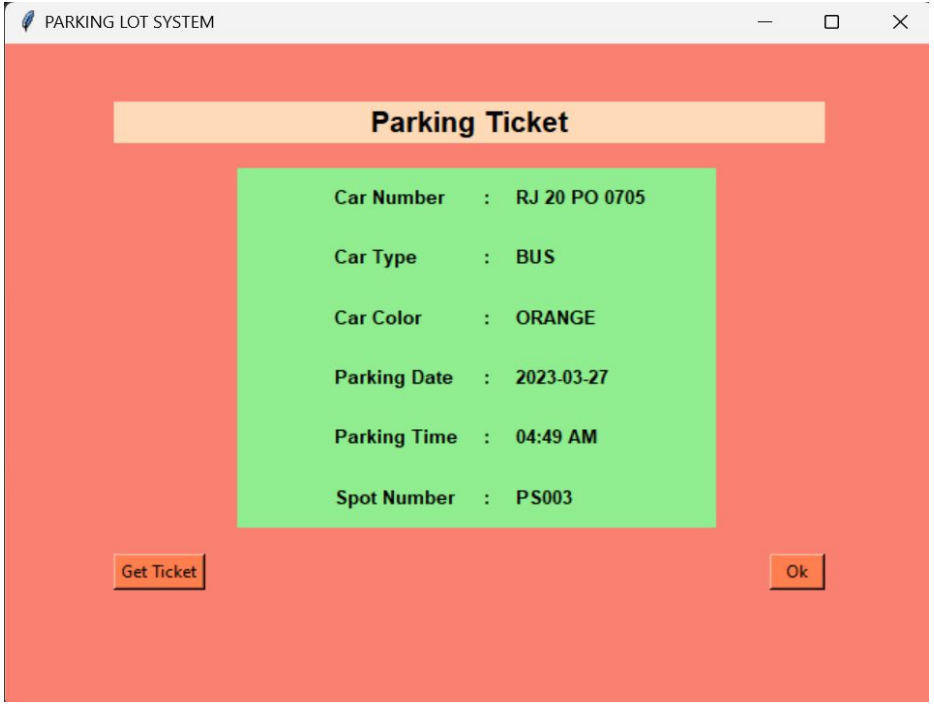
Note the SQL statement:

```python
sql = "UPDATE ParkingSpot SET Spot = 'Parked' WHERE SrNum = %s"
val = [self.SpotNum]
```



Slotted-page structure

Step-6: Run MainParkingLot.py again and press “enter car”/”OK”.

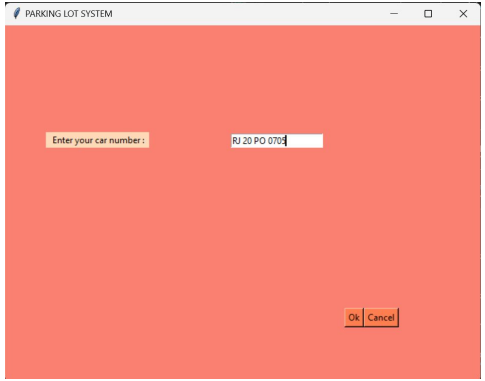


Step-7: Reflected in database because of:

```
mydb.commit()
```

CarNumber	CarColor	CarType	CardType	CardNum
Filter...	Filter...	Filter...	Filter...	Filte
RJ 20 PO 0705	ORANGE	BUS	CREDIT	37856924

Step-8: Exit your car: RJ 20 PO 0705



Step-9: Press OK.

