S.B.O.A. SCHOOL AND JUNIOR COLLEGE

Anna Nagar Western Extension, Chennai - 600 101

PROJECT REPORT

2020 – 2021

Name :

Standard : Sec:

Reg. No :

Title of the Project :

BONAFIDE CERTIFICATE

Certified to be the bonafide project work done by ..... of Std..... ... in the..... laboratory of S.B.O.A. School & Junior College, Chennai - 600 101 during the year 2020 - 2021.

Date : *Teacher-in-Charge*

Submitted for the ..... examination held in the year 2020 - 2021 at S.B.O.A. School and Junior College. Chennai - 600 101.

External Examiner *Internal Examiner*

PROJECT: HOTEL MANAGEMENT

Table of Contents

ACKNOWLEDGEMENTS

I would like to thank my teachers Mrs.Jayagowri and Mrs.Visalakshi for their continuous support and guidance throughout the learning of python and also in this project, without whom all this wouldn't be possible.

I would like to thank our principal Mr.Manoharan for giving us this golden opportunity to enhance our knowledge.

I would like to thank my family and friends who helped me in the completion of this project.

Also a big shout out to my group members, together we improved each other's skills and knowledge.

This project served as wonderful opportunity to increase our knowledge in python as well as to know more about computers.

This project totally put out our skill sets to the testing and I'm glad to say that we succeeded.

PROJECT DESCRIPTION:

This python project is a GRAPHICAL USER INTERFACE(GUI) based on hotel management with the name of the hotel taken as: '**HOTEL MAJESTIC'**.

This project consists of facilities like:

- CHECKIN
- CHECKOUT
- RESTAURANT
- LAUNDRY
- ROOM SERVICE
- EMPLOYEES
- PARKING

And a special button which can be used to reset the database. With each of the functionality being a separate python file.

The project has a main screen which acts like to menu to all the other service screen.

Each service is provided through a separate screen with the necessary details to be collected from the customer.

This project aims to make all the processes very simple and the data collected is saved in the Data base with appropriate popup messages to notify the customer about the status of their booking.

Each screen also contains all the details about pricing (with all the pricing predefined) within just a click of the mouse.

Bills are generated for the necessary services are saved in specific folders.

WORKING ENVIRONMENT:

Software Used:

- Python v.3.7.9
- Microsoft Visual Studio Code

Hardware Used:

- Intel core–i5 4570
- CPU @ 3.20Ghz
- 16 GB RAM

DATA DICTIONARY:

FUNCTIONS USED:

HOTEL(MAIN SCREEN)

1. check_pwd() – checks if the entered password matches the administrator's password
2. clear_data() – resets all the data in the SQL table

CHECKIN

1. get_ids() – returns a list of all customer ID's from the customer history table
2. check_date() – checks if the selected date is valid
3. common_values() – returns a list of common elements from two lists
4. get_rooms() – returns a list of all available rooms
5. generate-id() – generates a random unique ID
6. reset() – resets all the variables and resets the screen
7. confirm() – gets all the entered details and asks us to select an available room
8. submit() – updates the customer table with all the entered details

CHECKOUT

1. get_ids() – returns a list of all customer ID's from the customer history table
2. check_date() – checks if the selected date is valid

3. common_values() – returns a list of common elements from two lists
4. get_rooms() – returns a list of all available rooms
5. generate-id() – generates a random unique ID
6. reset() – resets all the variables and resets the screen
7. confirm() – gets all the entered details and asks us to select an available room
8. submit() – updates the customer table with all the entered details

LAUNDRY

1. menu()- to display the menu of the laundry
2. get_rooms()- to select the rooms that are occupied
3. laundry()- operations related to laundry
4. submit()- saving and taking order
5. reset()- resetting the data imputed from the guest

PARKING

1. rooms_slots() – returns the available rooms and slots
2. submit() – updates the parking table based

ROOM SERVICE

1. get rooms()- to get room number from customers
2. menu()-to display menu of restaurant
3. submit()-saving and taking order for the customer
4. roomvariable.get()-to select the room no.
5. service.get()-to get/select the service needed
6. cust_id.get()-to get customer id
7. localtime()-to get the local time of the place
8. reset()-resetting the data inputed from customer

9. roomservice()-to add column names
10. get_rooms()-to get rooms available for which the customer asks

RESTAURANT

1. findcustomertype() - to find which customer it is hotelguest or restaurant guest
2. menu()- to display the menu of the restaurant
3. Restaurantguest()- operations related to restaurantguest
4. Hotelguest()- operations related to hotelguest
5. submit1()- saving and taking order for a restaurantguest
6. reset1()- resetting the data inputed from the restaurantguest
7. previous1()- going to selection screen of restaurantguest or hotelguest
8. submit2()- saving and taking order for a hotelguest
9. reset2()- resetting the data imputed from the hotelguest
10. previous2()- going to selection screen of restaurantguest or hotelguest
11. get_ids()- to select the booking ids that are taken
12. generate_id()- to generate booking ids

EMPLOYEES

1. employees()- function to execute a treeview of employees

VARIABLES USED:

HOTEL

1. main_screen – The front screen containing all the buttons

CHECKIN

1. customerId – The random generated customer ID
2. c_name – Stored the name of the customer
3. occupants_no – Stores the entered number of occupants
4. c_phNo – Stores the entered phone number of the customer
5. list_of_ids – It's a list containing all the customer ID's from customer history
6. room_type_selected – Stores the type of the room selected by the customer
7. room_selected – Stores the room number selected from the available rooms

CHECKOUT

1. room_selected – Room to be checked out
2. laundry_total – Stores the price for all the laundry services
3. restaurant_total – Stores the price for all the restaurant services
4. checkin_date – Stores the date of Check-In
5. days_of_stay – Stores the number of days of stay
6. room_total – Stores the cost for stay at the room
7. parking_total – Stores the cost for the parking services
8. total_total – Stores the total amount to be paid for the stay at the hotel

LAUNDRY

1. cust_id - customer id taken in the entry box
2. roomvariable - variable for roomno selected in option menu
3. shirtvariable - variable for no.of shirts selected in option menu
4. pantvariable - variable for no.of pants selected in option menu
5. othervariable - variable for no.of others selected in option menu
6. correct_id - correct customer id present in the database

RESTAURANT

1. restaurant_label - Title label("RESTAURANT")
2. cust - Label("Customer type")
3. menubutton - Button("MENU")
4. confirm_button - Button("CONFIRM")
5. mealdropdown - OptionMenu for type of meal
6. meal - variable for meal in Optionmenu
7. meallist - list of types of meals
8. price_breakfast- 200*(no of veg adults)+100*(no of veg children)+250*(no of nonvegadults)+150*(no of nonveg children)
9. price_lunch- 250*(no of veg adults)+150*(no of veg children)+325*(no of nonvegadults)+200*(no of nonveg children)
10. price_dinner- 275*(no of veg adults)+200*(no of veg children)+350*(no of nonvegadults)+250*(no of nonveg children)

ROOMSERVICE

1. serviceneeded-to get the service needed
2. submit_ button-Button("SUBMIT")
3. reset_ button-Button("RESET")
4. menu _button-Button("MENU")
5. time-to get the time
6. response-to get the response

7. service-to get the service
8. cust_id-to generate customer id
9. identry-to enter id details
10. roomvariable-to get room details
11. available_rooms-to get the no of rooms available
12. roomservice_screen-option of room service to generate
13. Roomno label- to get specific room no
14. Roomnooption-option to choose room no
15. Idlabel-to label the id no
16. Servicelabel- option to label the service needed
17. Serviceselected-option to see the selected service

EMPLOYEES

1. window - Screen showing employee details
2. treev - Treeview object

MODULES / LIBRARIES USED:

1. tkinter
2. mysql.connector
3. random
4. os
5. time
6. datetime

ALGORITHM:

CHECK-IN:

STEP -1: Fill all the credentials and also select the room type

STEP -2: After filling all the credentials press confirm which will ask you to select a vacant room based on the selected room type

STEP -3: Press ‘RESET’ to clear all the data entered in the entry bar and sets the day, month and year to 0

STEP -4: After selecting the room type press submit which gets all the entered credentials and add them to the SQL table

CHECK-OUT:

STEP -1: Enter the room number and the check-out date and press generate bill which generates a check-out bill with the total amount to be paid based on the number of days stayed, laundry, parking, room service and restaurant services.

STEP -2: Once the bill is generated you can press ‘Restaurant orders’ or ‘Laundry Orders’ to review all your restaurant and laundry service received

STEP -3: After reviewing all the details press check-out which generates a bill and saves it in a folder to be printed and updates the SQL table

RESTAURANT:

STEP -1: Press ‘MENU’ for information about pricing

STEP -2: Select the type of guest hotel or restaurant guest and the type of meal (Breakfast, Lunch, Dinner) and press ‘CONFIRM’.

STEP -3: Display the appropriate entries for the selection.

STEP -4: If the customer is a hotel guest, Details like Room no, customer id, no of veg(adults), no of veg(children), no of nonveg(adults), no of nonveg(children) are collected.

STEP -5: After pressing ‘SUBMIT’, check whether the customer id matches with the room no. If it doesn’t display an error otherwise add all the details to the restaurant table with date and time.

STEP -6: Press ‘RESET’ to reset all the entries to 0 and ‘PREVIOUS’ to go back to selection screen.

STEP -7: If the customer is a restaurant guest, Details like no of veg(adults), no of veg(children), no of nonveg(adults), no of nonveg(children) are collected.

STEP -8: After pressing ‘SUBMIT’, show information box to confirm the order and cost. After confirmation generate a unique booking id and add all the details to the restaurant table with date and time. Generate a bill and save it in a bill folder.

STEP -9: Press ‘RESET’ to reset all the entries to 0 and ‘PREVIOUS’ to go back to selection screen.

LAUNDRY:

STEP -1: Press ‘MENU’ for information about pricing.

STEP -2: Display entries for data to be retrieved from the customer.

STEP -3: Details like Room no, customer id, no of pants, no of shirts, no of others are collected.

STEP -4: After pressing ‘SUBMIT’, check whether the entered customer id matches with the room no, if it doesn’t show error otherwise add the details to the laundry table with date and time.

STEP -5: Press ‘RESET’ to set all the entries to 0.

ROOMSERVICE:

STEP -1: Press ‘MENU’ for information about pricing.

STEP -2: Display entries for data to be retrieved from the customer.

STEP -3: Details like Room no, customer id, service required are collected.

STEP -4: After pressing ‘SUBMIT’, check whether the entered customer id matches with the room no, if it doesn’t show error otherwise add the details to the room service table with date and time.

STEP -5: Press ‘RESET’ to set all the entries to 0.

PARKING:

STEP -1: Display entries for data to be retrieved from the customer.

STEP -3: Details like Room no, car type, license plate are collected and a parking slot is selected from the available parking slot.

STEP -4: Add the details to the car service table and the selected parking slot is marked as unavailable.

EMPLOYEES:

STEP -1: Retrieve data from the employees table based on the column ‘SHIFT’.

STEP -2: Check the current time and display the details like employee name, employee id, duty, date of join who are working in the current shift.

STEP -3: A table with the employee details are displayed.

SOURCE CODE:

MAIN SCREEN:

```
from tkinter import *
from tkinter import messagebox
import mysql.connector
import Checkin
import laundry
import Checkout
import Restaurant
import parking
import employees
import roomservice
main_screen = Tk()
main_screen.config(bg = '#12232E')
main_screen.geometry('600x700')
main_screen.title('Hotel')
main_screen.iconbitmap('hotel.ico')

connector = mysql.connector.connect(host = 'localhost', password = 'akhash', user = 'root',
database = 'project')
cursor = connector.cursor()

date_selected = StringVar()
month_selected = StringVar()
year_selected = StringVar()
```

```
Label(text = 'HOTEL MAJESTIC', padx = 20, font = ("bahnschrift", 20, "bold"), bg = '#12232E',
fg = 'white').grid(row = 0, column = 0, columnspan = 2)


def check_pwd(pwd_entered, password, pwd):
    if pwd_entered == password:
        for i in [x for x in range(26)]:
            statement = 'update customers set NAME = NULL ,OCCUPANTS = NULL ,CHECK_IN
= NULL,CUSTOMER_ID = NULL,PH_NUMBER = NULL, ROOM_TYPE = NULL,
STATUS = "FREE" where ROOM_NO = ' + str(i+100)
            cursor.execute(statement)
            connector.commit()
            statement = 'update laundry set SHIRTS = 0, PANTS = 0, OTHERS = 0, TOTAL = 0
where ROOM_NO = ' + str(i+100)
            cursor.execute(statement)
            connector.commit()
            statement = 'delete from restaurant'
            cursor.execute(statement)
            connector.commit()
            statement = 'delete from customer_history'
            cursor.execute(statement)
            connector.commit()
            statement = 'update parking set CAR_TYPE = NULL, L_PLATE = NULL, P_SLOT =
NULL, STATUS = NULL WHERE ROOM_NO = ' + str(i+100)
            cursor.execute(statement)
            connector.commit()
            statement = 'delete from laundry_history'
            cursor.execute(statement)
            connector.commit()


        pwd.destroy()
```

```
        Checkin.list_of_ids = []
        messagebox.showinfo('', 'Table Resetted')
    else:
        messagebox.showwarning('', 'Wrong Password')


def clear_data():


    password = 'rt'
    pwd = Toplevel()
    pwd.config(bg = '#12232E')
    pwd_label = Label(pwd, text = 'Enter the password', padx = 20, font = ("Times", 13, "bold"),
bg = '#12232E', fg = 'white')
    pwd_label.pack()
    pwd_entered = Entry(pwd, show = '*')
    pwd_entered.pack()
    pwd_button = Button(pwd, text = 'Enter', command = lambda: check_pwd(pwd_entered.get(),
password, pwd))
    pwd_button.pack()
#-------------------------------------------------------Buttons-----------------------------------------------
---------
Button_checkin = Button(main_screen, text='Check-In', font = ("times", 13, "italic"), relief =
RIDGE, padx = 20, pady = 15, bg = '#12232E', fg = 'white', activebackground = 'cornflower
blue', activeforeground = 'white', command = lambda: Checkin.checkin(date_selected,
month_selected, year_selected))
Button_checkout = Button(main_screen, text='Check-Out', font = ("times", 13, "italic"), relief =
RIDGE, padx = 17, pady = 15, bg = '#12232E', fg = 'white', activebackground = 'cornflower
blue', activeforeground = 'white', command = Checkout.Checkout)
Button_laundry = Button(main_screen, text='Laundry', font = ("times", 13, "italic"), relief =
RIDGE, padx = 24, pady = 15, bg = '#12232E', fg = 'white', activebackground = 'cornflower
blue', activeforeground = 'white', command = laundry.laundry)
```

```
Button_parking = Button(main_screen, text='Parking', font = ("times", 13, "italic"), relief =
RIDGE, padx = 24, pady = 15, bg = '#12232E', fg = 'white', activebackground = 'cornflower
blue', activeforeground = 'white', command = parking.parking)

Button_employees = Button(main_screen, text='Employees', font = ("times", 13, "italic"), relief
= RIDGE, padx = 17, pady = 15, bg = '#12232E', fg = 'white', activebackground = 'cornflower
blue', activeforeground = 'white', command = employees.employees)

Button_restaurant = Button(main_screen, text='Restaurant', font = ("times", 13, "italic"), relief =
RIDGE, padx = 17, pady = 15, bg = '#12232E', fg = 'white', activebackground = 'cornflower
blue', activeforeground = 'white', command = Restaurant.restaurant)

reset_button = Button(main_screen, text = 'Reset Data', font = ("times", 13, "italic"), relief =
RIDGE, bg = '#12232E', fg = 'white', padx = 17, pady = 15, command = clear_data)

room_service_button = Button(main_screen, text = 'Room Service', font = ("times", 13, "italic"),
relief = RIDGE, bg = '#12232E', fg = 'white', padx = 17, pady = 15, command =
roomservice.roomservice)

#-----------------------------------------Placing Buttons-----------------------------------------------
------------

Button_checkin.grid(row = 1, column = 0, padx = 90, pady = (90,35))

Button_checkout.grid(row = 1, column = 1, padx = 90, pady = (90,35))

Button_laundry.grid(row = 2, column = 0, padx = 90, pady = 35)

Button_parking.grid(row = 2, column = 1, padx = 90, pady = 35)

room_service_button.grid(row = 3, column = 0, padx = 90, pady = 35)

Button_restaurant.grid(row = 3, column = 1, padx = 90, pady = 35)

Button_employees.grid(row = 4, column = 0, padx = 90, pady = 35)

reset_button.grid(row = 4, column = 1, padx = 90, pady = 35)

main_screen.mainloop()
```

