

**Report on**

AI in Smart Parking

**Submitted by:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Name** | **Roll No.** | **Reg. No** | **Section** | **Group** | | Swaraj Deep Masana | 64 | 12113993 | K21QT | 2 | | Vamshi Anantha | 65 | 12115075 | K21QT | 2 | | Dinesh Raj Joshi | 66 | 12111660 | K21QT | 2 | |  |  |
|  |  |  |
|  |  |  |

**Submitted to: Akshara Rana**

**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.

# MAIN REPORT

## 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.
* 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.

## Scope:

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.

## SYSTEM REQUIREMENT PHASE

1. **Project Title :**

Parking Management System

## Technology:

* + **Back End: SQL server 2008**

**Microsoft SQL Server** is a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft). As a [database server](https://en.wikipedia.org/wiki/Database_server), it is a [software product](https://en.wikipedia.org/wiki/Software_product) with the primary function of storing and retrieving data as requested by other [software](https://en.wikipedia.org/wiki/Software_application) [applications](https://en.wikipedia.org/wiki/Software_application)—which may run either on the same computer or on another computer across a network (including the Internet).

Microsoft markets at least a dozen different editions of Microsoft SQL Server, aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many [concurrent users.](https://en.wikipedia.org/wiki/Concurrent_user)

## Modules a).Data Records

**User Records**: - This record helps for the authorization for using Vehicle Parking Management System. It Provides the Username and Password for the User (staff).It also includes the level of authority that means it separates the normal users and administrator.

**Vehicle Records**: - This most important record which focuses in our Vehicle Parking Management System. It stores the essential Vehicle records like:

-Vehicle Number

-Vehicle Type

-Vehicle Entry Time

-Vehicle Exit Time

## Reports

**Vehicle Parking Detail**: - This report is very essential in this system. This report provides a brief summary of vehicle activities. It shows the overall Entry and Exit time. It shows the User at time of Entry and Exit .It also provides the facility for examining the total vehicle details according to date wise.

# SYSTEM ANALYSIS PHASE

## Information gathering

Information gathering is done by interviewing the users and reviewing the existing documents. For the development of Parking management system a lot of research and important input from various website and application user was needed. Hence the following questionnaires were provided to them and hence te need for our website arises

* + **Interviewing the users:**
    - What are the difficulties you are facing in the existing system ?
    - What all new things you want to be included in the proposed system

?

* + - In what way you are storing your information ?
    - Who all are the users of the system ?

## User Requirement

* Need for an application that makes communicating easy and comfortable.
* An application that enables user to park a vehicle with safe and secure.
* Need for an application that is easy to use and widely available and hence a web application
* Handling all functions done with organization in a computerized manner.
* Allowing the user to park the vehicle directly.

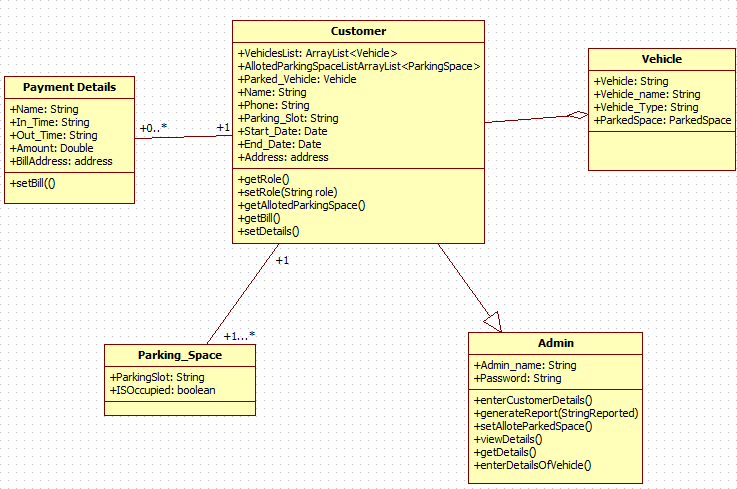
## Functional Requirement

* Admin need to enter all details for registration.
* Admin need to insert all details about customer and vehicle.
* Admin need to save all the details of customer and vehicle.
* Admin can retrieve the details of customer.
* Admin must generate a report for payment.

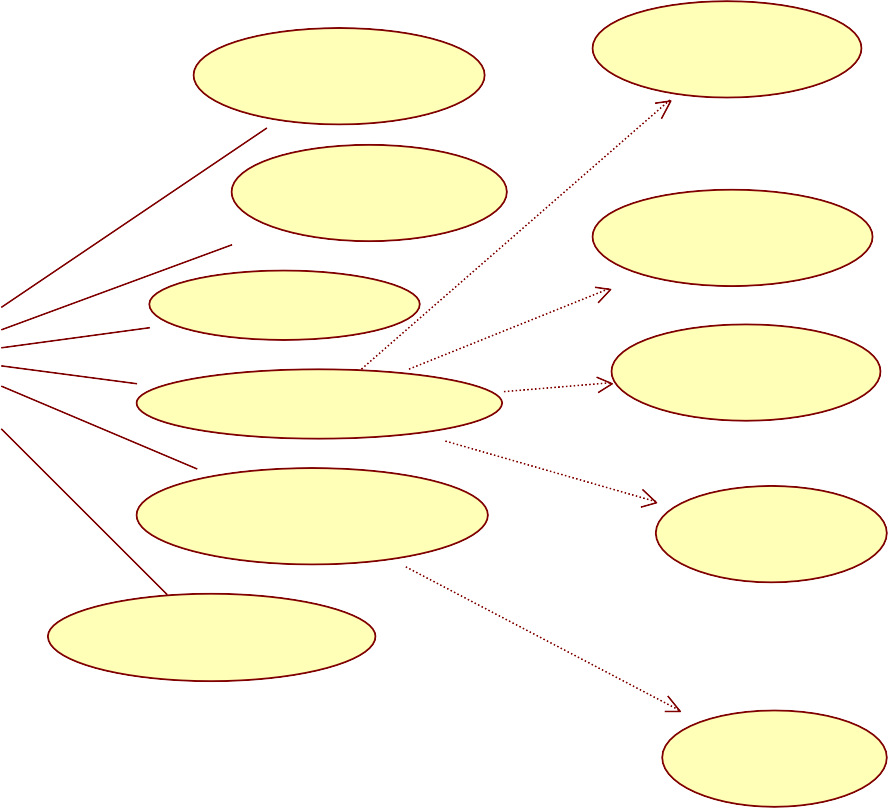
## Functional Requirement

* Admin need to enter all details for registration.
* Admin need to insert all details about customer and vehicle.
* Admin need to save all the details of customer and vehicle.
* Admin can retrieve the details of customer.
* Admin must generate a report for payment.