Amount is based on the

```
def AmountCal(hour, minute):
    HourAmount = 30
    TotalAmountForHour = hour * HourAmount
    TotalAmountForMinute = 0
    if (minute < 60) and (minute >= 30):
        TotalAmountForMinute = 20
    elif (minute < 30) and (minute >= 15):
        TotalAmountForMinute = 15
    elif (minute < 15) and (minute >= 1):
        TotalAmountForMinute = 10

    amount = TotalAmountForHour + TotalAmountForMinute
    return amount
```

Let's check if it is deleted in Db or not.

```
def DeleteDataDB(self):
    mycursor = mydb.cursor()
    sql = "DELETE FROM ParkingLot.ParkingInfo WHERE CarNumber = %s"
    val = [self.carNumber]
    mycursor.execute(sql, val)

    sql = "UPDATE ParkingLot.ParkingSpot SET Spot = 'Null' WHERE SrNum = %s"
    val = [self.SpotNum]
    mycursor.execute(sql, val)

    mydb.commit()
```

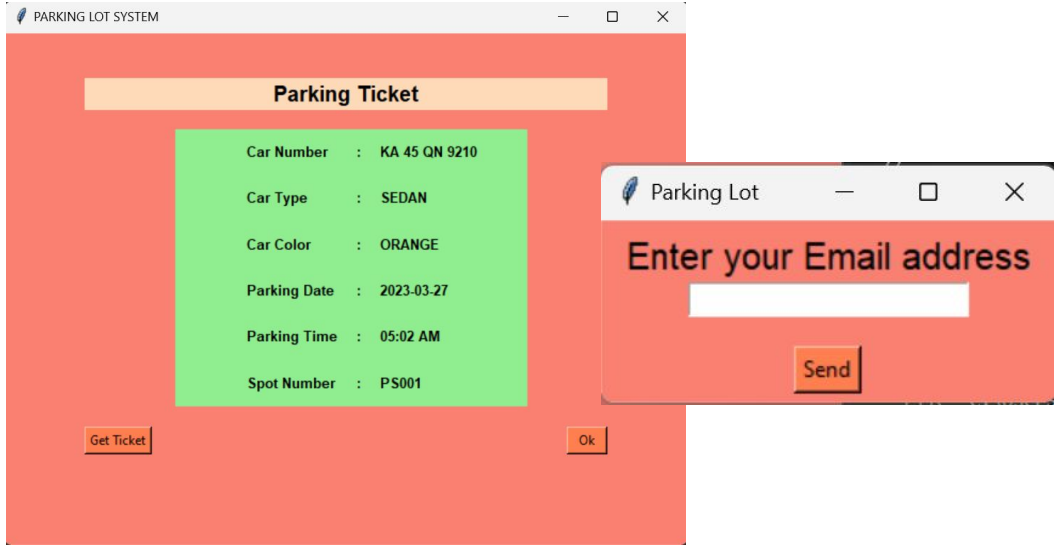
No ParkingInfo in Db.

CarNumber	CarColor	CarType	CardType	CardNum
abc Filter...	abc Filter...	abc Filter...	abc Filter...	abc Filte
No data				

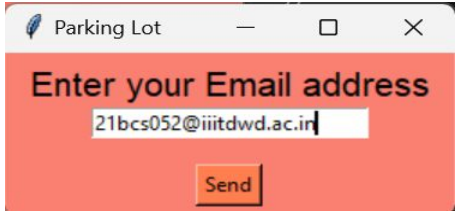
All ParkingSpot are “NULL”

SrNum	spot
abc Filter...	abc Filter...
PS001	Null
PS002	Null
PS003	Null
PS004	Null
PS005	Null

We will try Get Ticket button.



Step-10: Enter mail-id. Mail will be sent to the entered email address from the hardcoded email address in the file.