CHECKIN:

```
import mysql.connector
from tkinter import *
import datetime
import random
from tkinter import messagebox
connector = mysql.connector.connect(host = 'localhost', password = 'akhash', user = 'root',
database = 'project')
cursor = connector.cursor()
class labels:
    def __init__(self, screen, t, x, y):
        self = Label(screen, text = t, bg = 'wheat2', fg = 'black', font = 'bahnschrift 15 bold')
        self.place(x = x, y = y)


def get_ids():
    cursor.execute('select CUSTOMER_ID from customer_history where CUSTOMER_ID IS
NOT NULL')
    available_ids = [i[0] for i in cursor]
    return available_ids


def check_date(date, month, year):
    try:
        datetime.datetime(int(year), int(month), int(date))
        return True


    except ValueError:
        return False
```

```
def common_values(r,l):
    k = []
    for x in r:
        if x in l:
            k.append(x)
    return k
def get_rooms(t):
    statement = 'select ROOM_NO from customers where STATUS = "FREE"'
    cursor.execute(statement)
    r = [i[0] for i in cursor]
    if t == 'PREMIUM':
        l = common_values(r, [i for i in range(120,126)])
    else:
        l = common_values(r, [i for i in range(100, 120)])
    return l
def checkin(date_selected, month_selected, year_selected):

    customerId = StringVar()
    c_name = StringVar()
    occupants_no = StringVar()
    c_phNo = StringVar()

    checkin_screen = Toplevel()
    checkin_screen.config(bg = 'wheat2')
    checkin_screen.geometry('600x600')
    checkin_screen.title('Check-In')
```

```
list_of_ids = get_ids()
room_selected = IntVar()
room_type_selected = StringVar()

checkin_screen.iconbitmap('counter.ico')
def generate_id():
    s="1234567890"
    c_id = ''.join(random.sample('123456789',1)) + ''.join(random.sample(s,4))
    if c_id not in list_of_ids:
        list_of_ids.append(c_id)
        customerId.set(c_id)
    else:
        generate_id()

def reset():
    checkin_screen.destroy()
    date_selected = IntVar()
    month_selected = IntVar()
    year_selected = IntVar()
    checkin(date_selected, month_selected, year_selected)

def submit():
    customer_Id = customerId.get()
    name = c_name.get()
    occupants = occupants_no.get()
    room = room_selected.get()
    phNo = c_phNo.get()
    date = date_selected.get()
```

```
        month = month_selected.get()
        year = year_selected.get()

        if len(date) == 1:
            date = '0' + date
        if len(month) == 1:
            month = '0' + month

        statement = 'update customers set NAME = "' + str(name) + '",OCCUPANTS = ' +
str(occupants) + ',CHECK_IN = "' + str(year) + '/' + str(month) + '/' + str(date) +
'",CUSTOMER_ID = ' + str(customer_Id) + ',PH_NUMBER = ' + str(phNo) + ',ROOM_TYPE
= "' + str(room_type_selected.get()) + '", STATUS = "OCCUPIED" where ROOM_NO = ' +
str(room)

        try:
            cursor.execute(statement)
            connector.commit()
            statement = 'update parking set status = "OCCUPIED" where ROOM_NO = ' +
str(room_selected.get())
            cursor.execute(statement)
            connector.commit()
            checkin_screen.destroy()
            messagebox.showinfo('Check-In','Entry Added')
            checkin_screen.destroy()
        except:
            messagebox.showerror('Error', 'Enter proper credentials ')
    def confirm(self, t):
        customer_Id = customerId.get()
        name = c_name.get()
        occupants = occupants_no.get()
```

```
room = room_selected.get()
phNo = c_phNo.get()
date = date_selected.get()
month = month_selected.get()
year = year_selected.get()

if check_date(date, month, year) == False:
    year_selected.set(0)
    month_selected.set(0)
    date_selected.set(0)

    messagebox.showerror('Error', 'Enter a valid date')
else:
    l = get_rooms(t)

    customerId_entry.config(state = DISABLED)
    name_entry.config(state = DISABLED)
    room_type.config(state = DISABLED)
    phNo_entry.config(state = DISABLED)
    occupants_entry.config(state = DISABLED)
    date_option.config(state = DISABLED)
    month_option.config(state = DISABLED)
    year_option.config(state = DISABLED)

    room_list = OptionMenu(checkin_screen, room_selected, *l)
    room_list.place(x = 500, y = 300-30)
    room_selected.set(l[0])
    self.destroy()
```

```
        submit_button = Button(checkin_screen, text = 'Submit', bg = 'wheat2', fg = 'black', font = 'bahnschrift 15 bold', activebackground = 'wheat2', command = submit)

        submit_button.place(x = 80, y = 550-10)

#-----------------------------------------------------Labels-----------------------------------------------------

    Label(checkin_screen, text = 'CHECK-IN', bg = 'wheat2', fg = 'black', font = 'bahnschrift 25 bold').place(x = 240, y = 20)

    labels(checkin_screen, 'Customer Id',  50, 120)

    labels(checkin_screen, 'Enter your name',  50, 170)

    labels(checkin_screen, 'Number of occupants', 50, 220)

    labels(checkin_screen, 'Select a Room Type',  50, 270)

    labels(checkin_screen, 'Enter your Phone Number',  50, 320)

    labels(checkin_screen, 'Select date',  50, 370)

    labels(checkin_screen, 'Select month',  50, 420)

    labels(checkin_screen, 'Select year',  50, 470)

#-----------------------------------------------------Entries-----------------------------------------------------

    customerId_entry = Entry(checkin_screen, font = ("bahnschrift", 13, "bold"), textvariable = customerId, state = DISABLED)

    customerId_entry.place(x = 400, y = 150-30)

    name_entry = Entry(checkin_screen, font = ("bahnschrift", 13, "bold"), textvariable = c_name)

    name_entry.place(x = 400, y = 200-30)

    occupants_entry = Entry(checkin_screen, font = ("bahnschrift", 13, "bold"), textvariable = occupants_no)

    occupants_entry.place(x = 400, y = 250-30)

    phNo_entry = Entry(checkin_screen, font = ("bahnschrift", 13, "bold"), textvariable = c_phNo)

    phNo_entry.place(x = 400, y = 350-30)

#-----------------------------------------------------Option Menus-----------------------------------------------------
```

```
    room_type = OptionMenu(checkin_screen, room_type_selected, 'NORMAL', 'PREMIUM')

    room_type.place(x = 400, y = 270)

    date_option = OptionMenu(checkin_screen, date_selected, *[i for i in range(1, 32)])

    date_option.place(x = 400, y = 400-30)

    month_option = OptionMenu(checkin_screen, month_selected, *[i for i in range(1, 13)])

    month_option.place(x = 400, y = 450-30)

    year_option = OptionMenu(checkin_screen, year_selected, 2020,2021)

    year_option.place(x = 400, y = 500-30)

#--------------------------------------------------Buttons--------------------------------------------------
---

    confirm_button = Button(checkin_screen, text = 'Confirm', bg = 'wheat2', fg = 'black', font =
'bahnschrift 15 bold', activebackground = 'wheat2', command =
lambda:confirm(confirm_button, room_type_selected.get()))

    confirm_button.place(x = 80, y = 550-10)

    reset_button = Button(checkin_screen, text = 'Reset', bg = 'wheat2', fg = 'black', font =
'bahnschrift 15 bold', activebackground = 'wheat2', command = reset)

    reset_button.place(x = 400, y = 550-10)

    generate_id()
```

CHECKOUT:

```
from tkinter import *
from tkinter import messagebox
import mysql.connector
from datetime import datetime
from datetime import date
import csv
import Checkin
connector = mysql.connector.connect(host = 'localhost', password = 'akhash', user = 'root',
database = 'project')
cursor = connector.cursor()
class labels:
    def __init__(self, screen, t, x, y):
        self = Label(screen, text = t, bg = 'cadet blue', fg = 'black', font = 'bahnschrift 15 bold')
        self.place(x = x, y = y)
def check(l):
    if l == None:
        return 0
    else:
        return l
def get_rooms():
    s = 'SELECT ROOM_NO FROM customers WHERE STATUS = "OCCUPIED"'
    cursor.execute(s)
    l = [i[0] for i in cursor]
    return l
def validate_date(checkin_date, checkout_date):
```

```
    chk_out_date = date(int(checkout_date[:4]), int(checkout_date[5:7]),
int(checkout_date[8:10]))
    date_diff = chk_out_date - checkin_date
    if date_diff.days >= 0:
        return date_diff.days
    else:
        return False
def Checkout():

    if len(get_rooms()) == 0:
        messagebox.showerror('Error', 'No available rooms')
        return

    checkout_screen = Toplevel()
    checkout_screen.configure(bg = 'cadet blue')
    checkout_screen.geometry('600x600')
    checkout_screen.title('Check-Out')

    room_selected = IntVar()
    date_selected = StringVar()
    month_selected = StringVar()
    year_selected = StringVar()

    checkout_screen.iconbitmap('counter.ico')
    def reset():
        checkout_screen.destroy()
        Checkout()

    def submit(checkout_button, reset_button):
```

```
    cursor.execute('SELECT NAME FROM customers where ROOM_NO = ' +
str(room_selected.get()))
    name = [i[0] for i in cursor][0]


    cursor.execute('SELECT OCCUPANTS FROM customers where ROOM_NO = ' +
str(room_selected.get()))
    occupants = [i[0] for i in cursor][0]


    cursor.execute('SELECT CUSTOMER_ID FROM customers where ROOM_NO = ' +
str(room_selected.get()))
    customer_Id = [i[0] for i in cursor][0]


    cursor.execute('SELECT PH_NUMBER FROM customers where ROOM_NO = ' +
str(room_selected.get()))
    phNo = check([i[0] for i in cursor][0])


    cursor.execute('SELECT CHECK_IN FROM customers WHERE ROOM_NO = ' +
str(room_selected.get()))
    checkin_date = [i[0] for i in cursor][0]


    date = date_selected.get()
    month = month_selected.get()
    year = year_selected.get()


    if len(date) == 1:
        date = '0' + date
    if len(month) == 1:
        month = '0' + month
```

```
    statement = 'SELECT TOTAL FROM LAUNDRY WHERE ROOM_NO = ' +
str(room_selected.get())

    cursor.execute(statement)

    laundry_total = check([i[0] for i in cursor][0])


    statement = 'SELECT SUM(TOTAL) FROM RESTAURANT WHERE CUSTOMER_ID =
' + str(customer_Id)

    cursor.execute(statement)

    restaurant_total = check([i[0] for i in cursor][0])


    statement = 'SELECT SUM(TOTAL) FROM ROOMSERVICE WHERE ROOM_NO = ' +
str(room_selected.get())

    cursor.execute(statement)

    roomservice_total = check([i[0] for i in cursor][0])


    statement = 'SELECT SUM(ADULTS_VEG) FROM RESTAURANT WHERE
CUSTOMER_ID = ' + str(customer_Id)

    cursor.execute(statement)

    adult_veg = check([i[0] for i in cursor][0])


    statement = 'SELECT SUM(CHILDREN_VEG) FROM RESTAURANT WHERE
CUSTOMER_ID = ' + str(customer_Id)

    cursor.execute(statement)

    child_veg = check([i[0] for i in cursor][0])


    statement = 'SELECT SUM(ADULTS_NONVEG) FROM RESTAURANT WHERE
CUSTOMER_ID = ' + str(customer_Id)

    cursor.execute(statement)

    adult_nv = check([i[0] for i in cursor][0])
```

```
    statement = 'SELECT SUM(CHILDREN_NONVEG) FROM RESTAURANT WHERE
CUSTOMER_ID = ' + str(customer_Id)
    cursor.execute(statement)
    child_nv = check([i[0] for i in cursor][0])



    veg = child_veg + adult_veg
    nv = child_nv + adult_nv

    def show_restbill():
        restbill = Toplevel()
        restbill.geometry('400x400')
        restbill.config(bg = 'snow3')
        restbill.title('Restaurant Bill')

        restbill.iconbitmap('buffet.ico')

        change_x = 20
        change_y = 30

        Label(restbill, text = 'Restaurant Bill', font = 'bahnschrift 20 roman', bg = 'snow3').place(x
= 107, y = 10)
        Label(restbill, text = 'Veg Adult Meals:', font = 'bahnschrift 15 roman', bg =
'snow3').place(x = 30 + change_x, y = 80 + change_y)
        Label(restbill, text = 'Non-Veg Adult Meals:', font = 'bahnschrift 15 roman', bg =
'snow3').place(x = 30 + change_x, y = 130 + change_y)
        Label(restbill, text = 'Veg Kids Meals:', font = 'bahnschrift 15 roman', bg =
'snow3').place(x = 30 + change_x, y = 180 + change_y)
```

```
        Label(restbill, text = 'Non-Veg Kids Meals:', font = 'bahnschrift 15 roman', bg =
'snow3').place(x = 30 + change_x, y = 230 + change_y)

        Label(restbill, text = str(adult_veg), font = 'bahnschrift 15 roman', bg = 'snow3').place(x
= 320 + change_x, y = 80 + change_y)

        Label(restbill, text = str(adult_nv), font = 'bahnschrift 15 roman', bg = 'snow3').place(x =
320 + change_x, y = 130 + change_y)

        Label(restbill, text = str(child_veg), font = 'bahnschrift 15 roman', bg = 'snow3').place(x
= 320 + change_x, y = 180 + change_y)

        Label(restbill, text = str(child_nv), font = 'bahnschrift 15 roman', bg = 'snow3').place(x =
320 + change_x, y = 230 + change_y)


    def show_laundrybill():

        laundrybill = Toplevel()

        laundrybill.geometry('400x400')

        laundrybill.config(bg = 'snow3')

        laundrybill.title('Laundry Bill')


        laundrybill.iconbitmap('laundry.ico')


        statement = 'SELECT SHIRTS FROM LAUNDRY WHERE ROOM_NO = ' +
str(room_selected.get())

        cursor.execute(statement)

        shirts = [i[0] for i in cursor][0]


        statement = 'SELECT PANTS FROM LAUNDRY WHERE ROOM_NO = ' +
str(room_selected.get())

        cursor.execute(statement)

        pants = [i[0] for i in cursor][0]


        statement = 'SELECT OTHERS FROM LAUNDRY WHERE ROOM_NO = ' +
str(room_selected.get())
```

```
        cursor.execute(statement)
        others = [i[0] for i in cursor][0]

        change_x = 20
        change_y = 60

        Label(laundrybill, text = 'Laundry Bill', font = 'bahnschrift 20 roman', bg =
'snow3').place(x = 130, y = 20)
        Label(laundrybill, text = 'Shirts:', font = 'bahnschrift 17 roman', bg = 'snow3').place(x =
30 + change_x, y = 80 + change_y)
        Label(laundrybill, text = 'Pants:', font = 'bahnschrift 17 roman', bg = 'snow3').place(x =
30 + change_x, y = 130 + change_y)
        Label(laundrybill, text = 'Others:', font = 'bahnschrift 17 roman', bg = 'snow3').place(x =
30 + change_x, y = 180 + change_y)
        # Label(laundrybill, text = 'Total:', font = 'bahnschrift 15 roman', bg = 'snow3').place(x =
30 + 20, y = 230 + 30)
        Label(laundrybill, text = str(shirts), font = 'bahnschrift 17 roman', bg = 'snow3').place(x =
320 + change_x, y = 80 + change_y)
        Label(laundrybill, text = str(pants), font = 'bahnschrift 17 roman', bg = 'snow3').place(x =
320 + change_x, y = 130 + change_y)
        Label(laundrybill, text = str(others), font = 'bahnschrift 17 roman', bg = 'snow3').place(x
= 320 + change_x, y = 180 + change_y)
        # Label(laundrybill, text = str(laundry_total), font = 'bahnschrift 15 roman', bg =
'snow3').place(x = 320 + 20, y = 230 + 30)

    def car_fee(x):
        if x == 'SUV':
            return 100
        if x == 'SEDAN':
            return 200
        if x == 'HATCHBACK':
```

```
            return 300
        if x == 'CONVERTIBLE':
            return 400
        if x == 'SPORT':
            return 500
        else:
            return 0

    if restaurant_total == None:
        restaurant_total = 0

    statement = 'SELECT ROOM_TYPE FROM CUSTOMERS WHERE ROOM_NO = ' +
str(room_selected.get())
    cursor.execute(statement)
    room_type = [i[0] for i in cursor][0]

    if room_type == 'NORMAL':
        cost = 2500
    else:
        cost = 4000

    statement = 'SELECT CAR_TYPE FROM PARKING WHERE ROOM_NO = ' +
str(room_selected.get())
    cursor.execute(statement)
    car_type = [i[0] for i in cursor][0]

    car_cost = car_fee(car_type)

    tax = 8
```

```
checkout_date = str(year) + '-' + str(month) + '-' + str(date)

days_of_stay = validate_date(checkin_date, checkout_date)

if days_of_stay == 0:
    days_of_stay = 1

room_total = days_of_stay * cost
parking_total = days_of_stay * car_cost
total_total = str(float((room_total + laundry_total + restaurant_total + parking_total)) * (1 + tax/100))
l = total_total.split('.')
total_total = l[0] + '.' + l[1][0:2]


def chkout():
    statement = 'insert into customer_history values (''' + str(name) + ''',' + str(room_selected.get()) + ',' + str(occupants) + ',''' + str(checkin_date) + ''','''' + str(checkout_date) + '''),' + str(customer_Id) + ',' + str(phNo) + ',' + str(days_of_stay) + ',' + str(total_total) + ',''' + str(room_type) + ''')'
    cursor.execute(statement)
    connector.commit()
    statement = 'update customers set NAME = NULL ,OCCUPANTS = NULL ,CHECK_IN = NULL,CUSTOMER_ID = NULL,PH_NUMBER = NULL, ROOM_TYPE = NULL, STATUS = "FREE" where ROOM_NO = ' + str(room_selected.get())
    cursor.execute(statement)
    connector.commit()

    f = open('C:\\Users\\RK\\Desktop\\CS Project\\Hotel bill folder\\' + str(customer_Id) + ".csv","w")
```

```
        w = csv.writer(f)

        w.writerow(['S.No' , 'Product Description', 'Quantity', 'Price'])
        w.writerow(['1' , str(room_type).title() + ' Room' ,days_of_stay, room_total])
        w.writerow(['2', 'Room Service', '-', roomservice_total])
        w.writerow(['2' , 'Car Service' ,'1', parking_total])
        w.writerow(['3' , 'Restaurant' ,'-', restaurant_total])
        w.writerow(['4' , 'Laundry' ,'-', laundry_total])
        w.writerow(['-' , 'SubTotal' ,'-', room_total + laundry_total + restaurant_total +
parking_total])
        w.writerow(['-' , 'Total' ,'-', room_total + laundry_total + restaurant_total + parking_total])

        checkout_screen.destroy()
        messagebox.showinfo('Check-Out', 'Successfully Checked-Out !')

      if Checkin.check_date(date_selected.get(), month_selected.get(), year_selected.get()):
        if validate_date(checkin_date, checkout_date) == False and validate_date(checkin_date,
checkout_date) != 0:
            messagebox.showerror('Error', 'Check-Out date was before Check-In')
        else:
            rooms_option.config(state = DISABLED)
            date_menu.config(state = DISABLED)
            month_menu.config(state = DISABLED)
            year_menu.config(state = DISABLED)
            checkout_button.config(state = DISABLED)
#-------------------------------------------------------------Billing Screen-------------------------------------
----------------
            checkout_screen.geometry('1233x645')
```

```
restaurant_bill = Button(checkout_screen, text = 'Restaurant Orders', bg = 'cadet blue', fg = 'black', font = 'bahnschrift 15 bold', activebackground = 'cadet blue', command = show_restbill)

restaurant_bill.place(x = 65, y = 550)


laundry_button = Button(checkout_screen, text = 'Laundry Orders', bg = 'cadet blue', fg = 'black', font = 'bahnschrift 15 bold', activebackground = 'cadet blue', command = show_laundrybill)

laundry_button.place(x = 350, y = 550)


bill = Text(checkout_screen, height = 27, width = 61, borderwidth = 10, bg = 'gainsboro', font = 'bahnschrift 14 roman',relief = SUNKEN)

bill.insert(INSERT, '\n\n\nName: ' + name + '\t\t\t\t\t\t   Room No: ' + str(room_selected.get()) + '\n')

bill.insert(INSERT, 'Check-In Date: ' + str(checkin_date) + '\t\t\t\t\t        Customer Id: ' + str(customer_Id) + '\n')

bill.insert(INSERT, '-----------------------------------------------------------------\n')

bill.insert(INSERT, '   S.No\t\t     Product Description\t\t\t     Quantity\t      Price\n\n')

bill.insert(INSERT, '      1\t\t           ' + str(room_type).title() + ' Room\t\t\t           '+ str(days_of_stay) + '\t           ' + str(room_total) + '\n\n')

bill.insert(INSERT, '      2\t\t           Room Service\t\t\t          -\t          ' + str(roomservice_total) + '\n\n')

bill.insert(INSERT, '      3                                Car Service                        1                ' + str(parking_total) + '\n\n')

bill.insert(INSERT, '      4\t\t              Restaurant\t\t\t            -\t          '  + str(restaurant_total) + '\n\n')

bill.insert(INSERT, '      5\t\t\tLaundry\t\t            -\t           ' + str(laundry_total) + '\n\n')

bill.insert(INSERT, '*****************************************************************************\n')

bill.insert(INSERT, '  SubTotal :  Rs ' + str(room_total + laundry_total + restaurant_total + parking_total) + '\n')
```

```
            bill.insert(INSERT, '  Tax : ' + str(tax) + '%\n')
            bill.insert(INSERT, '  Total :  Rs ' + total_total + '\n\n')
            bill.insert(INSERT, '*' * 32 + ' Thank You ' + '*' * 32 + '\n\n')
            bill.insert(INSERT, '*' * 20 + ' Hope you Enjoyed your stay with Us ' + '*' * 17)
            bill.tag_add('total_tag','20.0','22.30')
            bill.tag_config('total_tag', font = 'bahnschrift 19 roman')
            # bill.config(state = DISABLED)
            bill.place(x = 600, y = 0)
            Label(checkout_screen, text = 'Majestic Hotel', font = 'bahnschrift 20 roman', bg =
'gainsboro').place(x = 830, y = 10)
            chkout_button = Button(checkout_screen, text = 'Check-Out', bg = 'gainsboro', font =
'bahnschrift 20 roman', activebackground = 'gainsboro', command = chkout)
            chkout_button.place(x = 1010, y = 463)


        else:
            messagebox.showerror('Error', 'Enter a valid date')
#-------------------------------------------------------------Labels-----------------------------------------
---------------
    Label(checkout_screen, text = 'CHECK-OUT', bg = 'cadet blue', fg = 'black', font =
'bahnschrift 25 bold').place(x = 200, y = 20)
    labels(checkout_screen, 'Select your Room Number:', 50, 120)
    labels(checkout_screen, 'Select the date:', 50, 200)
    labels(checkout_screen, 'Select the month:', 50, 280)
    labels(checkout_screen, 'Select the year:', 50, 360)
#-------------------------------------------------------------Menus-----------------------------------------
----------------
    rooms_option = OptionMenu(checkout_screen, room_selected, *get_rooms())
    rooms_option.place(x = 400, y = 120)


    room_selected.set(get_rooms()[0])
```

```
    date_menu = OptionMenu(checkout_screen, date_selected, *[i for i in range(1, 32)])
    date_menu.place(x = 400, y = 200)
    date_selected.set(21)

    month_menu = OptionMenu(checkout_screen, month_selected, *[i for i in range(1, 13)])
    month_menu.place(x = 400, y = 280)
    month_selected.set(1)

    year_menu = OptionMenu(checkout_screen, year_selected, 2020,2021)
    year_menu.place(x = 400, y = 360)
    year_selected.set(2020)
#-------------------------------------------------------------Buttons-----------------------------------------
---------------
    reset_button = Button(checkout_screen, text = 'Reset', bg = 'cadet blue', fg = 'black', font =
'bahnschrift 15 bold', activebackground = 'cadet blue', state = ACTIVE, command = reset)
    reset_button.place(x = 130, y = 450)
    generate_bill_button = Button(checkout_screen, text = 'Generate Bill', bg = 'cadet blue', fg =
'black', font = 'bahnschrift 15 bold', activebackground = 'cadet blue', state = ACTIVE, command
= lambda: submit(generate_bill_button, reset_button))
    generate_bill_button.place(x = 360, y = 450)
```

