INDEX

RESORT MANAGEMENT SYSTEM

Project Statement:

- Objective :-

    - The primary objective of our application is to design a customer-admin interface for a resort – Resort Ivory Bliss, where customers can avail different services at the resort and at the same time the admin can manage the day-to-day activities.

- Types of portals :-

  We have a common login portal for both the customers and admin staff.

    - Customer UI :
        - Customers must log in with their valid log in credentials after which they are given access to various options.
        - Customers are given an option to book their stay(s) in the resort.
        - Customers can also view their previous / upcoming stay(s) and can also avail services for their current stay(s), if any.
        - The customers can check out an overview of the resort which gives them information on the various aspects of the resort.
        - The customers can also view their profile details and are given an option to edit them.

    - Admin UI :
        - Admin staff must log in with their valid log in credentials given by the authority after which they are given various options depending on their posts in the resort.
        - The various options available are :

i. Editing the rooms' details
ii. Editing the restaurant menu
iii. Editing the laundry menu
iv. Editing the amenities menu
v. Viewing the record log

The staff would be given selected choices from the above, according to their post.

- Admin staff can also view their profile details and are given an option to edit them.

➢ <u>User-defined Functions:</u>

- ❖ Log-in Function
  - to accept the login credentials and give access to the various options of the application.

- ❖ Sign-up Function
  - to accept various details and credentials to register/create a new account as a customer.

- ❖ Booking Function
  - to book stays in the resort.

- ❖ Stays Function
  - to view previous or upcoming stays and avail services for current stay(s) if any.

- ❖ Admin Functions
  - to edit rooms' info or info about the various services available in the resort.

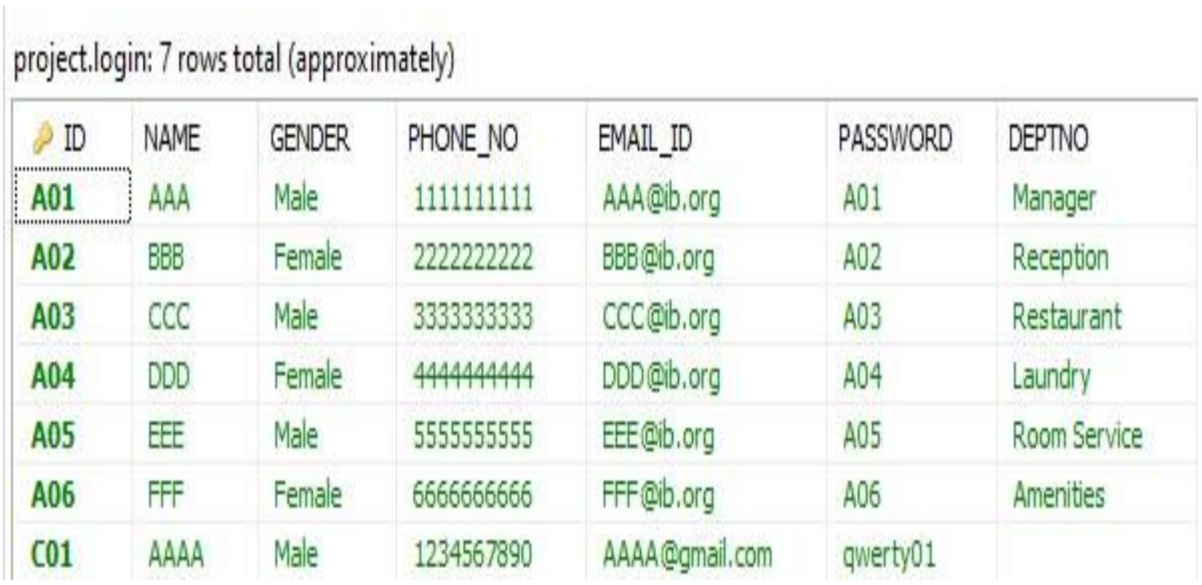
➢ <u>Tables Used</u>:

1. TABLE login :

    - Stores the login credentials of both the customers and admin staff.
    - Database : project
    - Creation Code :

```
CREATE TABLE login (
  ID VARCHAR(4) PRIMARY KEY,
  NAME VARCHAR(20),
  GENDER VARCHAR(7),
  PHONE_NO CHAR(10),
  EMAIL_ID VARCHAR(50),
  PASSWORD VARCHAR(20),
  DEPTNO VARCHAR(15) ) ;
```

    - Table Data:



project.login: 7 rows total (approximately)

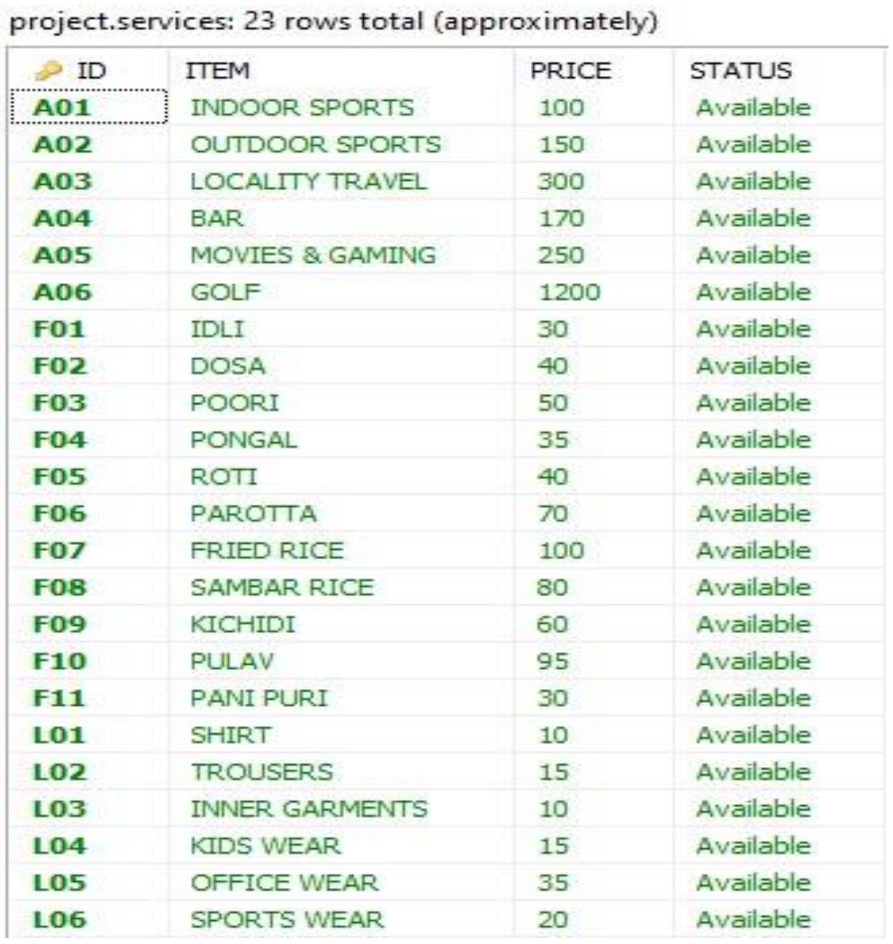
ID	NAME	GENDER	PHONE_NO	EMAIL_ID	PASSWORD	DEPTNO
A01	AAA	Male	1111111111	AAA@ib.org	A01	Manager
A02	BBB	Female	2222222222	BBB@ib.org	A02	Reception
A03	CCC	Male	3333333333	CCC@ib.org	A03	Restaurant
A04	DDD	Female	4444444444	DDD@ib.org	A04	Laundry
A05	EEE	Male	5555555555	EEE@ib.org	A05	Room Service
A06	FFF	Female	6666666666	FFF@ib.org	A06	Amenities
C01	AAAA	Male	1234567890	AAAA@gmail.com	qwerty01	

2. TABLE services :

- Stores details of the restaurant, laundry and amenities menus.
- Database : project
- Creation Code :

```
CREATE TABLE services(
  ID CHAR(4) PRIMARY KEY,
  ITEM VARCHAR(25),
  PRICE VARCHAR(4),
  STATUS VARCHAR(15) ) ;
```

- Table Data :



project.services: 23 rows total (approximately)

ID	ITEM	PRICE	STATUS
A01	INDOOR SPORTS	100	Available
A02	OUTDOOR SPORTS	150	Available
A03	LOCALITY TRAVEL	300	Available
A04	BAR	170	Available
A05	MOVIES & GAMING	250	Available
A06	GOLF	1200	Available
F01	IDLI	30	Available
F02	DOSA	40	Available
F03	POORI	50	Available
F04	PONGAL	35	Available
F05	ROTI	40	Available
F06	PAROTTA	70	Available
F07	FRIED RICE	100	Available
F08	SAMBAR RICE	80	Available
F09	KICHIDI	60	Available
F10	PULAV	95	Available
F11	PANI PURI	30	Available
L01	SHIRT	10	Available
L02	TROUSERS	15	Available
L03	INNER GARMENTS	10	Available
L04	KIDS WEAR	15	Available
L05	OFFICE WEAR	35	Available
L06	SPORTS WEAR	20	Available

3. TABLE room_info :

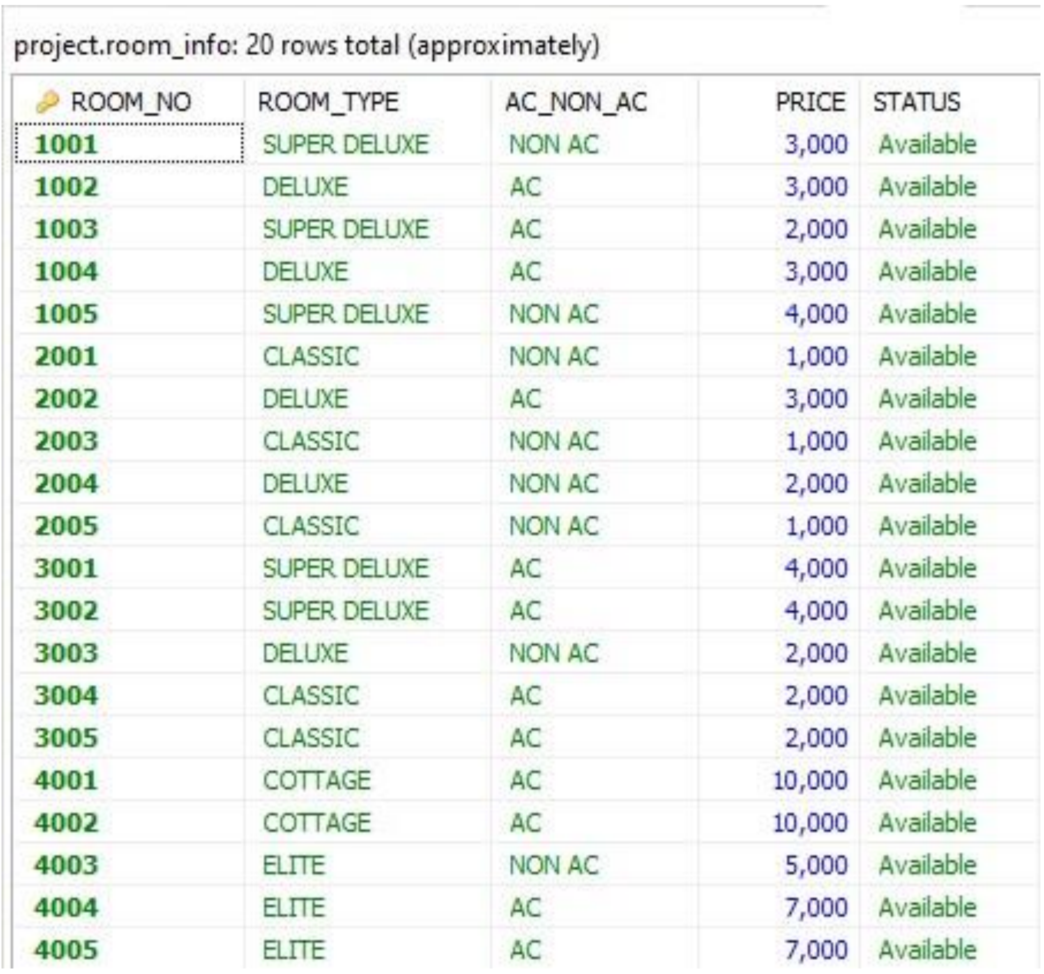
- Stores details of the rooms in the resort.

- Database : project

- Creation Code :

```
CREATE TABLE room_info(
  ROOM_NO CHAR(4) PRIMARY KEY,
  ROOM_TYPE VARCHAR(12),
  AC_NON_AC VARCHAR(6),
  PRICE INT(11),
 STATUS VARCHAR(15) ) ;
```

- Table Data :



project.room_info: 20 rows total (approximately)

ROOM_NO	ROOM_TYPE	AC_NON_AC	PRICE	STATUS
1001	SUPER DELUXE	NON AC	3,000	Available
1002	DELUXE	AC	3,000	Available
1003	SUPER DELUXE	AC	2,000	Available
1004	DELUXE	AC	3,000	Available
1005	SUPER DELUXE	NON AC	4,000	Available
2001	CLASSIC	NON AC	1,000	Available
2002	DELUXE	AC	3,000	Available
2003	CLASSIC	NON AC	1,000	Available
2004	DELUXE	NON AC	2,000	Available
2005	CLASSIC	NON AC	1,000	Available
3001	SUPER DELUXE	AC	4,000	Available
3002	SUPER DELUXE	AC	4,000	Available
3003	DELUXE	NON AC	2,000	Available
3004	CLASSIC	AC	2,000	Available
3005	CLASSIC	AC	2,000	Available
4001	COTTAGE	AC	10,000	Available
4002	COTTAGE	AC	10,000	Available
4003	ELITE	NON AC	5,000	Available
4004	ELITE	AC	7,000	Available
4005	ELITE	AC	7,000	Available

4. TABLE record_log :

- Stores details of all the bookings done in the resort.

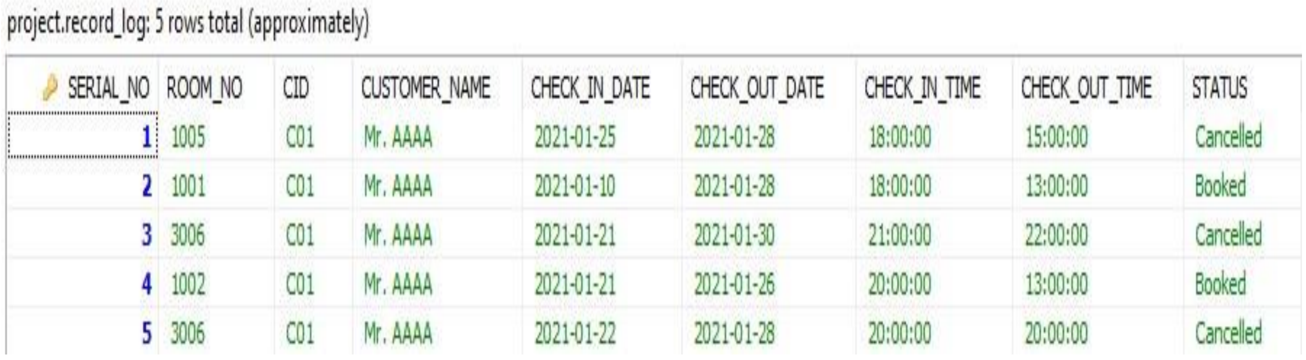
- Database : project

- Creation Code :

```
CREATE TABLE record_log(
  SERIAL_NO INT(11) PRIMARY KEY,
  ROOM_NO VARCHAR(4),
  CID VARCHAR(4),
  CUSTOMER_NAME VARCHAR(30),
  CHECK_IN_DATE VARCHAR(10),
  CHECK_OUT_DATE VARCHAR(10),
  CHECK_IN_TIME VARCHAR(10),
  CHECK_OUT_TIME VARCHAR(10),
  STATUS VARCHAR(10) ) ;
```

- Table Data :

For example, when there are some bookings



project.record_log: 5 rows total (approximately)

SERIAL_NO	ROOM_NO	CID	CUSTOMER_NAME	CHECK_IN_DATE	CHECK_OUT_DATE	CHECK_IN_TIME	CHECK_OUT_TIME	STATUS
1	1005	C01	Mr. AAAA	2021-01-25	2021-01-28	18:00:00	15:00:00	Cancelled
2	1001	C01	Mr. AAAA	2021-01-10	2021-01-28	18:00:00	13:00:00	Booked
3	3006	C01	Mr. AAAA	2021-01-21	2021-01-30	21:00:00	22:00:00	Cancelled
4	1002	C01	Mr. AAAA	2021-01-21	2021-01-26	20:00:00	13:00:00	Booked
5	3006	C01	Mr. AAAA	2021-01-22	2021-01-28	20:00:00	20:00:00	Cancelled