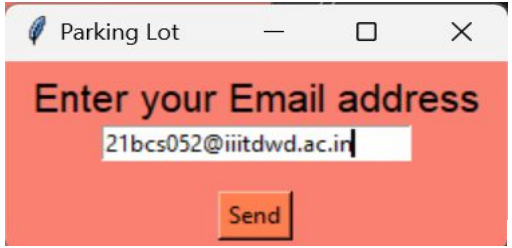
We need to enter our mail-id and password in the file “SendEmail.py”.

```
def send_Email(self):

    self.receiverEmail = self.mailAddress.get()
    try:
        ob = smtplib.SMTP('imap.gmail.com', 587)
        ob.starttls()
        ob.login("21bcs052@iiitdwd.ac.in", "<Your password>")
        subject = "PARKING TICKET"

        body = self.str
        message = 'Subject : {}\n\n{}'.format(subject, body)
        ob.sendmail("21bcs052@iiitdwd.ac.in", self.receiverEmail, message)
        # print("send successful..")
        ob.quit()
```

We get error:

```
PS C:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\Parki
ngManagement> python -u "c:\Users\Karthik Avinash\OneDr
ive\Desktop\DEMO\ParkingManagement\MainParkingLot.py"
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Program Files\Python311\Lib\tkinter\__init__
.py", line 1948, in __call__
    return self.func(*args)
           ^^^^^^^^^^^^^^^^
  File "c:\Users\Karthik Avinash\OneDrive\Desktop\DEMO\
ParkingManagement\SendEmail.py", line 41, in send_Email
    ob.login("MailID@gmail.com", "password")
  File "C:\Program Files\Python311\Lib\smtplib.py", lin
e 750, in login
    raise last_exception
  File "C:\Program Files\Python311\Lib\smtplib.py", lin
e 739, in login
    (code, resp) = self.auth(
                   ^^^^^^^^^^
  File "C:\Program Files\Python311\Lib\smtplib.py", lin
e 662, in auth
    raise SMTPAuthenticationError(code, resp)
smtplib.SMTPAuthenticationError: (535, b'5.7.8 Username
 and Password not accepted. Learn more at\n5.7.8  https
://support.google.com/mail/?p=BadCredentials e15-20020a
a7824f000000b006089fb79f1asm3578767pfn.208 - gsmtp')
```

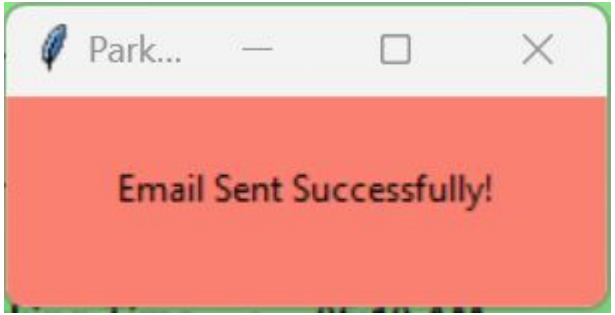
Step-11: Enter mail-id. It does not matter what you enter. Make sure the syntax is correct. As they are hardcoded.



We need to enter our mail-id and password in the file “SendEmail.py”.

Car Number : ML 19 AQ 5935
Car Type : SEDAN
Car Color : VIOLET
Date : 2023-03-27
Parking Time : 05:10 AM
Spot : PS003

Success in sending email:



21bcs052@iiitdwd.ac.in
to

from:	**21bcs052@iiitdwd.ac.in**
to:	
date:	27 Mar 2023, 05:13
subject:	PARKING TICKET
mailed-by:	iiitdwd.ac.in
Signed by:	iiitdwd.ac.in
security:	Standard encryption (TLS) Learn more

https://myaccount.google.com/lesssecureapps

Google Account

← Less secure app access

Some apps and devices use less secure sign-in technology, which makes your account vulnerable. You can turn off access for these apps, which we recommend, or turn it on if you want to use them despite the risks. Google will automatically turn this setting OFF if it's not being used. Learn more

Allow less secure apps: ON

Questions to be answered:

(We have 5 slots)

1. What if all slots are filled?
2. Will I be able to book a slot as soon as someone exit the parking?
3. Can we exit the same car more than once.

We have seen:

1. Delete records and insert records have committed to the database (Used mydb.commit()).
2. But not seen the UPDATES made to the records in the database.