LAUNDRY:

```
from tkinter import *
from tkinter import messagebox
import mysql.connector
import time
import datetime
import os
connector = mysql.connector.connect(host = 'localhost', password = 'akhash', user = 'root',
database = 'project')
cursor = connector.cursor()
def get_rooms():
    statement = 'select ROOM_NO from customers where STATUS != "FREE"'
    cursor.execute(statement)
    r = [i[0] for i in cursor]
    return r
def menu():
      messagebox.showinfo("Laundry"," \tPRICING\n\n\nShirts --> RS 15 \nPants --> RS 25
\nOthers --> RS 10 \n \n \n* Sarees are also counted as shirts\n ")
def submit():

      room=roomvariable.get()
      shirts=shirtvariable.get()
      pants=pantvariable.get()
      others=othervariable.get()
      statement1='SELECT CUSTOMER_ID FROM CUSTOMERS WHERE ROOM_NO='
+  str(room)
      cursor.execute(statement1)
      correct_id= cursor.fetchone()


      price=15*(shirts)+25*(pants)+10*(others)
      if price !=0 and cust_id.get()!=0:
            if correct_id[0] == cust_id.get():
                  a=time.localtime()
                  hours=a[3]
                  mins=a[4]
                  sec=a[5]
                  ctime=datetime.time(hours,mins,sec)
                  cdate= datetime.date.today()
                  ctime=str(ctime)
                  cdate=str(cdate)
```

```
            submitbutton.configure(state=DISABLED)
            resetbutton.configure(state=DISABLED)

            response=messagebox.askokcancel('Laundry',"Click OK to confirm\n\nCOST
:"+str(price))
            if response==1:
                statement2='insert into laundry_history
(ROOM_NO,CUSTOMER_ID,SHIRTS,PANTS,OTHERS,TOTAL,DATE,TIME)
values('+str(room)+','+str(cust_id.get())+','+
str(shirts)+','+str(pants)+','+str(others)+','+str(price)+',"'+cdate+'","'+ctime+'")'
                statement3='UPDATE LAUNDRY SET SHIRTS=SHIRTS+'+ str(shirts)+
',PANTS=PANTS+' +str(pants)+ ',OTHERS=OTHERS+' +str(others)+ ',TOTAL=TOTAL+'
+ str(price)+ ' WHERE ROOM_NO=' + str(room)
                cursor.execute(statement2)
                connector.commit()
                cursor.execute(statement3)
                connector.commit()
                messagebox.showinfo("Laundry",message="Entry successfully added")
                resetbutton.configure(state=ACTIVE)

            else:
                submitbutton.configure(state=ACTIVE)
                resetbutton.configure(state=ACTIVE)





        else:
            messagebox.showerror('Landry',"Room Number or Customer ID wrong")

    else:
        messagebox.showerror('Laundry',"Enter proper credentials")
def reset():
    shirtvariable.set(0)

    pantvariable.set(0)

    othervariable.set(0)

    cust_id.set(0)

    roomvariable.set(available_rooms[0])
```

```
    submitbutton.configure(state=ACTIVE)
def laundry():

    global identry
    global roomvariable
    global shirtvariable
    global pantvariable
    global othervariable
    global cust_id
    global submitbutton
    global resetbutton
    global available_rooms



    available_rooms=get_rooms()

    shirtvariable=IntVar()
    shirtvariable.set(0)

    pantvariable=IntVar()
    pantvariable.set(0)

    othervariable=IntVar()
    othervariable.set(0)

    roomvariable=IntVar()
    cust_id=IntVar()
    cust_id.set(0)

    laundry_screen=Toplevel()
    laundry_screen.geometry("600x600")
    laundry_screen.config(bg='salmon2')
    laundry_screen.title("Laundry")
    laundry_screen.iconbitmap(str(os.getcwd())+'/icons/'+'laundry.ico')

    try:

        roomvariable.set(available_rooms[0])
    except:
        messagebox.showerror("No customers available")
        laundry_screen.destroy()
        return
```

```
    # title label
    Laundry_label=Label(laundry_screen,text="LAUNDRY",font = 'bahnschrift 25
bold',bg='salmon2',fg='black',padx=20)
    Laundry_label.place(x=240-30,y=50-30)
    # for getting roomno
    roomnolabel=Label(laundry_screen,text='Room
No',fg='black',bg='salmon2',font='bahnschrift 15 bold',padx=20)
    roomnolabel.place(x=80,y=150-30)


menu_button=Button(laundry_screen,text='MENU',fg='black',bg='salmon2',font='bahnschrif
t 15 bold',padx=20,command=menu)
    menu_button.place(x=480,y=50)


    roomnooption=OptionMenu(laundry_screen,roomvariable,*available_rooms)
    roomnooption.place(x=400,y=150-30)

    # for getting customerid
    idlabel=Label(laundry_screen,text='Customer
id',fg='black',bg='salmon2',font='bahnschrift 15 bold',padx=20)
    idlabel.place(x=80,y=220-30)

    identry=Entry(laundry_screen,textvariable=cust_id)
    identry.place(x=400,y=220-30)


    #for getting no of shirts
    shirtlabel=Label(laundry_screen,text='Shirts',fg='black',bg='salmon2',font='bahnschrift
15 bold',padx=20)
    shirtlabel.place(x=80,y=280-30)

    shirtoption=OptionMenu(laundry_screen,shirtvariable,*[i for i in range(0,20)])
    shirtoption.place(x=400,y=280-30)


    #for getting no of pants
    pantlabel=Label(laundry_screen,text='Pants',fg='black',bg='salmon2',font='bahnschrift
15 bold',padx=20)
    pantlabel.place(x=80,y=350-30)
```

```
    pantoption=OptionMenu(laundry_screen,pantvariable,*[i for i in range(0,20)])
    pantoption.place(x=400,y=350-30)

    #for getting no of others
    otherlabel=Label(laundry_screen,text='Others',fg='black',bg='salmon2',font='bahnschrift
15 bold',padx=20)
    otherlabel.place(x=80,y=420-30)

    otheroption=OptionMenu(laundry_screen,othervariable,*[i for i in range(0,20)])
    otheroption.place(x=400,y=420-30)

    #submit button

submitbutton=Button(laundry_screen,text='SUBMIT',fg='black',bg='white',font='bahnschrift
15 bold',padx=20,command=submit)
    submitbutton.place(x=150,y=540-30)

    #reset button

resetbutton=Button(laundry_screen,text="RESET",fg='black',bg='white',font='bahnschrift
15 bold',padx=20,command=reset)
    resetbutton.place(x=300,y=540-30)
```

PARKING:

```
from tkinter import *
from tkinter import messagebox
import mysql.connector
connector = mysql.connector.connect(host = 'localhost', password = 'akhash', user = 'root',
database = 'project')
cursor = connector.cursor()
class labels:
    def __init__(self, screen, t, x, y):
        self = Label(screen, text = t, bg = 'light slate grey', fg = 'black', font = 'bahnschrift 18 bold')
        self.place(x = x, y = y)
def check_none(x):
    for i in x:
        if i == None:
            return True
    return False
def rooms_slots():
    statement = 'SELECT ROOM_NO FROM PARKING WHERE STATUS = "OCCUPIED"'
    cursor.execute(statement)
    rooms = [i[0] for i in cursor]
    statement = 'SELECT P_SLOT FROM PARKING WHERE P_SLOT IS NOT NULL'
    cursor.execute(statement)
    slots = [i[0] for i in cursor]
    if check_none(slots):
        slots = []

    slots2 = [j for j in range(1,26)]
```

```
    for x in slots:
        if x in slots2:
            slots2.remove(x)

    return slots2, rooms
def parking():
    parking_screen = Toplevel()
    parking_screen.configure(bg = 'light slate grey')
    parking_screen.title("Parking")
    parking_screen.geometry('600x600')

    room_selected = IntVar()
    type_selected = StringVar()
    slot_selected = IntVar()

    def submit():
        if type_selected.get() == None or plateNo_entry.get == None or slot_selected.get() == None:
            messagebox.showerror('', 'Enter proper credentials')
            parking_screen.destroy()
            parking()
        statement = 'UPDATE PARKING SET CAR_TYPE = ''' + str(type_selected.get()) + ''', L_PLATE = ''' + str(plateNo_entry.get()) + ''', P_SLOT = ' + str(slot_selected.get()) + ' WHERE ROOM_NO = ' + str(room_selected.get())
        cursor.execute(statement)
        connector.commit()
        parking_screen.destroy()
        messagebox.showinfo('', 'Updated')
```

```
    car_types = ['SUV', 'SEDAN', 'HATCHBACK', 'CONVERTIBLE', 'SPORT']

    Label(parking_screen, text = 'Parking', font = 'bahnschrift 25 bold', bg = 'light slate
grey').place(x = 235, y = 5)
    labels(parking_screen, 'Room Number', 50, 150)
    labels(parking_screen, 'Car Type', 50, 240)
    labels(parking_screen, 'Licence Plate Number', 50, 330)
    labels(parking_screen, 'Parking Slot', 50, 420)

    slots, rooms = rooms_slots()

    if rooms == []:
        parking_screen.destroy()
        messagebox.showerror('','No customers available')
        return

    room_menu = OptionMenu(parking_screen, room_selected, *rooms)
    room_menu.place(x = 470, y = 150)
    type_menu = OptionMenu(parking_screen, type_selected, *car_types)
    type_menu.place(x = 470, y = 240)
    plateNo_entry = Entry(parking_screen, font = 'bahnschrift 13 bold', width = 13)
    plateNo_entry.place(x = 470, y = 330)
    slots_menu = OptionMenu(parking_screen, slot_selected, *slots)
    slots_menu.place(x = 470, y = 420)
    submit_button = Button(parking_screen, text = 'Submit', font = 'bahnschrift 18 bold', bg =
'light slate grey', activebackground = 'light slate grey', command = submit)
    submit_button.place(x = 250, y = 520)
```

ROOM SERVICE:

```
from tkinter import *
from tkinter import messagebox
import mysql.connector
import time
import datetime
import os
connector = mysql.connector.connect(host = 'localhost', password = 'akhash', user = 'root',
database = 'project')
cursor = connector.cursor()
def get_rooms():
    statement = 'select ROOM_NO from customers where STATUS != "FREE"'
    cursor.execute(statement)
    r = [i[0] for i in cursor]
    return r
def menu():
    messagebox.showinfo("Room service"," \tPRICING\n\n\n Coffee--> RS 20 \nTea --> RS 20
\nMilk --> RS 10 \n \n \n* Other services are free of cost\n ")
def submit():


    room=roomvariable.get()
    serviceneeded= service.get()
    statement1='SELECT CUSTOMER_ID FROM CUSTOMERS WHERE ROOM_NO=' +
str(room)
    cursor.execute(statement1)
    correct_id= cursor.fetchone()


    if serviceneeded=='Cleaning'or serviceneeded=='Water' or serviceneeded=='Hotwater':
```

```
        price=0
    if serviceneeded=='Milk':
        price=10
    if serviceneeded=='Coffee':
        price=20
    if serviceneeded=='Tea':
        price=20
    if correct_id[0]==cust_id.get():
        a=time.localtime()
        hours=a[3]
        mins=a[4]
        sec=a[5]
        ctime=datetime.time(hours,mins,sec)
        cdate= datetime.date.today()
        ctime=str(ctime)
        cdate=str(cdate)

        submitbutton.configure(state=DISABLED)
        resetbutton.configure(state=DISABLED)
        response=messagebox.askokcancel("Room service","Click OK to confirm your order\n\n
COST:"+str(price))
        if response==1:
            statement2=  "insert into roomservice (ROOM_NO,SERVICE,TOTAL,DATE,TIME)
values('%s' ,'%s' ,%s,'%s','%s')" %(room,serviceneeded,price,cdate,ctime)
            cursor.execute(statement2)
            connector.commit()
            messagebox.showinfo("Room service",message='Entry successfully added')
            resetbutton.configure(state=ACTIVE)
        else:
```

```
        submitbutton.configure(state=ACTIVE)
        resetbutton.configure(state=ACTIVE)
    else:
      messagebox.showerror('Restaurant',"Room Number or Customer ID wrong")
def reset():
    service.set('Cleaning')
    cust_id.set(0)
    roomvariable.set(available_rooms[0])
    submitbutton.configure(state=ACTIVE)
def roomservice():
    global identry
    global roomvariable
    global service
    global cust_id
    global submitbutton
    global resetbutton
    global available_rooms
    available_rooms=get_rooms()
    servicelist=['Cleaning','Milk','Tea','Coffee','Water','Hotwater']
    service= StringVar()
    service.set(servicelist[0])
    roomvariable=IntVar()
    cust_id=IntVar()
    cust_id.set(0)
    roomservice_screen=Toplevel()
    roomservice_screen.geometry("600x600")
    roomservice_screen.config(bg='RosyBrown1')
    roomservice_screen.title("Room service")
```

```
    #roomservice_screen.iconbitmap(str(os.getcwd())+'/icons/'+'roomservice.ico')
    try:

        roomvariable.set(available_rooms[0])
    except:
        messagebox.showerror("No customers available")
        roomservice_screen.destroy()
        return
    # title label
    roomservice_label=Label(roomservice_screen,text="ROOM SERVICE",font = 'bahnschrift 24 bold',bg='RosyBrown1',fg='black',padx=20)
    roomservice_label.place(x=240-30,y=50-30)
    # for getting roomno
    roomnolabel=Label(roomservice_screen,text='Room No',fg='black',bg='RosyBrown1',font='bahnschrift 15 bold',padx=20)
    roomnolabel.place(x=80,y=170)


menu_button=Button(roomservice_screen,text='MENU',fg='black',bg='RosyBrown1',font='bahnschrift 15 bold',padx=20,command=menu)
    menu_button.place(x=480,y=100)

    roomnooption=OptionMenu(roomservice_screen,roomvariable,*available_rooms)
    roomnooption.place(x=400,y=170)

    # for getting customerid
    idlabel=Label(roomservice_screen,text='Customer id',fg='black',bg='RosyBrown1',font='bahnschrift 15 bold',padx=20)
    idlabel.place(x=80,y=240)
```

```
    identry=Entry(roomservice_screen,textvariable=cust_id)
    identry.place(x=400,y=240)

    #service required

    servicelabel=Label(roomservice_screen,text='Service
required',fg='black',bg='RosyBrown1',font='bahnschrift 15 bold',padx=20,pady=20)
    servicelabel.place(x=80,y=310)

    serviceselected=OptionMenu(roomservice_screen,service,*servicelist)
    serviceselected.place(x=400,y=310)

    #submit button

submitbutton=Button(roomservice_screen,text='SUBMIT',fg='black',bg='white',font='bahnschrif
t 15 bold',padx=20,command=submit)
    submitbutton.place(x=150,y=540-30)

    #reset button
    resetbutton=Button(roomservice_screen,text="RESET",fg='black',bg='white',font='bahnschrift
15 bold',padx=20,command=reset)
    resetbutton.place(x=300,y=540-30)
```