**Class diagram:-**



**Use case Diagram**



System

**Display parking space availability**

**Insert Vehicle\_no**

**System Configuration**

<<include>>

**Generate Report**

**Insert Vehicle\_name**

**Manage member**

<<include>>

<<include>>

**Insert owner\_name**

**Insert the details of vehicle**

<<include>>

**Transaction Details**

**Insert\_Intime**

<<include>>

**Print Transaction details**

**Insert\_Outtime**

**Pay's Money**

**Space Availability**

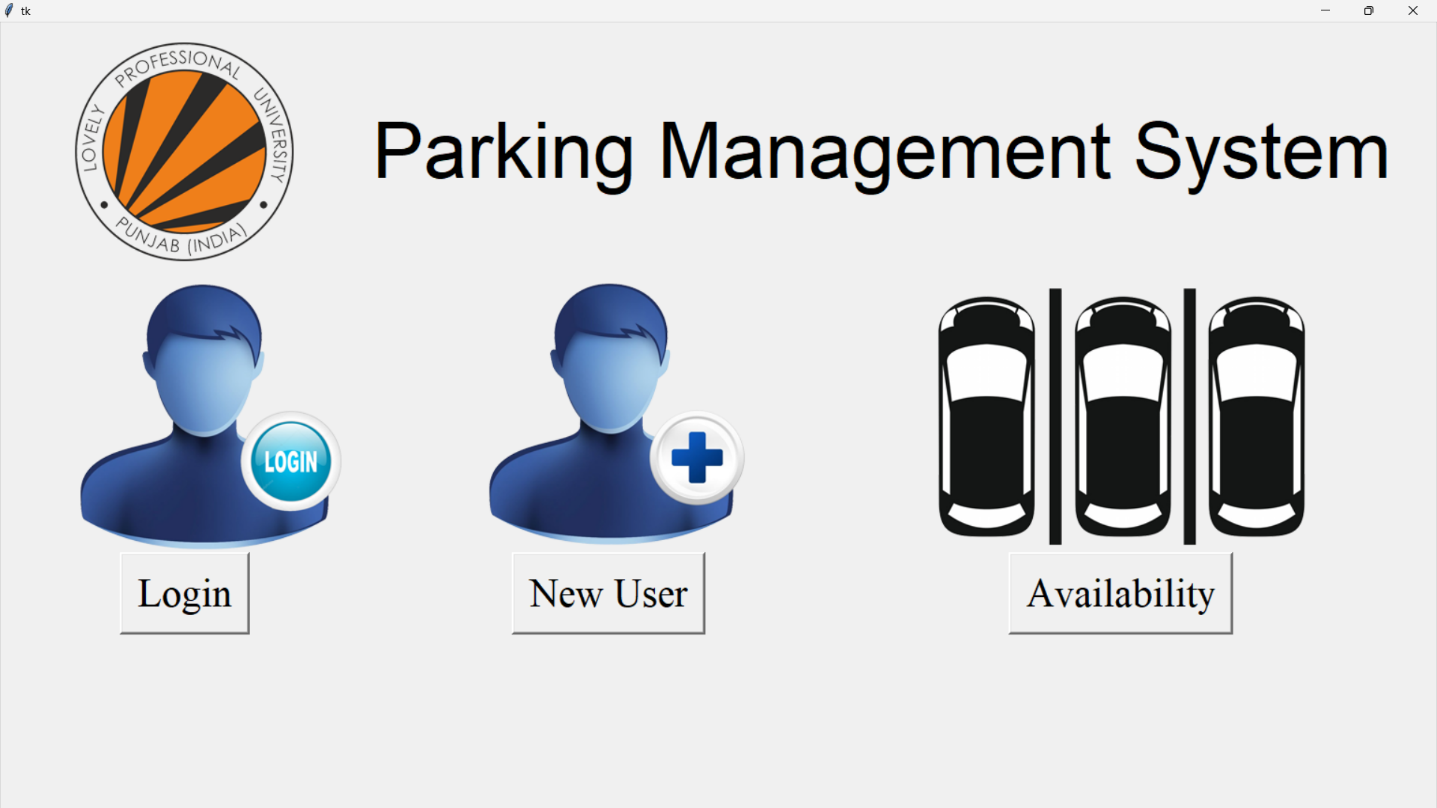
**Display**

**Admin**

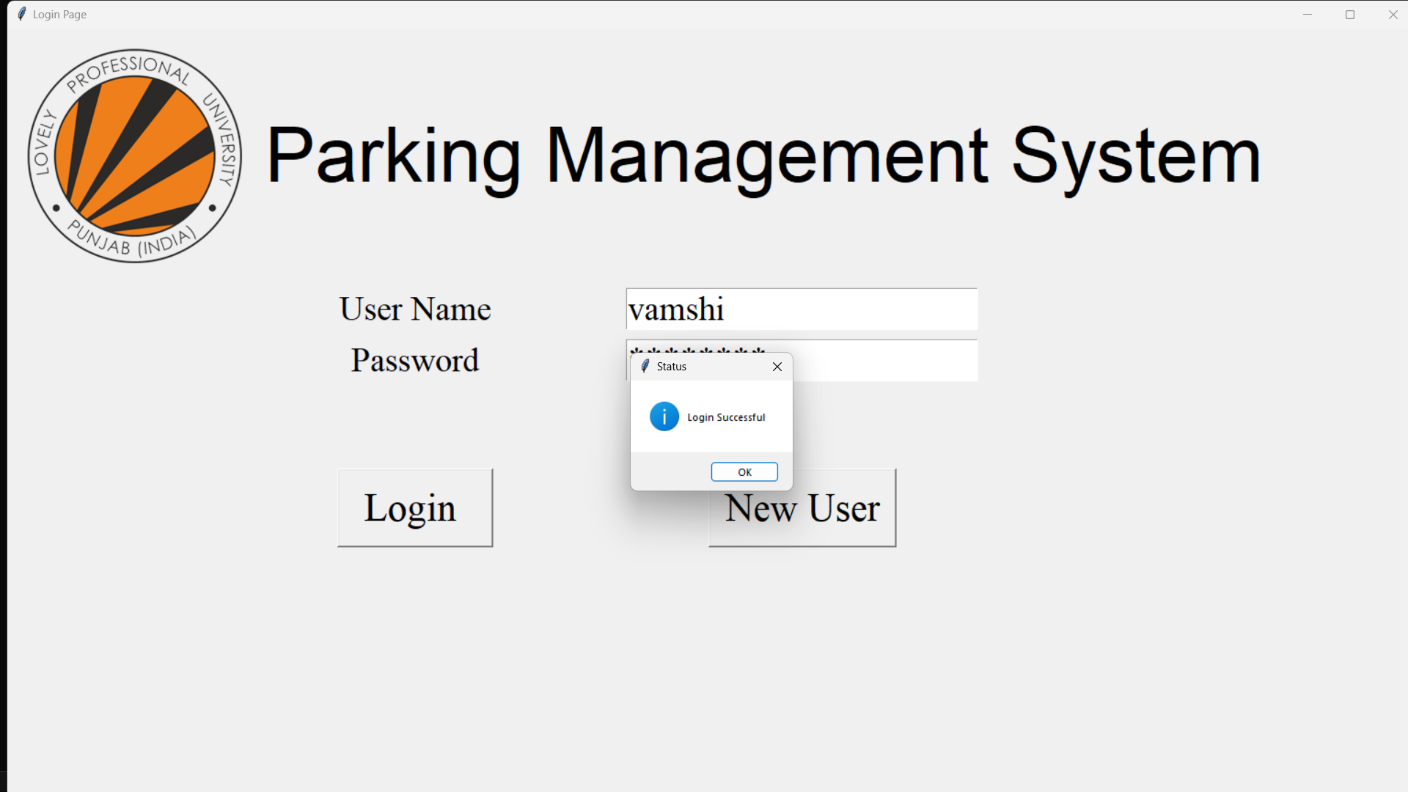
**Customer**

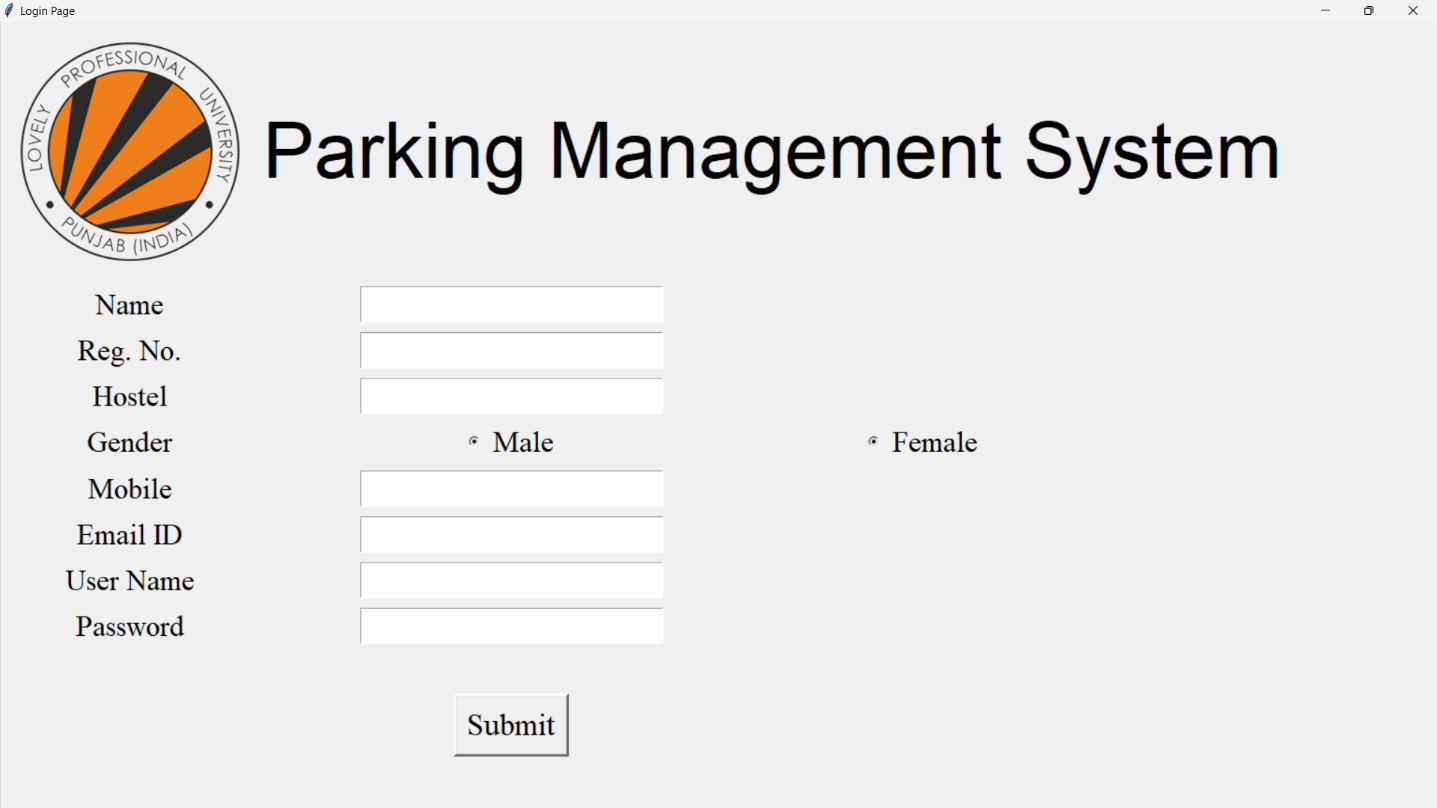