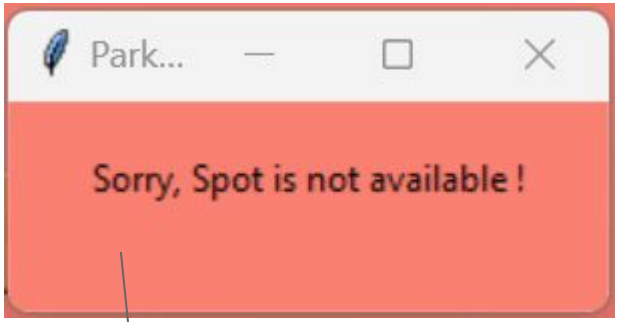
All Basic Functionality of the project is complete

1. Fill All Parking lots

Enter new car until below message pops up

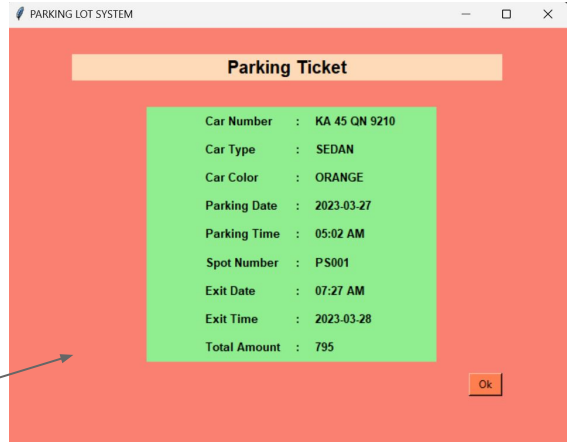
Database:

SrNum	spot
Filter...	Filte
PS001	Parked
PS002	Parked
PS003	Parked
PS004	Parked
PS005	Parked



Exit Car

Exited successfully! (1 lot is free)

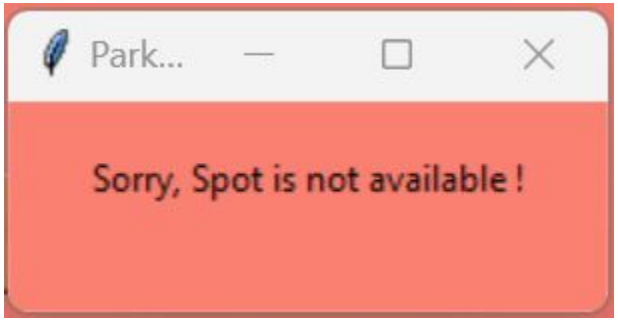


Now enter a new car

Database:

SrNum	spot
Filter...	Filter...
PS001	Parked
PS002	Null
PS003	Parked
PS004	Parked
PS005	Parked

Enter new car (Still same message)



Note: DB is consistent. But not the cached data in the frontend.

Solution 1: create a new connection.

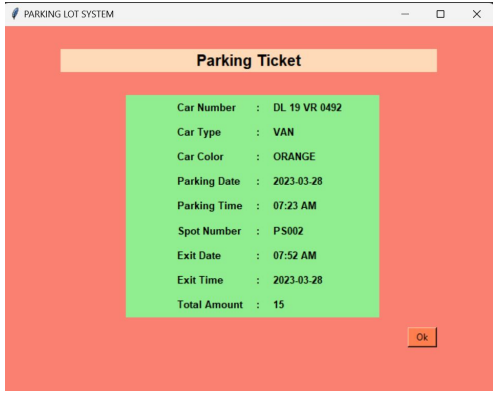
```python
def getSpotNumDB():

    mydb = mysql.connector.Connect(
    host='localhost',
    user='vscode',
    password='1234',
    database='ParkingLot'
    )
    mycursor = mydb.cursor()

    sql = "SELECT SrNum FROM ParkingLot

    mycursor.execute(sql)
    li = []
    for i in mycursor:
        li.append(list(i)[0])

    if len(li) == 0:
        return None

    # print(li)
    mydb.commit()
    return random.choice(li)
```