Source Code :

```
# Resort Management System


Name = Status = ID = Post = ''

import mysql.connector


import pyglet


from tkinter import *

from tkinter import messagebox

from tkcalendar import Calendar

from tkinter import ttk


from datetime import *


from PIL import Image, ImageTk

# ---------------------------------------------------------------------------------------------------------------

pyglet.font.add_file("D:\\pythonProject\\Century Gothic\\Century Gothic.ttf")

pyglet.font.add_file("D:\\pythonProject\\Century Gothic\\GOTHIC.TTF")

pyglet.font.add_file("D:\\pythonProject\\Century Gothic\\GOTHICB.TTF")

pyglet.font.add_file("D:\\pythonProject\\Century Gothic\\GOTHICB0.TTF")

pyglet.font.add_file("D:\\pythonProject\\Century Gothic\\GOTHICBI.TTF")
pyglet.font.add_file("D:\\pythonProject\\Century Gothic\\GOTHICI.TTF")

pyglet.font.add_file("D:\\pythonProject\\Bookman Old Style\\BKMNOS.ttf")

pyglet.font.add_file("D:\\pythonProject\\Bookman Old Style\\bookman old style
fett kursiv.ttf")

pyglet.font.add_file("D:\\pythonProject\\Bookman Old Style\\bookman old
style.ttf")

pyglet.font.add_file("D:\\pythonProject\\Bookman Old Style\\BOOKOS.TTF")

pyglet.font.add_file("D:\\pythonProject\\Bookman Old Style\\BOOKOSB.TTF")

pyglet.font.add_file("D:\\pythonProject\\Bookman Old Style\\BOOKOSBI.TTF")

pyglet.font.add_file("D:\\pythonProject\\Bookman Old Style\\BOOKOSI.TTF")

#------------------------------------------------------------------------------------------------------------

Account_Found_During_Login = Account_Found_During_Sign_Up = 2

# ----------------------------------------------------------------------------------------------------------

# Close Function

def close(page):

   page.destroy()

# ----------------------------------------------------------------------------------------------------------

# Login Page

def Login_Page_Func():

   global Account_Found_During_Sign_Up

   Login_Page_Func.Login_Page = Tk()


   sw = Login_Page_Func.Login_Page.winfo_screenwidth()

   sh = Login_Page_Func.Login_Page.winfo_screenheight()
```

```
    wf = sw / 1920

    hf = sh / 1080


    Login_Page_Func.Login_Page.title("Resort Ivory Bliss - Login")

    Login_Page_Func.Login_Page.config(bg="#00FFFF")

    Login_Page_Func.Login_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")

    Login_Page_Func.Login_Page.resizable(0,0)

    Login_Page_Func.Login_Page.state('zoomed')


    if Account_Found_During_Sign_Up == 1:

        messagebox.showerror("Resort Ivory Bliss", "You already have an account with
us. Please login.")

        Account_Found_During_Sign_Up = 2


    label_heading = Label(Login_Page_Func.Login_Page,text="Resort  Ivory
Bliss",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(60*hf)))

    button_login = Button(Login_Page_Func.Login_Page, text="Login",
command=log_in,bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(20*hf)))

    button_sign_up = Button(Login_Page_Func.Login_Page, text="Sign Up",
command=lambda: [close(Login_Page_Func.Login_Page),
Sign_Up_Func()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Century Gothic bold',  int(20*hf)))
```

```
    label_mail = Label(Login_Page_Func.Login_Page, text="Enter your mail
id",bg="#00FFFF",fg = "navy", font=('Century Gothic bold',  int(16*hf)))

    label_passwd = Label(Login_Page_Func.Login_Page, text="Enter your
password",bg="#00FFFF",fg = "navy", font=('Century Gothic bold',  int(16*hf)))

    Login_Page_Func.entry_mail = Entry(Login_Page_Func.Login_Page,fg = "navy",
font=('Century Gothic',  int(14*hf)),width=600)

    Login_Page_Func.entry_passwd = Entry(Login_Page_Func.Login_Page,fg = "navy",
show="*", font=('Century Gothic',  int(14*hf)),width=600)

    button_exit =
Button(Login_Page_Func.Login_Page,text="Quit",bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Century Gothic bold',
int(16*hf)),width=15,command = lambda : close(Login_Page_Func.Login_Page))

    button_login.place(x=565*wf, y=680*hf, height=50*hf, width=337.5*wf)

    button_sign_up.place(x=987.5*wf, y=680*hf, height=50*hf, width=337.5*wf)

    label_heading.place(x=565*wf,y=230*hf)

    label_mail.place(x=540*wf, y=430*hf, height=50*hf, width=225*wf)

    label_passwd.place(x=540*wf, y=530*hf, height=50*hf, width=225*wf)

    Login_Page_Func.entry_mail.place(x=815*wf, y=430*hf, height=50*hf,
width=550*wf)

    Login_Page_Func.entry_passwd.place(x=815*wf, y=530*hf, height=50*hf,
width=550*wf)

    button_exit.place(x=840*wf,y=800*hf)


    Login_Page_Func.Login_Page.mainloop()

# ------------------------------------------------------------------------------------------------------------------

# Login Func
```

```
def log_in():
    global Name, Account_Found_During_Login, Status, ID, Post


    con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")
    mycursor = con.cursor()
    mycursor.execute("SELECT * FROM login")
    login_data = mycursor.fetchall()
    con.close()


    if Login_Page_Func.entry_mail.get() != "" and
Login_Page_Func.entry_passwd.get() != "" :
        if '@' in Login_Page_Func.entry_mail.get() and
(Login_Page_Func.entry_mail.get().endswith(".com") or
Login_Page_Func.entry_mail.get().endswith(".org")) and (" " not in
Login_Page_Func.entry_mail.get()) and
Login_Page_Func.entry_mail.get().count("@")==1:
            email_id = Login_Page_Func.entry_mail.get()
            Account_Found_During_Login = 0
            for data in login_data:
                if email_id in data:
                    Account_Found_During_Login = 1
                    record = data
                    break
            if Account_Found_During_Login == 0:
                close(Login_Page_Func.Login_Page)
                Sign_Up_Func()


            else:
                ID = record[0]
                Post = record[-1]
                Status = ID[0]


                passwd = Login_Page_Func.entry_passwd.get()
                if passwd == record[5]:
                    if record[2] == "Male":
                        Name = "Mr. " + record[1]
                    else:
                        Name = "Mrs. " + record[1]
                    close(Login_Page_Func.Login_Page)
                    Main_Menu_Func()
                else:
                    messagebox.showerror("Resort Ivory Bliss", "Incorrect password. Please
enter the password again.")
                    Login_Page_Func.entry_passwd.delete(0, END)
        else:
            messagebox.showerror("Resort Ivory Bliss", " Please enter a valid email id.")
    else:
        messagebox.showerror("Resort Ivory Bliss", " Please enter all the details.")


#------------------------------------------------------------------------------------------------------------------------------------------------------

# Sign Up Page
def Sign_Up_Func():
    global Account_Found_During_Login


    Sign_Up_Func.Sign_Up_Page = Tk()


    sw = Sign_Up_Func.Sign_Up_Page.winfo_screenwidth()
    sh = Sign_Up_Func.Sign_Up_Page.winfo_screenheight()


    wf = sw / 1920
    hf = sh / 1080


    Sign_Up_Func.Sign_Up_Page.config(bg="#00FFFF")


    Sign_Up_Func.Sign_Up_Page.resizable(0, 0)
    Sign_Up_Func.Sign_Up_Page.state('zoomed')


    Sign_Up_Func.Sign_Up_Page.title("Resort Ivory Bliss - Sign Up")
    Sign_Up_Func.Sign_Up_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")
    label_heading = Label(Sign_Up_Func.Sign_Up_Page, text="Sign Up", bg="#00FFFF",
fg="navy",
                        font=('Bookman Old Style bold', int(60 * hf)))
    button_main_sign_up = Button(Sign_Up_Func.Sign_Up_Page, text="Sign Up",
command=Sign_Up, bg="navy",
                              activebackground="#00FFFF", fg="#00FFFF",
activeforeground="navy",
                              font=('Century Gothic bold', int(20 * hf)), width=30)
    label_mail = Label(Sign_Up_Func.Sign_Up_Page, text="Enter your mail id ",
bg="#00FFFF", fg="navy",
                     font=('Century Gothic bold', int(16 * hf)))
    label_passwd = Label(Sign_Up_Func.Sign_Up_Page, text="Enter your password",
bg="#00FFFF", fg="navy",
                       font=('Century Gothic bold', int(16 * hf)))
    label_name = Label(Sign_Up_Func.Sign_Up_Page, text="Enter your name ",
bg="#00FFFF", fg="navy",
                     font=('Century Gothic bold', int(16 * hf)))
    label_gender = Label(Sign_Up_Func.Sign_Up_Page, text="Gender (M/F) ",
bg="#00FFFF", fg="navy",
                       font=('Century Gothic bold', int(16 * hf)))
    label_phone_no = Label(Sign_Up_Func.Sign_Up_Page, text="Enter your 10-digit
mobile number", bg="#00FFFF", fg="navy",
                         font=('Century Gothic bold', int(16 * hf)))
    Sign_Up_Func.entry_mail = Entry(Sign_Up_Func.Sign_Up_Page, fg="navy",
width=50,
                              font=('Century Gothic', int(14 * hf)))
    Sign_Up_Func.entry_passwd = Entry(Sign_Up_Func.Sign_Up_Page, fg="navy",
width=50,
                                font=('Century Gothic', int(14 * hf)))
    Sign_Up_Func.entry_name = Entry(Sign_Up_Func.Sign_Up_Page, fg="navy",
width=50,
                              font=('Century Gothic', int(14 * hf)))
    Sign_Up_Func.entry_phone_no = Entry(Sign_Up_Func.Sign_Up_Page, fg="navy",
width=50,
                                  font=('Century Gothic', int(14 * hf)))
    button_back_to_login = Button(Sign_Up_Func.Sign_Up_Page, text="Back",
                             command=lambda: [close(Sign_Up_Func.Sign_Up_Page),
Login_Page_Func()], bg="navy",
                             activebackground="#00FFFF", fg="#00FFFF",
activeforeground="navy",
                             font=('Century Gothic bold', int(20 * hf)), width=15)

    list_gender = ["Male", "Female"]
    Sign_Up_Func.gender = StringVar()
    Sign_Up_Func.gender.set("Male")

    option_gender = OptionMenu(Sign_Up_Func.Sign_Up_Page,
Sign_Up_Func.gender, *list_gender)
    option_gender.config(width=46, bg="navy", activebackground="#00FFFF",
fg="#00FFFF", activeforeground="navy",
                    font=('Century Gothic', int(14 * hf)))

    if Account_Found_During_Login == 0:
        messagebox.showerror("Resort Ivory Bliss", "You don't have an account with us yet. Please sign up.")
        Account_Found_During_Login = 2

    label_heading.place(x=825 * wf, y=55 * hf)
    button_main_sign_up.place(x=910 * wf, y=800 * hf)
    button_back_to_login.place(x=500 * wf, y=800 * hf)
    label_mail.place(x=550 * wf, y=550 * hf, height=50 * hf)
    label_passwd.place(x=530 * wf, y=650 * hf, height=50 * hf)
    label_name.place(x=550 * wf, y=250 * hf, height=50 * hf)
    label_gender.place(x=550 * wf, y=350 * hf, height=50 * hf)
    label_phone_no.place(x=465 * wf, y=450 * hf, height=50 * hf)
    Sign_Up_Func.entry_mail.place(x=875 * wf, y=550 * hf, height=50 * hf)
    Sign_Up_Func.entry_passwd.place(x=875 * wf, y=650 * hf, height=50 * hf)
    Sign_Up_Func.entry_name.place(x=875 * wf, y=250 * hf, height=50 * hf)
    Sign_Up_Func.entry_phone_no.place(x=875 * wf, y=450 * hf, height=50 * hf)

    option_gender.place(x=878 * wf, y=350 * hf)

    Sign_Up_Func.Sign_Up_Page.mainloop()
```

```
#-----------------------------------------------------------------------------------------------------------------

# Sign Up Func

def Sign_Up():

   global Name, Account_Found_During_Sign_Up, ID, Status


   con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")

   mycursor = con.cursor()

   mycursor.execute("SELECT * FROM login")

   login_data = mycursor.fetchall()


   if Sign_Up_Func.entry_name.get()!="" and Sign_Up_Func.entry_mail.get()!="" and
Sign_Up_Func.entry_phone_no.get()!="" and Sign_Up_Func.entry_passwd.get()!="":

      if Sign_Up_Func.entry_phone_no.get().isdigit() and
len(Sign_Up_Func.entry_phone_no.get()) == 10:

         if '@' in Sign_Up_Func.entry_mail.get() and
(Sign_Up_Func.entry_mail.get().endswith(".com") or
Sign_Up_Func.entry_mail.get().endswith(".org")) and (" " not in
Sign_Up_Func.entry_mail.get()) and
(not(Sign_Up_Func.entry_mail.get().endswith("@ib.org"))) and
Sign_Up_Func.entry_mail.get().count("@")==1:

            new_login = Sign_Up_Func.entry_mail.get()

            Account_Found_During_Sign_Up = 0

            for data in login_data:

               if new_login in data:
```

```
            Account_Found_During_Sign_Up = 1
            break
    if Account_Found_During_Sign_Up == 0:
        id_no = 0
        for data in login_data:
            if data[0][0] == "C":
                id_no += 1
        new_name = Sign_Up_Func.entry_name.get()
        new_gender = Sign_Up_Func.gender.get()
        new_phone_no = Sign_Up_Func.entry_phone_no.get()
        new_passwd = Sign_Up_Func.entry_passwd.get()

        if new_gender == "Male":
            Name = "Mr. " + new_name
        else:
            Name = "Mrs. " + new_name

        if id_no + 1 < 10:
            n = "0" + str(id_no + 1)
        else:
            n = str(id_no + 1)

        ID = "C" + n
```

```
                Status = "C"


                command = "INSERT INTO login VALUES ('C" + n + "','" + new_name + "','" +
new_gender + "','" + new_phone_no + "','" + new_login + "','" + new_passwd + "','')"

                mycursor.execute(command)

                con.commit()

                con.close()

                close(Sign_Up_Func.Sign_Up_Page)

                Main_Menu_Func()


            else:

                close(Sign_Up_Func.Sign_Up_Page)

                Login_Page_Func()

        else:

            messagebox.showerror("Resort Ivory Bliss", "Please enter a valid email
id.",parent=Sign_Up_Func.Sign_Up_Page)

    else:

        messagebox.showerror("Resort Ivory Bliss", "Please enter a valid phone
number.", parent=Sign_Up_Func.Sign_Up_Page)

  else:

    messagebox.showerror("Resort Ivory Bliss","Please enter all the details.",
parent=Sign_Up_Func.Sign_Up_Page)

# -------------------------------------------------------------------------------------------------------------------

# Main Menu Window
```

```
def Main_Menu_Func():
    global Status, Name, Post
    #-----------------------------------------------------------------------------------------------------------------------
    def overview():
        Overview_Page = Tk()
        sw = Overview_Page.winfo_screenwidth()
        sh = Overview_Page.winfo_screenheight()

        wf = sw / 1920
        hf = sh / 1080

        Overview_Page.config(bg="#00FFFF")
        Overview_Page.state('zoomed')
        Overview_Page.title("Resort Ivory Bliss - Overview")
        Overview_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")
        Overview_Page.resizable(0, 0)

        main_frame = Frame(Overview_Page, bg="#00FFFF")
        main_frame.pack(fill=BOTH, expand=1)
        my_canvas = Canvas(main_frame, bg="#00FFFF")
        my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
        my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)
```

```
        my_scrollbar.pack(side=RIGHT, fill=Y)

        my_canvas.configure(yscrollcommand=my_scrollbar.set)

        my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))

        second_frame = Frame(my_canvas, bg="#00FFFF")

        my_canvas.create_window((0, 0), window=second_frame, anchor=NW)


        label_heading_overview = Label(second_frame, text="Overview",
pady=35*hf,bg="#00FFFF", fg="navy",

                                   font=('Bookman Old Style bold', int(60 * hf)),

                                   justify=CENTER)

        label_heading_overview.grid(row=0, column=1,columnspan=3)


        photo_1 = Image.open(

            "D:\\pythonProject\\Resort Photos\\1.jpg")

        resized_1 = photo_1.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)

        new_photo_1 = ImageTk.PhotoImage(resized_1)

        photo_2 = Image.open(

            "D:\\pythonProject\\Resort Photos\\2.jpg")

        resized_2 = photo_2.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)

        new_photo_2 = ImageTk.PhotoImage(resized_2)

        photo_3 = Image.open(

            "D:\\pythonProject\\Resort Photos\\3.jpg")

        resized_3 = photo_3.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
```

```
new_photo_3 = ImageTk.PhotoImage(resized_3)
photo_4 = Image.open(
    "D:\\pythonProject\\Resort Photos\\4.jpg")
resized_4 = photo_4.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_4 = ImageTk.PhotoImage(resized_4)
photo_5 = Image.open(
    "D:\\pythonProject\\Resort Photos\\5.jpg")
resized_5 = photo_5.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_5 = ImageTk.PhotoImage(resized_5)
photo_6 = Image.open(
    "D:\\pythonProject\\Resort Photos\\6.jpg")
resized_6 = photo_6.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_6 = ImageTk.PhotoImage(resized_6)
photo_7 = Image.open(
    "D:\\pythonProject\\Resort Photos\\7.jpg")
resized_7 = photo_7.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_7 = ImageTk.PhotoImage(resized_7)
photo_8 = Image.open(
    "D:\\pythonProject\\Resort Photos\\8.jpg")
resized_8 = photo_8.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_8 = ImageTk.PhotoImage(resized_8)
photo_9 = Image.open(
    "D:\\pythonProject\\Resort Photos\\19.jpg")
```

```
resized_9 = photo_9.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_9 = ImageTk.PhotoImage(resized_9)
photo_10 = Image.open(
    "D:\\pythonProject\\Resort Photos\\10.jpg")
resized_10 = photo_10.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_10 = ImageTk.PhotoImage(resized_10)
photo_11 = Image.open(
    "D:\\pythonProject\\Resort Photos\\11.jpg")
resized_11 = photo_11.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_11 = ImageTk.PhotoImage(resized_11)
photo_12 = Image.open(
    "D:\\pythonProject\\Resort Photos\\12.jpg")
resized_12 = photo_12.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_12 = ImageTk.PhotoImage(resized_12)
photo_13 = Image.open(
    "D:\\pythonProject\\Resort Photos\\13.jpg")
resized_13 = photo_13.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_13 = ImageTk.PhotoImage(resized_13)
photo_14 = Image.open(
    "D:\\pythonProject\\Resort Photos\\14.jpg")
resized_14 = photo_14.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_14 = ImageTk.PhotoImage(resized_14)
photo_15 = Image.open(
    "D:\\pythonProject\\Resort Photos\\15.jpg")
resized_15 = photo_15.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_15 = ImageTk.PhotoImage(resized_15)
photo_16 = Image.open(
    "D:\\pythonProject\\Resort Photos\\16.jpg")
resized_16 = photo_16.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_16 = ImageTk.PhotoImage(resized_16)
photo_17 = Image.open(
    "D:\\pythonProject\\Resort Photos\\17.jpg")
resized_17 = photo_17.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_17 = ImageTk.PhotoImage(resized_17)
photo_18 = Image.open(
    "D:\\pythonProject\\Resort Photos\\18.jpg")
resized_18 = photo_18.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_18 = ImageTk.PhotoImage(resized_18)
photo_19 = Image.open(
    "D:\\pythonProject\\Resort Photos\\19.jpg")
resized_19 = photo_19.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_19 = ImageTk.PhotoImage(resized_19)
photo_20 = Image.open(
    "D:\\pythonProject\\Resort Photos\\20.jpg")
resized_20 = photo_20.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
new_photo_20 = ImageTk.PhotoImage(resized_20)
```

```
        photo_21 = Image.open(
            "D:\\pythonProject\\Resort Photos\\21.jpg")
        resized_21 = photo_21.resize((int(1768 * wf), int(874 * hf)), Image.ANTIALIAS)
        new_photo_21 = ImageTk.PhotoImage(resized_21)

        list_images = [new_photo_1, new_photo_2, new_photo_3, new_photo_4,
new_photo_5, new_photo_6, new_photo_7,
                       new_photo_8, new_photo_9, new_photo_10,
                       new_photo_11, new_photo_12, new_photo_13, new_photo_14,
new_photo_15, new_photo_16, new_photo_17,
                       new_photo_18, new_photo_19,
                       new_photo_20, new_photo_21]

        list_images_label = []
        overview.image_index = 0

        def next_image(list):
            if overview.image_index < len(list) - 1:
                list[overview.image_index].grid_forget()
                overview.image_index += 1
                list[overview.image_index].grid(row=1, column=1, sticky=NSEW)
            elif overview.image_index == len(list) - 1:
                list[overview.image_index].grid_forget()
                overview.image_index = 0
            list[overview.image_index].grid(row=1, column=1, sticky=NSEW)


        def previous_image(list):
            if overview.image_index > 0:
                list[overview.image_index].grid_forget()
                overview.image_index -= 1
                list[overview.image_index].grid(row=1, column=1, sticky=NSEW)
            elif overview.image_index == 0:
                list[overview.image_index].grid_forget()
                overview.image_index = len(list) - 1
                list[overview.image_index].grid(row=1, column=1, sticky=NSEW)


        for photo in list_images:
            label_photo = Label(second_frame, image=photo, bg="#00FFFF")
            list_images_label.append(label_photo)


        next_button = Button(second_frame, text=">", command=lambda:
next_image(list_images_label), width=1, height=8,
                             bg="#00FFFF", activebackground="#00FFFF", fg="navy",
activeforeground="navy",
                             font=('Century Gothic bold', int(60 * hf)), pady=30 * hf,relief=FLAT)
        previous_button = Button(second_frame, text="<", command=lambda:
previous_image(list_images_label), width=1,
                    height=8, bg="#00FFFF", activebackground="#00FFFF", fg="navy",
activeforeground="navy",
                    font=('Century Gothic bold', int(60 * hf)), pady=30 *
hf,relief=FLAT)
    list_images_label[0].grid(row=1, column=1, sticky=NSEW)
    next_button.grid(row=1, column=2, sticky=W)
    previous_button.grid(row=1, column=0, sticky=W)


    label_data_1 = Label(second_frame, text="HOTEL IVORY BLISS", bg="#00FFFF",
fg="navy",
                  font=('Century Gothic bold', int(17 * hf)), pady=20 * hf)
    label_data_2 = Label(second_frame, text="Ivory Bliss is a  premier 5 star deluxe
beach front resort hotel. Its consists of 3 separate buildings with a contemporary
style architecture: The main building has 8 floors and an impressive\n\
atrium, 3 panoramic glass elevators and a large garden lagoon on the ground
level; Two separate additional wings with 6 and 7 floors respectively. Ivory Bliss is
distinguished by its elegant\n\
décor and high aesthetics in both its public areas and guest rooms. Luxury, style
and quality of services are the core values that make Ivory Bliss stand in a class on
its own.",
                  justify=LEFT, bg="#00FFFF", fg="navy", font=('Century Gothic', int(15 *
hf)), pady=20 * hf)
    label_data_3 = Label(second_frame, text="LOCATION", bg="#00FFFF",
fg="navy",
                  font=('Century Gothic bold', int(17 * hf)), pady=20 * hf)
```

```
        label_data_4 = Label(second_frame, text="Located along the East Coast Road
(ECR), Ivory Bliss makes it easy to escape the bustle of Chennai for a relaxing
getaway in Mahabalipuram. Our 44 acres of landscaped grounds are\n\

set right against the Bay of Bengal, inviting you to unwind and recharge with your
family or loved one. For travelers who want to see the sights, attractions like the
famous Shore Temple\n\

as well as other UNESCO World Heritage sites are less than a mile (lessthan two
kilometers) from the resort. A little farther out, Pondicherry and Kanchipuram make
great day-trip\n\

destinations", justify=LEFT, bg="#00FFFF", fg="navy", font=('Century Gothic', int(15 *
hf)), pady=20 * hf)

        label_data_5 = Label(second_frame, text="DISTANCES FROM", bg="#00FFFF",
fg="navy",

                          font=('Century Gothic bold', int(17 * hf)),

                          pady=20 * hf)

        label_data_6 = Label(second_frame, text="● Chennai International Airport
(MAA) : 55 km\n\

● Chennai Central Railway Station : 55 km", justify=LEFT, bg="#00FFFF", fg="navy",

                          font=('Century Gothic', int(15 * hf)), pady=20 * hf)

        label_data_7 = Label(second_frame, text="THE HOTEL HIGHLIGHTS",
bg="#00FFFF", fg="navy",

                          font=('Century Gothic bold', int(17 * hf)),

                          pady=20 * hf)

        label_data_8 = Label(second_frame, text="1. Accommodation:\n\n\

  ● Large variety of room types with modern design & elegant décor\n\

  ● All rooms with sea view\n\
    ● Spacious rooms with large balconies\n\n\
2. Food & Beverage:\n\n\
    ● Roof top panoramic gourmet restaurant\n\
    ● 24h Room Service\n\
    ● Beach Service\n\
    ● Atmospheric bars\n\n\
3. Internet Access:\n\n\
    ● Free Wi-Fi in all rooms and public areas\n\n\
4. Lifts:\n\n\
    ● Three panoramic glass elevators offering spectacular views to the sea and the atrium", justify=LEFT,
                         bg="#00FFFF", fg="navy",
                         font=('Century Gothic', int(15 * hf)), pady=20 * hf)
       label_data_9 = Label(second_frame, text="ACCOMMODATION", bg="#00FFFF", fg="navy",
                         font=('Century Gothic bold', int(17 * hf)),
                         pady=20 * hf)
       label_data_10 = Label(second_frame, text="1. Classic Rooms (Approx: 30 m2 / 323 ft2 *.Max. occupancy: 3 adults or 2 adults and 1 child):\n\n\
    A room of high aesthetics and modern design, furnished with Wenge or Zebrano furniture. The Classis Rooms are available with twin beds or a Queen-sized bed, tile flooring, a seating\n\
    area and a spacious balcony of 8 m2 / 86 ft2 all comfortably furnished. The bathroom is finished with chocolate brown elements, and is equipped with a bathtub, hairdryer, magnifying\n\
```