#### User Interface:-

1. Admin Login

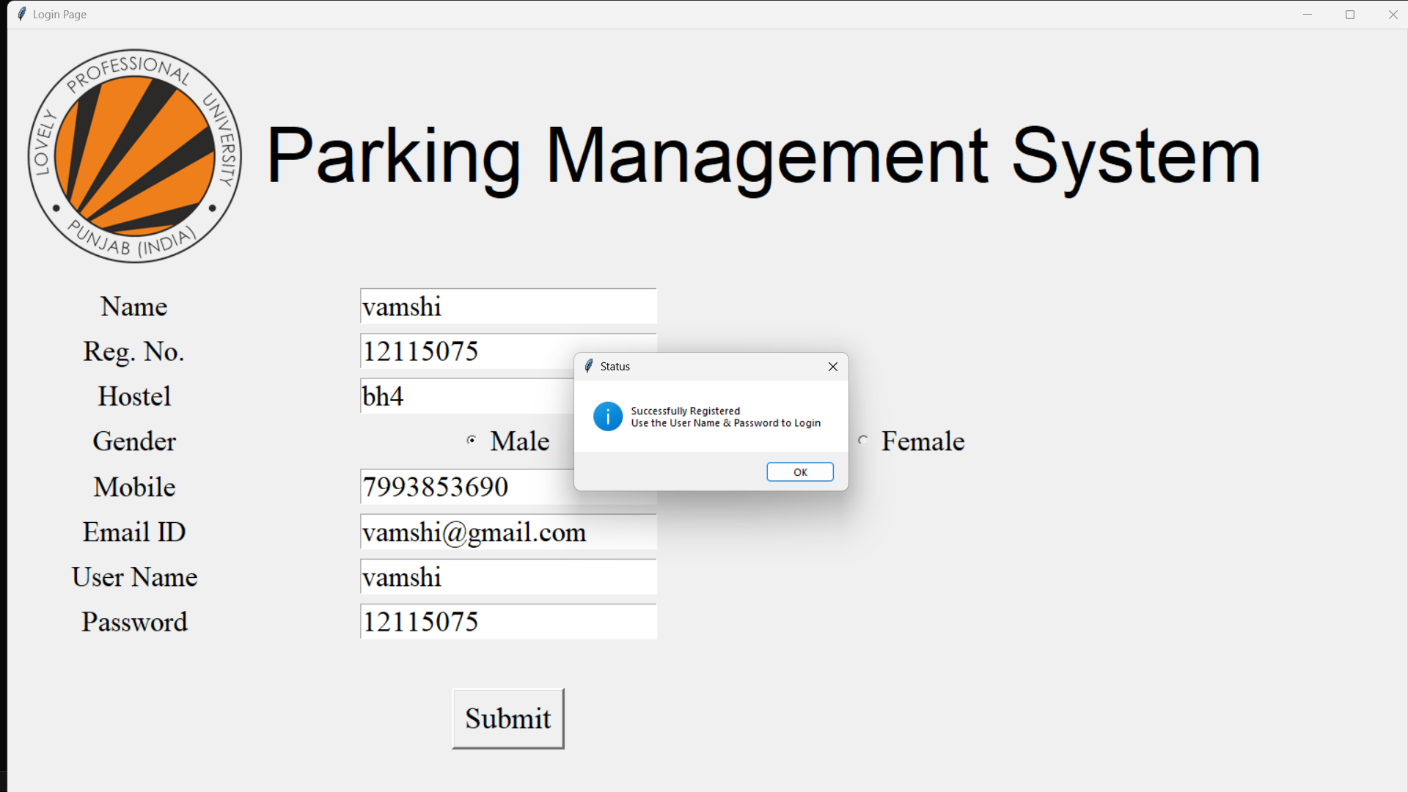


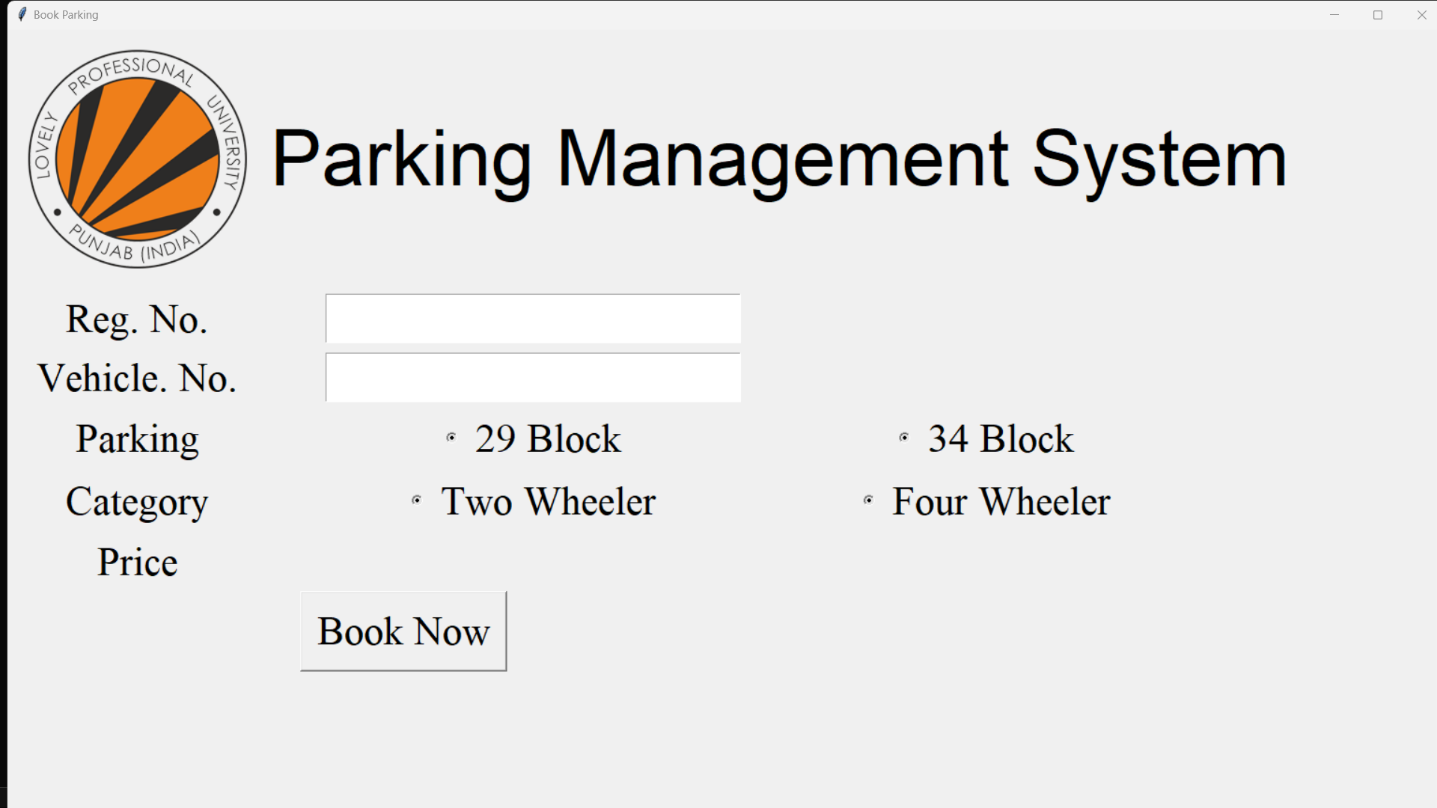
1. **login and password**



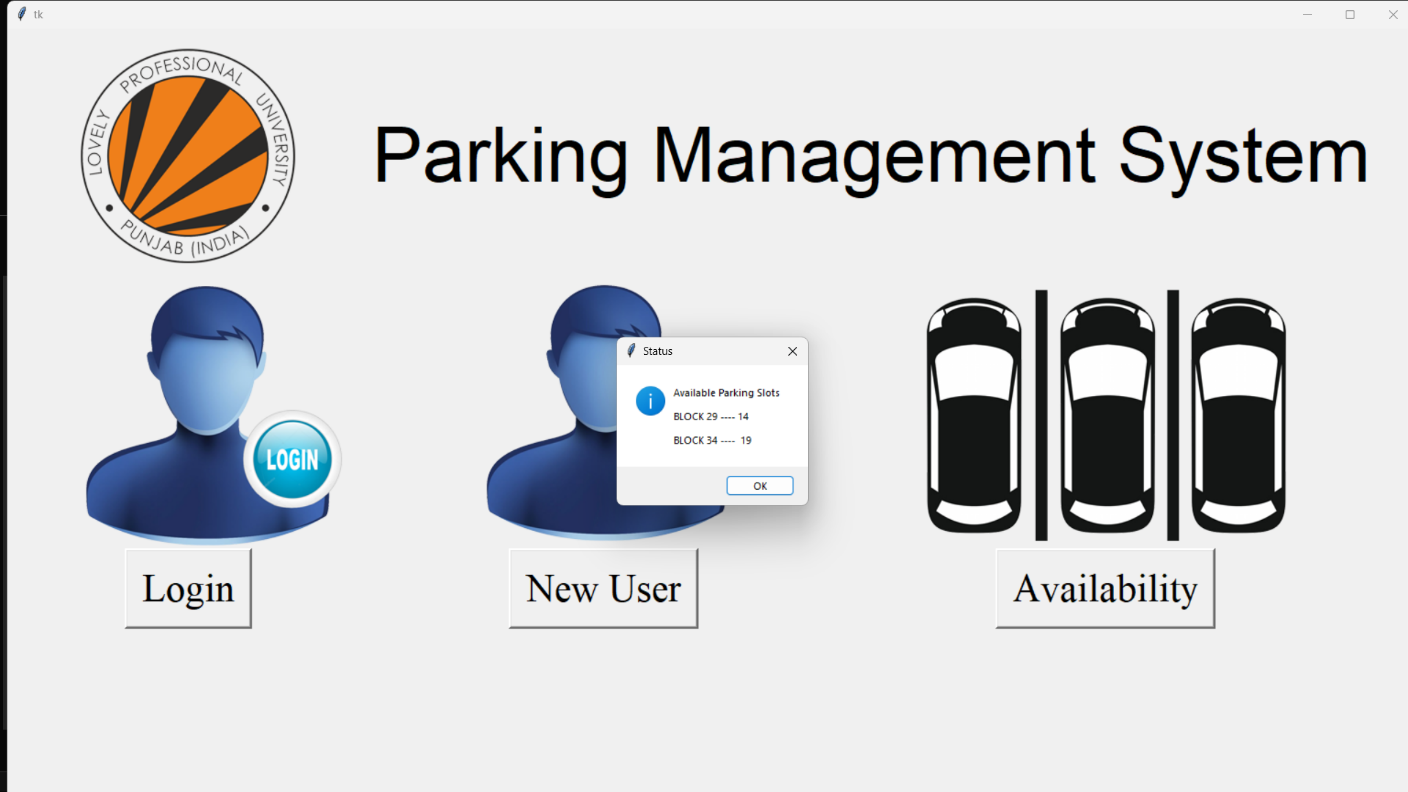
* 1. **Register page**

##### Registration successful prompt

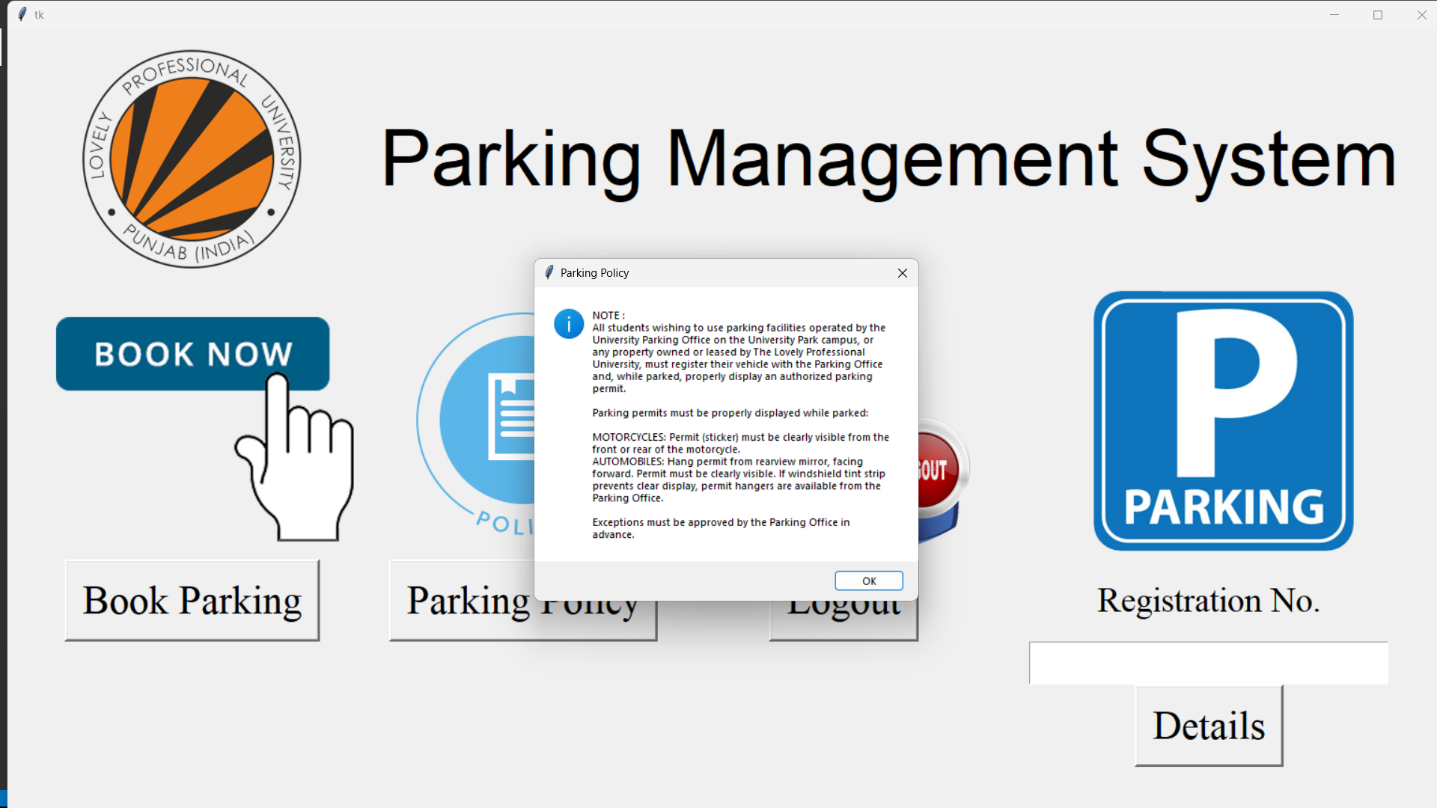


* 1. **Add and vehicle details**

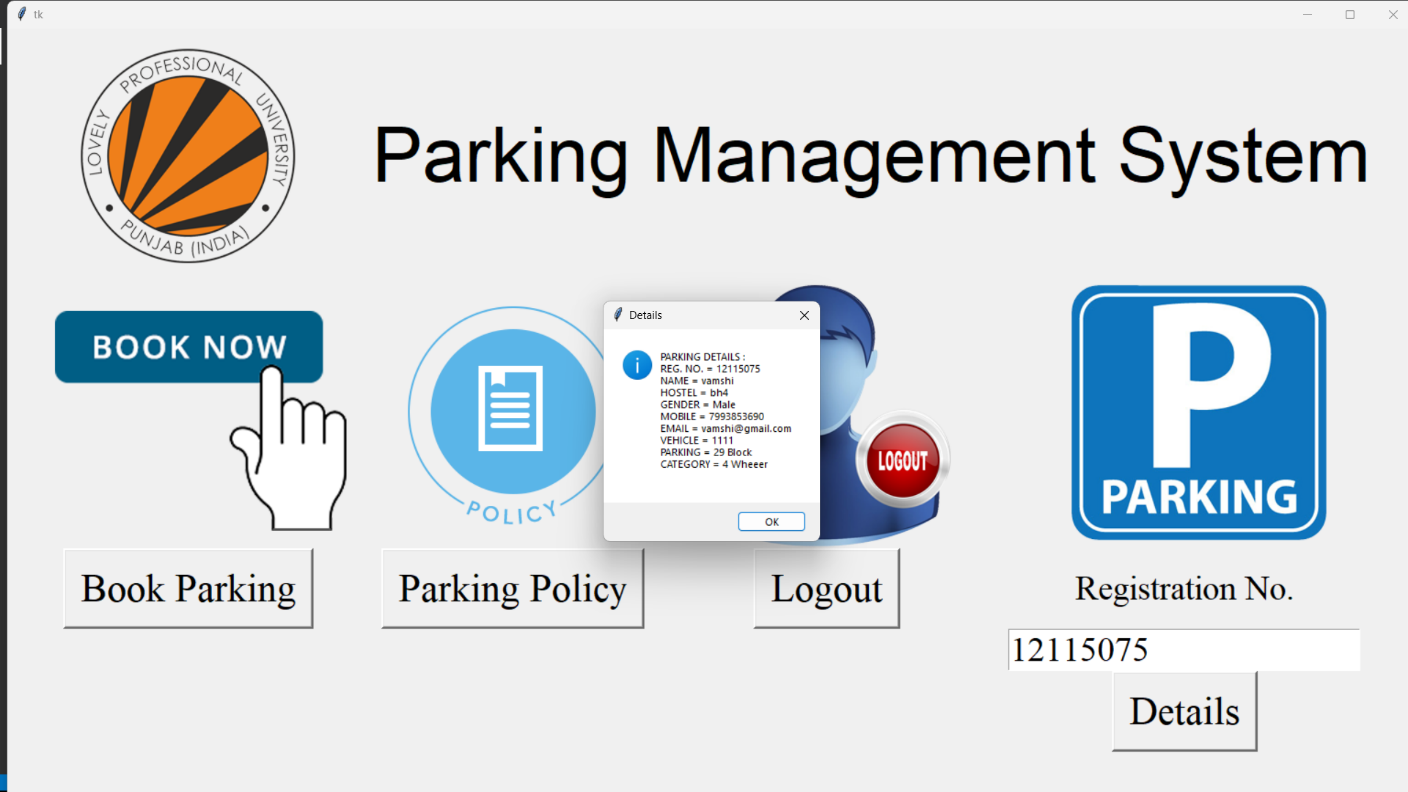
### avaliability:-



* 1. **policy**



### Detalis



**Conclusion**

#### Conclusion

This Project is minimizing the task of parking a vehicle by paying and saying some details about customer and vehicle to save data .In this the vehicle is parked as a safe and secure. This project is done as Efficient as possible

We also conclude that this project has helped us gain more knowledge about the topic that we are indulged ourselves into “ Visual Studio ”. We would be glad to enhance and promote this project if given chance and help ourselves and society in the near future

The developed application is tested with sample inputs and outputs obtained in according to the requirement. Even though we have tried our level best to make it a dream project. Due to time constraints we could not add more facilities to it.