RESTAURANT:

```
from tkinter import *
from tkinter import messagebox
import mysql.connector
import random
import time
import datetime
import os
connector = mysql.connector.connect(host = 'localhost', password = 'akhash', user = 'root',
database = 'project')
cursor = connector.cursor()
def get_ids():
    cursor.execute('select BOOKING_ID from restaurant where BOOKING_ID IS NOT NULL')
    available_ids = [i[0] for i in cursor]
    return available_ids
idlist=get_ids()
def generate_id():
        s="123456789"
        b_id = ''.join(random.sample('123456789',1)) + ''.join(random.sample(s,4))
        if b_id not in idlist:
            idlist.append(b_id)
            bookingid.set(b_id)
        else:
            generate_id()
def restaurant():

    def menu():
```

```
    messagebox.showinfo("Restaurant","\tPRICING\n\nBREAKFAST:(8 AM-12
AM)\nAdult(VEG)--> RS 200\nChild(VEG)--> RS 100\nAdult(NONVEG)--> RS
250\nChild(NONVEG)--> RS 150"+
                                    "\n\nLUNCH:(12 PM-5 PM)\nAdult(VEG)--> RS
250\nChild(VEG)--> RS 150\nAdult(NONVEG)--> RS 325\nChild(NONVEG)--> RS 200"+
                                    "\n\nDINNER:(7 PM-12 PM)\nAdult(VEG)--> RS
275\nChild(VEG)--> RS 200\nAdult(NONVEG)--> RS 350\nChild(NONVEG)--> RS 250"+
                                    "\n\n*Please note that we only offer BUFFET"+
                                    "\n*BUFFETS are open only during the time mentioned
above"+
                                    "\n*Food cannot be provided directly through ROOM
SERVICE"+
                                    "\n*CHILD - 14 AND BELOW")


  def findcustomertype():
      if meal.get() not in["Breakfast","Dinner","Lunch"]:
        confirm_button.configure(state=DISABLED)
        messagebox.showerror(","Meal not selected")
      else:
        global type1
        type1=r.get()
        #print(type1)
        if type1=="Hotel Guest":
          hotelguest()


        elif type1=='Restaurant Guest':
          Restaurantguest()
        else:
          messagebox.showerror(","Please select customer type")
```

```
global r



# for hotel guest


def submit2():
        a=time.localtime()
        hours=a[3]
        price_breakfast=
200*(vegadult1no.get())+100*(vegchild1no.get())+250*(nonvegadult1no.get())+150*(nonvegchi
ld1no.get())
        price_lunch=
250*(vegadult1no.get())+150*(vegchild1no.get())+325*(nonvegadult1no.get())+200*(nonvegchi
ld1no.get())
        price_dinner=
275*(vegadult1no.get())+200*(vegchild1no.get())+350*(nonvegadult1no.get())+250*(nonvegchi
ld1no.get())
        statement1='SELECT CUSTOMER_ID FROM CUSTOMERS WHERE ROOM_NO=' +
str(roomnovariable.get())
        cursor.execute(statement1)
        correct_id= cursor.fetchone()
        if meal.get()=='Breakfast':
            price=price_breakfast
        if meal.get()=='Lunch':
            price=price_lunch
        if meal.get()=='Dinner':
            price=price_dinner


        if  price!=0:
```

```
            if correct_id[0] == cust_id.get() :
                a=time.localtime()
                hours=a[3]
                mins=a[4]
                sec=a[5]
                ctime=datetime.time(hours,mins,sec)
                cdate= datetime.date.today()
                ctime=str(ctime)
                cdate=str(cdate)

                submit1button.configure(state=DISABLED)
                reset1button.configure(state=DISABLED)
                previous1button.configure(state=DISABLED)

                response=messagebox.askokcancel("Restaurant","Click OK to confirm your
order\n\n COST:"+str(price))
                if response==1:
                    statement2=  'insert into restaurant
(CUSTOMER_ID,CUSTOMER_TYPE,ADULTS_VEG,CHILDREN_VEG,ADULTS_NONVE
G,CHILDREN_NONVEG,TOTAL,DATE,TIME) values('+str(identry.get())+','"+
type1+'","+str(vegadult1no.get())+','+str(vegchild1no.get())+','+str(nonvegadult1no.get())+','+str(
nonvegchild1no.get())+','+str(price)+','"+cdate+'","'+ctime+'")'
                    cursor.execute(statement2)
                    connector.commit()
                    messagebox.showinfo("Restaurant",message='Entry successfully added')
                    reset1button.configure(state=ACTIVE)
                    previous1button.configure(state=ACTIVE)
                else:
                    submit1button.configure(state=ACTIVE)
```

```
            reset1button.configure(state=ACTIVE)
            previous1button.configure(state=ACTIVE)
        else:

            messagebox.showerror('Restaurant',"Room Number or Customer ID wrong")

    else:

        messagebox.showerror('Restaurant',"Enter proper credentials")

def reset2():

    vegadult1no.set(0)
    nonvegadult1no.set(0)
    vegchild1no.set(0)
    nonvegchild1no.set(0)
    cust_id.set(0)
    submit1button.configure(state=ACTIVE)

def previous2():
    confirm_button.config(state=ACTIVE)
    mealdropdown.config(state=ACTIVE)

    vegadult1label.destroy()
    vegadult1option.destroy()

    vegchild1label.destroy()
    vegchild1option.destroy()
```

```
        nonvegadult1label.destroy()
        nonvegadult1option.destroy()

        nonvegchild1label.destroy()
        nonvegchild1option.destroy()

        roomnolabel.destroy()
        roomnooption.destroy()
        idlabel.destroy()
        identry.destroy()

        submit1button.destroy()
        reset1button.destroy()
        previous1button.destroy()

def hotelguest():
    global identry

    global vegadult1no
    global vegchild1no
    global nonvegadult1no
    global nonvegchild1no

    global vegadult1label
    global vegchild1label
    global nonvegadult1label
    global nonvegchild1label
```

```
global vegadult1option
global vegchild1option
global nonvegadult1option
global nonvegchild1option

global previous1button
global reset1button
global submit1button

global roomnolabel
global roomnooption
global cust_id
global idlabel

global roomnovariable




vegadult1no=IntVar()

vegchild1no=IntVar()

nonvegadult1no=IntVar()

nonvegchild1no=IntVar()
```

```
cust_id=IntVar()
cust_id.set(0)

confirm_button.config(state=DISABLED)
mealdropdown.config(state=DISABLED)

def get_rooms():
    statement = 'select ROOM_NO from customers where OCCUPANTS is not NULL'
    cursor.execute(statement)
    r = [i[0] for i in cursor]
    return r

available_rooms=get_rooms()
roomnovariable= IntVar()
try:
    roomnovariable.set(available_rooms[0])
except:
    messagebox.showerror("No customers available")
    Restaurant_screen.destroy()
    return




roomnolabel=Label(Restaurant_screen,text="Room
no",fg='darkgoldenrod1',bg='firebrick2',font = 'bahnschrift 15 bold',padx=20)
roomnolabel.place(x=80,y=210)

roomnooption=OptionMenu(Restaurant_screen,roomnovariable,*available_rooms)
roomnooption.place(x=400,y=210)
```

```
idlabel=Label(Restaurant_screen,text='Customer
id',fg='darkgoldenrod1',bg='firebrick2',font='bahnschrift 15 bold',padx=20)
idlabel.place(x=80,y=270)

identry=Entry(Restaurant_screen,textvariable= cust_id)
identry.place(x=400,y=270)

vegadult1label=Label(Restaurant_screen,text="VEG(Adults)",fg='darkgoldenrod1',bg='firebrick
2',font = 'bahnschrift 15 bold',padx=20)
vegadult1label.place(x=80,y=320)

vegadult1option=OptionMenu(Restaurant_screen,vegadult1no,*[i for i in range(0,5)])
vegadult1option.place(x=400,y=320)

vegchild1label=Label(Restaurant_screen,text="VEG(Kids)",fg='darkgoldenrod1',bg='firebrick2',
font = 'bahnschrift 15 bold',padx=20)
vegchild1label.place(x=80,y=370)

vegchild1option=OptionMenu(Restaurant_screen,vegchild1no,*[i for i in range(0,5)])
vegchild1option.place(x=400,y=370)

nonvegadult1label=Label(Restaurant_screen,text="NONVEG(Adults)",fg='darkgoldenrod1',bg='
firebrick2',font = 'bahnschrift 15 bold',padx=20)
nonvegadult1label.place(x=80,y=420)
```

```
    nonvegadult1option=OptionMenu(Restaurant_screen,nonvegadult1no,*[i for i in
range(0,5)])

    nonvegadult1option.place(x=400,y=420)



nonvegchild1label=Label(Restaurant_screen,text="NONVEG(Kids)",fg='darkgoldenrod1',bg='fi
rebrick2',font = 'bahnschrift 15 bold',padx=20)

    nonvegchild1label.place(x=80,y=470)


    nonvegchild1option=OptionMenu(Restaurant_screen,nonvegchild1no,*[i for i in
range(0,5)])

    nonvegchild1option.place(x=400,y=470)



submit1button=Button(Restaurant_screen,text="SUBMIT",fg='darkgoldenrod1',bg='firebrick2',f
ont='bahnschrift 15 bold',activebackground=
'firebrick2',activeforeground='darkgoldenrod1',padx=20,command=submit2)

    submit1button.place(x=100,y=550)


    #reset button


reset1button=Button(Restaurant_screen,text="RESET",fg='darkgoldenrod1',bg='firebrick2',font=
'bahnschrift 15 bold',activebackground=
'firebrick2',activeforeground='darkgoldenrod1',padx=20,command=reset2)

    reset1button.place(x=220,y=550)


    #previous page



previous1button=Button(Restaurant_screen,text="PREVIOUS",fg='darkgoldenrod1',bg='firebric
k2',font='bahnschrift 15 bold',activebackground=
'firebrick2',activeforeground='darkgoldenrod1',padx=20,command=previous2)

    previous1button.place(x=330,y=550)
```

```
# for restaurant guest

def submit1():
    global bookingid
    global bookingidlabel
    global bookingidentry
    global costlabel
    global costlabel1
    global bill



    bookingid=StringVar()
    a=time.localtime()
    hours=a[3]
    price_breakfast=
200*(vegadultno.get())+100*(vegchildno.get())+250*(nonvegadultno.get())+150*(nonvegchildn
o.get())
    price_lunch=
250*(vegadultno.get())+150*(vegchildno.get())+325*(nonvegadultno.get())+200*(nonvegchildn
o.get())
    price_dinner=
275*(vegadultno.get())+200*(vegchildno.get())+350*(nonvegadultno.get())+250*(nonvegchildn
o.get())
    if meal.get()=='Breakfast':
        price=price_breakfast
    if meal.get()=='Lunch':
        price=price_lunch
    if meal.get()=='Dinner':
```

```
        price=price_dinner
    if price!=0:
        a=time.localtime()
        hours=a[3]
        mins=a[4]
        sec=a[5]
        ctime=datetime.time(hours,mins,sec)
        cdate=datetime.date.today()
        ctime=str(ctime)
        cdate=str(cdate)
        cost= price+ (5/100)*price



#cancelbutton=Button(Restaurant_screen,text='CANCEL',fg='darkgoldenrod1',bg='firebrick2',fo
nt = 'bahnschrift 15 bold',padx=20,command=cancel)
        #cancelbutton.place(x=250,y=540)

costlabel=Label(Restaurant_screen,text="COST",fg='darkgoldenrod1',bg='firebrick2',font =
'bahnschrift 12 bold',padx=20)
        costlabel.place(x=350,y=550)



costlabel1=Label(Restaurant_screen,text=str(cost),fg='darkgoldenrod1',bg='firebrick2',font =
'bahnschrift 12 bold',padx=20)
        costlabel1.place(x=450,y=550)
        submitbutton.configure(state=DISABLED)
        resetbutton.configure(state=DISABLED)
        previousbutton.configure(state=DISABLED)
        response=messagebox.askokcancel("Restaurant","Click OK to confirm your
order\n\nCOST(Tax included) :" +str(price+ ((5/100)*price)))
```

```
        if response == 1:

            bookingidlabel=Label(Restaurant_screen,text="BOOKING
ID",fg='darkgoldenrod1',bg='firebrick2',font = 'bahnschrift 12 bold',padx=20)
            bookingidlabel.place(x=10,y=550)

            bookingidentry=Entry(Restaurant_screen,textvariable= bookingid,font = 'bahnschrift
12 bold')
            bookingidentry.place (x=170,y=550)

            generate_id()

            bookingidentry.configure(state= DISABLED)

            statement=  'insert into restaurant (BOOKING_ID,CUSTOMER_TYPE,
ADULTS_VEG,CHILDREN_VEG,ADULTS_NONVEG,CHILDREN_NONVEG,TOTAL,DA
TE,TIME) values('+str(bookingid.get())+',"'+ type1
+'","'+str(vegadultno.get())+','+str(vegchildno.get())+','+str(nonvegadultno.get())+','+str(nonvegch
ildno.get())+','+str(cost)+',"'+cdate+'","'+ctime+'")'
            cursor.execute(statement)
            connector.commit()
            messagebox.showinfo("Restaurant",message="Entry successfully
added\nBookingID:"+ str(bookingid.get()))
#-----------------------------------------------------------------billscreen-----------------------------------
---------------------------
            Restaurant_screen.geometry("1250x600")
            bill=Text( Restaurant_screen, height = 28, width = 60, bg = 'white',fg='black', font =
'bahnschrift 15 bold ',relief = SUNKEN)
            bill.insert(INSERT,"\t\t\tHOTEL NAME\n\n"     )
            bill.insert(INSERT,"DATE : "+ cdate+"\n"    )
```

```
            bill.insert(INSERT,"TIME : "+ ctime+ "\n\n"  )
            bill.insert(INSERT,"  ----------------------------------------------------------------------------------
-------- \n")
            bill.insert(INSERT,"\t\t\tRESTAURANT\n\n\n")
            bill.insert(INSERT,"Booking ID : "+str(bookingid.get())+"\n\n")
            bill.insert(INSERT,"S.NO\tPRODUCT\t\tQUANTITY\t\tCOST\n\n")
            if price==price_breakfast:
                if vegadultno.get()!=0:

bill.insert(INSERT,"1\tVEG(ADULT)\t\t"+str(vegadultno.get())+"\t\t"+str(int(vegadultno.get())*
200)+'\n')
                if vegchildno.get()!=0:

bill.insert(INSERT,"2\tVEG(CHILD)\t\t"+str(vegchildno.get())+"\t\t"+str(int(vegchildno.get())*
100)+'\n')
                if nonvegadultno.get()!=0:
                    bill.insert(INSERT,"3\tNON-
VEG(ADULT)\t\t"+str(nonvegadultno.get())+"\t\t"+str(int(nonvegadultno.get())*250)+'\n')
                if nonvegchildno.get()!=0:
                    bill.insert(INSERT,"4\tNON-
VEG(CHILD)\t\t"+str(nonvegchildno.get())+"\t\t"+str(int(nonvegchildno.get())*150)+'\n')
            if price==price_lunch:
                if vegadultno.get()!=0:

bill.insert(INSERT,"1\tVEG(ADULT)\t\t"+str(vegadultno.get())+"\t\t"+str(int(vegadultno.get())*
250)+'\n')
                if vegchildno.get()!=0:

bill.insert(INSERT,"2\tVEG(CHILD)\t\t"+str(vegchildno.get())+"\t\t"+str(int(vegchildno.get())*
150)+'\n')
                if nonvegadultno.get()!=0:
```

```
                bill.insert(INSERT,"3\tNON-
VEG(ADULT)\t\t"+str(nonvegadultno.get())+"\t\t"+str(int(nonvegadultno.get())*325)+'\n')
            if nonvegchildno.get()!=0:
                bill.insert(INSERT,"4\tNON-
VEG(CHILD)\t\t"+str(nonvegchildno.get())+"\t\t"+str(int(nonvegchildno.get())*200)+'\n')


        if price==price_dinner:
            if vegadultno.get()!=0:

bill.insert(INSERT,"1\tVEG(ADULT)\t\t"+str(vegadultno.get())+"\t\t"+str(int(vegadultno.get())*
275)+'\n')
            if vegchildno.get()!=0:

bill.insert(INSERT,"2\tVEG(CHILD)\t\t"+str(vegchildno.get())+"\t\t"+str(int(vegchildno.get())*
200)+'\n')
            if nonvegadultno.get()!=0:
                bill.insert(INSERT,"3\tNON-
VEG(ADULT)\t\t"+str(nonvegadultno.get())+"\t\t"+str(int(nonvegadultno.get())*350)+'\n')
            if nonvegchildno.get()!=0:
                bill.insert(INSERT,"4\tNON-
VEG(CHILD)\t\t"+str(nonvegchildno.get())+"\t\t"+str(int(nonvegchildno.get())*250)+'\n')
        bill.insert(INSERT,"\t\t\t\t\t-----------\n")
        bill.insert(INSERT,"\t\t\tTOTAL\t\t" +str(price)+"\n")
        bill.insert(INSERT,"\t\t\t\t\t-----------\n")
        bill.insert(INSERT,"\tTAX = 5%(CGST+SGST)\n")
        bill.insert(INSERT,"\t\tCOST:\t\t\t" +str(((5/100)*price )+ price)+"\n\n")
        bill.insert(INSERT,"\t\tHOPE YOU ENJOYED OUR SERVICE!")
        bill.place(x=600,y=0)


        #saving the bill in a notepad
        billcontent=bill.get(1.0,END)
```

```
            billfile=open(str(os.getcwd())+'/Restaurant bill
folder'+"/"+str(bookingid.get())+".txt","w")
            billfile.write(billcontent)
            billfile.close()

            resetbutton.configure(state=ACTIVE)
            previousbutton.configure(state=ACTIVE)
#-------------------------------------------------------------------------------------------------------------------------
---------------------------
        else:
            costlabel.destroy()
            costlabel1.destroy()
            submitbutton.configure(state=ACTIVE)
            resetbutton.configure(state=ACTIVE)
            previousbutton.configure(state=ACTIVE)
    else:

        messagebox.showerror('Restaurant',"Enter proper credentials")

  def reset1():

    vegadultno.set(0)
    nonvegadultno.set(0)
    vegchildno.set(0)
    nonvegchildno.set(0)
    try:
        bill.destroy()
        bookingidlabel.destroy()
        bookingidentry.destroy()
```

```
        costlabel.destroy()
        costlabel1.destroy()
        Restaurant_screen.geometry('600x600')
        submitbutton.configure(state=ACTIVE)

    except:
        pass

def previous1():

    confirm_button.config(state=ACTIVE)
    mealdropdown.config(state=ACTIVE)

    vegadultlabel.destroy()
    vegadultoption.destroy()

    vegchildlabel.destroy()
    vegchildoption.destroy()

    nonvegadultlabel.destroy()
    nonvegadultoption.destroy()

    nonvegchildlabel.destroy()
    nonvegchildoption.destroy()

    submitbutton.destroy()
    resetbutton.destroy()
    previousbutton.destroy()
```

```
    try:
        bookingidlabel.destroy()
        bookingidentry.destroy()
        costlabel.destroy()
        costlabel1.destroy()
        bill.destroy()
        Restaurant_screen.geometry('600x600')
    except:
        pass

r=StringVar()

def Restaurantguest():
    global vegadultno
    global vegchildno
    global nonvegadultno
    global nonvegchildno

    global vegadultlabel
    global vegchildlabel
    global nonvegadultlabel
    global nonvegchildlabel



    global vegadultoption
    global vegchildoption
    global nonvegadultoption
    global nonvegchildoption
```

```
    global previousbutton
    global resetbutton
    global submitbutton
    vegadultno=IntVar()

    vegchildno=IntVar()

    nonvegadultno=IntVar()

    nonvegchildno=IntVar()

    confirm_button.config(state=DISABLED)
    mealdropdown.config(state=DISABLED)


vegadultlabel=Label(Restaurant_screen,text="VEG(Adults)",fg='darkgoldenrod1',bg='firebrick2'
,font = 'bahnschrift 15 bold',padx=20)
    vegadultlabel.place(x=110,y=250)

    vegadultoption=OptionMenu(Restaurant_screen,vegadultno,*[i for i in range(0,21)])
    vegadultoption.place(x=400,y=250)



vegchildlabel=Label(Restaurant_screen,text="VEG(Kids)",fg='darkgoldenrod1',bg='firebrick2',f
ont = 'bahnschrift 15 bold',padx=20)
    vegchildlabel.place(x=110,y=300)

    vegchildoption=OptionMenu(Restaurant_screen,vegchildno,*[i for i in range(0,21)])
```

```
vegchildoption.place(x=400,y=300)



nonvegadultlabel=Label(Restaurant_screen,text="NONVEG(Adults)",fg='darkgoldenrod1',bg='f
irebrick2',font = 'bahnschrift 15 bold',padx=20)
nonvegadultlabel.place(x=110,y=350)


nonvegadultoption=OptionMenu(Restaurant_screen,nonvegadultno,*[i for i in range(0,21)])
nonvegadultoption.place(x=400,y=350)



nonvegchildlabel=Label(Restaurant_screen,text="NONVEG(Kids)",fg='darkgoldenrod1',bg='fir
ebrick2',font = 'bahnschrift 15 bold',padx=20)
nonvegchildlabel.place(x=110,y=400)


nonvegchildoption=OptionMenu(Restaurant_screen,nonvegchildno,*[i for i in range(0,21)])
nonvegchildoption.place(x=400,y=400)


#submit button


submitbutton=Button(Restaurant_screen,text='SUBMIT',fg='darkgoldenrod1',bg='firebrick2',font
='bahnschrift 15 bold',activebackground=
'firebrick2',activeforeground='darkgoldenrod1',padx=20,command=submit1)
submitbutton.place(x=100,y=460)


#reset button


resetbutton=Button(Restaurant_screen,text="RESET",fg='darkgoldenrod1',bg='firebrick2',font='
bahnschrift 15 bold',activebackground=
'firebrick2',activeforeground='darkgoldenrod1',padx=20,command=reset1)
resetbutton.place(x=220,y=460)
```

```
        #previous page


previousbutton=Button(Restaurant_screen,text="PREVIOUS",fg='darkgoldenrod1',bg='firebrick
2',font='bahnschrift 15 bold',activebackground=
'firebrick2',activeforeground='darkgoldenrod1',padx=20,command=previous1)
        previousbutton.place(x=340,y=460)


    #main screen
    meal=StringVar()
    meal.set("Breakfast")
    meallist=["Breakfast","Lunch","Dinner"]
    a=time.localtime()
    if a[3] not in[7,8,9,10,11,12,13,14,15,16,17,19,20,21,22,23,24]:
        messagebox.showerror("Restaurant","Restaurant service currently not
available\nBREAKFAST:(8 AM-12 AM)\n\nLUNCH:(12 PM-5 PM)\n\nDINNER:(7 PM-12
PM)")


    else:


        Restaurant_screen=Toplevel()
        Restaurant_screen.geometry("600x600")
        Restaurant_screen.config(bg='firebrick2')
        Restaurant_screen.title('Restaurant')
        Restaurant_screen.iconbitmap(str(os.getcwd())+'/icons/'+'buffet.ico')


        # Restaurant label

restaurant_label=Label(Restaurant_screen,text='Restaurant',fg='darkgoldenrod1',bg='firebrick2',f
ont = 'bahnschrift 25 bold',padx=20)
```

```
restaurant_label.place(x=210,y=10)

cust=Label(Restaurant_screen,text="Customer
Type",fg='darkgoldenrod1',bg='firebrick2',font = 'bahnschrift 15 bold',padx=20)

cust.place(x=40,y=110)

Radiobutton(Restaurant_screen,text="Hotel Guest",font = 'bahnschrift 13
bold',variable=r,value="Hotel
Guest",bg='firebrick2',fg='gold2',activeforeground='black',activebackground='red2').place(x=270
,y=110)

Radiobutton(Restaurant_screen,text="Restaurant Guest",font = 'bahnschrift 13
bold',variable=r,value="Restaurant
Guest",bg='firebrick2',fg='gold2',activeforeground='black',activebackground='red2').place(x=400
,y=110)


menubutton=Button(Restaurant_screen,text="MENU",fg='gold2',bg='firebrick2',font='bahnschri
ft 15 bold',padx=20,command=menu)

menubutton.place(x=480,y=40)


confirm_button=Button(Restaurant_screen,text='Confirm',fg='darkgoldenrod1',bg='firebrick2',ac
tivebackground= 'firebrick2',activeforeground='darkgoldenrod1',font = 'bahnschrift 12
bold',command=findcustomertype)

confirm_button.place(x=370,y=170)

mealdropdown=OptionMenu(Restaurant_screen,meal,*meallist)

mealdropdown.place(x=130,y=170)

mealdropdown.config(bg="firebrick2",font="bahnschrift 13
bold",fg='gold2',activebackground= 'firebrick2',activeforeground='darkgoldenrod1')
```