mirror, telephone and a variety of deluxe bath amenities.\n\n\n\

2. Deluxe Rooms (Approx: 30 m2 / 323 ft2 * Max. occupancy: 2 adults): \n\n\

A room of high aesthetics and Modern design, furnished with Wenge or Zebrano furniture and wooden floor. The Deluxe Rooms are available with either twin beds or a Queen-sized bed,\n\

a seating area and a spacious comfortably furnished balcony of 8 m2 / 86 ft2. The bathroom is equipped with hydro massage bathtub, hairdryer, magnifying mirror with light, telephone\n\

and a variety of branded bath amenities.\n\n\n\

3. Super Deluxe Rooms (Approx: 36 m2 / 388 ft2 *. Max. occupancy: 2 adults and 1 child): \n\n\

This comfortable Luxury room is overwhelmed with sun light offering spectacular views of the Bay of Bengal and magnificent sunrise experiences. The Elite Club Superior shares\n\

the same wealth of facilities as the Elite Club Guestroom,yet comes enhanced with a larger sofa, a more spacious seating area with writing desk, and bathroom with a separate WC.\n\

Some rooms also feature a glass panel (upon availability) allowing guests to relax in their hydro massage bathtub while enjoying a unique view of the Aegean sea.\n\n\n\

4. Elite Rooms (Approx: 64 m2 / 690 ft2 *. Max. occupancy: 2 adults and 2 children): \n\n\

This luxurious suite of modern design consists of one bedroom and a large living room with its own dining area. The wooden floored bedroom features a King-sized bed, a walk-in closet,\n\

```
    private bathroom with shower cabin, hydro massage bathtub, two wash-basins, hairdryer, magnifying mirror with light, telephone and a separate WC. The comfortable living room leads\n\
    to a large balcony of 16 m2/ 172 ft2, furnished with restful armchairs from which guests can enjoy majestic views of the Bay of Bengal.\n\n\n\
5. Cottages (Approx: 115 m2 / 1238 ft2 *. Max. occupancy: 4 adults or 2 adults and 2 children): \n\n\
    These ultra luxurious suites are available with a white or grey interior and include one master bedroom, one secondary bedroom and a large living room. The master bedroom with\n\
    parquet floor is available with a King-sized bed, en-suite deep whirlpool bath overlooking the Bay of Bengal, Jacuzzi walk-in shower next to the twin wash-basins and a separate WC. The\n\
    second bedroom, also with parquet floor, is available with twin wash-basins and private bathroom with a shower cabin and a separate WC. The living room, furnished with Poltrona Frau\n\
    leather, features a 6 seat dining table and writing desk and offers breathtaking views to the Bay of Bengal through wide glass panels that also provide access to a panoramic balcony.\n\
    This large balcony of 43 m2 / 462 ft2 accommodates a lounge area with restful armchairs and sun beds where guests can relax enjoying majestic views over the sea.",
                              justify=LEFT, bg="#00FFFF", fg="navy", font=('Century Gothic', int(15 * hf)),
                              pady=20 * hf)
        label_data_11 = Label(second_frame, text="DINING & BARS", bg="#00FFFF", fg="navy",
                    font=('Century Gothic bold', int(17 * hf)), pady=20 * hf)
    label_data_12 = Label(second_frame, text="● A 24 hour full Room Service
menu is available.", justify=LEFT,
                    bg="#00FFFF", fg="navy", font=('Century Gothic', int(15 * hf)),
pady=20 * hf)
    label_data_13 = Label(second_frame, text="AMENITIES", bg="#00FFFF",
fg="navy",
                    font=('Century Gothic bold', int(17 * hf)), pady=20 * hf)
    label_data_14 = Label(second_frame, text="● Spa: State-of-the-art spa offering
the world-renowned ESPA treatments & products, with a dedicated area of 800 m2
featuring a reception lounge, 13 separate treatment rooms, Yoga\n\
     & Pilates studio, relaxing area and an outdoor pool with hydromassage. The
Spa offers a wide range of ESPA signature rituals, facials and body treatments
focused on releasing\n\
     stress and tension whilst promoting inner balance and harmony. All guests
having a treatment within the Spa will be offered complimentary use of the Spa
facilities, which include\n\
     the luxurious heat experiences and the outdoor hydro massage pool.\n\n\
● Sports: Cybex-equipped fitness center; floodlight quick tennis court; fitness
activities including yoga, pilates and aqua gym; mini golf, boccia court, beach
volley, and nearby water\n\
       sports station(operated by an independent professional). An 18-hole Golf
course is 8 km away.\n\n\
● Pools: Two outdoor pools: Lagoon style pool (1610 m2), as well as a second
separated smaller outdoor pool for kids and a pool with hydro massage (140 m2)
exclusively devoted to\n\
      Spa guests. All pools are with fresh water.\n\n\
● Beach: 400 m long sandy & shingle beach directly in front of the resort. Sun beds
& umbrellas, both at the pools and on the beach, as well as beach towels, are
available for resort\n\

            guests free of charge.\n\n\

● Entertainment: A mini-theatre with state-of-the-art AV technology with a capacity
of up to 30 people. A  state of the art hardware and, a wide selection of games,
and an\n\

                      unforgettable ambiance, is available at our Gaming Café.",
justify=LEFT, bg="#00FFFF",

                      fg="navy", font=('Century Gothic', int(15 * hf)), pady=20 * hf)

      label_data_15 = Label(second_frame, text="ENERGY CONSERVATION",
bg="#00FFFF", fg="navy",

                      font=('Century Gothic bold', int(17 * hf)), pady=20 * hf)

      label_data_16 = Label(second_frame, text="The resort operates under a BMS
(Building Management System) to enable efficient energy management. In
addition, the resort adopts a number of energy saving, eco-friendly solutions\n\

such as energy saving glass panels, use of gas, litter separation system, energy
saving bulbs and other. Besides, the property has received the "Green Key Award",
the \n\

""Resort Energy Efficiency Award" and the "Blue Flag Award", while it is also certified
with "ISO 14001:2004".",

                      justify=LEFT, bg="#00FFFF", fg="navy", font=('Century Gothic', int(15
* hf)),

                      pady=20 * hf)

      label_data_17 = Label(second_frame, text="OTHER SERVICES", bg="#00FFFF",
fg="navy",

                      font=('Century Gothic bold', int(17 * hf)), pady=20 * hf)
```

```
        label_data_18 = Label(second_frame, text="● Reception Desk on 24 hours
basis\n\
● Guest Relations / Concierge\n\
● Free pool & beach sunbeds and umbrellas\n\
● Free towels for the beach and the pools\n\
● Laundry and dry cleaning service\n\
● Free outdoor parking\n\
● Currency exchange\n\
● Medical Doctor on call (24 hours)\n\
● Rooms and public toilets for guests with reduced mobility", justify=LEFT,
bg="#00FFFF", fg="navy",
                              font=('Century Gothic', int(15 * hf)), pady=20 * hf)
        label_data_19 = Label(second_frame, text="AWARDS & CERTIFICATES",
bg="#00FFFF", fg="navy",
                              font=('Century Gothic bold', int(17 * hf)), pady=20 * hf)
        label_data_20 = Label(second_frame, text="Ivory Bliss has managed to build an
excellent reputation amongst the finest five star properties with its outstanding
performance being reflected by an ever increasing number of awards\n\
and distinctions received. Among those are awards from consumers' review sites
such as: the Tripadvisor's Travelers' Choice Award and Certificate of Excellence, the
Holidaycheck Award\n\
and Quality Selection, and the Zoover Award. In addition, a number of prestigious
Tour Operators have awarded Ivory Bliss with awards such as: TUI Nordic's Silver Blue
Award,\n\
Kuoni Apollo's Silver Customer Choice Award. Ivory Bliss is also certified with the
food safety management system"ISO 22000: 2005".",
                justify=LEFT, bg="#00FFFF", fg="navy", font=('Century Gothic', int(15
* hf)),
                pady=20 * hf)
    label_data_21 = Label(second_frame, text="CONTACT US ", bg="#00FFFF",
fg="navy",
                font=('Century Gothic bold', int(17 * hf)), pady=20 * hf)
    label_data_22 = Label(second_frame, text="● Toll Free Number : 1-8806-
53467\n\
● Reception :  04113 2816 0217 \n\
● Email Us : info@ib.gmail.com", justify=LEFT, bg="#00FFFF", fg="navy", font=('Century
Gothic', int(15 * hf)),
                pady=20 * hf)
    label_data_1.grid(row=2, column=0, columnspan=3, sticky=NSEW)
    label_data_2.grid(row=3, column=0, columnspan=3, sticky=NSEW)
    label_data_3.grid(row=4, column=0, columnspan=3, sticky=NSEW)
    label_data_4.grid(row=5, column=0, columnspan=3, sticky=NSEW)
    label_data_5.grid(row=6, column=0, columnspan=3, sticky=NSEW)
    label_data_6.grid(row=7, column=0, columnspan=3, sticky=NSEW)
    label_data_7.grid(row=8, column=0, columnspan=3, sticky=NSEW)
    label_data_8.grid(row=9, column=0, columnspan=3, sticky=NSEW)
    label_data_9.grid(row=10, column=0, columnspan=3, sticky=NSEW)
    label_data_10.grid(row=11, column=0, columnspan=3, sticky=NSEW)
    label_data_11.grid(row=12, column=0, columnspan=3, sticky=NSEW)
    label_data_12.grid(row=13, column=0, columnspan=3, sticky=NSEW)
    label_data_13.grid(row=14, column=0, columnspan=3, sticky=NSEW)

    label_data_14.grid(row=15, column=0, columnspan=3, sticky=NSEW)

    label_data_15.grid(row=16, column=0, columnspan=3, sticky=NSEW)

    label_data_16.grid(row=17, column=0, columnspan=3, sticky=NSEW)

    label_data_17.grid(row=18, column=0, columnspan=3, sticky=NSEW)

    label_data_18.grid(row=19, column=0, columnspan=3, sticky=NSEW)

    label_data_19.grid(row=20, column=0, columnspan=3, sticky=NSEW)

    label_data_20.grid(row=21, column=0, columnspan=3, sticky=NSEW)

    label_data_21.grid(row=22, column=0, columnspan=3, sticky=NSEW)

    label_data_22.grid(row=23, column=0, columnspan=3, sticky=NSEW)


    button_back_to_main_menu = Button(second_frame, command=lambda:
[close(Overview_Page), Main_Menu_Func()],

                                    text="Back", bg="navy", activebackground="#00FFFF",
fg="#00FFFF",

                                    activeforeground="navy", font=('Century Gothic bold',
int(18 * hf)))

    button_back_to_main_menu.grid(row=24, column=1)


    Overview_Page.mainloop()

  #------------------------------------------------------------------------------------------------------------

  # Displaying Profile Func

  def Display_Profile():

    #----------------------------------------------------------------------------------------------------------
    def edit_profile(record):
        global Status

        def back_to_profile_page(page):
            confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss",
                                                    "Are you sure you want to return to main menu ?
All unsaved changes will be lost.",
                                                    parent=Edit_Profile_Page)
            if confirm_yes_or_no == "yes":
                close(page)
                Display_Profile()
            else:
                pass

        def update_changes(record):
            global Name
            confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss", "Do you
want to save changes?", parent=Edit_Profile_Page)
            if confirm_yes_or_no == "yes":
                if entry_name.get()!="" and entry_mobile_no.get()!="" and
entry_email_id.get()!="" and entry_password.get()!="":
                    if entry_mobile_no.get().isdigit() and len(entry_mobile_no.get()) == 10:
                        if '@' in entry_email_id.get() and
(entry_email_id.get().endswith(".com") or entry_email_id.get().endswith(".org")) and
" " not in entry_email_id.get():
```

```
                if Status == "A":
                    command_1 = "UPDATE login SET  NAME ='" + entry_name.get()
+ "',GENDER = '" + \
                                record[
                                    2] + "',PHONE_NO='" + entry_mobile_no.get() + \
                                "',EMAIL_ID = '" + record[
                                    4] + "',PASSWORD = '" + entry_password.get() +
"',STATUS = '" + \
                                record[
                                    6] + "'WHERE ID ='" + ID + "'"
                    if record[2] == "Male":
                        Name = "Mr. " + entry_name.get()
                    else:
                        Name = "Mrs. " + entry_name.get()
                else:
                    if '@' in entry_email_id.get() and (
                            entry_email_id.get().endswith(".com") or
entry_email_id.get().endswith(
                            ".org")):
                        command_1 = "UPDATE login SET  NAME ='" +
entry_name.get() + "',GENDER = '" + \
                                    record[
                                        2] + "',PHONE_NO='" + entry_mobile_no.get() + \
                                    "',EMAIL_ID = '" + entry_email_id.get() + "',PASSWORD =
'" + entry_password.get() + "'WHERE ID ='" + ID + "'"
```

```
                        if  record[2] == "Male":
                            Name = "Mr. " + entry_name.get()
                        else:
                            Name = "Mrs. " + entry_name.get()

                        command_2 = "UPDATE record_log SET CUSTOMER_NAME ='"
+ Name + "'WHERE CID='"+ID+"'"
                    else:
                        messagebox.showerror("Resort Ivory Bliss", "Please Enter A
Valid Email ID !",
                                             parent=Edit_Profile_Page)

                con = mysql.connector.connect(host="localhost", user="root",
passwd="root",
                                              database="project")
                mycursor = con.cursor()

                mycursor.execute(command_1)
                mycursor.execute(command_2)
                con.commit()
                con.close()
                messagebox.showinfo("Resort Ivory Bliss", "Successfully Updated.",
                                    parent=Edit_Profile_Page)
                close(Edit_Profile_Page)

                place_name_label.label_heading_2.place_forget()

                place_name_label()

                Display_Profile()

             else:

                messagebox.showerror("Resort Ivory Bliss", "Please enter a valid
email id.",

                                             parent=Edit_Profile_Page)

                   else:

                messagebox.showerror("Resort Ivory Bliss", "Please enter a valid
phone number.",

                                             parent=Edit_Profile_Page)

                 else:

             messagebox.showerror("Resort Ivory Bliss", "Please enter all the
details.",

                                           parent=Edit_Profile_Page)

               else:

             pass



  Edit_Profile_Page = Tk()



  sw = Edit_Profile_Page.winfo_screenwidth()

  sh = Edit_Profile_Page.winfo_screenheight()
```

```
        wf = sw / 1920

        hf = sh / 1080


        Edit_Profile_Page.config(bg="#00FFFF")


        Edit_Profile_Page.resizable(0, 0)

        Edit_Profile_Page.state('zoomed')


        Edit_Profile_Page.title("Resort Ivory Bliss -  Edit Profile")

        Edit_Profile_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")


        label_heading = Label(Edit_Profile_Page, text="Your Profile Details",
bg="#00FFFF", fg="navy",

                              font=('Bookman Old Style bold', int(40 * hf)))


        label_name = Label(Edit_Profile_Page, text="Name", bg="#00FFFF", fg="navy",

                           font=('Century Gothic bold', int(16 * hf)))

        label_gender = Label(Edit_Profile_Page, text="Gender", bg="#00FFFF",
fg="navy",

                             font=('Century Gothic bold', int(16 * hf)))

        label_mobile_no = Label(Edit_Profile_Page, text="Mobile Number",
bg="#00FFFF", fg="navy",

                                font=('Century Gothic bold', int(16 * hf)))
        label_email_id = Label(Edit_Profile_Page, text="Email Id", bg="#00FFFF",
fg="navy",
                            font=('Century Gothic bold', int(16 * hf)))
        label_password = Label(Edit_Profile_Page, text="Password", bg="#00FFFF",
fg="navy",
                            font=('Century Gothic bold', int(16 * hf)))
        label_customer_gender = Label(Edit_Profile_Page, text=record[2], fg="navy",
                                font=('Century Gothic', int(16 * hf)), bg='white',
                                justify=CENTER, width=40)


        entry_name = Entry(Edit_Profile_Page, width=40, font=('Century Gothic',
int(16 * hf)), fg="navy",
                          justify=CENTER)
        entry_mobile_no = Entry(Edit_Profile_Page, width=40, font=('Century Gothic',
int(16 * hf)), fg="navy",
                             justify=CENTER)
        entry_password = Entry(Edit_Profile_Page, width=40, font=('Century Gothic',
int(16 * hf)), fg="navy",
                            justify=CENTER)


        button_update = Button(Edit_Profile_Page, text="Save Changes",
command=lambda: update_changes(record),
                            bg="navy",
                            activebackground="#00FFFF", fg="#00FFFF",
activeforeground="navy",
                            font=('Century Gothic bold', int(16 * hf)), width=40)
        button_back = Button(Edit_Profile_Page, text="Back",
                            command=lambda: back_to_profile_page(Edit_Profile_Page),
bg="navy",