The efficiency of the developed system can be enhanced with some minor modifications. Future development can be made in proposed system by integration more services like:

* It can be implemented through web pages.
* New effectives modules can be added time to time

**References**

#### Renferences

1.Huang, Y., & Zhai, X. (2019). A machine learning-based smart parking system for urban areas. IEEE Transactions on Intelligent Transportation Systems, 20(8), 2826-2835.

2.Zhang, Y., & Zheng, Y. (2019). Smart parking system: a survey. IEEE Transactions on Intelligent Transportation Systems, 20(11), 3924-3938.

3.Gao, Y., Li, L., Li, X., & Li, Y. (2018). An intelligent parking system based on computer vision and deep learning. IEEE Transactions on Intelligent Transportation Systems, 19(12), 4086-4097.

4.Wang, J., Zhang, J., & Lin, Y. (2019). A novel smart parking system based on blockchain and IoT. Future Generation Computer Systems, 101, 681-693.

**Code of whole program**

from tkinter import \*

from tkinter import messagebox

import sqlite3

import datetime

conn = sqlite3.connect('PMS.db')

conn.execute("""CREATE TABLE IF NOT EXISTS PARKING (REG INT PRIMARY KEY NOT NULL,NAME TEXT NOT NULL,HOSTEL CHAR(5) NOT NULL,SEX CHAR(8) NOT NULL,MOBILE CHAR(12) NOT NULL,EMAIL CHAR(50) NOT NULL,

USERNAME CHAR(50) NOT NULL,PASSWORD CHAR(50) NOT NULL,VEHICLE CHAR(15) NOT NULL,PARK CHAR(10) NOT NULL, TYPE CHAR(12) NOT NULL ,STATUS CHAR(5) NOT NULL,SLOT CHAR(5) NOT NULL,PB CHAR(5) NOT NULL,C CHAR(5),CIN CHAR(50),COUT CHAR(50))""")

conn.commit()

conn.close()

conn = sqlite3.connect('PPMS.db')

conn.execute("""CREATE TABLE IF NOT EXISTS PARKING (REG INT PRIMARY KEY NOT NULL,NAME TEXT NOT NULL,HOSTEL CHAR(5) NOT NULL,SEX CHAR(8) NOT NULL,MOBILE CHAR(12) NOT NULL,EMAIL CHAR(50) NOT NULL,

USERNAME CHAR(50) NOT NULL,PASSWORD CHAR(50) NOT NULL,VEHICLE CHAR(15) NOT NULL,PARK CHAR(10) NOT NULL, TYPE CHAR(12) NOT NULL ,STATUS CHAR(5) NOT NULL,SLOT CHAR(5) NOT NULL,PB CHAR(5) NOT NULL,C CHAR(5),CIN CHAR(50),COUT CHAR(50))""")

conn.commit()

conn.close()

def Show():

conn = sqlite3.connect('PMS.db')

cursor = conn.execute("SELECT \* FROM PARKING")

for row in cursor:

print(row)

conn.commit()

conn.close()

def Base\_Page():

global root

root=Tk()

w,h=root.winfo\_screenwidth(),root.winfo\_screenheight()

root.geometry("%dx%d+0+0" % (w,h))

icon=PhotoImage(file="LPU-Logo.png")

logo=Label(root,image=icon)

PMS=Label(root,text="Parking Management System",font=("Helvetica", 62))

logo.grid(row=1,column=0)

PMS.grid(row=1,column=1,columnspan=2)

icon1=PhotoImage(file="Login.png")

Log=Label(root,image=icon1)

Login\_Button=Button(root,text="Login",font=("Times New Roman",32),bd=3,command=Login\_Page)

icon2=PhotoImage(file="NewUser.png")

NUsr=Label(root,image=icon2)

NewUser\_Button=Button(root,text="New User",font=("Times New Roman",32),bd=3,command=BptoNu)

icon3=PhotoImage(file="Availability.png")

Avai=Label(root,image=icon3)

Availability\_Button=Button(root,text="Availability",font=("Times New Roman",32),bd=3,command=Availability)

Log.grid(row=2)

Login\_Button.grid(row=3,column=0)

NUsr.grid(row=2,column=1)

NewUser\_Button.grid(row=3,column=1)

Avai.grid(row=2,column=2)

Availability\_Button.grid(row=3,column=2)

root.mainloop()

def Login\_Page():

root.destroy()

global Login\_GUI

Login\_GUI=Tk()

Login\_GUI.title("Login Page")

w,h=Login\_GUI.winfo\_screenwidth(),Login\_GUI.winfo\_screenheight()

Login\_GUI.geometry("%dx%d+0+0" % (w,h))

icon=PhotoImage(file="LPU-Logo.png")

logo=Label(Login\_GUI,image=icon)

PMS=Label(Login\_GUI,text="Parking Management System",font=("Helvetica", 62))

logo.grid(row=1,column=0)

PMS.grid(row=1,column=1,columnspan=4)

lab1=Label(Login\_GUI,text="User Name",padx=20,pady=5,font=("Times New Roman",28))

lab2=Label(Login\_GUI,text="Password",padx=20,pady=5,font=("Times New Roman",28))

lab3=Label(Login\_GUI,text="\n",font=("Times New Roman",28))

global name

global pswd

name=Entry(Login\_GUI,font=("Times New Roman",28))

pswd=Entry(Login\_GUI,show="\*",font=("Times New Roman",28))

Login\_Button=Button(Login\_GUI,text=" Login ",command=validate,font=("Times New Roman",32))

NewUser\_Button=Button(Login\_GUI,text="New User",command=LptoNu,font=("Times New Roman",32))

lab1.grid(row=2,column=1)

lab2.grid(row=3,column=1)

name.grid(row=2,column=2)

pswd.grid(row=3,column=2)

lab3.grid(row=4)

Login\_Button.grid(row=5,column=1)

NewUser\_Button.grid(row=5,column=2)

Login\_GUI.mainloop()

def LptoNu():

Login\_GUI.destroy()

NewUser\_Page()

def BptoNu():

root.destroy()

NewUser\_Page()

def validate():

c=0

conn = sqlite3.connect('PMS.db')

cursor = conn.execute("SELECT USERNAME,PASSWORD FROM PARKING")

if((str(name.get())=="Admin" ) and (str(pswd.get())=="Admin")): # If Username and Password is of Admin, Login as Admin

c=2

for row in cursor:

if( (str(name.get())==row[0]) and (str(pswd.get())==row[1])): # If Username & Password matches, Break & Proceed

c=1

break

else: # If doesn't match, Warning

c=0

if(c==1):

messagebox.showinfo("Status", "Login Successful")

User\_Page()

if(c==0):

messagebox.showinfo("Status", "Login Failed.\nPlease Try Again")

if(c==2):

messagebox.showinfo("Status", "Admin Login")

LptoAdm()

conn.commit()

def NewUser\_Page():

global NewUser\_GUI

NewUser\_GUI=Tk()

NewUser\_GUI.title("Login Page")

w,h=NewUser\_GUI.winfo\_screenwidth(),NewUser\_GUI.winfo\_screenheight()

NewUser\_GUI.geometry("%dx%d+0+0" % (w,h))

icon=PhotoImage(file="LPU-Logo.png")

logo=Label(NewUser\_GUI,image=icon)

PMS=Label(NewUser\_GUI,text="Parking Management System",font=("Helvetica", 62))

logo.grid(row=1,column=0)

PMS.grid(row=1,column=1,columnspan=3)

lab1=Label(NewUser\_GUI,text="Name",padx=20,pady=5,font=("Times New Roman",23))

lab2=Label(NewUser\_GUI,text="Reg. No.",padx=20,pady=5,font=("Times New Roman",23))

lab3=Label(NewUser\_GUI,text="Hostel",padx=20,pady=5,font=("Times New Roman",23))

lab4=Label(NewUser\_GUI,text="Gender",padx=20,pady=5,font=("Times New Roman",23))

lab5=Label(NewUser\_GUI,text="Mobile",padx=20,pady=5,font=("Times New Roman",23))

lab6=Label(NewUser\_GUI,text="Email ID",padx=20,pady=5,font=("Times New Roman",23))

lab7=Label(NewUser\_GUI,text="User Name",padx=20,pady=5,font=("Times New Roman",23))

lab8=Label(NewUser\_GUI,text="Password",padx=20,pady=5,font=("Times New Roman",23))

global Name

global Register

global Hostel

global Mobile

global Email

global UserName

global Pass

Name=Entry(NewUser\_GUI,font=("Times New Roman",23))