Solution 2: Use a function reconnect()

```python
def getSpotNumDB():
    mydb.reconnect()
    mycursor = mydb.cursor()

    sql = "SELECT SrNum FROM Par

    mycursor.execute(sql)
    li = []
    for i in mycursor:
        li.append(list(i)[0])

    if len(li) == 0:
        return None

    # print(li)
    mydb.commit()
    return random.choice(li)
```

2. Exiting same car more than once:

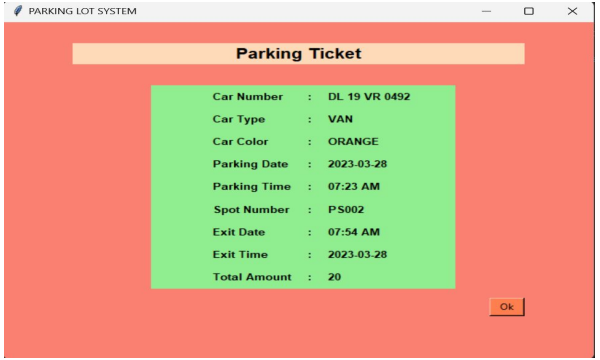
Exit car(DL 19 VR 0492)=>(₹15):



Database:

CarNumber	CarColor	CarType
abc Filter...	abc Filter...	abc Filter...
BR 35 PO 2634	VIOLET	HATCHBACK
PB 19 QN 2341	PINK	SEDAN
UP 19 AK 8322	YELLOW	SEDAN
UT 19 VR 6991	ORANGE	SEDAN

Exit car(DL 19 VR 0492) again=>(₹20) !!



It will work in desired way if we close the running application and run the program again.

3. Problem: Enter after exiting a car more than once.

CarNumber	CarColor	CarType	CardType	CardNumber	ParkingTime	ParkingDate	SpotNum
MH 02 TY 2552	VIOLET	VAN	DEBIT	0788990207206214	08:27 AM	2023-03-28	PS003
ML 19 QN 8254	ORANGE	SEDAN	CREDIT	1562793018875103	08:29 AM	2023-03-28	PS004
PB 20 TY 8429	BROWN	BUS	DEBIT	5695675066264553	08:28 AM	2023-03-28	PS004
PB 45 VR 7233	ORANGE	SEDAN	CREDIT	7798433074102577	08:23 AM	2023-03-28	PS002
TG 02 QN 6266	BLACK	PICKUP-TRUCK	CREDIT	1751201277887245	08:27 AM	2023-03-28	PS005
TN 19 VR 2604	RED	VAN	DEBIT	7238176760458238	08:27 AM	2023-03-28	PS004
UT 19 VR 6991	ORANGE	SEDAN	CREDIT	9379131770920192	07:45 AM	2023-03-28	PS001
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

=> Exit car (say “xyz” which was in PS004) and Enter a car (It works), Again exit “xyz” and Enter a car (It works), again exit “xyz” and enter a car (It works).

To solve this problem, at database level: Add a UNIQUE Constraint to SpotNum attribute. We can prevent duplicate values from being inserted into the table, ensuring data integrity.

Solution 1:(Program Level)

Check if the carNumber is present in the table “ParkingLot.ParkingInfo” before exiting a car.

```python
def DeleteDataAndTicket(self):
    #____NEWELY ADDED STARTS_____________________________________________________
    mydb.reconnect()
    mycursor = mydb.cursor()
    sql = "SELECT * FROM ParkingLot.ParkingInfo WHERE CarNumber = %s"
    # car = self.cardNumber
    car = [self.carNumber]
    mycursor.execute(sql,car)
    rows = mycursor.fetchall()
    if len(rows) == 0:
        print("Car not found in database!!!")
        messagebox.showwarning(
            "Parking Warning", "🚨 Uh-oh! Your car seems to have gone on a joyride without you! 🚗😜")
        return
    #____NEWELY ADDED ENDS_______________________________________________________
    self.DeleteDataDB()
    self.TicketFrame()
```

Solution 2:(Database Level)

Use “Unique” Integrity Constraint when you are creating the DB. Additional functions need to be written(shown in next slide) to handle the error produced from the database connections if the constraint fails in the insert query.