                            activebackground="#00FFFF",
                            fg="#00FFFF", activeforeground="navy", font=('Century Gothic
bold', int(16 * hf)),
                            width=10)


        entry_name.insert(0, record[1])
        entry_mobile_no.insert(0, record[3])
        entry_password.insert(0, record[5])


        if Status == "A":
            label_post = Label(Edit_Profile_Page, text="Your Post", bg="#00FFFF",
fg="navy",
                            font=('Century Gothic bold', int(16 * hf)))
            label_admin_post = Label(Edit_Profile_Page, text=record[6], width=40,
justify=CENTER,
                            font=('Century Gothic', int(16 * hf)), fg="navy", bg='white')
            label_email_id_admin = Label(Edit_Profile_Page, text=record[4], fg="navy",
                                  font=('Century Gothic', int(16 * hf)), bg='white',
                                  justify=CENTER, width=40)
            label_heading.place(x=720 * wf, y=95 * hf)


            label_name.place(x=650 * wf, y=235 * hf)

            label_gender.place(x=650 * wf, y=335 * hf)

            label_mobile_no.place(x=650 * wf, y=435 * hf)

            label_email_id.place(x=650 * wf, y=535 * hf)

            label_password.place(x=650 * wf, y=635 * hf)


            entry_name.place(x=855 * wf, y=235 * hf)

            label_customer_gender.place(x=855 * wf, y=335 * hf)

            entry_mobile_no.place(x=855 * wf, y=435 * hf)

            label_email_id_admin.place(x=855 * wf, y=535 * hf)

            entry_password.place(x=855 * wf, y=635 * hf)

            label_post.place(x=655 * wf, y=735 * hf)

            label_admin_post.place(x=855 * wf, y=735 * hf)


            button_update.place(x=835 * wf, y=835 * hf)

            button_back.place(x=635 * wf, y=835 * hf)


        else:

            entry_email_id = Entry(Edit_Profile_Page, width=40, font=('Century Gothic',
int(16 * hf)), fg="navy",

                               justify=CENTER)
```

```
            entry_email_id.insert(0, record[4])


            label_heading.place(x=720 * wf, y=145 * hf)


            label_name.place(x=650 * wf, y=285 * hf)

            label_gender.place(x=650 * wf, y=385 * hf)

            label_mobile_no.place(x=650 * wf, y=485 * hf)

            label_email_id.place(x=650 * wf, y=585 * hf)

            label_password.place(x=650 * wf, y=685 * hf)


            entry_name.place(x=855 * wf, y=285 * hf)

            label_customer_gender.place(x=855 * wf, y=385 * hf)

            entry_mobile_no.place(x=855 * wf, y=485 * hf)

            entry_email_id.place(x=855 * wf, y=585 * hf)

            entry_password.place(x=855 * wf, y=685 * hf)


            button_update.place(x=835 * wf, y=785 * hf)

            button_back.place(x=635 * wf, y=785 * hf)


        Edit_Profile_Page.mainloop()

        #---------------------------------------------------------------------------------------------------------------

    con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")
```

```
mycursor = con.cursor()
mycursor.execute("SELECT * FROM login")
profile_data = mycursor.fetchall()
con.close()

global ID

Profile = Tk()

sw = Profile.winfo_screenwidth()
sh = Profile.winfo_screenheight()
wf = sw / 1920
hf = sh / 1080

Profile.config(bg="#00FFFF")

Profile.resizable(0, 0)
Profile.state('zoomed')

Profile.title("Resort Ivory Bliss - Profile")
Profile.iconbitmap("D:\\pythonProject\\HOTEL.ico")

for data in profile_data:
```

```
            if data[0] == ID:
                record = data

        label_heading = Label(Profile, text="Your Profile Details", bg="#00FFFF",
fg="navy",
                              font=('Bookman Old Style bold', int(40 * hf)))

        label_name = Label(Profile, text="Name", bg="#00FFFF", fg="navy",
font=('Century Gothic bold', int(16 * hf)))
        label_gender = Label(Profile, text="Gender", bg="#00FFFF", fg="navy",
                             font=('Century Gothic bold', int(16 * hf)))
        label_mobile_no = Label(Profile, text="Mobile Number", bg="#00FFFF",
fg="navy",
                                font=('Century Gothic bold', int(16 * hf)))
        label_email_id = Label(Profile, text="Email Id", bg="#00FFFF", fg="navy",
                               font=('Century Gothic bold', int(16 * hf)))
        label_password = Label(Profile, text="Password", bg="#00FFFF", fg="navy",
                               font=('Century Gothic bold', int(16 * hf)))

        label_customer_name = Label(Profile, text=record[1], fg="navy", bg="white",
width=40,
                                    font=('Century Gothic', int(16 * hf)),
                                    justify=CENTER)
        label_customer_gender = Label(Profile, text=record[2], fg="navy", bg="white",
width=40,
                              font=('Century Gothic', int(16 * hf)),
                              justify=CENTER)
    label_customer_mobile_no = Label(Profile, text=record[3], fg="navy",
bg="white", width=40,
                                font=('Century Gothic', int(16 * hf)),
                                justify=CENTER)
    label_customer_email_id = Label(Profile, text=record[4], fg="navy", bg="white",
width=40,
                               font=('Century Gothic', int(16 * hf)),
                               justify=CENTER)
    label_customer_password = Label(Profile, text=record[5], fg="navy", bg="white",
width=40,
                               font=('Century Gothic', int(16 * hf)),
                               justify=CENTER)


    button_logout = Button(Profile, text="Logout",
                          command=lambda: [close(Profile), close(main_menu),
Login_Page_Func()], bg="navy",
                          activebackground="#00FFFF", fg="#00FFFF",
activeforeground="navy",
                          font=('Century Gothic bold', int(20 * hf)))
    button_edit_profile = Button(Profile, text="Edit Profile",
                                command=lambda: [close(Profile), edit_profile(record)],
bg="navy",
                              activebackground="#00FFFF", fg="#00FFFF",
activeforeground="navy",
                              font=('Century Gothic bold', int(20 * hf)), width=40)
      button_close = Button(Profile, text="Close", command=lambda: close(Profile),
                         font=('Century Gothic bold', int(20 * hf)),
                         bg="navy", activebackground="#00FFFF", fg="#00FFFF",
activeforeground="navy", width=10)

      if Status == "A":
         label_heading.place(x=725 * wf, y=80 * hf)

         label_name.place(x=650 * wf, y=220 * hf)
         label_gender.place(x=650 * wf, y=320 * hf)
         label_mobile_no.place(x=650 * wf, y=420 * hf)
         label_email_id.place(x=650 * wf, y=520 * hf)
         label_password.place(x=650 * wf, y=620 * hf)

         label_customer_name.place(x=850 * wf, y=220 * hf)
         label_customer_gender.place(x=850 * wf, y=320 * hf)
         label_customer_mobile_no.place(x=850 * wf, y=420 * hf)
         label_customer_email_id.place(x=850 * wf, y=520 * hf)
         label_customer_password.place(x=850 * wf, y=620 * hf)

         label_post = Label(Profile, text="Your Post", bg="#00FFFF", fg="navy",
                     font=('Century Gothic bold', int(16 * hf)))
         label_post.place(x=650 * wf, y=720 * hf)

         label_admin_post = Label(Profile, text=record[6], width=40, justify=CENTER,
fg="navy",
                              font=('Century Gothic', int(16 * hf)),
                              bg='white')
         label_admin_post.place(x=850 * wf, y=720 * hf)

         button_logout.place(x=560 * wf, y=920 * hf, width=850.5 * wf)
         button_edit_profile.place(x=760 * wf, y=820 * hf)
         button_close.place(x=560 * wf, y=820 * hf)

      else:
         label_heading.place(x=725 * wf, y=110 * hf)

         label_name.place(x=650 * wf, y=250 * hf)
         label_gender.place(x=650 * wf, y=350 * hf)
         label_mobile_no.place(x=650 * wf, y=450 * hf)
         label_email_id.place(x=650 * wf, y=550 * hf)
         label_password.place(x=650 * wf, y=650 * hf)

         label_customer_name.place(x=850 * wf, y=250 * hf)
         label_customer_gender.place(x=850 * wf, y=350 * hf)

         label_customer_mobile_no.place(x=850 * wf, y=450 * hf)

         label_customer_email_id.place(x=850 * wf, y=550 * hf)

         label_customer_password.place(x=850 * wf, y=650 * hf)


         button_logout.place(x=560 * wf, y=850 * hf, width=850.5 * wf)

         button_edit_profile.place(x=760 * wf, y=750 * hf)

         button_close.place(x=560 * wf, y=750 * hf)


      Profile.mainloop()
  #-----------------------------------------------------------------------------------------------------------------
  main_menu = Tk()

  sw = main_menu.winfo_screenwidth()

  sh = main_menu.winfo_screenheight()


  wf = sw / 1920

  hf = sh / 1080

  main_menu.config(bg="#00FFFF")

  main_menu.resizable(0, 0)

  main_menu.state('zoomed')

  main_menu.title("Resort Ivory Bliss - Main Menu")

  main_menu.iconbitmap("D:\\pythonProject\\HOTEL.ico")
```

```
    def place_name_label():

        place_name_label.label_heading_2 = Label(main_menu, text=Name,
bg="#00FFFF", fg="navy", font=('Century Gothic bold', int(30*hf)),justify=CENTER)

        place_name_label.label_heading_2.place(x=0*wf, y=175*hf, width=1920*wf)


    if Status == "C" :

        icon = Image.open("D:\\pythonProject\\HOTEL_BG.jpg")

        resized_icon = icon.resize((int(sw), int(sh)), Image.ANTIALIAS)

        new_icon = ImageTk.PhotoImage(resized_icon)

        label_icon = Label(main_menu, image=new_icon, justify=CENTER)

        label_icon.place(x=0*wf, y=0*hf)


        label_heading_1 = Label(main_menu, text="Welcome to Resort Ivory Bliss",
bg="#00FFFF", fg="navy",

                            font=('Century Gothic bold', int(30*hf)), justify=CENTER)

        label_heading_1.place(x=0*wf, y=100*hf, width=1920*wf)


        place_name_label()


        button_profile = Button(main_menu, text=" Your Profile",
command=Display_Profile,bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold', int(20*hf)))

        button_booking = Button(main_menu, text="Book a Stay", command=lambda:
[close(main_menu), Booking_Page_Func()],bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold',
int(20*hf)))
        button_your_stays = Button(main_menu, text="Your Stays", command=lambda:
[close(main_menu), stays()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold', int(20*hf)))
        button_overview = Button(main_menu, text="Overview", command = lambda :
[close(main_menu),overview()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",font=('Bookman Old Style bold', int(20*hf)))


        button_profile.place(x=180*wf, y=412.5*hf, width=350*wf)
        button_overview.place(x=180*wf, y=777.5*hf, width=350*wf)
        button_booking.place(x=1390*wf, y=412.5*hf, width=350*wf)
        button_your_stays.place(x=1390*wf, y=777.5*hf, width=350*wf)


    else:
        if Post == "Manager" :
            icon =Image.open("D:\\pythonProject\\HOTEL_BG.jpg")
            resized_icon = icon.resize((int(sw), int(sh)), Image.ANTIALIAS)
            new_icon = ImageTk.PhotoImage(resized_icon)
            label_icon = Label(main_menu, image=new_icon, justify=CENTER)
            label_icon.place(x=0 * wf, y=0 * hf)


            label_heading_1 = Label(main_menu, text="Welcome to Resort Ivory Bliss",
bg="#00FFFF", fg="navy",
                              font=('Century Gothic bold', int(30*hf)), justify=CENTER)
        label_heading_1.place(x=0*wf, y=100*hf, width=1920*wf)

        place_name_label()

        button_profile = Button(main_menu, text=" Your Profile", 
command=Display_Profile,bg="navy",activebackground = "#00FFFF",fg = 
"#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold', int(20*hf)) )
        button_edit_rooms_info = Button(main_menu, text="Edit Rooms' 
Details",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground = 
"navy",font=('Bookman Old Style bold', int(20*hf)), command=lambda : 
[close(main_menu), Room_Info_Page_Admin_Func()])
        button_edit_restaurant_menu = Button(main_menu, text="Edit Restaurant 
Menu",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground = 
"navy", font=('Bookman Old Style bold', int(20*hf)), command=lambda : 
[close(main_menu), Restaurant_Menu_Page_Admin_Func()])
        button_edit_laundry_menu = Button(main_menu, text="Edit Laundry 
Menu",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground = 
"navy", font=('Bookman Old Style bold', int(20*hf)), command=lambda : 
[close(main_menu), Laundry_Service_Page_Admin_Func()])
        button_edit_amenities_menu = Button(main_menu, text="Edit Amenities 
Menu",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground = 
"navy", font=('Bookman Old Style bold', int(20*hf)), command=lambda : 
[close(main_menu), Amenities_Page_Admin_Func()])
        button_view_record_log = Button(main_menu, text="View Record 
Log",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground = 
"navy",  font=('Bookman Old Style bold', int(20*hf)), command=lambda : 
[close(main_menu), Admin_View_Record_Log()])
```

```
        button_profile.place(x=180*wf, y=350*hf, width=350*wf)
        button_edit_rooms_info.place(x=180*wf, y=575*hf, width=350*wf)
        button_edit_restaurant_menu.place(x=1390*wf, y=350*hf, width=350*wf)
        button_edit_laundry_menu.place(x=1390*wf, y=575*hf, width=350*wf)
        button_edit_amenities_menu.place(x=1390*wf, y=800*hf, width=350*wf)
        button_view_record_log.place(x=180*wf, y=800*hf, width=350*wf)

    elif Post == "Reception" :
        icon = Image.open("D:\\pythonProject\\HOTEL_BG_HIGH.jpg")
        resized_icon = icon.resize((int(sw), int(sh)), Image.ANTIALIAS)
        new_icon = ImageTk.PhotoImage(resized_icon)
        label_icon = Label(main_menu, image=new_icon, justify=CENTER)
        label_icon.place(x=0 * wf, y=0 * hf)

        label_heading_1 = Label(main_menu, text="Welcome to Resort Ivory Bliss",
bg="#00FFFF", fg="navy",
                              font=('Century Gothic bold', int(30*hf)), justify=CENTER)
        label_heading_1.place(x=0*wf, y=100*hf, width=1920*wf)

        place_name_label()

        button_profile = Button(main_menu, text=" Your Profile",
command=Display_Profile,bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold', int(20*hf)) )
```

```
            button_edit_rooms_info = Button(main_menu, text="Edit Rooms'
Details",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy",font=('Bookman Old Style bold', int(20*hf)), command=lambda :
[close(main_menu), Room_Info_Page_Admin_Func()])

            button_view_record_log = Button(main_menu, text="View Record
Log",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy",  font=('Bookman Old Style bold', int(20*hf)), command=lambda :
[close(main_menu), Admin_View_Record_Log()])


            button_profile.place(x=370*wf, y=875*hf)

            button_edit_rooms_info.place(x=820*wf, y=875*hf)

            button_view_record_log.place(x=1350*wf, y=875*hf)


        elif Post == "Restaurant" :

            icon = Image.open("D:\\pythonProject\\HOTEL_BG_HIGH.jpg")

            resized_icon = icon.resize((int(sw), int(sh)), Image.ANTIALIAS)

            new_icon = ImageTk.PhotoImage(resized_icon)

            label_icon = Label(main_menu, image=new_icon, justify=CENTER)

            label_icon.place(x=0 * wf, y=0 * hf)


            label_heading_1 = Label(main_menu, text="Welcome to Resort Ivory Bliss",
bg="#00FFFF", fg="navy",

                        font=('Century Gothic bold', int(30*hf)), justify=CENTER)

            label_heading_1.place(x=0*wf, y=100*hf, width=1920*wf)
            place_name_label()


            button_profile = Button(main_menu, text=" Your
Profile",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy", command=Display_Profile, font=('Bookman Old Style bold', int(20*hf)))

            button_edit_restaurant_menu = Button(main_menu, text="Edit Restaurant
Menu",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy", font=('Bookman Old Style bold', int(20*hf)), command=lambda :
[close(main_menu), Restaurant_Menu_Page_Admin_Func()])


            button_profile.place(x=620*wf, y=875*hf)

            button_edit_restaurant_menu.place(x=970*wf, y=875*hf)


        elif Post == "Laundry" :

            icon = Image.open("D:\\pythonProject\\HOTEL_BG_HIGH.jpg")

            resized_icon = icon.resize((int(sw), int(sh)), Image.ANTIALIAS)

            new_icon = ImageTk.PhotoImage(resized_icon)

            label_icon = Label(main_menu, image=new_icon, justify=CENTER)

            label_icon.place(x=0 * wf, y=0 * hf)


            label_heading_1 = Label(main_menu, text="Welcome to Resort Ivory Bliss",
bg="#00FFFF", fg="navy",

                        font=('Century Gothic bold', int(30*hf)), justify=CENTER)

            label_heading_1.place(x=0*wf, y=100*hf, width=1920*wf)
```

```
            place_name_label()


            button_profile = Button(main_menu, text=" Your Profile",
command=Display_Profile,bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold', int(20*hf)))
            button_edit_laundry_menu = Button(main_menu, text="Edit Laundry
Menu",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy", font=('Bookman Old Style bold', int(20*hf)), command=lambda :
[close(main_menu), Laundry_Service_Page_Admin_Func()])


            button_profile.place(x=640*wf, y=875*hf)