Register=Entry(NewUser\_GUI,font=("Times New Roman",23))

Hostel=Entry(NewUser\_GUI,font=("Times New Roman",23))

global Gender

Gender = StringVar()

G1 = Radiobutton(NewUser\_GUI, text="Male", variable=Gender, value="Male",font=("Times New Roman",23))

G2 = Radiobutton(NewUser\_GUI, text="Female", variable=Gender, value="Female",font=("Times New Roman",23))

Mobile=Entry(NewUser\_GUI,font=("Times New Roman",23))

Email=Entry(NewUser\_GUI,font=("Times New Roman",23))

UserName=Entry(NewUser\_GUI,font=("Times New Roman",23))

Pass=Entry(NewUser\_GUI,font=("Times New Roman",23))

lab1.grid(row=2,column=0)

lab2.grid(row=3,column=0)

lab3.grid(row=4,column=0)

lab4.grid(row=5,column=0)

lab5.grid(row=6,column=0)

lab6.grid(row=7,column=0)

lab7.grid(row=8,column=0)

lab8.grid(row=9,column=0)

Name.grid(row=2,column=1)

Register.grid(row=3,column=1)

Hostel.grid(row=4,column=1)

G1.grid(row=5,column=1)

G2.grid(row=5,column=2)

Mobile.grid(row=6,column=1)

Email.grid(row=7,column=1)

UserName.grid(row=8,column=1)

Pass.grid(row=9,column=1)

lab10=Label(NewUser\_GUI,text="",font=("Times New Roman",28))

lab10.grid(row=10)

Submit\_Button=Button(NewUser\_GUI,text="Submit",command=DB\_Reg,font=("Times New Roman",25),bd=3)

Submit\_Button.grid(row=11,column=1)

NewUser\_GUI.mainloop()

def DB\_Reg():

dbReg=int(Register.get())

dbName=str(Name.get())

dbHostel=str(Hostel.get())

dbMobile=str(Mobile.get())

dbGender=str(Gender.get())

dbEmail=str(Email.get())

dbUserName=str(UserName.get())

dbPass=str(Pass.get())

dbVehicle=str("NIL")

dbPark=str("NIL")

dbType=str("NIL")

dbStatus=str("Not Parked")

dbSlot=str("NIL")

dbPb=str("NIL")

if(str(Register.get())=="" or str(Name.get())=="" or str(Hostel.get())=="" or str(Mobile.get())=="" or str(Gender.get())=="" or str(Email.get())=="" or str(UserName.get())=="" or str(Pass.get())==""):

messagebox.showinfo("Warning", "Fields can not be Empty !!!")

else:

conn = sqlite3.connect('PMS.db')

cott = sqlite3.connect('PPMS.db')

cursor=conn.execute("INSERT INTO PARKING (REG,NAME,HOSTEL,SEX,MOBILE,EMAIL,USERNAME,PASSWORD,VEHICLE,PARK,TYPE,STATUS,SLOT,PB) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?)",(dbReg,dbName,dbHostel,dbMobile,dbGender,dbEmail,dbUserName,dbPass,dbVehicle,dbPark,dbType,dbStatus,dbSlot,dbPb))

conn.commit()

cursor=cott.execute("INSERT INTO PARKING (REG,NAME,HOSTEL,SEX,MOBILE,EMAIL,USERNAME,PASSWORD,VEHICLE,PARK,TYPE,STATUS,SLOT,PB) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?)",(dbReg,dbName,dbHostel,dbMobile,dbGender,dbEmail,dbUserName,dbPass,dbVehicle,dbPark,dbType,dbStatus,dbSlot,dbPb))

cott.commit()

messagebox.showinfo("Status", "Successfully Registered\nUse the User Name & Password to Login")

NewUser\_GUI.destroy()

Base\_Page()

def User\_Page():

Login\_GUI.destroy()

global User\_GUI

User\_GUI=Tk()

w,h=User\_GUI.winfo\_screenwidth(),User\_GUI.winfo\_screenheight()

User\_GUI.geometry("%dx%d+0+0" % (w,h))

icon=PhotoImage(file="LPU-Logo.png")

logo=Label(User\_GUI,image=icon)

PMS=Label(User\_GUI,text="Parking Management System",font=("Helvetica", 62))

logo.grid(row=1,column=0)

PMS.grid(row=1,column=1,columnspan=3)

icon1=PhotoImage(file="Book.png")

Book=Label(User\_GUI,image=icon1)

BookParking\_Button=Button(User\_GUI,text="Book Parking",font=("Times New Roman",32),bd=3,command=Book\_Page)

icon2=PhotoImage(file="Policy.png")

Policy=Label(User\_GUI,image=icon2)

Policy\_Button=Button(User\_GUI,text="Parking Policy",font=("Times New Roman",32),bd=3,command=ParkingPolicy)

icon3=PhotoImage(file="Logout.png")

Logout=Label(User\_GUI,image=icon3)

Logout\_Button=Button(User\_GUI,text="Logout",font=("Times New Roman",32),bd=3,command=UptoBp)

icon4=PhotoImage(file="Parking.png")

Y=Label(User\_GUI,text="Registration No.",font=("Times New Roman",28),padx=20,pady=5)

global Z

Z=Entry(User\_GUI,font=("Times New Roman",28))

Details=Label(User\_GUI,image=icon4)

Details\_Button=Button(User\_GUI,text="Details",font=("Times New Roman",32),bd=3,command=De)

Book.grid(row=2)

BookParking\_Button.grid(row=3,column=0)

Policy.grid(row=2,column=1)

Policy\_Button.grid(row=3,column=1)

Logout.grid(row=2,column=2)

Logout\_Button.grid(row=3,column=2)

Details.grid(row=2,column=3)

Y.grid(row=3,column=3)

Z.grid(row=4,column=3)

Details\_Button.grid(row=5,column=3)

User\_GUI.mainloop()

User\_GUI.mainloop()

def De():

zz=int(Z.get())

conn = sqlite3.connect('PMS.db')

cursor = conn.execute("SELECT REG,NAME,HOSTEL,SEX,MOBILE,EMAIL,VEHICLE,PARK,TYPE,STATUS,SLOT,PB FROM PARKING")

Flag=0

for row in cursor:

if(row[0]==zz and row[11]=="P"): # if reg exist & parking is booked

S="PARKING DETAILS : \nREG. NO. = "+str(row[0])+"\nNAME = "+row[1]+"\nHOSTEL = "+row[2]+"\nGENDER = "+row[4]+"\nMOBILE = "+row[3]+"\nEMAIL = "+row[5]+"\nVEHICLE = "+row[6]+"\nPARKING = "+row[7]+"\nCATEGORY = "+row[8]+"\nSTATUS = "+row[9]+"\nSLOT = "+row[10]+"\n\n"

messagebox.showinfo("Details",S)

Flag=1

if(Flag==0): # if reg does'nt exist or parking is not booked

S="NOTE : Invalid Registration Number/Parking Not Booked"

messagebox.showinfo("Note",S)

conn.commit()

def ParkingPolicy():

messagebox.showinfo("Parking Policy", "NOTE : \nAll students wishing to use parking facilities operated by the University Parking Office on the University Park campus, or any property owned or leased by The Lovely Professional University, must register their vehicle with the Parking Office and, while parked, properly display an authorized parking permit.\n\nParking permits must be properly displayed while parked:\n\nMOTORCYCLES: Permit (sticker) must be clearly visible from the front or rear of the motorcycle.\nAUTOMOBILES: Hang permit from rearview mirror, facing forward. Permit must be clearly visible. If windshield tint strip prevents clear display, permit hangers are available from the Parking Office.\n\nExceptions must be approved by the Parking Office in advance.")

def UptoBp():

messagebox.showinfo("Status","Successfully Logged Off")

User\_GUI.destroy()

Base\_Page()

def Book\_Page():

User\_GUI.destroy()

global Book\_GUI

Book\_GUI=Tk()

Book\_GUI.title("Book Parking")

w, h = Book\_GUI.winfo\_screenwidth(),Book\_GUI.winfo\_screenheight()

Book\_GUI.geometry("%dx%d+0+0" % (w, h))

icon=PhotoImage(file="LPU-Logo.png")

logo=Label(Book\_GUI,image=icon)

PMS=Label(Book\_GUI,text="Parking Management System",font=("Helvetica", 62))

logo.grid(row=1,column=0)

PMS.grid(row=1,column=1,columnspan=3)

lab1=Label(Book\_GUI,text="Reg. No.",padx=20,pady=5,font=("Times New Roman",32))

lab2=Label(Book\_GUI,text="Vehicle. No.",padx=20,pady=5,font=("Times New Roman",32))

lab3=Label(Book\_GUI,text="Parking",padx=20,pady=5,font=("Times New Roman",32))

lab4=Label(Book\_GUI,text="Category",padx=20,pady=5,font=("Times New Roman",32))

lab5=Label(Book\_GUI,text="Price",padx=20,pady=5,font=("Times New Roman",32))

global Reg

global Veh

Reg=Entry(Book\_GUI,font=("Times New Roman",32))