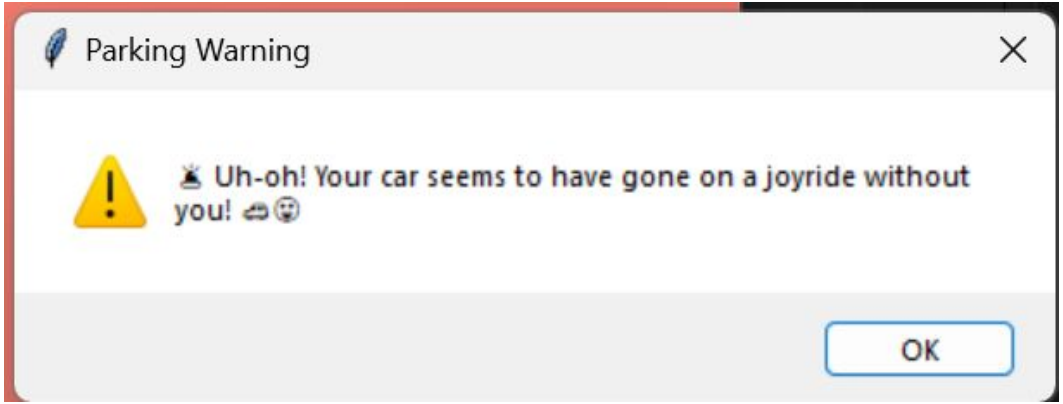
```sql
CREATE TABLE ParkingInfo (
  CarNumber varchar(250) Primary Key,
  CarColor VARCHAR(255),
  CarType VARCHAR(255) ,
  CardType varchar(200) ,
  CardNumber varchar(200) ,
  ParkingTime varchar(200),
  ParkingDate varchar(200),
  SpotNum varchar(200) unique
);
```

Both can also be used. (It will check for any error while insertion and also when we delete same record from the DB more than once)

Handling error when failing integrity constraint.

Using try and except block in python.

```python
def storeInfoInDB(self):
    myc = mydb.cursor()
    try:
        sql = """INSERT INTO ParkingLot.ParkingInfo
        (CarNumber, CarColor, CarType, CardType, CardNumber,
        ParkingTime, ParkingDate, SpotNum)
        VALUES(%s, %s, %s, %s, %s, %s, %s, %s)"""
        val = [str(self.carNumber),
               str(self.carColor), str(self.carType),
               str(self.__cardType), str(self.__cardNumber),
               str(self.timeNow), str(self.date),self.SpotNum]
        myc.execute(sql, val)
        sql = "UPDATE ParkingSpot SET Spot = 'Parked' WHERE SrNum = %s"
        val = [self.SpotNum]
        myc.execute(sql, val)
        mydb.commit()
        return True
    except IntegrityError as e:
        return False
```



Output of User Level Error Handling

Shown when carNumber is not found in DB.

LIVE DEMO

(If Time Permits)

All questions are answered...

1. Mysql.connector library: caching property (Connection Pool).
2. Variable Size Records: “Slotted Page Structure”. (Many more such structures)
3. Indexing: (CarNumber), Sparse Index, Dense Index.
4. Hashing: static hashing, extended hashing, bitmap index
5. 2-3-4 Trees, B-Tree, and B+ Trees: DS (STORE AND RETRIEVE RECORDS)
6. ACID Properties of Transaction: (Atomicity, Consistency, Isolation, and Durability)
7. Stable Storage (concept: durable and persistent data), Input/Output, Read/Write, Temporary Buffer, Private Work Area
8. log-based recovery and caching can improve performance but should be used with caution.

Conclusion + (Additional Information for large dataset)

Complete solution:

https://github.com/KarthikAvinash/ParkingManagementCaseStudySolution

THANK YOU

QUESTIONS?