            button_edit_laundry_menu.place(x=970*wf, y=875*hf)


        elif Post == "Amenities" :
            icon = Image.open("D:\\pythonProject\\HOTEL_BG_HIGH.jpg")
            resized_icon = icon.resize((int(sw), int(sh)), Image.ANTIALIAS)
            new_icon = ImageTk.PhotoImage(resized_icon)
            label_icon = Label(main_menu, image=new_icon, justify=CENTER)
            label_icon.place(x=0 * wf, y=0 * hf)


            label_heading_1 = Label(main_menu, text="Welcome to Resort Ivory Bliss",
bg="#00FFFF", fg="navy",
                         font=('Century Gothic bold', int(30*hf)), justify=CENTER)
            label_heading_1.place(x=0*wf, y=100*hf, width=1920*wf)
            place_name_label()


            button_profile = Button(main_menu, text=" Your Profile",
command=Display_Profile,bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold', int(20*hf)))

            button_edit_amenities_menu = Button(main_menu, text="Edit Amenities
Menu",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy", font=('Bookman Old Style bold', int(20*hf)), command=lambda :
[close(main_menu), Amenities_Page_Admin_Func()])


            button_profile.place(x=620*wf, y=875*hf)

            button_edit_amenities_menu.place(x=960*wf, y=875*hf)


        elif Post == "Room Service" :

            icon = Image.open("D:\\pythonProject\\HOTEL_BG_HIGH.jpg")

            resized_icon = icon.resize((int(sw), int(sh)), Image.ANTIALIAS)

            new_icon = ImageTk.PhotoImage(resized_icon)

            label_icon = Label(main_menu, image=new_icon, justify=CENTER)

            label_icon.place(x=0 * wf, y=0 * hf)


            label_heading_1 = Label(main_menu, text="Welcome to Resort Ivory Bliss",
bg="#00FFFF", fg="navy",

                         font=('Century Gothic bold', int(30*hf)), justify=CENTER)

            label_heading_1.place(x=0*wf, y=100*hf, width=1920*wf)
```

```
            place_name_label()


            button_profile = Button(main_menu, text=" Your Profile",
command=Display_Profile,bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold', int(20*hf)))
            button_edit_restaurant_menu = Button(main_menu, text="Edit Restaurant
Menu",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy", font=('Bookman Old Style bold', int(20*hf)), command=lambda :
[close(main_menu), Restaurant_Menu_Page_Admin_Func()])
            button_edit_laundry_menu = Button(main_menu, text="Edit Laundry
Menu",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy", font=('Bookman Old Style bold', int(20*hf)), command=lambda :
[close(main_menu), Laundry_Service_Page_Admin_Func()])


            button_profile.place(x=370*wf, y=875*hf)
            button_edit_restaurant_menu.place(x=780*wf, y=875*hf)
            button_edit_laundry_menu.place(x=1280*wf, y=875*hf)



    main_menu.mainloop()
#--------------------------------------------------------------------------------------------------------------
# Booking
def Booking_Page_Func():
    def confirm_details(d1, d2):
```

```
    if d1 == "YYYY-MM-DD" or d2 == "YYYY-MM-DD" or
Booking_Page_Func.check_in_time.get() == "Please Select Check In Time" or
Booking_Page_Func.check_out_time.get() == "Please Select Check Out Time":
        messagebox.showerror("Resort Ivory Bliss", "Please fill all existing fields.")


    else:
        close(Booking_Page)


        booked_room_numbers = []


        con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")


        mycursor_room_info = con.cursor()
        mycursor_room_info.execute("SELECT * FROM room_info WHERE STATUS =
'Available'")
        room_info_data = mycursor_room_info.fetchall()


        now_date = str(date.today() + timedelta(days=1))


        mycursor_room_info.execute(
            "SELECT * FROM record_log WHERE CHECK_OUT_DATE >='" + now_date + "'
AND STATUS = 'Booked' ORDER BY ROOM_NO,CHECK_IN_DATE,CHECK_IN_TIME")
        confirm_details.room_info_booked_data = mycursor_room_info.fetchall()
        mycursor_room_info.execute(
            "SELECT DISTINCT(ROOM_NO) FROM record_log WHERE CHECK_OUT_DATE
>='" + now_date + "' AND STATUS = 'Booked'")

        con.close()

        for booking_id in mycursor_room_info.fetchall():
            booked_room_numbers.append(booking_id[0])

        confirm_details.rooms_available = []
        confirm_details.rooms_available_dictionary = []

        confirm_details.req_date_in_1 = d1
        req_date_list_1 = confirm_details.req_date_in_1.split("-")
        confirm_details.req_date_out_2 = d2
        req_date_list_2 = confirm_details.req_date_out_2.split("-")

        req_time_in_1 = Booking_Page_Func.check_in_time.get()
        req_time_list_1 = req_time_in_1.split(":")
        req_time_out_2 = Booking_Page_Func.check_out_time.get()
        req_time_list_2 = req_time_out_2.split(":")

        for each_room in room_info_data:
            if each_room[0] in booked_room_numbers:

                for each_booking in confirm_details.room_info_booked_data:
                    if each_room[0] == each_booking[1]:
                        if len(confirm_details.room_info_booked_data) > 1 and \
                                each_booking != confirm_details.room_info_booked_data[-1]
and \
                                each_booking[1] ==
confirm_details.room_info_booked_data[confirm_details.room_info_booked_data.i
ndex(each_booking) + 1][1] :

                            req_date_in_3 =
confirm_details.room_info_booked_data[confirm_details.room_info_booked_data.i
ndex(each_booking) + 1][4]
                            req_date_list_3 = req_date_in_3.split("-")

                            req_date_out_4 = each_booking[5]
                            req_date_list_4 = req_date_out_4.split("-")

                            req_time_in_3 =
confirm_details.room_info_booked_data[confirm_details.room_info_booked_data.i
ndex(each_booking) + 1][6]
                            req_time_list_3 = req_time_in_3.split(":")

                            req_time_out_4 = each_booking[7]
                        req_time_list_4 = req_time_out_4.split(":")


                        if datetime(int(req_date_list_1[0]), int(req_date_list_1[1]),
int(req_date_list_1[2]),int(req_time_list_1[0]),
int(req_time_list_1[1]),int(req_time_list_1[2])) > \
                              datetime(int(req_date_list_4[0]),int(req_date_list_4[1]),
int(req_date_list_4[2]), int(req_time_list_4[0]),
int(req_time_list_4[1]),int(req_time_list_4[2])) and \
                              datetime(int(req_date_list_2[0]), int(req_date_list_2[1]),
int(req_date_list_2[2]),int(req_time_list_2[0]),
int(req_time_list_2[1]),int(req_time_list_2[2])) < \
                              datetime(int(req_date_list_3[0]),int(req_date_list_3[1]),
int(req_date_list_3[2]),int(req_time_list_3[0]),
int(req_time_list_3[1]),int(req_time_list_3[2])):


                           confirm_details.rooms_available.append(each_room)
                           room__dictionary = {}
                           room__dictionary["Room Number"] = each_room[0]
                           room__dictionary["Room Type"] = each_room[1]
                           room__dictionary["AC / Non AC"] = each_room[2]
                           room__dictionary["Rate"] = each_room[3]

confirm_details.rooms_available_dictionary.append(room__dictionary)
                           break
```

```
                    elif each_booking == confirm_details.room_info_booked_data[-1]
or \
                        each_booking[1] !=
confirm_details.room_info_booked_data[confirm_details.room_info_booked_data.i
ndex(each_booking) + 1][1]:

                        req_date_out_4 = each_booking[5]
                        req_date_list_4 = req_date_out_4.split("-")
                        req_time_out_4 = each_booking[7]
                        req_time_list_4 = req_time_out_4.split(":")

                        if datetime(int(req_date_list_1[0]),
int(req_date_list_1[1]),int(req_date_list_1[2]), int(req_time_list_1[0]),
int(req_time_list_1[1]),int(req_time_list_1[2])) > \

datetime(int(req_date_list_4[0]),int(req_date_list_4[1]),int(req_date_list_4[2]),int(req_
time_list_4[0]),int(req_time_list_4[1]),int(req_time_list_4[2])):

                            confirm_details.rooms_available.append(each_room)
                            room__dictionary = {}
                            room__dictionary["Room Number"] = each_room[0]
                            room__dictionary["Room Type"] = each_room[1]
                            room__dictionary["AC / Non AC"] = each_room[2]
                            room__dictionary["Rate"] = each_room[3]
```

```
confirm_details.rooms_available_dictionary.append(room__dictionary)
                        break

            else:
                confirm_details.rooms_available.append(each_room)
                room__dictionary = {}
                room__dictionary["Room Number"] = each_room[0]
                room__dictionary["Room Type"] = each_room[1]
                room__dictionary["AC / Non AC"] = each_room[2]
                room__dictionary["Rate"] = each_room[3]
                confirm_details.rooms_available_dictionary.append(room__dictionary)

        if len(confirm_details.rooms_available):
            Room_Display_Page = Tk()
            sw = Room_Display_Page.winfo_screenwidth()
            sh = Room_Display_Page.winfo_screenheight()

            wf = sw / 1920
            hf = sh / 1080

            Room_Display_Page.config(bg="#00FFFF")
            Room_Display_Page.state('zoomed')
            Room_Display_Page.title("Resort Ivory Bliss - Available Rooms")

            Room_Display_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")

            Room_Display_Page.resizable(0, 0)


            main_frame = Frame(Room_Display_Page, bg="#00FFFF")

            main_frame.pack(fill=BOTH, expand=1)

            my_canvas = Canvas(main_frame, bg="#00FFFF")

            my_canvas.pack(side=LEFT, fill=BOTH, expand=1)

            my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)

            my_scrollbar.pack(side=RIGHT, fill=Y)

            my_canvas.configure(yscrollcommand=my_scrollbar.set)

            my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))

            second_frame = Frame(my_canvas,bg="#00FFFF")

            my_canvas.create_window((0, 0), window=second_frame, anchor=NW)


            confirm_details.button_book_list = []


            label_main_heading = Label(second_frame, text='AVAILABLE
ROOMS',bg="#00FFFF",fg = "navy", font=("Bookman Old Style bold", int(40*hf)),

                                pady=50*hf)

            label_main_heading.grid(row=0, column=0, columnspan=4, rowspan=2)
            label_heading_room = Label(second_frame, text="Room",bg="#00FFFF",fg
= "navy", font=('Bookman Old Style bold', int(30*hf)), pady=50*hf,
                            padx=225*wf)
            label_book = Label(second_frame, text="Book",bg="#00FFFF",fg = "navy",
font=('Bookman Old Style bold', int(30*hf)), pady=50*hf, padx=225*wf)
            label_heading_room.grid(row=2, column=0)
            label_book.grid(row=2, column=1)

            label_filters = Label(second_frame, text="Filters",bg="#00FFFF",fg = "navy",
font=('Bookman Old Style bold', int(30*hf)), pady=50*hf,
                        padx=225*wf)
            label_filters.grid(row=2, column=2)

            list_sorting = ["None", "Price : High To Low", "Price : Low To High"]
            list_filters = ["None", "AC", "NON AC"]

            list_type_filter = ["None","CLASSIC", "DELUXE", "SUPER DELUXE", "ELITE",
"COTTAGE"]

            filter_selection_1 = StringVar()
            filter_selection_1.set("None")

            option_selection_1 = OptionMenu(second_frame, filter_selection_1,
*list_sorting)
            option_selection_1.grid(row=4, column=2)
```

```
        option_selection_1.configure(bg="navy",activebackground = "#00FFFF",fg
= "#00FFFF",activeforeground = "navy",font=('Century Gothic', int(20*hf)), width=16)


        filter_selection_2 = StringVar()

        filter_selection_2.set("None")


        option_selection_2 = OptionMenu(second_frame, filter_selection_2,
*list_filters)

        option_selection_2.grid(row=6, column=2)

        option_selection_2.configure(bg="navy",activebackground = "#00FFFF",fg
= "#00FFFF",activeforeground = "navy",font=('Century Gothic', int(20*hf)), width=16)


        filter_selection_3 = StringVar()

        filter_selection_3.set("None")


        option_selection_3 = OptionMenu(second_frame, filter_selection_3,
*list_type_filter)

        option_selection_3.grid(row=8, column=2)

        option_selection_3.configure(bg="navy",activebackground = "#00FFFF",fg
= "#00FFFF",activeforeground = "navy",font=('Century Gothic', int(20*hf)), width=16)


        update_list = []

        update_final_list = []


        for id_1 in range(len(confirm_details.rooms_available)):
            update_list.append(confirm_details.rooms_available[id_1])


        for id_2 in update_list:
            update_final_list.append(id_2)


        confirm_details.label_list = []
        confirm_details.button_book_list = []


        Placement_Row_For_Label_Room = 4
        for label_room in range(len(update_list)):
            req_text = "Room No : " + str(
                update_list[label_room][0]) + "\n" + "Room Type : " + str(
                update_list[label_room][1]) + "\n"  + "AC / NON AC : " + str(
                update_list[label_room][2]) + "\n" + "Tariff : " +
str(update_list[label_room][3])


            label_room = Label(second_frame, text=req_text,bg="#00FFFF",fg =
"navy", justify=CENTER, font=('Century Gothic', int(20*hf)),
                               pady=30*hf)
            confirm_details.label_list.append(label_room)
            label_room.grid(row=Placement_Row_For_Label_Room, column=0)
            Placement_Row_For_Label_Room += 2


        Placement_Row_For_Button_Book = 4
```

```
            for button_book in range(len(update_list)):
                button_book = Button(second_frame, text="BOOK
ROOM",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy", font=('Century Gothic', int(20*hf)),
                                 command=lambda button_book=button_book:
[close(Room_Display_Page),
                                                      room_bill_display(str(

Booking_Page_Func.cal_checkin_date.selection_get()),
                                                        str(

Booking_Page_Func.cal_checkout_date.selection_get()),
                                                        str(

Booking_Page_Func.check_in_time.get()),
                                                        str(

Booking_Page_Func.check_out_time.get()),
                                                        button_book,
                                                        update_final_list)])
                button_book.grid(row=Placement_Row_For_Button_Book, column=1)
                confirm_details.button_book_list.append(button_book)
                Placement_Row_For_Button_Book += 2
```

```
        button_back_to_details = Button(second_frame,
text="Back",bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(20*hf)),
                                command=lambda: [close(Room_Display_Page),
Booking_Page_Func()])
        button_back_to_details.grid(row=6, column=3)


        def remove_label():
            for label_room in confirm_details.label_list:
                label_room.grid_forget()


            for button_book in confirm_details.button_book_list:
                button_book.grid_forget()


        def refresh_click():
            selected_filter_1 = filter_selection_1.get()
            selected_filter_2 = filter_selection_2.get()
            selected_filter_3 = filter_selection_3.get()


            update_list = []
            update_final_list = []


            for id in confirm_details.rooms_available:
                update_list.append(id)
```

```
if selected_filter_2 == "None" and selected_filter_3 == "None":
    for id_1 in update_list:
        update_final_list.append(id_1)

elif selected_filter_2 != "None" and selected_filter_3 == "None":
    for id_2 in update_list:
        if id_2[2] == selected_filter_2:
            update_final_list.append(id_2)

elif selected_filter_2 == "None" and selected_filter_3 != "None":
    for id_3 in update_list:
        if id_3[1] == selected_filter_3:
            update_final_list.append(id_3)

elif selected_filter_2 != "None" and selected_filter_3 != "None":
    for id_4 in update_list:
        if id_4[2] == selected_filter_2 and id_4[1] == selected_filter_3:
            update_final_list.append(id_4)

if selected_filter_1 == "None":
    pass
```

```
elif selected_filter_1 == "Price : High To Low":
    def sort_high_to_low(list):
        return list[3]


    update_final_list.sort(key=sort_high_to_low, reverse=True)


elif selected_filter_1 == "Price : Low To High":
    def sort_low_to_high(list):
        return list[3]


    update_final_list.sort(key=sort_low_to_high)


if len(update_final_list) != 0:
    confirm_details.label_list = []
    confirm_details.button_book_list = []


    Placement_Row_For_Label_Room = 4
    for label_room in range(len(update_final_list)):
        req_text = "Room No : " + str(
            update_final_list[label_room][0]) + "\n" + "Room Type : " + str(
            update_final_list[label_room][1]) + "\n" + "AC / NON AC : " + str(