Veh=Entry(Book\_GUI,font=("Times New Roman",32))

global var

var = StringVar()

R1 = Radiobutton(Book\_GUI, text="29 Block", variable=var, value="29 Block",font=("Times New Roman",32))

R2 = Radiobutton(Book\_GUI, text="34 Block", variable=var, value="34 Block",font=("Times New Roman",32))

global VehTyp

VehTyp = StringVar()

R3 = Radiobutton(Book\_GUI, text="Two Wheeler", variable=VehTyp, value="2 Wheeler",command=CalPrice2,font=("Times New Roman",32))

R4 = Radiobutton(Book\_GUI, text="Four Wheeler", variable=VehTyp, value="4 Wheeer",command=CalPrice4,font=("Times New Roman",32))

Book\_Button=Button(Book\_GUI,text="Book Now",command=Park,font=("Times New Roman",32))

lab1.grid(row=2,column=0)

Reg.grid(row=2,column=1)

lab2.grid(row=3,column=0)

Veh.grid(row=3,column=1)

lab3.grid(row=4,column=0)

R1.grid(row=4,column=1)

R2.grid(row=4,column=2)

lab4.grid(row=5,column=0)

R3.grid(row=5,column=1)

R4.grid(row=5,column=2)

lab5.grid(row=6,column=0)

global amt

amt = Label(Book\_GUI,font=("Times New Roman",32))

amt.grid(row=6,column=1)

Book\_Button.grid(row=7,columnspan=2)

Book\_GUI.mainloop()

def CalPrice2():

amt.config(text="1000")

def CalPrice4():

amt.config(text="1500")

def Park():

dbR=str(Reg.get())

dbV=str(Veh.get())

dbA=str(var.get())

dbT=str(VehTyp.get())

dbS=str("Not Parked")

dbP=str("P")

if(str(Reg.get())=="" or str(Veh.get()) =="" or str(var.get())=="" or str(VehTyp.get())==""):

messagebox.showinfo("Warning","Fields Cannot Be Empty !!!")

else:

conn = sqlite3.connect('PMS.db')

cott = sqlite3.connect('PPMS.db')

conn.execute("UPDATE PARKING SET VEHICLE=?,PARK=?,TYPE=?,STATUS=?,PB=? WHERE REG=?", (dbV,dbA,dbT,dbS,dbP,dbR))

cott.execute("UPDATE PARKING SET VEHICLE=?,PARK=?,TYPE=?,STATUS=?,PB=? WHERE REG=?", (dbV,dbA,dbT,dbS,dbP,dbR))

conn.commit()

conn.close()

cott.commit()

cott.close()

messagebox.showinfo("Status","Successfully Booked Parking")

Book\_GUI.destroy()

Base\_Page()

def Availability():

Tot34=20

Tot29=15

C34=0

C29=0

conn = sqlite3.connect('PMS.db')

cursor = conn.execute("SELECT PARK FROM PARKING")

for row in cursor:

if(row[0]=="34 Block"):

C34=C34+1

if(row[0]=="29 Block"):

C29=C29+1

conn.commit()

Av34=str(Tot34-C34)

Av29=str(Tot29-C29)

messagebox.showinfo("Status","Available Parking Slots\n\nBLOCK 29 ---- "+Av29+"\n\nBLOCK 34 ---- "+Av34)

def LptoAdm():

Login\_GUI.destroy()

SuperUser\_Page()

def SuperUser\_Page():

global SuperUser\_GUI

SuperUser\_GUI=Tk()

w,h=SuperUser\_GUI.winfo\_screenwidth(),SuperUser\_GUI.winfo\_screenheight()

SuperUser\_GUI.geometry("%dx%d+0+0" % (w,h))

# Empty Laber at Column 1

Emp=Label(SuperUser\_GUI,text=" ",font=("Times New Roman",17),padx=17,pady=5)

Emp.grid(row=1,column=0)

# LPU Logo & Heading

icon=PhotoImage(file="LPU-LogoDetails.png")

logo=Label(SuperUser\_GUI,image=icon)

PMS=Label(SuperUser\_GUI,text="Parking Management System",font=("Helvetica", 48))

logo.grid(row=1,column=1)

PMS.grid(row=1,column=2,columnspan=4)

# Parking Details

icon1=PhotoImage(file="ParkingDetails.png")

Details=Label(SuperUser\_GUI,image=icon1)

Y=Label(SuperUser\_GUI,text="Registration No.",font=("Times New Roman",17),padx=17,pady=3)

global Z

Z=Entry(SuperUser\_GUI,font=("Times New Roman",17))

Y1=Label(SuperUser\_GUI,text="Vehicle No.",font=("Times New Roman",17),padx=17,pady=3)

global Z1

Z1=Entry(SuperUser\_GUI,font=("Times New Roman",17))

Details\_Button=Button(SuperUser\_GUI,text="Fetch Details",font=("Times New Roman",15),bd=3,width=12,command=FetchDetails)

Details.grid(row=2,column=1)

Y.grid(row=3,column=1)

Z.grid(row=4,column=1)

Y1.grid(row=5,column=1)

Z1.grid(row=6,column=1)

Details\_Button.grid(row=7,column=1,pady=4,rowspan=2)

# CheckIN

icon2=PhotoImage(file="CheckIN.png")

ChIN=Label(SuperUser\_GUI,image=icon2)

A=Label(SuperUser\_GUI,text="Registration No.",font=("Times New Roman",17),padx=17,pady=3)

global B

B=Entry(SuperUser\_GUI,font=("Times New Roman",17))

A1=Label(SuperUser\_GUI,text="Vehicle No.",font=("Times New Roman",17),padx=17,pady=3)

global B1

B1=Entry(SuperUser\_GUI,font=("Times New Roman",17))

A2=Label(SuperUser\_GUI,text="Slot Number",font=("Times New Roman",17),padx=17,pady=3)

global B2

B2=Entry(SuperUser\_GUI,font=("Times New Roman",17))

R=Label(SuperUser\_GUI,text="Block",font=("Times New Roman",25),padx=17,pady=3)

global pd

pd = StringVar()

R1 = Radiobutton(SuperUser\_GUI, text="29 Block", variable=pd, value="29 Block",font=("Times New Roman",17))

R2 = Radiobutton(SuperUser\_GUI, text="34 Block", variable=pd, value="34 Block",font=("Times New Roman",17))

CheckIN\_Button=Button(SuperUser\_GUI,text="Check IN",font=("Times New Roman",17),bd=3,width=12,command=CheckIN)

ChIN.grid(row=2,column=2,columnspan=2)

A.grid(row=3,column=2,columnspan=2)

B.grid(row=4,column=2,columnspan=2)

A1.grid(row=5,column=2,columnspan=2)

B1.grid(row=6,column=2,columnspan=2)

A2.grid(row=7,column=2,columnspan=2)

B2.grid(row=8,column=2,columnspan=2)

R.grid(row=9,column=2,columnspan=2)

R1.grid(row=10,column=2)

R2.grid(row=10,column=3)

CheckIN\_Button.grid(row=11,column=2,columnspan=2)

# CheckOUT

icon3=PhotoImage(file="CheckOUT.png")

ChOUT=Label(SuperUser\_GUI,image=icon3)

C=Label(SuperUser\_GUI,text="Registration No.",font=("Times New Roman",17),padx=17,pady=5)

global D

D=Entry(SuperUser\_GUI,font=("Times New Roman",17))

C1=Label(SuperUser\_GUI,text="Vehicle No.",font=("Times New Roman",17),padx=17,pady=5)

global D1

D1=Entry(SuperUser\_GUI,font=("Times New Roman",17))

C2=Label(SuperUser\_GUI,text="Slot Number",font=("Times New Roman",17),padx=17,pady=5)

global D2

D2=Entry(SuperUser\_GUI,font=("Times New Roman",17))

CheckOUT\_Button=Button(SuperUser\_GUI,text="Check OUT",font=("Times New Roman",17),bd=3,width=12,command=CheckOUT)

ChOUT.grid(row=2,column=4)

C.grid(row=3,column=4)

D.grid(row=4,column=4)

C1.grid(row=5,column=4)

D1.grid(row=6,column=4)

C2.grid(row=7,column=4)

D2.grid(row=8,column=4)

CheckOUT\_Button.grid(row=10,column=4,rowspan=1)

# Other Features

icon4=PhotoImage(file="Features.png")

Feat=Label(SuperUser\_GUI,image=icon4)

OverView\_Button=Button(SuperUser\_GUI,text="Overview",font=("Times New Roman",17),bd=3,width=12,command=OverView)

History\_Button=Button(SuperUser\_GUI,text="History",font=("Times New Roman",17),bd=3,width=12,command=History)

Slots\_Button=Button(SuperUser\_GUI,text="Available Slots",font=("Times New Roman",17),bd=3,width=12,command=AvailSlots)

Logout\_Button=Button(SuperUser\_GUI,text="Logout",font=("Times New Roman",17),bd=3,width=12,command=SptoBp)

Feat.grid(row=2,column=5)

OverView\_Button.grid(row=4,column=5,pady=4)

History\_Button.grid(row=6,column=5,pady=4)

Slots\_Button.grid(row=8,column=5,pady=4)