EMPLOYEES:

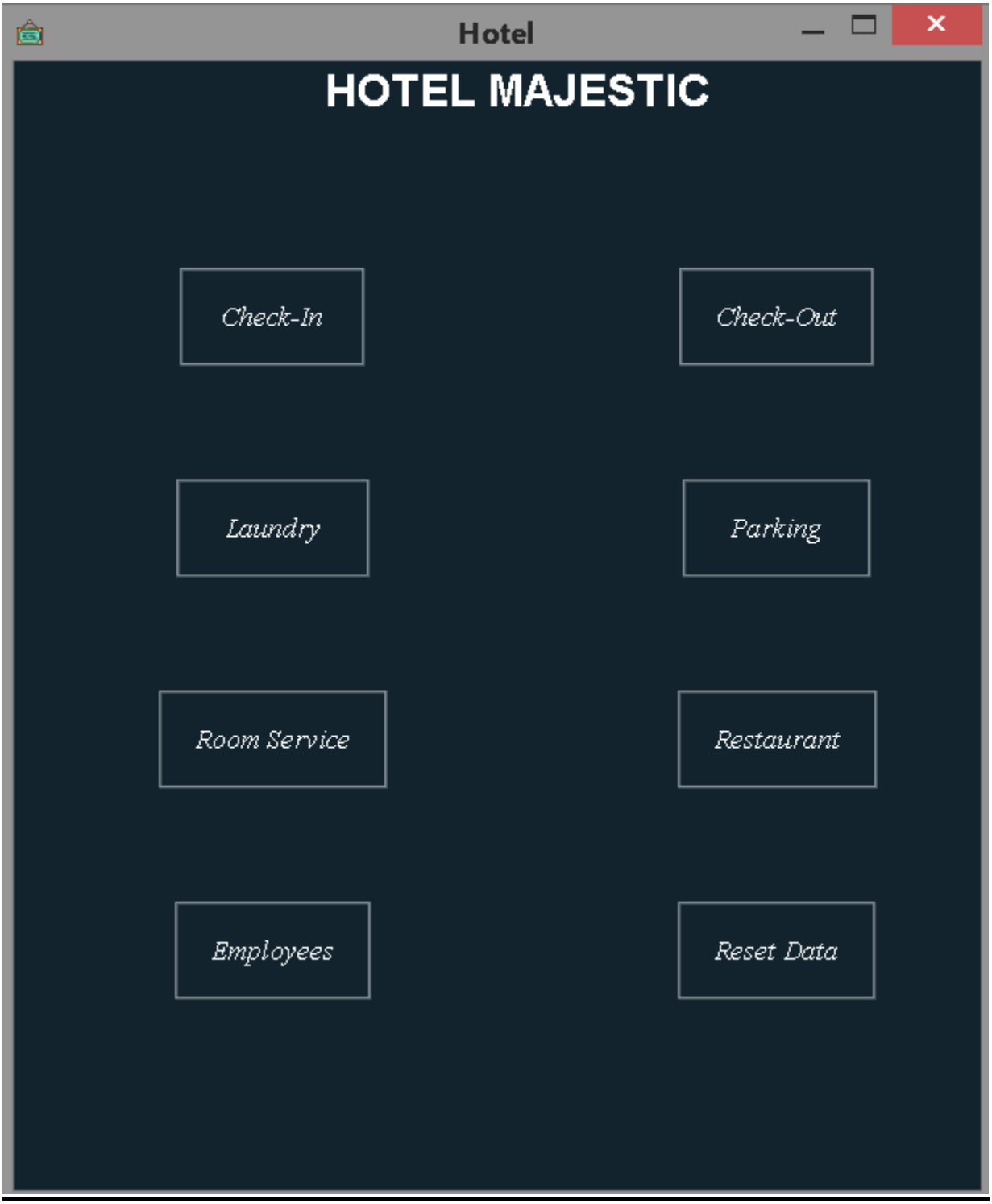
```
from tkinter import ttk
import tkinter as tk
import mysql.connector
import time
connector = mysql.connector.connect(host = 'localhost', password = 'akhash', user = 'root',
database = 'project')
cursor = connector.cursor()
def employees():
    window=tk.Toplevel()
    window.title("Employees")
    window.resizable(False,False)
    window.geometry('600x600')
    window.configure(bg="slategray1")

tk.Label(window,text="EMPLOYEE",font=("bahnschrift",20,"bold"),pady=20,bg="slategray1").
pack()
    treev=ttk.Treeview(window,show="headings",height="15")
    treev.pack()
    mystyle=ttk.Style()
    mystyle.theme_use("alt")
    mystyle.configure("Treeview",background="light
cyan",fieldbackground="silver",rowheight="25",foreground="white")
    mystyle.map("Treeview",background=[('selected', 'dodger blue')])
    treev["columns"]=("1","2","3","4","5")
    treev['show']='headings'
    treev.column("1", width=100, anchor='c')
    treev.column("2", width=100, anchor='c')
```

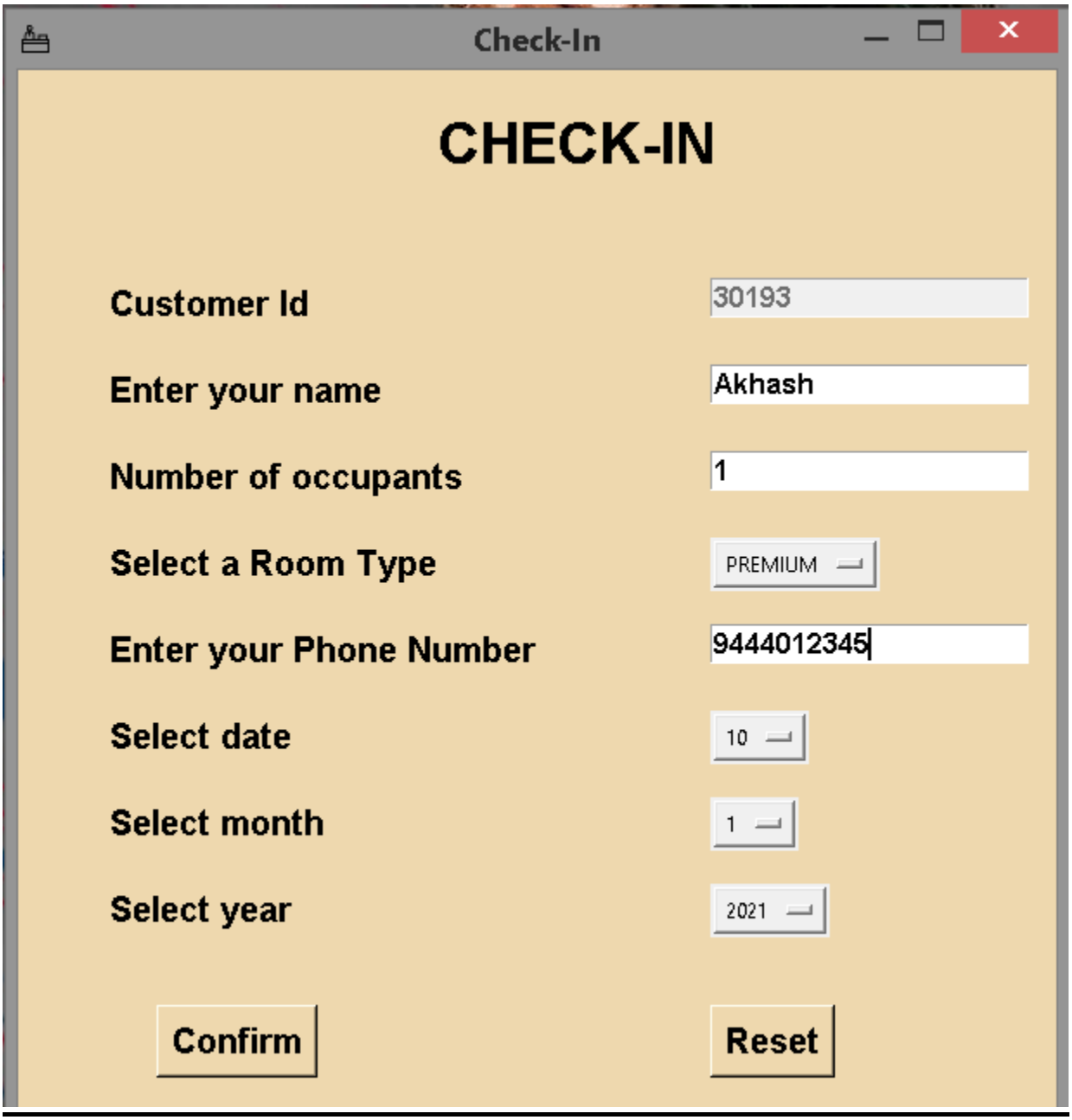
```
    treev.column("3", width=100, anchor='c')
    treev.column("4", width=100, anchor='c')
    treev.column("5", width=100,anchor='c')
    treev.heading("1", text="EMP_ID")
    treev.heading("2", text="EMP_NAME")
    treev.heading("3", text="DUTY")
    treev.heading("4", text="DATE OF JOIN")
    treev.heading("5", text="SHIFT")
    #sql queries
    a=time.localtime()
    hours=a[3]
    if hours in [8,9,10,11,12,13,14,15,16]:
        statement1='select EMP_ID,EMP_NAME,DUTY,DATE_OF_JOIN,SHIFT from
employees where SHIFT ="DAY";'
        cursor.execute(statement1)
    if hours in [17,18,19,20,21,22,23,24]:
        statement2='select EMP_ID,EMP_NAME,DUTY,DATE_OF_JOIN,SHIFT from
employees where SHIFT ="EVENING";'
        cursor.execute(statement2)
    if hours in [0,1,2,3,4,5,6,7]:
        statement3='select EMP_ID,EMP_NAME,DUTY,DATE_OF_JOIN,SHIFT from
employees where SHIFT ="NIGHT";'
        cursor.execute(statement3)
    details=cursor.fetchall()
    # adding details
    for j in details:
        treev.insert("", 'end', values=j)
```