            update_final_list[label_room][2]) + "\n" + "Tariff : Rs." + str(
            update_final_list[label_room][3])
```

```
                    label_room = Label(second_frame, text=req_text,bg="#00FFFF",fg =
"navy", justify=CENTER, font=('Century Gothic', int(20*hf)),
                                  pady=30*hf)
                    confirm_details.label_list.append(label_room)
                    label_room.grid(row=Placement_Row_For_Label_Room, column=0)
                    Placement_Row_For_Label_Room += 2


                  Placement_Row_For_Button_Book = 4
                  for button_book in range(len(update_final_list)):
                    button_book = Button(second_frame, text="BOOK
ROOM",bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy", font=('Century Gothic', int(20*hf)),
                                   command=lambda button_book=button_book:
[close(Room_Display_Page),
                                                         room_bill_display(str(

Booking_Page_Func.cal_checkin_date.selection_get()),
                                                           str(

Booking_Page_Func.cal_checkout_date.selection_get()),
                                                           str(

Booking_Page_Func.check_in_time.get()),
                                                           str(
Booking_Page_Func.check_out_time.get()),
                                                                        button_book,
                                                                        update_final_list)])
                      button_book.grid(row=Placement_Row_For_Button_Book,
column=1)
                      confirm_details.button_book_list.append(button_book)
                      Placement_Row_For_Button_Book += 2


                      if len(update_final_list) == 1:
                         label_spacing_1 = Label(second_frame, text="\t",bg="#00FFFF",fg
= "navy", pady=30*hf)
                         label_spacing_2 = Label(second_frame, text="\t",bg="#00FFFF",fg
= "navy", pady=30*hf)
                         label_spacing_1.grid(row=7, column=2)
                         label_spacing_2.grid(row=9, column=2)


                      elif len(update_final_list) == 2:
                         label_spacing_1 = Label(second_frame, text="\t",bg="#00FFFF",fg
= "navy", pady=15*hf)
                         label_spacing_1.grid(row=7, column=2)
                         label_spacing_2 = Label(second_frame, text='\t',bg="#00FFFF",fg
= "navy", pady=45*hf)
                         label_spacing_2.grid(row=9, column=2)
```

```
                else:
                    label_spacing_1 = Label(second_frame, text="\t", bg="#00FFFF",
fg="navy", pady=45*hf)
                    label_spacing_1.grid(row=5, column=2)
                    label_spacing_2 = Label(second_frame, text='\t', bg="#00FFFF",
fg="navy", pady=45*hf)
                    label_spacing_2.grid(row=7, column=2)
                    label_spacing_3 = Label(second_frame, text='\t', bg="#00FFFF",
fg="navy", pady=45*hf)
                    label_spacing_3.grid(row=9, column=2)
                    messagebox.showerror('Resort Ivory Bliss', "Sorry no such rooms
available.")

                button_refresh = Button(second_frame,
text="Refresh",bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Century Gothic bold',int(20*hf)),
                                command=lambda: [remove_label(), refresh_click()])
                button_refresh.grid(row=10, column=2)
                Room_Display_Page.mainloop()

            else:
                messagebox.showerror('Resort Ivory Bliss', "Sorry no such rooms available")

    def room_bill_display(d1, d2, t1, t2, n, list):
        def Book_Room_Func(n, list):
```

```
        confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss", "Do you
want to confirm booking?",
                                               parent=Room_Bill_Display_Page)


        if confirm_yes_or_no == "yes":

            con = mysql.connector.connect(host="localhost", user="root",
passwd="root", database="project")

            mycursor_book_room = con.cursor()

            key = list[n][0]

            mycursor_book_room.execute("SELECT * FROM record_log")

            s_no = len(mycursor_book_room.fetchall()) + 1

            insert_booking_command = " INSERT INTO record_log VALUES('" + str(s_no)
+ "','" + str(

                key) + "','" + ID + "','" + Name \

                                + "','" + str(confirm_details.req_date_in_1) + "','" \

                                + str(

                confirm_details.req_date_out_2) + "','" +
Booking_Page_Func.check_in_time.get() + "','" +
Booking_Page_Func.check_out_time.get() + "', 'Booked')"

            mycursor_book_room.execute(insert_booking_command)

            con.commit()

            con.close()

            messagebox.showinfo("Resort Ivory Bliss",

                            "Booking successful. A copy of the bill has been sent to your
email. Have a nice stay.",
                         parent=Room_Bill_Display_Page)
            close(Room_Bill_Display_Page)
            Main_Menu_Func()
        else:
            pass

    def back_to_booking():
        confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss", "Do you
want to return to booking page?",parent=Room_Bill_Display_Page)
        if confirm_yes_or_no == "yes":
            close(Room_Bill_Display_Page)
            Booking_Page_Func()
        else:
            pass

    key = 0
    for each_data in confirm_details.rooms_available:
        if list[n] == each_data:
            key = confirm_details.rooms_available.index(each_data)

    req_datein_1 = d1
    req_date_list_1 = req_datein_1.split("-")
```

```
        req_date_2 = d2
        req_date_list_2 = req_date_2.split("-")

        req_time_1 = t1
        req_time_list_1 = req_time_1.split(":")

        req_time_2 = t2
        req_time_list_2 = req_time_2.split(":")

        x = datetime(int(req_date_list_2[0]), int(req_date_list_2[1]),
int(req_date_list_2[2]),int(req_time_list_2[0]), int(req_time_list_2[1]),
int(req_time_list_2[2])) - \
            datetime(int(req_date_list_1[0]), int(req_date_list_1[1]), int(req_date_list_1[2]),
int(req_time_list_1[0]),int(req_time_list_1[1]), int(req_time_list_1[2]))

        no_of_days = (x).days
        no_of_hours = str(x).partition(", ")[2]
        req_hours = int(no_of_hours.split(":")[0])

        req_data_dictionary = confirm_details.rooms_available_dictionary[key]
        if req_hours < 12:
            price_before_gst = (req_data_dictionary["Rate"] * no_of_days) +
((req_data_dictionary["Rate"] / 2) + 200)
        elif req_hours >= 12:
            price_before_gst = confirm_details.rooms_available_dictionary[key]["Rate"] *
(no_of_days + 1)


        x = str(datetime.now())

        date_today = x[0:10]

        time_now = x[11:19]


        con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")


        mycursor_prev_stays = con.cursor()

        mycursor_prev_stays.execute("SELECT * FROM record_log WHERE CID='" + ID + "'
AND CHECK_OUT_DATE<'" + date_today + "' AND STATUS='Booked'")

        prev_stays_list = mycursor_prev_stays.fetchall()

        mycursor_prev_stays.execute("SELECT * FROM record_log WHERE CID='" + ID + "'
AND CHECK_OUT_DATE='" + date_today + "' AND CHECK_OUT_TIME<'" + time_now +
"' AND STATUS='Booked'")

        prev_stays_list.extend(mycursor_prev_stays.fetchall())

        con.close()


        Room_Bill_Display_Page = Tk()

        sw = Room_Bill_Display_Page.winfo_screenwidth()

        sh = Room_Bill_Display_Page.winfo_screenheight()


        wf = sw / 1920
```

```
    hf = sh / 1080


    Room_Bill_Display_Page.config(bg="#00FFFF")

    Room_Bill_Display_Page.resizable(0, 0)

    Room_Bill_Display_Page.state('zoomed')


    Room_Bill_Display_Page.title("Resort Ivory Bliss - Payment For Booking")

    Room_Bill_Display_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")


    global
new_resized_photo_classic,new_resized_photo_deluxe,new_resized_photo_super_d
eluxe,new_resized_photo_elite,new_resized_photo_cottage

    if req_data_dictionary.get("Room Type") == "CLASSIC":

        photo_classic = Image.open("D:\\pythonProject\\Room
Photos\\CLASSIC.jpg")

        resized_photo_classic = photo_classic.resize((int(600*wf), int(400*hf)),
Image.ANTIALIAS)

        new_resized_photo_classic = ImageTk.PhotoImage(resized_photo_classic)

        label_room_photo = Label(Room_Bill_Display_Page,
image=new_resized_photo_classic, bg="#00FFFF")

    elif req_data_dictionary.get("Room Type") == "DELUXE":

        photo_deluxe = Image.open("D:\\pythonProject\\Room
Photos\\DELUXE.jpg")

        resized_photo_deluxe = photo_deluxe.resize((int(600*wf), int(400*hf)),
Image.ANTIALIAS)
```

```
            new_resized_photo_deluxe = ImageTk.PhotoImage(resized_photo_deluxe)
            label_room_photo = Label(Room_Bill_Display_Page,
image=new_resized_photo_deluxe, bg="#00FFFF")
        elif req_data_dictionary.get("Room Type") == "SUPER DELUXE":
            photo_super_deluxe = Image.open("D:\\pythonProject\\Room
Photos\\SUPER DELUXE.jpg")
            resized_photo_super_deluxe = photo_super_deluxe.resize((int(600*wf),
int(400*hf)), Image.ANTIALIAS)
            new_resized_photo_super_deluxe =
ImageTk.PhotoImage(resized_photo_super_deluxe)
            label_room_photo = Label(Room_Bill_Display_Page,
image=new_resized_photo_super_deluxe, bg="#00FFFF")
        elif req_data_dictionary.get("Room Type") == "ELITE":
            photo_elite = Image.open("D:\\pythonProject\\Room Photos\\ELITE.jpg")
            resized_photo_elite = photo_elite.resize((int(600*wf), int(400*hf)),
Image.ANTIALIAS)
            new_resized_photo_elite = ImageTk.PhotoImage(resized_photo_elite)
            label_room_photo = Label(Room_Bill_Display_Page,
image=new_resized_photo_elite, bg="#00FFFF")
        elif req_data_dictionary.get("Room Type") == "COTTAGE":
            photo_cottage = Image.open("D:\\pythonProject\\Room
Photos\\COTTAGE.jpg")
            resized_photo_cottage = photo_cottage.resize((int(600*wf), int(400*hf)),
Image.ANTIALIAS)
            new_resized_photo_cottage =
ImageTk.PhotoImage(resized_photo_cottage)
```

```
            label_room_photo = Label(Room_Bill_Display_Page,
image=new_resized_photo_cottage, bg="#00FFFF")


        label_heading_pay = Label(Room_Bill_Display_Page,text="PAY  FOR  YOUR
STAY",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(60*hf)))


        label_heading_room_number = Label(Room_Bill_Display_Page, text="Room
Number : ",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(20*hf)))

        label_heading_room_type = Label(Room_Bill_Display_Page, text="Room Type :
",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(20*hf)))

        label_heading_ac_non_ac_type = Label(Room_Bill_Display_Page, text="AC /
Non AC : ",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(20*hf)))

        label_heading_room_rate = Label(Room_Bill_Display_Page, text="Tariff in Rs.  :
",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(20*hf)))

        label_heading_number_of_days = Label(Room_Bill_Display_Page,
text="Number Of Days : ",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold',
int(20*hf)))

        label_heading_price_before_gst = Label(Room_Bill_Display_Page, text="Price :
",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(20*hf)))

        label_heading_GST = Label(Room_Bill_Display_Page, text="GST :
",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(20*hf)))

        label_heading_amount_payable_for_room = Label(Room_Bill_Display_Page,
text="Total Amount Payable : ",bg="#00FFFF",fg = "navy", font=('Bookman Old Style
bold', int(20*hf)))
```

```
        label_room_number_data = Label(Room_Bill_Display_Page,
text=req_data_dictionary.get("Room Number"),bg="#00FFFF",fg = "navy",
font=('Bookman Old Style', int(20*hf)))

        label_room_type_data = Label(Room_Bill_Display_Page,
text=req_data_dictionary.get("Room Type"),bg="#00FFFF",fg = "navy",
font=('Bookman Old Style', int(20*hf)))

        label_ac_non_ac_type_data = Label(Room_Bill_Display_Page,
text=req_data_dictionary.get("AC / Non AC"),bg="#00FFFF",fg = "navy",
font=('Bookman Old Style', int(20*hf)))

        label_room_rate_data = Label(Room_Bill_Display_Page,
text=req_data_dictionary.get("Rate"),bg="#00FFFF",fg = "navy", font=('Bookman Old
Style', int(20*hf)))

        label_number_of_days_data =
Label(Room_Bill_Display_Page,text=str(no_of_days) + " (" + no_of_hours + "
Hours)",bg="#00FFFF",fg = "navy", font=('Bookman Old Style', int(20*hf)))

        label_price_before_gst = Label(Room_Bill_Display_Page,
text=str(float(price_before_gst)),bg="#00FFFF",fg = "navy", font=('Bookman Old Style',
int(20*hf)))

        label_GST_data = Label(Room_Bill_Display_Page, text="18%",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style', int(20*hf)))


        button_back_to_booking_page =
Button(Room_Bill_Display_Page,text="Back",command =
back_to_booking,bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold', int(20*hf)))

        button_pay = Button(Room_Bill_Display_Page, text="PAY",
width=20,command=lambda: Book_Room_Func(n,
list),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy", font=('Bookman Old Style bold', int(20*hf)))


    if len(prev_stays_list) < 10:
        label_room_photo.place(x=1150*wf, y=290*hf)


        label_amount_payable_for_room_data = Label(Room_Bill_Display_Page,
text=str(price_before_gst*1.18),bg="#00FFFF",fg = "navy", font=('Bookman Old Style',
int(20*hf)))


        label_heading_pay.place(x=475*wf, y=50*hf)


        label_heading_room_number.place(x=150*wf, y=250*hf)
        label_heading_room_type.place(x=150*wf, y=290*hf)
        label_heading_ac_non_ac_type.place(x=150*wf, y=330*hf)
        label_heading_room_rate.place(x=150*wf, y=430*hf)
        label_heading_number_of_days.place(x=150*wf, y=470*hf)
        label_heading_price_before_gst.place(x=150*wf, y=510*hf)
        label_heading_GST.place(x=150*wf, y=610*hf)
        label_heading_amount_payable_for_room.place(x=150*wf, y=710*hf)


        label_room_number_data.place(x=750*wf, y=250*hf)
        label_room_type_data.place(x=750*wf, y=290*hf)
        label_ac_non_ac_type_data.place(x=750*wf, y=330*hf)
```

```
            label_room_rate_data.place(x=750*wf, y=430*hf)
            label_number_of_days_data.place(x=750*wf, y=470*hf)
            label_price_before_gst.place(x=750*wf, y=510*hf)
            label_GST_data.place(x=750*wf, y=610*hf)
            label_amount_payable_for_room_data.place(x=750*wf, y=710*hf)


            button_back_to_booking_page.place(x=675*wf, y=875*hf)
            button_pay.place(x=910*wf, y=875*hf)


        elif len(prev_stays_list) < 30:
            label_room_photo.place(x=1150*wf, y=315*hf)


            discount = (len(prev_stays_list)//10) * 5
            discounted_price = round(price_before_gst*(1-(discount/100)), 2)


            label_heading_discount = Label(Room_Bill_Display_Page, text="Discount : 
",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(20*hf)))
            label_heading_discounted_price = Label(Room_Bill_Display_Page, 
text="Discounted Price : ",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', 
int(20*hf)))


            label_discount = Label(Room_Bill_Display_Page, 
text=str(discount)+"%",bg="#00FFFF",fg = "navy", font=('Bookman Old Style', 
int(20*hf)))
```

```
        label_discounted_price = Label(Room_Bill_Display_Page,
text=str(discounted_price),bg="#00FFFF",fg = "navy", font=('Bookman Old Style',
int(20*hf)))

        label_amount_payable_for_room_data = Label(Room_Bill_Display_Page,
text=str(discounted_price * 1.18),bg="#00FFFF",fg = "navy", font=('Bookman Old
Style', int(20*hf)))

        label_heading_pay.place(x=475*wf, y=50*hf)

        label_heading_room_number.place(x=150*wf, y=200*hf)
        label_heading_room_type.place(x=150*wf, y=240*hf)
        label_heading_ac_non_ac_type.place(x=150*wf, y=280*hf)
        label_heading_room_rate.place(x=150*wf, y=380*hf)
        label_heading_number_of_days.place(x=150*wf, y=420*hf)
        label_heading_price_before_gst.place(x=150*wf, y=460*hf)
        label_heading_discount.place(x=150*wf, y=560*hf)
        label_heading_discounted_price.place(x=150*wf, y=600*hf)
        label_heading_GST.place(x=150*wf, y=700*hf)
        label_heading_amount_payable_for_room.place(x=150*wf, y=800*hf)

        label_room_number_data.place(x=750*wf, y=200*hf)
        label_room_type_data.place(x=750*wf, y=240*hf)
        label_ac_non_ac_type_data.place(x=750*wf, y=280*hf)
        label_room_rate_data.place(x=750*wf, y=380*hf)

        label_number_of_days_data.place(x=750*wf, y=420*hf)

        label_price_before_gst.place(x=750*wf, y=460*hf)

        label_discount.place(x=750*wf, y=560*hf)

        label_discounted_price.place(x=750*wf, y=600*hf)

        label_GST_data.place(x=750*wf, y=700*hf)

        label_amount_payable_for_room_data.place(x=750*wf, y=800*hf)


        button_back_to_booking_page.place(x=675*wf, y=965*hf)

        button_pay.place(x=910*wf, y=965*hf)


    else:

        label_room_photo.place(x=1150*wf, y=315*hf)


        discount = 15

        discounted_price = round(price_before_gst * (1 - (discount / 100)), 2)


        label_heading_discount = Label(Room_Bill_Display_Page, text="Discount : ",
bg="#00FFFF", fg="navy",