Logout\_Button.grid(row=10,column=5)

SuperUser\_GUI.mainloop()

def SptoBp():

messagebox.showinfo("Status","Successfully Logged Off")

SuperUser\_GUI.destroy()

Base\_Page()

def FetchDetails():

if(str(Z.get())=="" and str(Z1.get())=="" ):

S="Please Fill - Registration/Vehicle"

messagebox.showinfo("Warning",S)

return

if(str(Z.get())==""):

zz=0

else:

zz=int(Z.get())

v=str(Z1.get())

conn = sqlite3.connect('PMS.db')

cursor = conn.execute("SELECT REG,NAME,HOSTEL,SEX,MOBILE,EMAIL,VEHICLE,PARK,TYPE,STATUS,SLOT FROM PARKING")

Flag=0

for row in cursor:

if(row[0]==zz or row[6]==v):

S="PARKING DETAILS : \nREG. NO. = "+str(row[0])+"\nNAME = "+row[1]+"\nHOSTEL = "+row[2]+"\nGENDER = "+row[4]+"\nMOBILE = "+row[3]+"\nEMAIL = "+row[5]+"\nVEHICLE = "+row[6]+"\nPARKING = "+row[7]+"\nCATEGORY = "+row[8]+"\nSTATUS = "+row[9]+"\nSLOT = "+row[10]+"\n\n"

messagebox.showinfo("Details",S)

Show()

Flag=1

if(Flag==0):

S="NOTE : Invalid Details Provided/Parking Not Booked"

messagebox.showinfo("Note",S)

conn.commit()

def CheckIN():

if(str(B.get())=="" and str(B1.get())==""):

S="Please Fill - Registration/Vehicle"

messagebox.showinfo("Warning",S)

return

if((str(B.get())=="" or str(B1.get())=="" ) and str(B2.get())==""):

S="Please Fill Slot Number"

messagebox.showinfo("Warning",S)

return

if((str(B.get())=="" or str(B1.get())=="" ) and str(B2.get())!="" and str(pd.get())==""):

S="Please Select Block"

messagebox.showinfo("Warning",S)

return

if(str(B.get())==""):

reg=0

else:

reg=int(B.get())

veh=str(B1.get())

slot=str(B2.get())

blck=str(pd.get())

Flag=0

conn = sqlite3.connect('PMS.db')

cursor = conn.execute("SELECT REG,VEHICLE,STATUS,SLOT,PB,PARK FROM PARKING")

for row in cursor:

if((row[0]==reg or row[1]==veh) and row[4]=="NIL"):

S="NOTE : Book Parking First"

messagebox.showinfo("Note",S)

return

if((row[0]==reg or row[1]==veh) and row[4]=="P" and row[2]=="Parked" ):

S="NOTE : Vehicle Already Parked\nProceed With Checkout First"

messagebox.showinfo("Note",S)

return

if((row[0]==reg or row[1]==veh) and row[4]=="P" and row[2]=="Not Parked" and row[5]==blck): # Proceed with CheckIN

Flag=1

print("Proceed")

break

conn.commit()

conn.close()

print("DB Close")

if(Flag==1):

conn = sqlite3.connect('PMS.db')

cott = sqlite3.connect('PPMS.db')

stateP=str("Parked")

stateC=str("C")

dbCin=datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

if(str(reg)!=0):

conn.execute("UPDATE PARKING SET SLOT=?, STATUS=?,C=?,CIN=? WHERE REG=?",(slot,stateP,stateC,dbCin,reg))

cott.execute("UPDATE PARKING SET SLOT=?, STATUS=?,C=?,CIN=? WHERE REG=?",(slot,stateP,stateC,dbCin,reg))

S="NOTE : Vehicle Parked"

messagebox.showinfo("Note",S)

if(str(veh)!=""):

conn.execute("UPDATE PARKING SET SLOT=?, STATUS=?,C=?,CIN=? WHERE VEHICLE=?",(slot,stateP,stateC,dbCin,veh))

cott.execute("UPDATE PARKING SET SLOT=?, STATUS=?,C=?,CIN=? WHERE VEHICLE=?",(slot,stateP,stateC,dbCin,veh))

S="NOTE : Vehicle Parked"

messagebox.showinfo("Note",S)

conn.commit()

conn.close()

cott.commit()

cott.close()

if(Flag==0):

S="NOTE : Invalid Details/Block or Mismatching Entries"

messagebox.showinfo("Note",S)

Show()

def CheckOUT():

if(str(D.get())=="" and str(D1.get())=="" and str(D2.get())==""):

S="Please Fill - Registration/Vehicle/Slot"

messagebox.showinfo("Warning",S)

return

if(str(D.get())==""):

reg=0

else:

reg=int(D.get())

veh=str(D1.get())

slot=str(D2.get())

Flag=0

conn = sqlite3.connect('PMS.db')

cursor = conn.execute("SELECT REG,VEHICLE,STATUS,SLOT,PB FROM PARKING")

for row in cursor:

if((row[0]==reg or row[1]==veh or row[3]==slot) and row[4]=="NIL"):

S="NOTE : Book Parking First"

messagebox.showinfo("Note",S)

return

if((row[0]==reg or row[1]==veh or row[3]==slot) and row[4]=="P" and row[2]=="Not Parked" ):

S="NOTE : Vehicle Not Parked\nProceed With CheckIN First"

messagebox.showinfo("Note",S)

return

if((row[0]==reg or row[1]==veh or row[3]==slot) and row[4]=="P" and row[2]=="Parked"): # Proceed with CheckOUT

Flag=1

print("Proceed out")

break

conn.commit()

conn.close()

print("DB Close")

if(Flag==1):

conn = sqlite3.connect('PMS.db')

cott = sqlite3.connect('PPMS.db')

stateP=str("Not Parked")

stateS=str("NIL")

stateC=str("")

dbCout=datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

if(str(reg)!=0):

conn.execute("UPDATE PARKING SET STATUS=?,SLOT=?,C=?,COUT=? WHERE REG=?",(stateP,stateS,stateC,dbCout,reg))

cott.execute("UPDATE PARKING SET STATUS=?,SLOT=?,C=?,COUT=? WHERE REG=?",(stateP,stateS,stateC,dbCout,reg))

S="NOTE : Vehicle Checked OUT"

messagebox.showinfo("Note",S)

if(str(veh)!=""):

conn.execute("UPDATE PARKING SET STATUS=?,SLOT=?,C=?,COUT=? WHERE VEHICLE=?",(stateP,stateS,stateC,dbCout,veh))

cott.execute("UPDATE PARKING SET STATUS=?,SLOT=?,C=?,COUT=? WHERE VEHICLE=?",(stateP,stateS,stateC,dbCout,veh))

S="NOTE : Vehicle Cheked OUT"

messagebox.showinfo("Note",S)

if(str(slot)!=""):

conn.execute("UPDATE PARKING SET STATUS=?,SLOT=?,C=?,COUT=? WHERE SLOT=?",(stateP,stateS,stateC,dbCout,slot))

cott.execute("UPDATE PARKING SET STATUS=?,SLOT=?,C=?,COUT=? WHERE SLOT=?",(stateP,stateS,stateC,dbCout,slot))

S="NOTE : Vehicle Cheked OUT"

messagebox.showinfo("Note",S)

conn.commit()

conn.close()

cott.commit()

cott.close()

Show()

if(Flag==0):

S="NOTE : Invalid Details or Mismatching Entries"

messagebox.showinfo("Note",S)

def OverView():

S=""

conn = sqlite3.connect('PMS.db')

cursor = conn.execute("SELECT REG,NAME,VEHICLE,PARK,TYPE,SLOT,C FROM PARKING")

for row in cursor:

if(row[6]=="C"):

S=S+str(row[0])+" - "+row[1]+" - "+row[2]+" - "+row[3]+" - "+row[4]+" - "+row[5]+"\n"

messagebox.showinfo("Note",S)

conn.commit()

conn.close()

def History():

S=""

cott = sqlite3.connect('PPMS.db')

cursor = cott.execute("SELECT REG,NAME,VEHICLE,PARK,TYPE,SLOT,C FROM PARKING")

for row in cursor:

S=S+str(row[0])+" - "+row[1]+" - "+row[2]+" - "+row[3]+" - "+row[4]+" - "+row[5]+"\n"

messagebox.showinfo("Note",S)

cott.commit()

cott.close()

def AvailSlots():

S34=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

S29=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

A34=""

A29=""

conn = sqlite3.connect('PMS.db')

cursor = conn.execute("SELECT PARK,STATUS,SLOT FROM PARKING")

for row in cursor:

if(row[0]=="34 Block" and row[1]=="Parked" and row[2]!="NIL"):

S34[int(row[2])]=-1

if(row[0]=="29 Block" and row[1]=="Parked" and row[2]!="NIL"):

S29[int(row[2])]=-1

for i in S34:

if(i!=-1 and i!=0):

A34=A34+str(i)+" "

for i in S29:

if(i!=-1 and i!=0):

A29=A29+str(i)+" "

conn.commit()

messagebox.showinfo("Status","Available Parking Slots\n\nBLOCK 29 ---- "+A29+"\n\nBLOCK 34 ---- "+A34)

Base\_Page()##############################