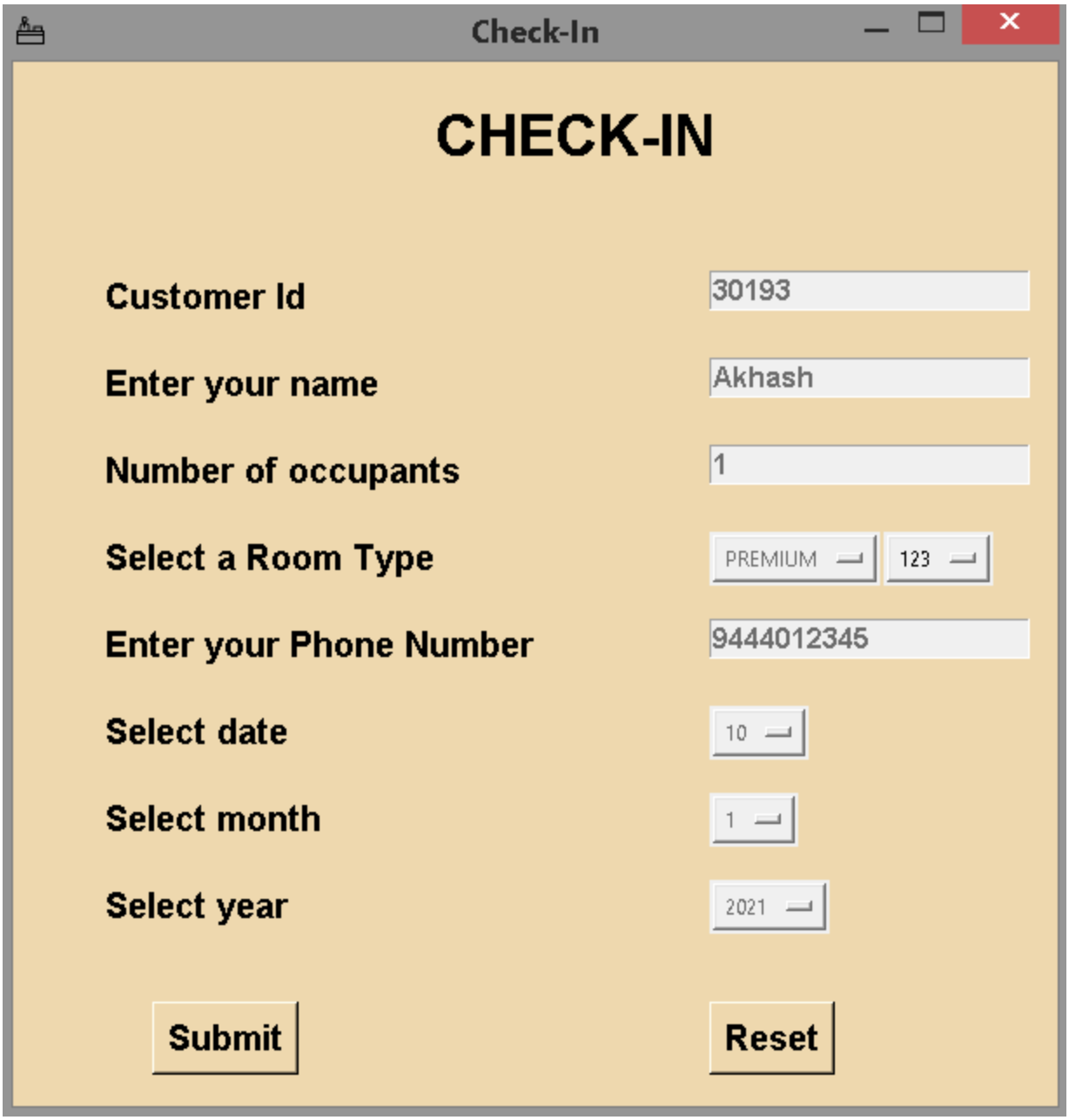
SAMPLE OUTPUT:

MAIN SCREEN:

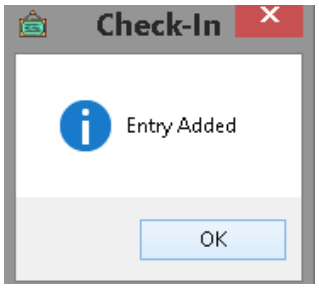


CHECKIN:

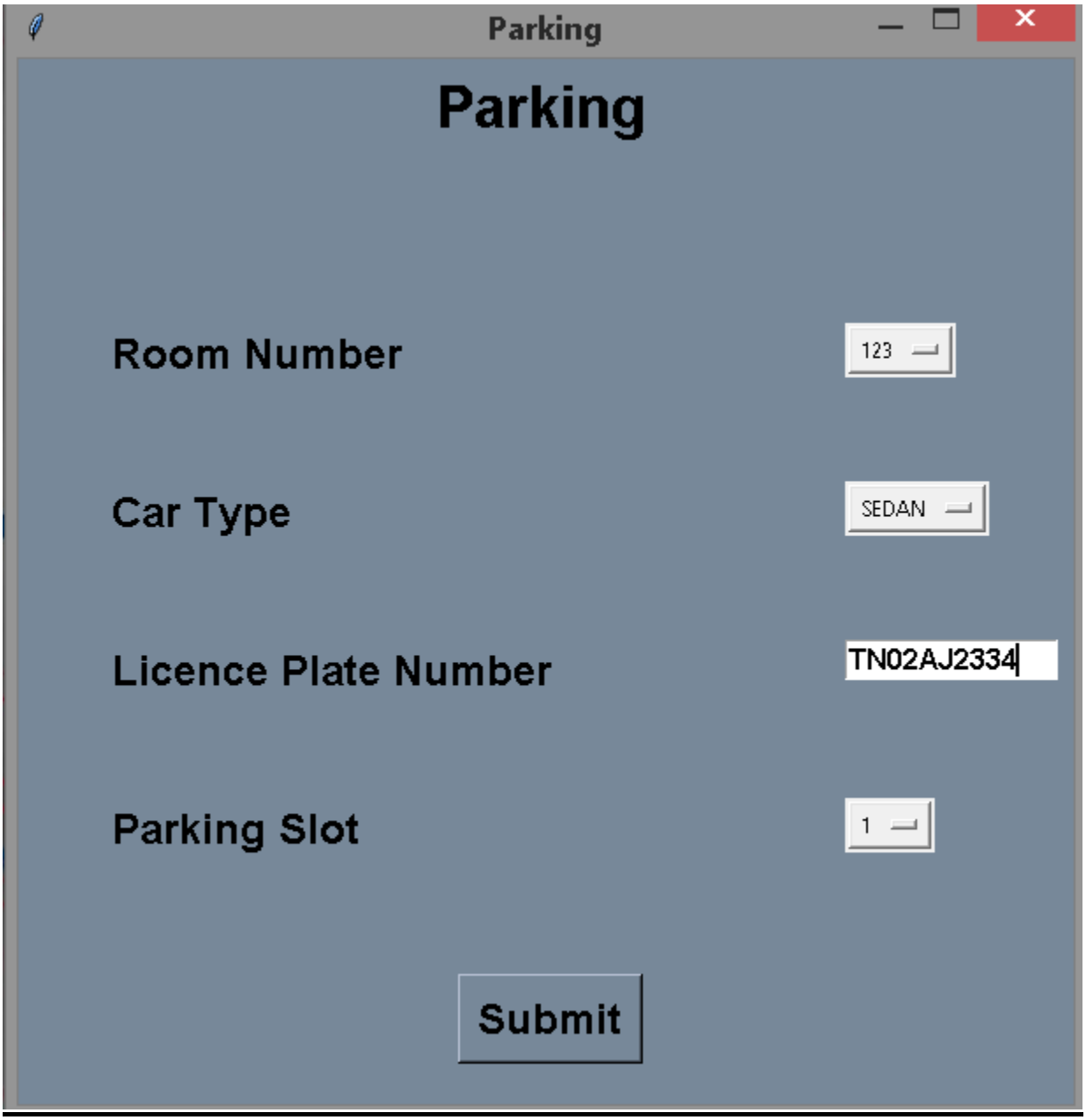

Check-In
CHECK-IN
Customer Id
30193
Enter your name
Akhash
Number of occupants
1
Select a Room Type
PREMIUM
Enter your Phone Number
9444012345
Select date
10
Select month
1
Select year
2021
Confirm
Reset



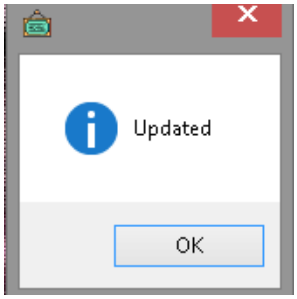
After pressing “SUBMIT”,



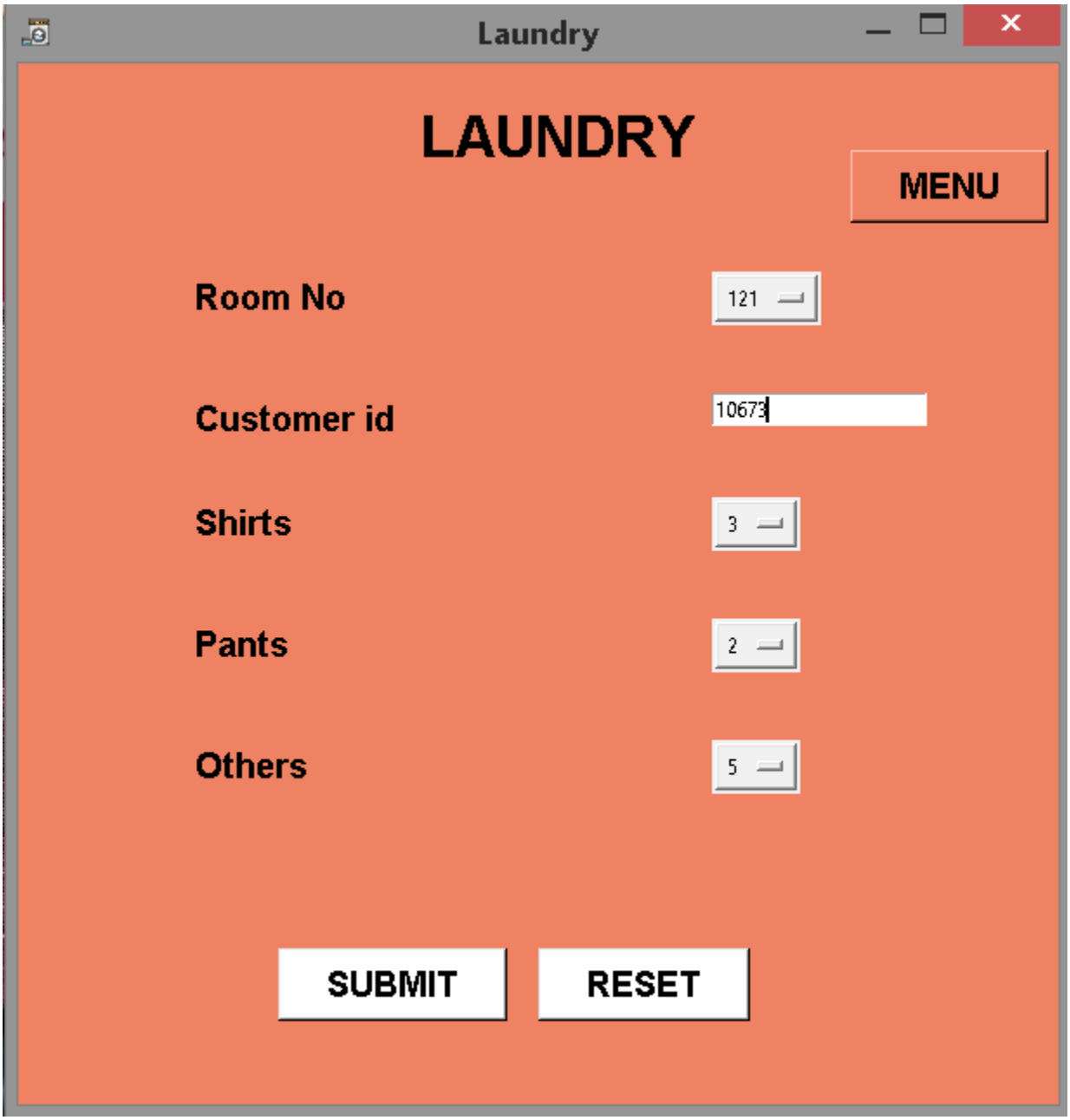
PARKING:



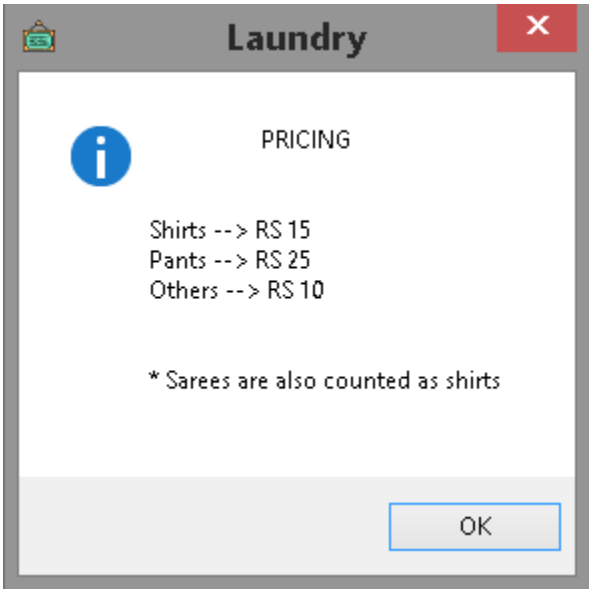
After pressing “SUBMIT”,



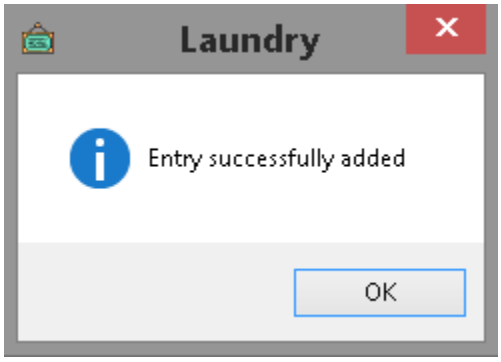
LAUNDRY:


Laundry
LAUNDRY
MENU
Room No
121
Customer id
10673
Shirts
3
Pants
2
Others
5
SUBMIT
RESET

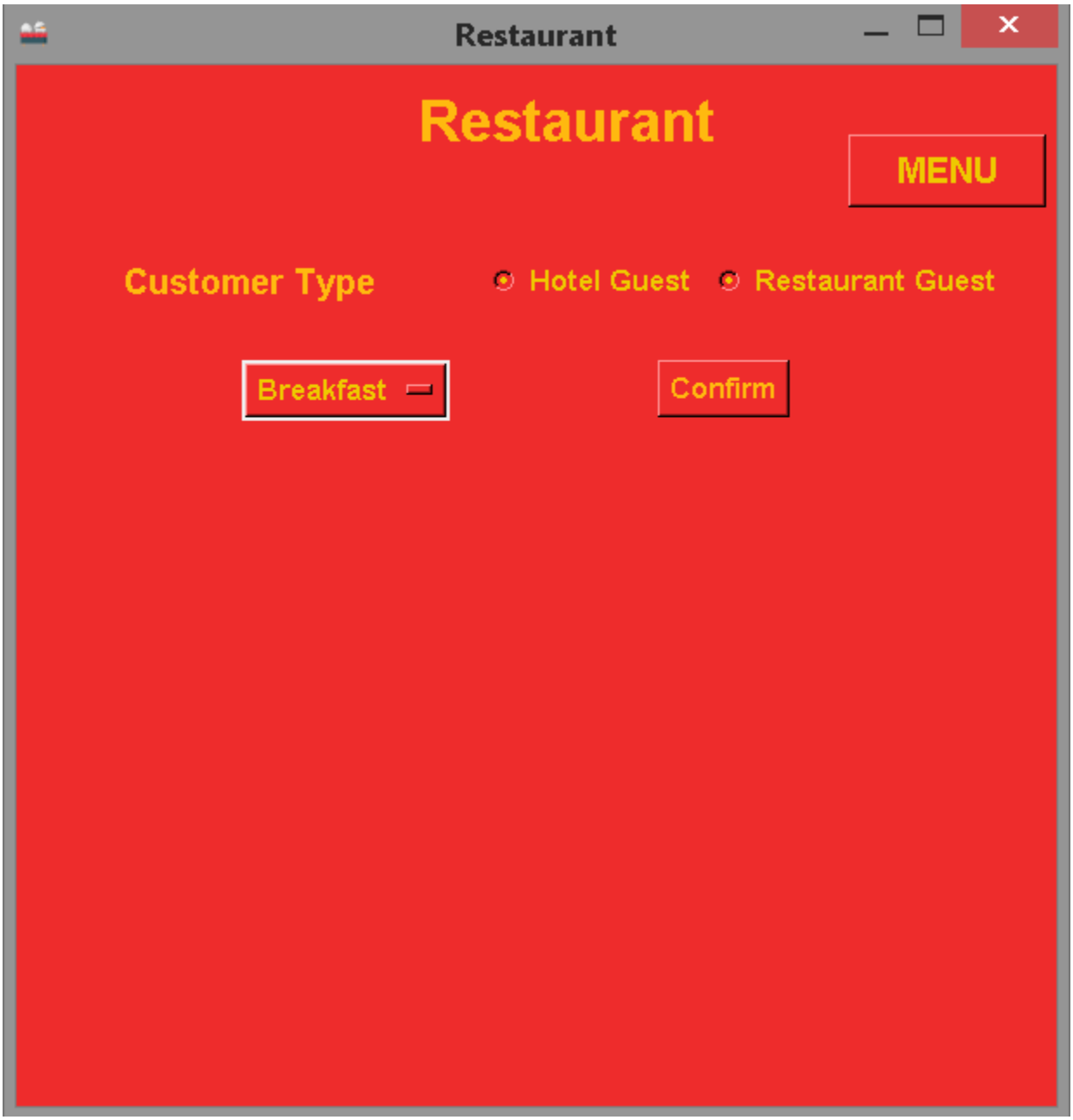
After pressing “MENU”,



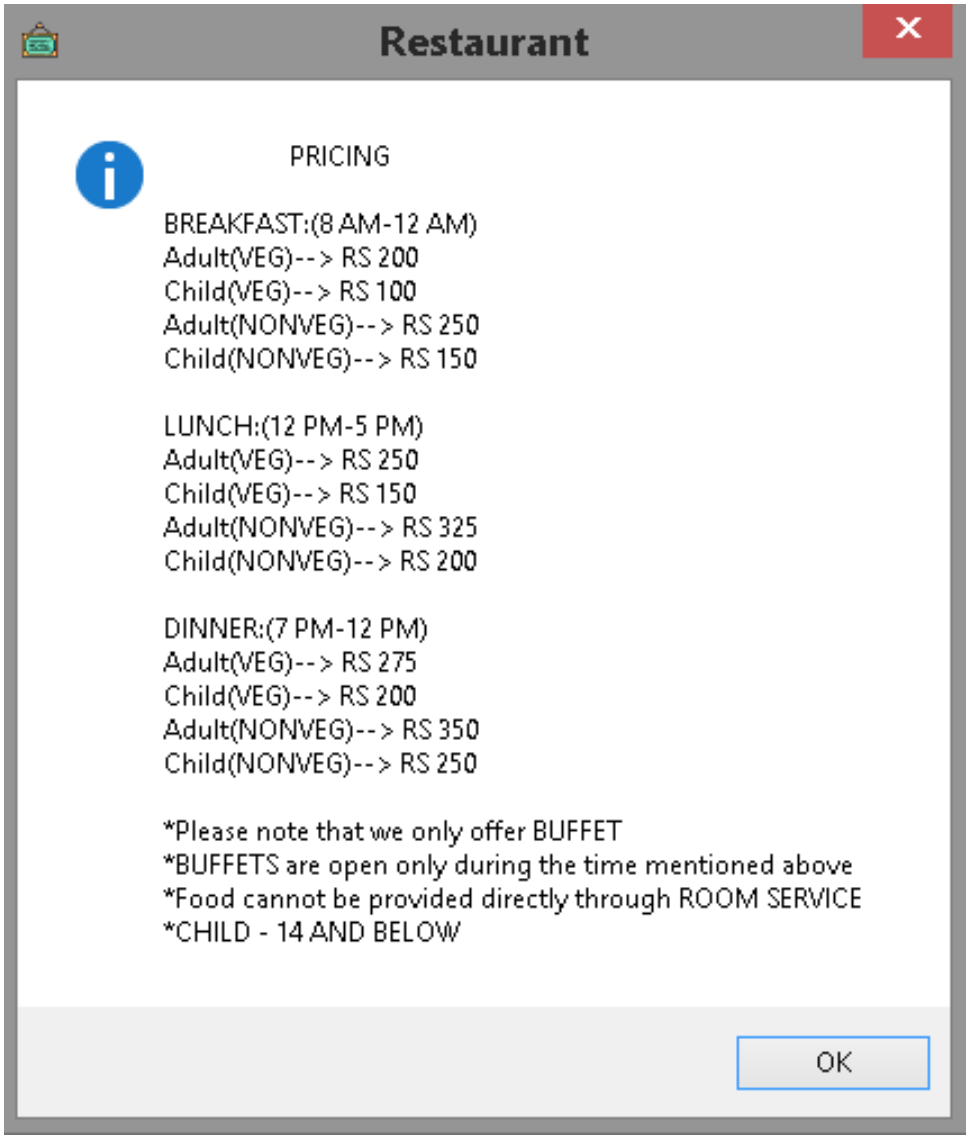
After pressing “SUBMIT”,



RESTAURANT:

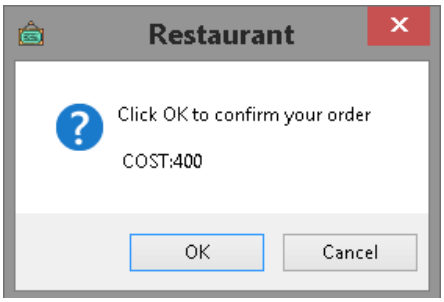

Restaurant
Restaurant
MENU
Customer Type
Hotel Guest
Restaurant Guest
Breakfast
Confirm

After pressing “MENU”,

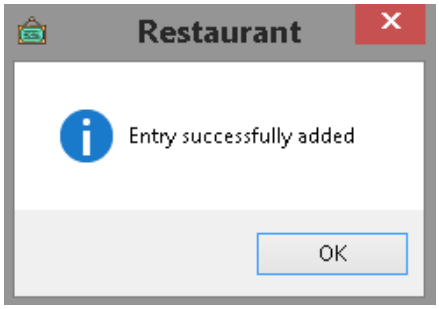


After selecting hotel guest and pressing “CONFIRM”,

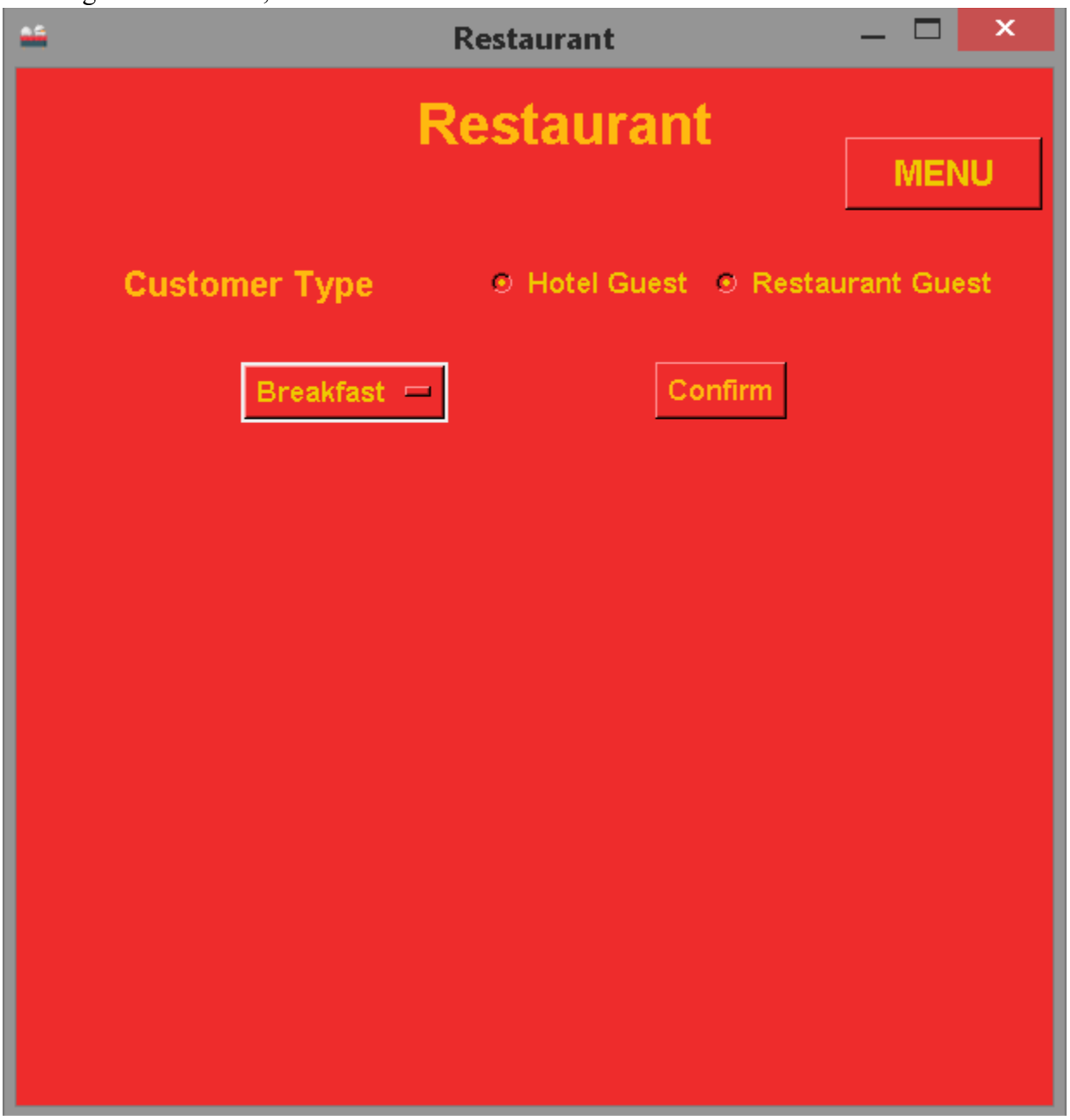
After pressing “SUBMIT”,



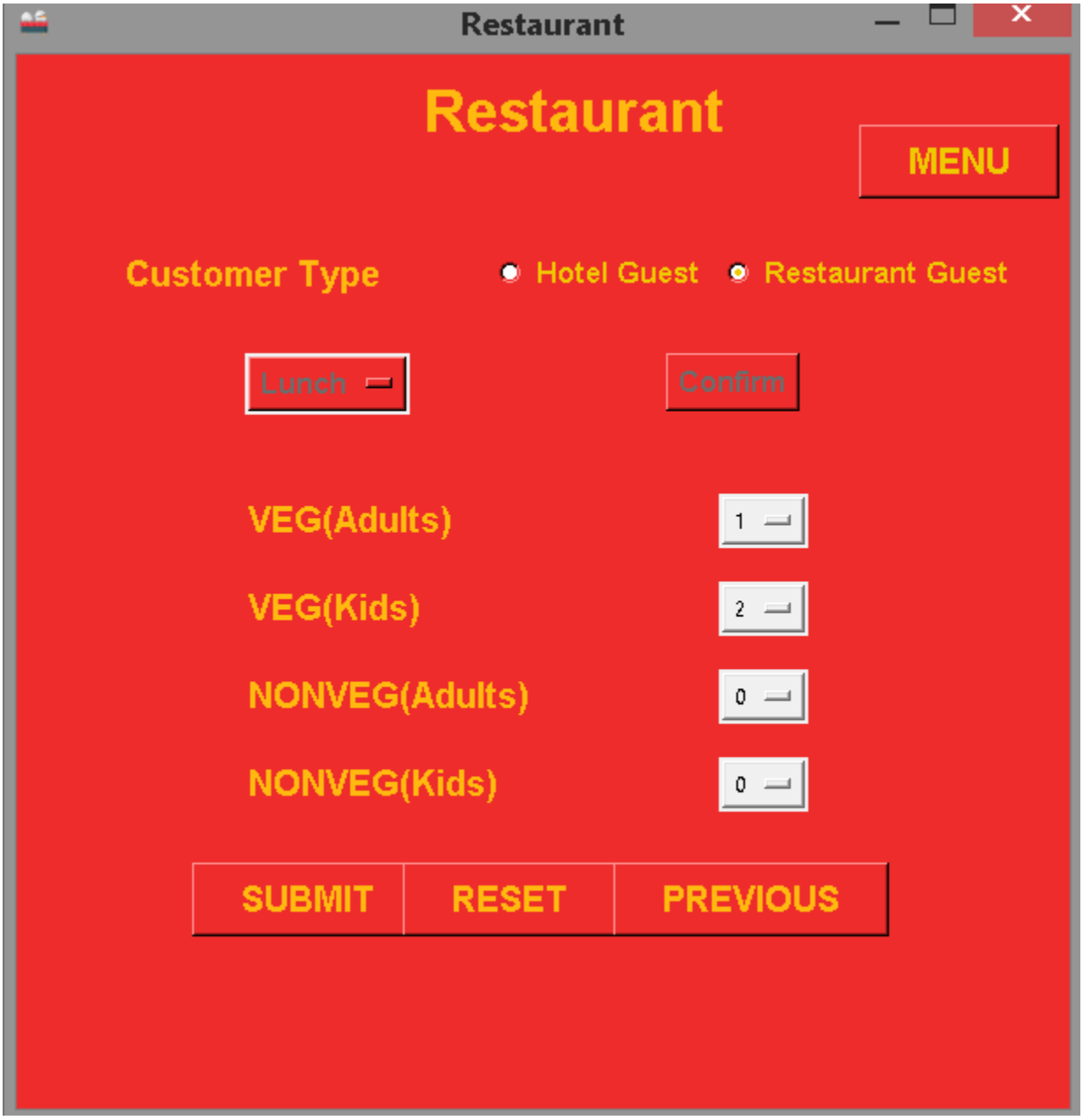
After pressing “OK”,



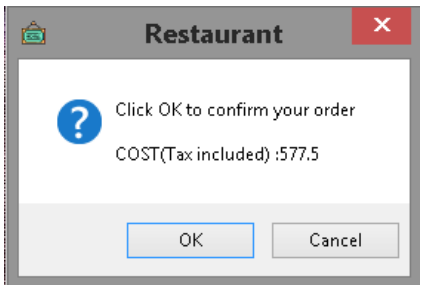
Pressing “PREVIOUS”,



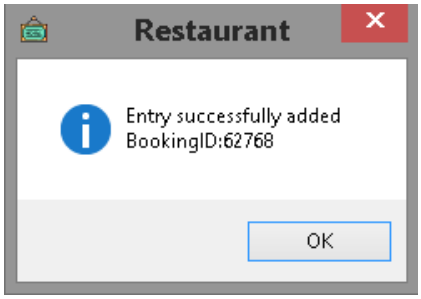
Pressing “CONFIRM” after selecting “Restaurant Guest”,



After pressing “SUBMIT”,

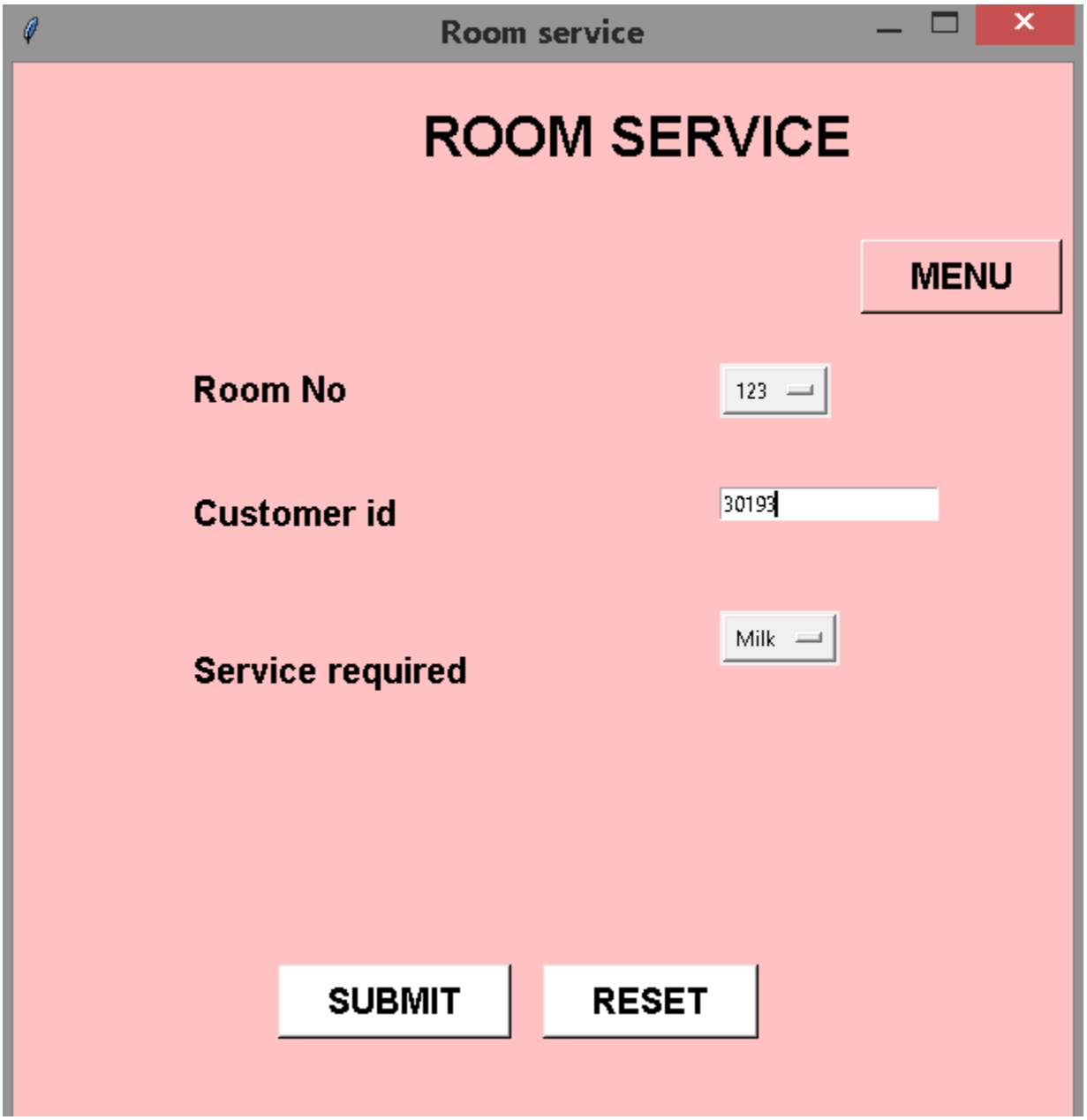


After pressing “OK”

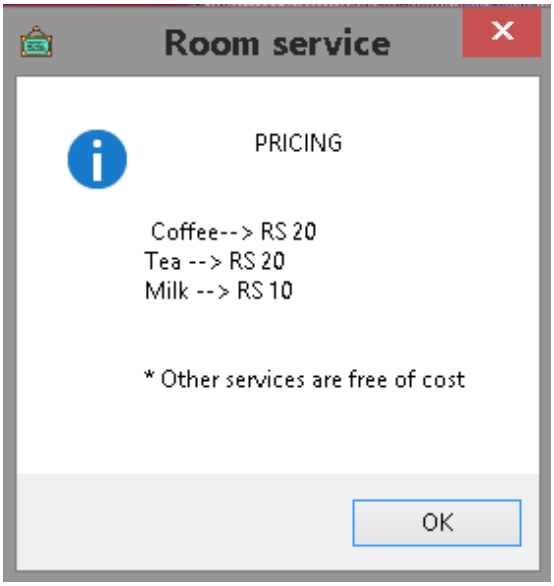


Generated bill:

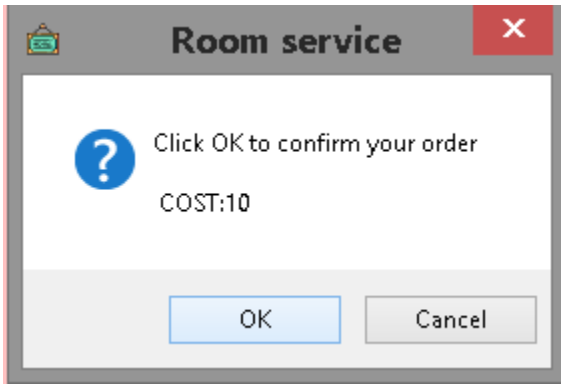
ROOM SERVICE:



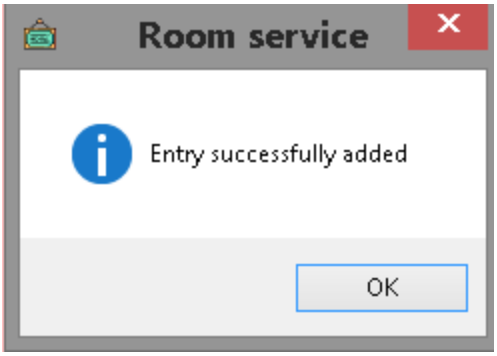
After pressing “MENU”,



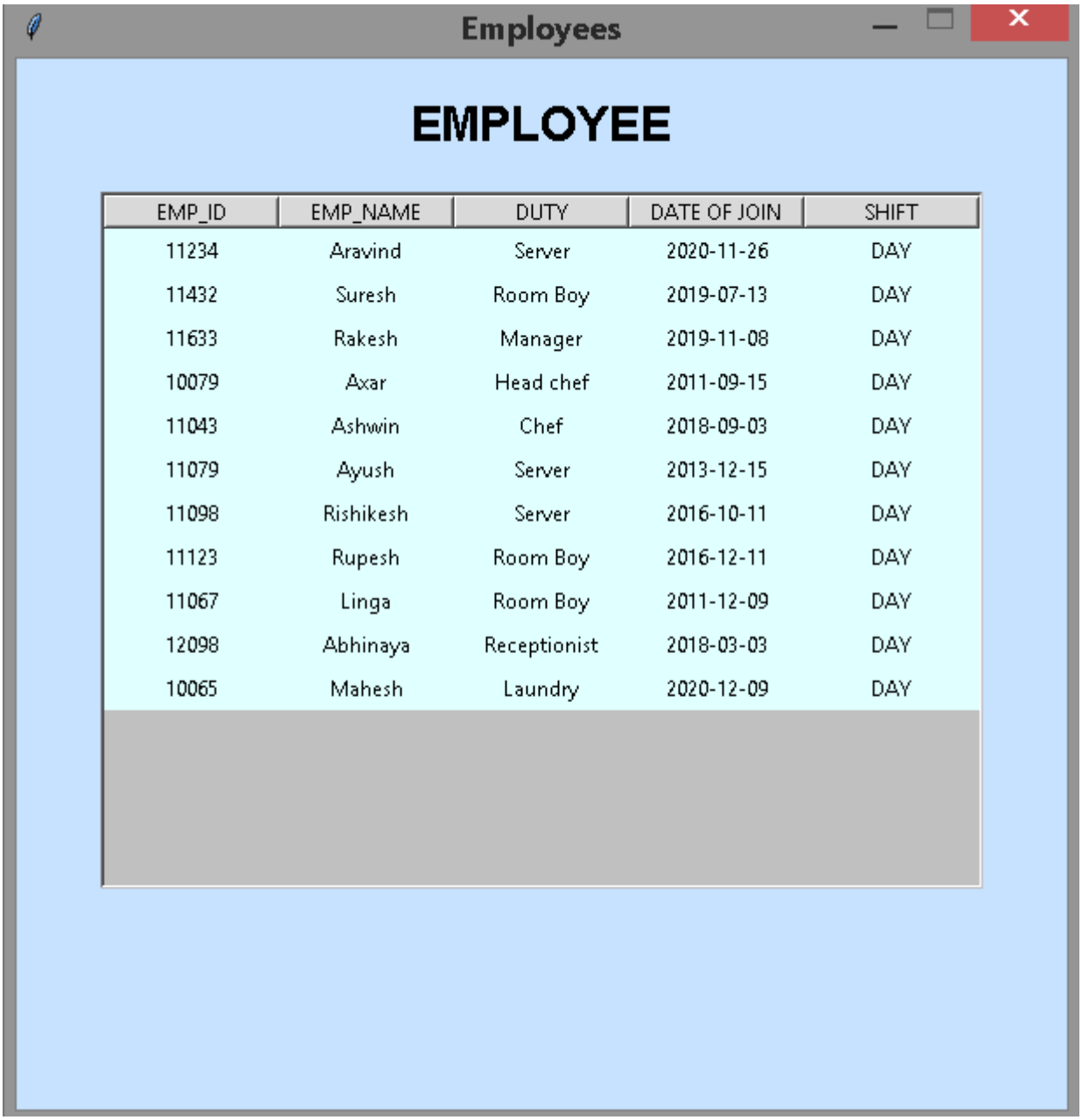
After pressing ”SUBMIT”,



After pressing “OK”,



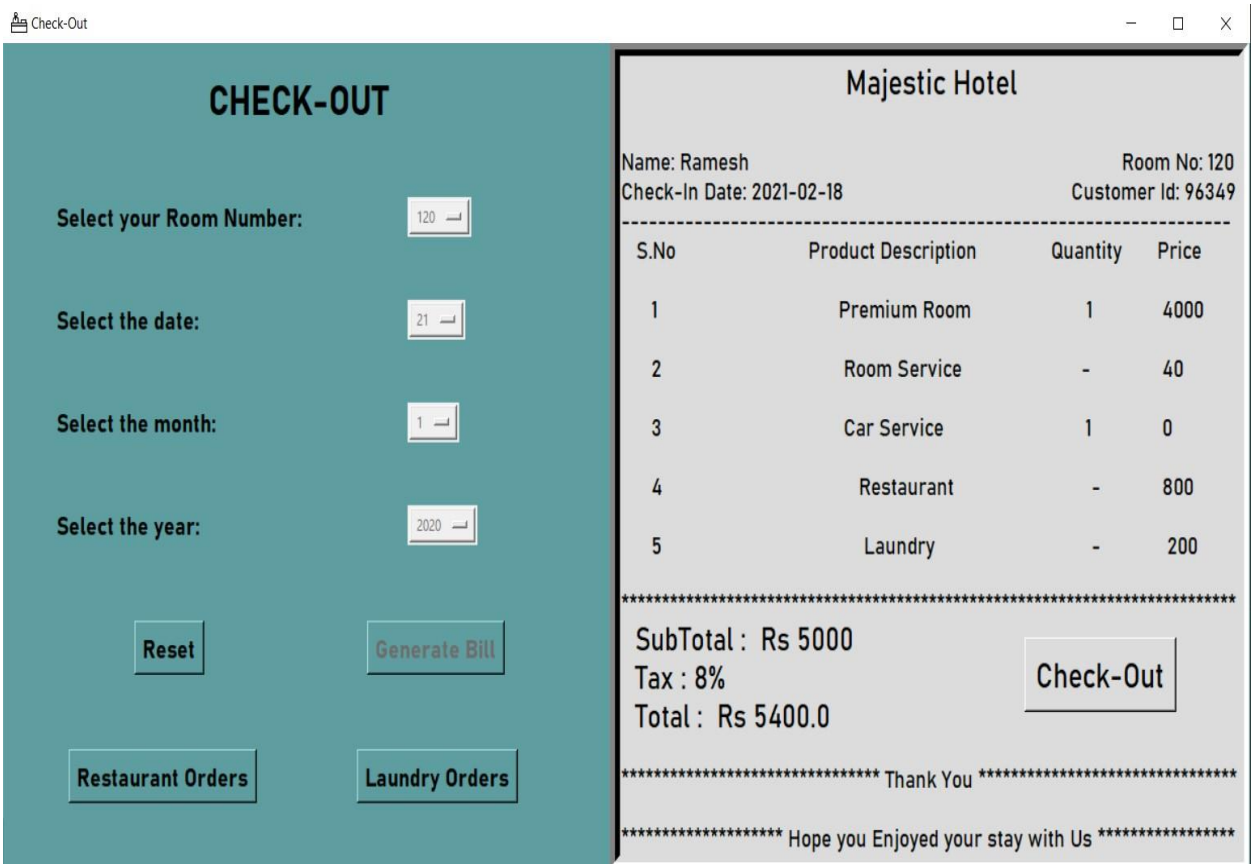
EMPLOYEES:



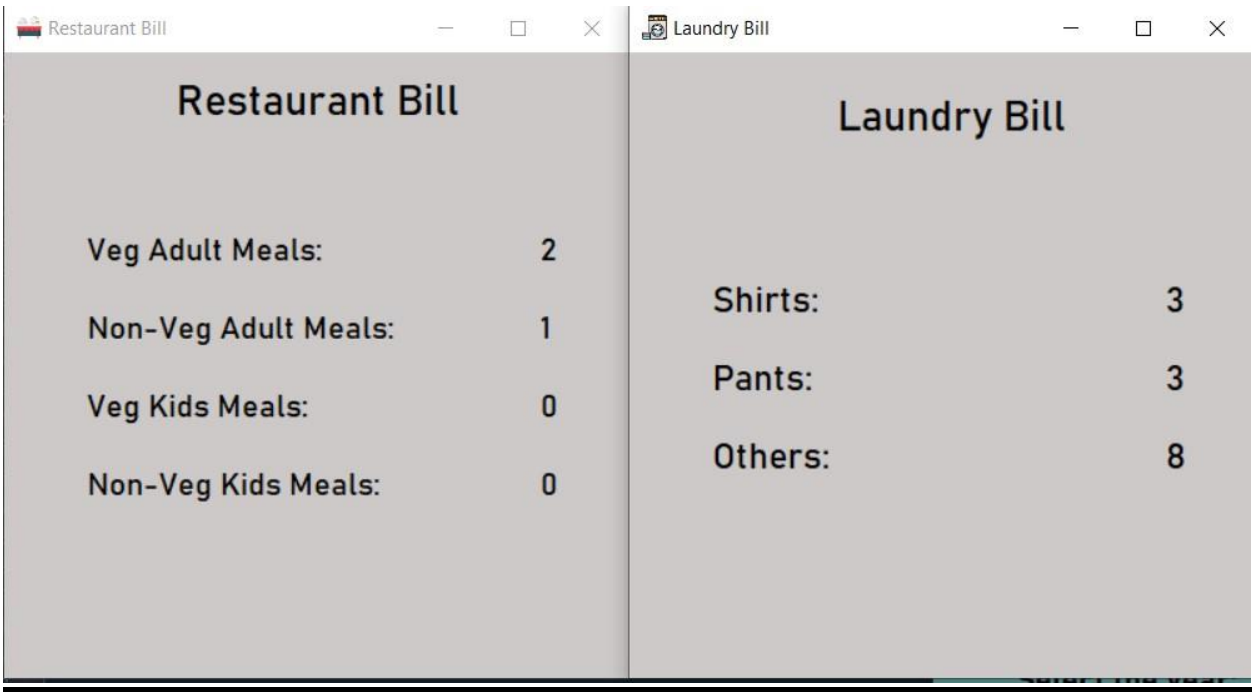
EMP_ID	EMP_NAME	DUTY	DATE OF JOIN	SHIFT
11234	Aravind	Server	2020-11-26	DAY
11432	Suresh	Room Boy	2019-07-13	DAY
11633	Rakesh	Manager	2019-11-08	DAY
10079	Axar	Head chef	2011-09-15	DAY
11043	Ashwin	Chef	2018-09-03	DAY
11079	Ayush	Server	2013-12-15	DAY
11098	Rishikesh	Server	2016-10-11	DAY
11123	Rupesh	Room Boy	2016-12-11	DAY
11067	Linga	Room Boy	2011-12-09	DAY
12098	Abhinaya	Receptionist	2018-03-03	DAY
10065	Mahesh	Laundry	2020-12-09	DAY

Employees belonging to the shift based on the time of the day is displayed in the screen.

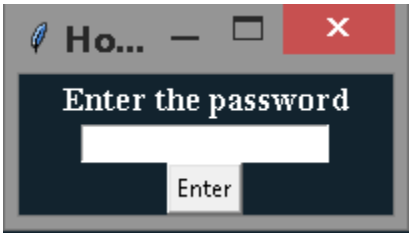
CHECKOUT:



Pressing Restaurant orders and Laundry orders,



RESET DATA:



If the correct password is entered, the entire data is erased.

SQL TABLES:

Tables present in the project database with example contents.

Tables_in_project
customer_history
customers
employees
laundry
laundry_history
parking
restaurant
roomservice

TABLE: customers

NAME	ROOM_NO	OCCUPANTS	CHECK_IN	CUSTOMER_ID	PH_NUMBER	STATUS	ROOM_TYPE
NULL	100	NULL	NULL	NULL	NULL	FREE	NULL
abc	101	1	2020-02-08	35267	273937584	OCCUPIED	NORMAL
NULL	102	NULL	NULL	NULL	NULL	FREE	NULL
NULL	103	NULL	NULL	NULL	NULL	FREE	NULL
NULL	104	NULL	NULL	NULL	NULL	FREE	NULL
NULL	105	NULL	NULL	NULL	NULL	FREE	NULL
NULL	106	NULL	NULL	NULL	NULL	FREE	NULL
NULL	107	NULL	NULL	NULL	NULL	FREE	NULL
NULL	108	NULL	NULL	NULL	NULL	FREE	NULL
NULL	109	NULL	NULL	NULL	NULL	FREE	NULL
NULL	110	NULL	NULL	NULL	NULL	FREE	NULL
NULL	111	NULL	NULL	NULL	NULL	FREE	NULL
NULL	112	NULL	NULL	NULL	NULL	FREE	NULL
NULL	113	NULL	NULL	NULL	NULL	FREE	NULL
NULL	114	NULL	NULL	NULL	NULL	FREE	NULL
NULL	115	NULL	NULL	NULL	NULL	FREE	NULL
NULL	116	NULL	NULL	NULL	NULL	FREE	NULL
NULL	117	NULL	NULL	NULL	NULL	FREE	NULL
NULL	118	NULL	NULL	NULL	NULL	FREE	NULL
NULL	119	NULL	NULL	NULL	NULL	FREE	NULL
NULL	120	NULL	NULL	NULL	NULL	FREE	NULL
a	121	1	2021-01-06	10673	9444709124	OCCUPIED	PREMIUM
NULL	122	NULL	NULL	NULL	NULL	FREE	NULL
Akhash	123	1	2021-01-10	30193	9444012345	OCCUPIED	PREMIUM
NULL	124	NULL	NULL	NULL	NULL	FREE	NULL
NULL	125	NULL	NULL	NULL	NULL	FREE	NULL

TABLE: parking

CAR_TYPE	L_PLATE	P_SLOT	STATUS	ROOM_NO
NULL	NULL	NULL	OCCUPIED	101
NULL	NULL	NULL	NULL	102
NULL	NULL	NULL	NULL	103
NULL	NULL	NULL	NULL	104
NULL	NULL	NULL	NULL	105
NULL	NULL	NULL	NULL	106
NULL	NULL	NULL	NULL	107
NULL	NULL	NULL	NULL	108
NULL	NULL	NULL	NULL	109
NULL	NULL	NULL	NULL	110
NULL	NULL	NULL	NULL	111
NULL	NULL	NULL	NULL	112
NULL	NULL	NULL	NULL	113
NULL	NULL	NULL	NULL	114
NULL	NULL	NULL	NULL	115
NULL	NULL	NULL	NULL	116
NULL	NULL	NULL	NULL	117
NULL	NULL	NULL	NULL	118
NULL	NULL	NULL	NULL	119
NULL	NULL	NULL	NULL	120
HATCHBACK	TN03AP...	5	OCCUPIED	121
NULL	NULL	NULL	NULL	122
SEDAN	TN02AJ...	1	OCCUPIED	123
NULL	NULL	NULL	NULL	124
NULL	NULL	NULL	NULL	125

TABLE: employees

EMP_ID	EMP_NAME	DUTY	DATE_OF_JOIN	SALARY	SHIFT
11234	Aravind	Server	2020-11-26	7500	DAY
11233	Shekar	Manager	2020-10-27	35000	EVENING
11322	Shubash	Room Boy	2019-07-23	5000	NIGHT
11432	Suresh	Room Boy	2019-07-13	5000	DAY
11071	Ramesh	Room Boy	2018-08-03	5000	EVENING
11633	Rakesh	Manager	2019-11-08	35000	DAY
10079	Axar	Head chef	2011-09-15	50000	DAY
11043	Ashwin	Chef	2018-09-03	25000	DAY
11079	Ayush	Server	2013-12-15	7500	DAY
11098	Rishikesh	Server	2016-10-11	7500	DAY
11123	Rupesh	Room Boy	2016-12-11	5000	DAY
11067	Linga	Room Boy	2011-12-09	5000	DAY
12098	Abhinaya	Receptio...	2018-03-03	30000	DAY
10065	Mahesh	Laundry	2020-12-09	5000	DAY

TABLE: laundry

ROOM_NO	SHIRTS	PANTS	OTHERS	TOTAL
100	0	0	0	0
101	0	0	0	0
102	0	0	0	0
103	0	0	0	0
104	0	0	0	0
105	0	0	0	0
106	0	0	0	0
107	0	0	0	0
108	0	0	0	0
109	0	0	0	0
110	0	0	0	0
111	0	0	0	0
112	0	0	0	0
113	0	0	0	0
114	0	0	0	0
115	0	0	0	0
116	0	0	0	0
117	0	0	0	0
118	0	0	0	0
119	0	0	0	0
120	0	0	0	0
121	6	4	10	290
122	0	0	0	0
123	0	0	0	0
124	0	0	0	0
125	0	0	0	0

TABLE: laundry_history

ROOM_NO	SHIRTS	PANTS	OTHERS	TOTAL	DATE	TIME	CUSTOMER_ID
100	3	1	0	70	2020-11-16	08:24:57	12345
100	6	1	7	185	2020-11-16	08:25:11	12345
100	3	0	0	45	2020-11-19	15:03:28	12345
100	1	1	0	40	2020-11-20	20:14:16	12345
100	2	0	0	30	2020-11-23	15:31:01	12345
100	3	0	0	45	2020-11-23	21:42:03	12345
100	3	0	13	175	2020-12-07	10:56:28	12345
100	3	0	1	55	2020-12-07	10:58:48	12345
100	2	0	1	40	2021-01-05	11:08:57	12345
121	3	2	5	145	2021-02-13	14:12:30	10673
121	3	2	5	145	2021-02-13	14:18:56	10673

TABLE: restaurant

BOOKING_ID	CUSTOMER_TYPE	CUSTOMER_ID	ADULTS_VEG	CHILDREN_VEG	ADULTS_NONVEG	CHILDREN_NONVEG	TOTAL	DATE	TIME
NULL	Hotel Guest	10673	1	0	0	0	200	2021-01-12	16:21:57
NULL	Hotel Guest	10673	1	2	0	0	400	2021-01-12	16:26:41
NULL	Hotel Guest	10673	1	2	0	0	430	2021-01-12	16:38:46
NULL	Hotel Guest	10673	1	0	0	0	200	2021-01-13	08:08:14
NULL	Hotel Guest	10673	1	0	0	0	200	2021-01-13	08:22:40
NULL	Hotel Guest	10673	1	2	0	0	400	2021-02-13	14:23:14
62768	Restaurant Guest	NULL	1	2	0	0	578	2021-02-13	14:23:48
59461	Restaurant Guest	NULL	1	2	0	0	420	2021-02-13	15:06:09

TABLE: roomservice

ROOM_NO	SERVICE	TOTAL	DATE	TIME
121	Milk	10	2021-03-04	11:25:08
121	Coffee	20	2021-03-04	11:25:42
123	Milk	10	2021-03-04	11:26:10
123	Tea	20	2021-03-04	11:26:27

Once customers checkout from the hotel, their details are added to customer_history.

TABLE: customer_history

NAME	ROOM_NO	OCCUPANTS	CHECK_IN	CHECK_OUT	CUSTOMER_ID	PH_NUMBER	DAYS_STAYED	TOTAL	ROOM_TYPE
abc	101	1	2020-02-08	2021-03-04	35267	273937584	390	975000	NORMAL
a	121	1	2021-01-06	2021-03-04	10673	9444709124	57	266998	PREMIUM
Akhash	123	1	2021-01-10	2021-03-05	30193	9444012345	54	244944	PREMIUM

BIBLIOGRAPHY

- ✓ **COMPUTER SCIENCE WITH PYTHON** for class 12 by Sumita Arora
- ✓ https://docs.python.org/3/library/tkinter.html
- ✓ https://codemy.com/intro-tkinter-python-gui-apps/
- ✓ https://www.tutorialspoint.com/python/python_gui_programming.htm