                                       font=('Bookman Old Style bold', int(20*hf)))

        label_heading_discounted_price = Label(Room_Bill_Display_Page,
text="Discounted Price : ", bg="#00FFFF",

                                               fg="navy", font=('Bookman Old Style bold', int(20*hf)))
        label_discount = Label(Room_Bill_Display_Page, text=str(discount) + "%",
bg="#00FFFF", fg="navy",
                           font=('Bookman Old Style', int(20*hf)))
        label_discounted_price = Label(Room_Bill_Display_Page,
text=str(discounted_price), bg="#00FFFF", fg="navy",
                                  font=('Bookman Old Style', int(20*hf)))

        label_amount_payable_for_room_data = Label(Room_Bill_Display_Page,
text=str(discounted_price * 1.18),
                                              bg="#00FFFF", fg="navy", font=('Bookman Old Style',
int(20*hf)))

        label_heading_pay.place(x=475*wf, y=50*hf)

        label_heading_room_number.place(x=150*wf, y=200*hf)
        label_heading_room_type.place(x=150*wf, y=240*hf)
        label_heading_ac_non_ac_type.place(x=150*wf, y=280*hf)
        label_heading_room_rate.place(x=150*wf, y=380*hf)
        label_heading_number_of_days.place(x=150*wf, y=420*hf)
        label_heading_price_before_gst.place(x=150*wf, y=460*hf)
        label_heading_discount.place(x=150*wf, y=560*hf)
        label_heading_discounted_price.place(x=150*wf, y=600*hf)
        label_heading_GST.place(x=150*wf, y=700*hf)
        label_heading_amount_payable_for_room.place(x=150*wf, y=800*hf)
        label_room_number_data.place(x=750*wf, y=200*hf)

        label_room_type_data.place(x=750*wf, y=240*hf)

        label_ac_non_ac_type_data.place(x=750*wf, y=280*hf)

        label_room_rate_data.place(x=750*wf, y=380*hf)

        label_number_of_days_data.place(x=750*wf, y=420)*hf

        label_price_before_gst.place(x=750*wf, y=460*hf)

        label_discount.place(x=750*wf, y=560*hf)

        label_discounted_price.place(x=750*wf, y=600*hf)

        label_GST_data.place(x=750*wf, y=700*hf)

        label_amount_payable_for_room_data.place(x=750*wf, y=800*hf)


        button_back_to_booking_page.place(x=675*wf, y=965*hf)

        button_pay.place(x=910*wf, y=965*hf)

    #------------------------------------------------------------------------------------------------------------

Booking_Page = Tk()


sw = Booking_Page.winfo_screenwidth()

sh = Booking_Page.winfo_screenheight()


wf = sw / 1920

hf =sh / 1080
```

```
    Booking_Page.config(bg="#00FFFF")

    Booking_Page.resizable(0, 0)

    Booking_Page.state('zoomed')


    Booking_Page.title("Resort Ivory Bliss - Booking")

    Booking_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")


    label_heading = Label(Booking_Page,text="BOOK YOUR STAY",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold', int(60*hf)))

    label_heading.place(x=550*wf,y=50*hf)


    def select_checkin_date():

        Booking_Page_Func.entry_checkout_date.configure(state=NORMAL)

        Booking_Page_Func.entry_checkout_date.delete(0, END)

        Booking_Page_Func.entry_checkout_date.insert(0, "YYYY-MM-DD")

        Booking_Page_Func.entry_checkout_date.configure(state=DISABLED)

        button_select_checkout_date.configure(state=NORMAL)

        Booking_Page_Func.entry_checkin_date.configure(state=NORMAL)

        Booking_Page_Func.entry_checkin_date.delete(0, END)

        Booking_Page_Func.entry_checkin_date.insert(0,
Booking_Page_Func.cal_checkin_date.selection_get())

        Booking_Page_Func.entry_checkin_date.configure(state=DISABLED)

        button_select_checkin_date.configure(state=NORMAL)
        label_check_in_3.place(x=425*wf,y=425*hf)
        label_check_in_4.place(x=300*wf, y=475*hf)


    def select_checkout_date():
        button_select_checkin_date.configure(state=NORMAL)
        Booking_Page_Func.entry_checkout_date.configure(state=NORMAL)
        Booking_Page_Func.entry_checkout_date.delete(0, END)
        Booking_Page_Func.entry_checkout_date.insert(0,
Booking_Page_Func.cal_checkout_date.selection_get())
        Booking_Page_Func.entry_checkout_date.configure(state=DISABLED)
        button_select_checkout_date.configure(state=NORMAL)
        label_check_out_3.place(x=1125*wf, y=425*hf)
        label_check_out_4.place(x=1025*wf, y=475*hf)


    def show_checkin_calendar():
        label_check_in_1.place_forget()
        label_check_in_2.place_forget()
        label_check_in_3.place_forget()
        label_check_in_4.place_forget()
        button_select_checkin_date.configure(state = DISABLED)
        button_select_checkout_date.configure(state=DISABLED)
        Booking_Page_Func.cal_checkin_date = Calendar(Booking_Page,
font=("Century Gothic",int(16*hf)), selectmode="day",cursor="hand2",
mindate=date.today() + timedelta(days=1))
```

```
        Booking_Page_Func.cal_checkin_date.place(x=375*wf,y=300*hf)

        button_choose_checkin_date.place(x=525*wf,y=600*hf)


    def show_checkout_calendar():

        if Booking_Page_Func.entry_checkin_date.get() != "YYYY-MM-DD":

            label_check_out_1.place_forget()

            label_check_out_2.place_forget()

            label_check_out_3.place_forget()

            label_check_out_4.place_forget()

            button_select_checkout_date.configure(state=DISABLED)

            button_select_checkin_date.configure(state=DISABLED)

            Booking_Page_Func.cal_checkout_date = Calendar(Booking_Page,
font=("Century Gothic",int(16*hf)),
selectmode="day",cursor="hand2",mindate=Booking_Page_Func.cal_checkin_date.
selection_get() + timedelta(days=1))

            Booking_Page_Func.cal_checkout_date.place(x=1075*wf,y=300*hf)

            button_choose_checkout_date.place(x=1225*wf, y=600*hf)

        else:

            messagebox.showerror("Resort Ivory Bliss","Please select check in date first.")



    label_check_in_1 = Label(Booking_Page,text="Please Select Check In
Date",bg="#00FFFF",fg = "navy", font=('Century Gothic', int(16*hf)))

    label_check_in_2 = Label(Booking_Page,text="Use The Above Button 'Select
Check-In Date'",bg="#00FFFF",fg = "navy", font=('Century Gothic', int(16*hf)))
```

```
    label_check_out_1 = Label(Booking_Page,text="Please Select Check Out Date",bg="#00FFFF",fg = "navy", font=('Century Gothic', int(16*hf)))
    label_check_out_2 = Label(Booking_Page,text="Use The Above Button 'Select Check-Out Date'",bg="#00FFFF",fg = "navy", font=('Century Gothic', int(16*hf)))
    label_check_in_3 = Label(Booking_Page, text="You Have Selected Check In Date",bg="#00FFFF",fg = "navy",font=('Century Gothic', int(16*hf)))
    label_check_in_4 = Label(Booking_Page,text="Click The 'Select Check-In Date' Button Again To Change",bg="#00FFFF",fg = "navy",font=('Century Gothic', int(16*hf)))
    label_check_out_3 = Label(Booking_Page, text="You have Selected Check Out Date",bg="#00FFFF",fg = "navy", font=('Century Gothic', int(16*hf)))
    label_check_out_4 = Label(Booking_Page, text="Click The 'Select Check-Out Date' Button Again To Change",bg="#00FFFF",fg = "navy",font=('Century Gothic', int(16*hf)))
    label_check_in_1.place(x=425*wf,y=425*hf)
    label_check_in_2.place(x=375*wf, y=475*hf)
    label_check_out_1.place(x=1125*wf, y=425*hf)
    label_check_out_2.place(x=1050*wf, y=475*hf)


    Booking_Page_Func.entry_checkin_date = Entry(Booking_Page, width=20, justify="center",disabledbackground="white",disabledforeground = "navy", font=('Century Gothic bold', int(16*hf)))
    Booking_Page_Func.entry_checkout_date = Entry(Booking_Page, width=20, justify="center",disabledbackground="white",disabledforeground = "navy", font=('Century Gothic bold', int(16*hf)))
    Booking_Page_Func.entry_checkin_date.configure(state=DISABLED)
    Booking_Page_Func.entry_checkout_date.configure(state=DISABLED)


    button_choose_checkin_date = Button(Booking_Page, text="Select",
command=lambda:
[select_checkin_date(),Booking_Page_Func.cal_checkin_date.place_forget(),butto
n_choose_checkin_date.place_forget()],bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold',
int(16*hf)))
    button_choose_checkout_date = Button(Booking_Page, text="Select",
command=lambda:
[select_checkout_date(),Booking_Page_Func.cal_checkout_date.place_forget(),b
utton_choose_checkout_date.place_forget()],bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold',
int(16*hf)))


    button_select_checkin_date = Button(Booking_Page, text="Select Check-In
Date", command= lambda :
show_checkin_calendar(),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold', int(16*hf)))
    button_select_checkout_date = Button(Booking_Page, text="Select Check-Out
Date", command= lambda :
show_checkout_calendar(),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Bookman Old Style bold', int(16*hf)))


    button_select_checkin_date.place(x=450*wf,y=200*hf)
    button_select_checkout_date.place(x=1150*wf,y=200*hf)
```

```
Booking_Page_Func.entry_checkin_date.place(x=450*wf,y=700*hf)
Booking_Page_Func.entry_checkout_date.place(x=1150*wf,y=700*hf)

Booking_Page_Func.entry_checkin_date.configure(state=NORMAL)
Booking_Page_Func.entry_checkout_date.configure(state=NORMAL)
Booking_Page_Func.entry_checkin_date.insert(0, "YYYY-MM-DD")
Booking_Page_Func.entry_checkout_date.insert(0, "YYYY-MM-DD")
Booking_Page_Func.entry_checkin_date.configure(state=DISABLED)
Booking_Page_Func.entry_checkout_date.configure(state=DISABLED)

time_am = []
time_pm = []
time_add_12_am = datetime.strptime("12AM", "%I%p")
time_add_12_pm = datetime.strptime("12PM", "%I%p")
time_am.append(time_add_12_am.time())
time_pm.append(time_add_12_pm.time())

for t_am in range(1, 12):
    time_add_am = datetime.strptime(str(t_am) + "AM", '%I%p')
    time_am.append(time_add_am.time())

for t_pm in range(1, 12):
    time_add_pm = datetime.strptime(str(t_pm) + "PM", '%I%p')
```

```
        time_pm.append(time_add_pm.time())


    total_time = time_am + time_pm

    Booking_Page_Func.check_in_time = StringVar()

    Booking_Page_Func.check_in_time.set("Please Select Check In Time")


    option_chk_in_time = OptionMenu(Booking_Page,
Booking_Page_Func.check_in_time, *total_time)

    option_chk_in_time.config(width=40,bg="navy",activebackground = "#00FFFF",fg
= "#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(16*hf)))

    option_chk_in_time.place(x=310*wf,y=800*hf)


    Booking_Page_Func.check_out_time = StringVar()

    Booking_Page_Func.check_out_time.set("Please Select Check Out Time")


    option_chk_out_time = OptionMenu(Booking_Page,
Booking_Page_Func.check_out_time, *total_time)

    option_chk_out_time.config(width=40,bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Century Gothic bold',
int(16*hf)))

    option_chk_out_time.place(x=1015*wf,y=800*hf)


    button_confirm = Button(Booking_Page, text="Confirm",
width=50,command=lambda:
confirm_details(Booking_Page_Func.entry_checkin_date.get(),Booking_Page_Func.
entry_checkout_date.get()),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(16*hf)))

    button_back_to_main_menu = Button(Booking_Page,text="Back",width=20,command = lambda : [close(Booking_Page),Main_Menu_Func()],bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(16*hf)))

    button_back_to_main_menu.place(x=400*wf,y=900*hf)

    button_confirm.place(x=850*wf,y=900*hf)


    Booking_Page.mainloop()

# ----------------------------------------------------------------------------------------------------------------------------

# Restaurant Menu Customer

def Restaurant_Menu_Page_Func():

    con = mysql.connector.connect(host="localhost", user="root", passwd="root", database="project")

    mycursor_services = con.cursor()

    mycursor_services.execute("SELECT ITEM, PRICE FROM services WHERE ID LIKE 'F%' AND STATUS = 'Available'")

    restaurant_menu = mycursor_services.fetchall()

    con.close()


    Restaurant_Menu_Page_Func.Entry_Quantity = []


    Restaurant_Menu_Page = Tk()
    sw = Restaurant_Menu_Page.winfo_screenwidth()
    sh = Restaurant_Menu_Page.winfo_screenheight()

    wf = sw / 1920
    hf = sh / 1080

    Restaurant_Menu_Page.config(bg="#00FFFF")
    Restaurant_Menu_Page.title("Resort Ivory Bliss - Restaurant Menu")
    Restaurant_Menu_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")
    Restaurant_Menu_Page.resizable(0,0)
    Restaurant_Menu_Page.state('zoomed')

    main_frame = Frame(Restaurant_Menu_Page, bg="#00FFFF")
    main_frame.pack(fill=BOTH, expand=1)
    my_canvas = Canvas(main_frame, bg="#00FFFF")
    my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
    my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)
    my_scrollbar.pack(side=RIGHT, fill=Y)
    my_canvas.configure(yscrollcommand=my_scrollbar.set)
    my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))
    second_frame = Frame(my_canvas,bg="#00FFFF")
    my_canvas.create_window((0, 0), window=second_frame, anchor=NW)
    label_main_heading = Label(second_frame, text="ORDER
FOOD",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(50*hf)),
pady=50*hf)

    label_main_heading.grid(row=0, column=0, columnspan=5)


    label_heading_dish = Label(second_frame, text="Dish",bg="#00FFFF",fg = "navy",
font=('Bookman Old Style bold',int(40*hf)), padx=200*wf, pady=50*hf)

    label_heading_price = Label(second_frame, text="Price",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold',int(40*hf)), padx=260*wf, pady=50*hf)

    label_heading_quantity = Label(second_frame, text="Quantity",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold',int(40*hf)), padx=210*wf, pady=50*hf)


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

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

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


    button_order = Button(second_frame, text="ORDER", width=30,

                    command=lambda: view_order("R", restaurant_menu,
Restaurant_Menu_Page_Func.Entry_Quantity,


Restaurant_Menu_Page),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",

                    font=('Century Gothic bold',int(30*hf)))

    button_back_to_services_page = Button(second_frame, text="Back",
                                command=lambda: [close(Restaurant_Menu_Page),
Services_Page_Func()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",
                                font=('Century Gothic bold',int(30*hf)))


    Placement_Row_For_Label_Dish = 4
    for label_dish in range(len(restaurant_menu)):
        label_dish = Label(second_frame,
text=restaurant_menu[label_dish][0],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(25*hf)), pady=30*hf)
        label_dish.grid(row=Placement_Row_For_Label_Dish, column=0)
        Placement_Row_For_Label_Dish += 2


    Placement_Row_For_Label_Price = 4
    for label_price in range(len(restaurant_menu)):
        label_price = Label(second_frame, text="Rs. " +
restaurant_menu[label_price][1],bg="#00FFFF",fg = "navy", font=('Century Gothic',
int(25*hf)), pady=30*hf)
        label_price.grid(row=Placement_Row_For_Label_Price, column=1)
        Placement_Row_For_Label_Price += 2


    Placement_Row_For_Button_Add = 4
    for button_add in range(len(restaurant_menu)):
        button_add = Button(second_frame, text="+", command=lambda
button_add=button_add: qty_add("R", button_add,
Restaurant_Menu_Page_Func.Entry_Quantity),bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy",
                    font=('Berlin Sans FB', int(25*hf)))
        button_add.grid(row=Placement_Row_For_Button_Add, column=4)
        Placement_Row_For_Button_Add += 2

    Placement_Row_For_Button_Sub = 4
    for button_sub in range(len(restaurant_menu)):
        button_sub = Button(second_frame, text="-", command=lambda
button_sub=button_sub: qty_sub("R", button_sub,

Restaurant_Menu_Page_Func.Entry_Quantity),bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy",
                    font=('Berlin Sans FB', int(25*hf)))
        button_sub.grid(row=Placement_Row_For_Button_Sub, column=2)
        Placement_Row_For_Button_Sub += 2

    Placement_Row_For_Entry_Quantity = 4
    for Entries in range(len(restaurant_menu)):
        Entries = Entry(second_frame, width=15, justify=CENTER, font=('Century Gothic',
int(25*hf)), disabledforeground="navy", disabledbackground="white")
        Restaurant_Menu_Page_Func.Entry_Quantity.append(Entries)
        Entries.grid(row=Placement_Row_For_Entry_Quantity, column=3)
        Entries.insert(0, "0")
        Entries.configure(state=DISABLED)

        Placement_Row_For_Entry_Quantity += 2


    label_space = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy",
pady=30*hf)

    label_space.grid(row=Placement_Row_For_Entry_Quantity, column=0,
columnspan=7, rowspan=2)


    button_order.grid(row=Placement_Row_For_Entry_Quantity + 2, column=1,
columnspan=4)

    button_back_to_services_page.grid(row=Placement_Row_For_Entry_Quantity + 2,
column=0)


    Restaurant_Menu_Page.mainloop()

# -----------------------------------------------------------------------------------------------------------

# Laundry Service Menu - Customer

def Laundry_Service_Page_Func():

    con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")

    mycursor_services = con.cursor()

    mycursor_services.execute("SELECT ITEM, PRICE FROM services WHERE ID LIKE 'L%'
AND STATUS = 'Available'")

    laundry_service = mycursor_services.fetchall()

    con.close()
    Laundry_Service_Page_Func.Entry_Quantity = []


    Laundry_Service_Page = Tk()

    sw = Laundry_Service_Page.winfo_screenwidth()

    sh = Laundry_Service_Page.winfo_screenheight()


    wf = sw / 1920

    hf = sh / 1080


    Laundry_Service_Page.config(bg="#00FFFF")

    Laundry_Service_Page.title("Resort Ivory Bliss - Laundry Menu")

    Laundry_Service_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")

    Laundry_Service_Page.resizable(0,0)

    Laundry_Service_Page.state('zoomed')


    main_frame = Frame(Laundry_Service_Page, bg="#00FFFF")

    main_frame.pack(fill=BOTH, expand=1)

    my_canvas = Canvas(main_frame, bg="#00FFFF")

    my_canvas.pack(side=LEFT, fill=BOTH, expand=1)

    my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)

    my_scrollbar.pack(side=RIGHT, fill=Y)

    my_canvas.configure(yscrollcommand=my_scrollbar.set)
    my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))

    second_frame = Frame(my_canvas,bg="#00FFFF")

    my_canvas.create_window((0, 0), window=second_frame, anchor=NW)


    label_main_heading = Label(second_frame, text="ORDER  LAUNDRY
SERVICE",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold',int(45*hf)),
pady=50*hf)

    label_main_heading.grid(row=0, column=0, columnspan=5)


    label_heading_type = Label(second_frame, text="Type of
Clothing",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold',int(35*hf)),
padx=120*wf, pady=50*hf)

    label_heading_price = Label(second_frame, text="Price per
Clothing",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold',int(35*hf)),
padx=130*wf, pady=50*hf)

    label_heading_quantity = Label(second_frame, text="Quantity",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold',int(35*hf)), padx=155*wf, pady=50*hf)


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

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

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


    button_book = Button(second_frame, text="BOOK", width=30,

                     command=lambda: view_order("L", laundry_service,
Laundry_Service_Page_Func.Entry_Quantity,
                                    Laundry_Service_Page),bg="navy",activebackground
= "#00FFFF",fg = "#00FFFF",activeforeground = "navy",
                    font=('Century Gothic bold',int(25*hf)))
    button_back_to_services_page = Button(second_frame, text="Back",
                                command=lambda: [close(Laundry_Service_Page),
Services_Page_Func()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",
                                font=('Century Gothic bold',int(25*hf)))


    Placement_Row_For_Label_Dish = 4
    for label_dish in range(len(laundry_service)):
        label_dish = Label(second_frame,
text=laundry_service[label_dish][0],bg="#00FFFF",fg = "navy", font=('Century Gothic',
int(20*hf)), pady=30*hf)
        label_dish.grid(row=Placement_Row_For_Label_Dish, column=0)
        Placement_Row_For_Label_Dish += 2


    Placement_Row_For_Label_Price = 4
    for label_price in range(len(laundry_service)):
        label_price = Label(second_frame, text="Rs. " +
laundry_service[label_price][1],bg="#00FFFF",fg = "navy", font=('Century Gothic',
int(20*hf)), pady=30*hf)
        label_price.grid(row=Placement_Row_For_Label_Price, column=1)
        Placement_Row_For_Label_Price += 2
    Placement_Row_For_Button_Add = 4
    for button_add in range(len(laundry_service)):
        button_add = Button(second_frame, text="+", font=('Berlin Sans FB',
int(25*hf)),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground
= "navy",command=lambda button_add=button_add: qty_add("L", button_add,

Laundry_Service_Page_Func.Entry_Quantity))
        button_add.grid(row=Placement_Row_For_Button_Add, column=4)
        Placement_Row_For_Button_Add += 2


    Placement_Row_For_Button_Sub = 4
    for button_sub in range(len(laundry_service)):
        button_sub = Button(second_frame, text="-",font=('Berlin Sans FB',
int(25*hf)),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground
= "navy", command=lambda button_sub=button_sub: qty_sub("L", button_sub,

Laundry_Service_Page_Func.Entry_Quantity))
        button_sub.grid(row=Placement_Row_For_Button_Sub, column=2)
        Placement_Row_For_Button_Sub += 2


    Placement_Row_For_Entry_Quantity = 4
    for Entries in range(len(laundry_service)):
        Entries = Entry(second_frame, width=10, justify=CENTER, font=('Century Gothic',
int(20*hf)), disabledforeground="navy", disabledbackground="white")
        Laundry_Service_Page_Func.Entry_Quantity.append(Entries)
        Entries.grid(row=Placement_Row_For_Entry_Quantity, column=3)

        Entries.insert(0, "0")

        Entries.configure(state=DISABLED)

        Placement_Row_For_Entry_Quantity += 2


    label_space = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy",
pady=30*hf)

    label_space.grid(row=Placement_Row_For_Entry_Quantity, column=0,
rowspan=2, columnspan=7)


    button_book.grid(row=Placement_Row_For_Entry_Quantity + 2, column=1,
columnspan=4)

    button_back_to_services_page.grid(row=Placement_Row_For_Entry_Quantity + 2,
column=0)


    Laundry_Service_Page.mainloop()

# ---------------------------------------------------------------------------------------------------------------

# Amenities Menu - Customer

def Amenities_Page_Func():

    con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")

    mycursor_services = con.cursor()

    mycursor_services.execute("SELECT ITEM, PRICE FROM services WHERE ID LIKE
'A%'")

    amenities_menu = mycursor_services.fetchall()
    con.close()


    Amenities_Page_Func.Entry_Quantity = []


    Amenities_Page = Tk()
    sw = Amenities_Page.winfo_screenwidth()
    sh = Amenities_Page.winfo_screenheight()


    wf = sw / 1920
    hf = sh / 1080


    Amenities_Page.config(bg="#00FFFF")
    Amenities_Page.title("Resort Ivory Bliss - Amenities Menu")
    Amenities_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")
    Amenities_Page.resizable(0,0)
    Amenities_Page.state('zoomed')


    main_frame = Frame(Amenities_Page, bg="#00FFFF")
    main_frame.pack(fill=BOTH, expand=1)
    my_canvas = Canvas(main_frame, bg="#00FFFF")
    my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
    my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)
```

```
    my_scrollbar.pack(side=RIGHT, fill=Y)

    my_canvas.configure(yscrollcommand=my_scrollbar.set)

    my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))

    second_frame = Frame(my_canvas,bg="#00FFFF")

    my_canvas.create_window((0, 0), window=second_frame, anchor=NW)


    label_main_heading = Label(second_frame, text="BOOK
AMENITIES",bg="#00FFFF",fg = "navy",  font=('Bookman Old Style bold',int(50*hf)),
pady=50*hf)

    label_main_heading.grid(row=0, column=0, columnspan=5)


    label_heading_amenity = Label(second_frame, text="Amenity",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold',int(40*hf)), padx=200*wf, pady=50*hf)

    label_heading_price = Label(second_frame, text="Price",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold',int(40*hf)),padx=220*wf, pady=50*hf)

    label_heading_hours = Label(second_frame, text="Hours",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold',int(40*hf)), padx=250*wf, pady=50*hf)


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

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

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


    button_book = Button(second_frame, text="BOOK", width=30,
                    command=lambda: view_order("A", amenities_menu,
Amenities_Page_Func.Entry_Quantity,
                                        Amenities_Page),bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy",
                    font=('Century Gothic', int(30*hf)))
   button_back_to_services_page = Button(second_frame, text="Back",
                                command=lambda: [close(Amenities_Page),
Services_Page_Func()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",
                                font=('Century Gothic', int(30*hf)))


   Placement_Row_For_Label_Dish = 4
   for label_dish in range(len(amenities_menu)):
      label_dish = Label(second_frame,
text=amenities_menu[label_dish][0],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(25*hf)), pady=30*hf)
      label_dish.grid(row=Placement_Row_For_Label_Dish, column=0)
      Placement_Row_For_Label_Dish += 2


   Placement_Row_For_Label_Price = 4
   for label_price in range(len(amenities_menu)):
      label_price = Label(second_frame, text="Rs. " +
amenities_menu[label_price][1],bg="#00FFFF",fg = "navy", font=('Century Gothic',
int(25*hf)), pady=30*hf)
      label_price.grid(row=Placement_Row_For_Label_Price, column=1)
```

```
        Placement_Row_For_Label_Price += 2


    Placement_Row_For_Button_Add = 4
    for button_add in range(len(amenities_menu)):
        button_add = Button(second_frame, text="+",
                        command=lambda button_add=button_add: qty_add("A",
button_add,Amenities_Page_Func.Entry_Quantity),
                        font=('Berlin Sans FB', int(25*hf)),bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy")
        button_add.grid(row=Placement_Row_For_Button_Add, column=4)
        Placement_Row_For_Button_Add += 2


    Placement_Row_For_Button_Sub = 4
    for button_sub in range(len(amenities_menu)):
        button_sub = Button(second_frame, text="-",
                        command=lambda button_sub=button_sub: qty_sub("A",
button_sub,Amenities_Page_Func.Entry_Quantity),
                        font=('Berlin Sans FB', int(25*hf)),bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy")
        button_sub.grid(row=Placement_Row_For_Button_Sub, column=2)
        Placement_Row_For_Button_Sub += 2


    Placement_Row_For_Entry_Quantity = 4
    for Entries in range(len(amenities_menu)):
        Entries = Entry(second_frame, width=10, justify=CENTER, font=('Century Gothic',
int(25*hf)), disabledforeground="navy", disabledbackground="white")
        Amenities_Page_Func.Entry_Quantity.append(Entries)
        Entries.grid(row=Placement_Row_For_Entry_Quantity, column=3)
        Entries.insert(0, "0.0")
        Entries.configure(state=DISABLED)
        Placement_Row_For_Entry_Quantity += 2

    label_space = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy",
pady=30*hf)
    label_space.grid(row=Placement_Row_For_Entry_Quantity, column=0,
rowspan=2, columnspan=7)

    button_book.grid(row=Placement_Row_For_Entry_Quantity + 2, column=1,
columnspan=4)
    button_back_to_services_page.grid(row=Placement_Row_For_Entry_Quantity + 2,
column=0)

    Amenities_Page.mainloop()
# ------------------------------------------------------------------------------------------------------------------
# View Order Func
def view_order(id, data_list, entry_list, page):
    order_service = []
    total = 0
    def back_to_ordering(id_req):
        if id_req == "R":
            close(Order)
            Restaurant_Menu_Page_Func()
        elif id_req == "L":
            close(Order)
            Laundry_Service_Page_Func()
        elif id_req == "A":
            close(Order)
            Amenities_Page_Func()

    for data in range(len(entry_list)):
        if float(entry_list[data].get()) != 0.0:
            order_service.append([data_list[data][0], data_list[data][1],
entry_list[data].get(),
                                  str(float(entry_list[data].get()) * (float(data_list[data][1])))])
            total += float(entry_list[data].get()) * (float(data_list[data][1]))

        gst = round((0.18 * total), 2)
        gtotal = total + gst

    if len(order_service) != 0:
        close(page)
        Order = Tk()


        sw = Order.winfo_screenwidth()

        sh = Order.winfo_screenheight()


        wf = sw / 1920

        hf = sh / 1080


        Order.config(bg="#00FFFF")

        Order.state('zoomed')

        Order.resizable(0,0)


        main_frame = Frame(Order, bg="#00FFFF")

        main_frame.pack(fill=BOTH, expand=1)

        my_canvas = Canvas(main_frame, bg="#00FFFF")

        my_canvas.pack(side=LEFT, fill=BOTH, expand=1)

        my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)

        my_scrollbar.pack(side=RIGHT, fill=Y)

        my_canvas.configure(yscrollcommand=my_scrollbar.set)

        my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))

        second_frame = Frame(my_canvas,bg="#00FFFF")

        my_canvas.create_window((0, 0), window=second_frame, anchor=NW)
```

```
    def confirm_payment(id_req):
        if id_req == "R" or id_req == "L":
            confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss", "Do you
want to confirm your order?",
                                                     parent=Order)
        elif id_req == "A":
            confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss", "Do you
want to confirm your booking?",
                                                     parent=Order)
        if confirm_yes_or_no == "yes":
            if id_req == "R" or id_req == "L":
                messagebox.showinfo("Resort Ivory Bliss",
                                    "Order placed successfully. A copy of the bill has been sent
to your mail.",
                                    parent=Order)
            elif id_req == "A":
                messagebox.showinfo("Resort Ivory Bliss",
                                    "Booking successfully. A copy of the bill has been sent to
your mail.",
                                    parent=Order)
            close(Order)
            stays()
        else:
            pass

        if id == "R":

            Order.title("Resort Ivory Bliss - Food Order")
            Order.iconbitmap("D:\\pythonProject\\HOTEL.ico")

            label_main_heading = Label(second_frame, text="YOUR
ORDER",bg="#00FFFF",fg = "navy", font=("Bookman Old Style bold", int(50*hf)),
pady=50*hf)
            label_main_heading.grid(row=0, column=0, columnspan=4)

            label_heading_1 = Label(second_frame, text="Dish",bg="#00FFFF",fg = "navy",
font=("Bookman Old Style bold", int(40*hf)), pady=50*hf, padx=160*wf)
            label_heading_2 = Label(second_frame, text="Rate",bg="#00FFFF",fg =
"navy", font=("Bookman Old Style bold", int(40*hf)), pady=50*hf, padx=125*wf)
            label_heading_3 = Label(second_frame, text="Quantity",bg="#00FFFF",fg =
"navy", font=("Bookman Old Style bold", int(40*hf)), pady=50*hf, padx=125*wf)
            label_heading_4 = Label(second_frame, text="Price in Rs.",bg="#00FFFF",fg =
"navy", font=("Bookman Old Style bold", int(40*hf)), pady=50*hf,padx=125*wf)

        elif id == "L":

            Order.title("Resort Ivory Bliss - Laundry Order")
            Order.iconbitmap("D:\\pythonProject\\HOTEL.ico")
            label_main_heading = Label(second_frame, text="YOUR
ORDER",bg="#00FFFF",fg = "navy", font=("Bookman Old Style bold", int(50*hf)),
pady=50*hf)

            label_main_heading.grid(row=0, column=0, columnspan=4)


            label_heading_1 = Label(second_frame, text="Type of
clothing",bg="#00FFFF",fg = "navy", font=("Bookman Old Style bold", int(28*hf)),
pady=50*hf, padx=60*wf)

            label_heading_2 = Label(second_frame, text="Cost per Piece of
Clothing",bg="#00FFFF",fg = "navy", font=("Bookman Old Style bold", int(28*hf)),
pady=50*hf, padx=50*wf)

            label_heading_3 = Label(second_frame, text="Number of
Clothes",bg="#00FFFF",fg = "navy", font=("Bookman Old Style bold", int(28*hf)),
pady=50*hf, padx=55*wf)

            label_heading_4 = Label(second_frame, text="Price in Rs.",bg="#00FFFF",fg =
"navy", font=("Bookman Old Style bold", int(28*hf)), pady=50*hf, padx=50*wf)


        elif id == "A":


            Order.title("Resort Ivory Bliss - Amenities Booked")

            Order.iconbitmap("D:\\pythonProject\\HOTEL.ico")


            label_main_heading = Label(second_frame, text="YOUR
BOOKINGS",bg="#00FFFF",fg = "navy", font=("Bookman Old Style bold", int(50*hf)),
pady=50*hf)
```

```
        label_main_heading.grid(row=0, column=0, columnspan=4)


        label_heading_1 = Label(second_frame, text="Amenity",bg="#00FFFF",fg =
"navy", font=("Bookman Old Style bold", int(30*hf)), pady=50*hf, padx=110*wf)
        label_heading_2 = Label(second_frame, text="Price per
hour",bg="#00FFFF",fg = "navy", font=("Bookman Old Style bold", int(30*hf)),
pady=50*hf, padx=100*wf)
        label_heading_3 = Label(second_frame, text="Number of
hours",bg="#00FFFF",fg = "navy", font=("Bookman Old Style bold", int(30*hf)),
pady=50*hf, padx=100*wf)
        label_heading_4 = Label(second_frame, text="Price in Rs.",bg="#00FFFF",fg =
"navy", font=("Bookman Old Style bold", int(30*hf)), pady=50*hf, padx=100*wf)


    label_heading_1.grid(row=2, column=0)
    label_heading_2.grid(row=2, column=1)
    label_heading_3.grid(row=2, column=2)
    label_heading_4.grid(row=2, column=3)


    Placement_Row_For_Label_1 = 4
    for label_1 in range(len(order_service)):
        label_1 = Label(second_frame,
text=order_service[label_1][0],bg="#00FFFF",fg = "navy", font=("Century Gothic",
int(20*hf)), pady=30*hf)
        label_1.grid(row=Placement_Row_For_Label_1, column=0)
        Placement_Row_For_Label_1 += 2
    Placement_Row_For_Label_2 = 4
    for label_2 in range(len(order_service)):
        label_2 = Label(second_frame,
text=order_service[label_2][1],bg="#00FFFF",fg = "navy", font=("Century Gothic",
int(20*hf)), pady=30*hf)
        label_2.grid(row=Placement_Row_For_Label_2, column=1)
        Placement_Row_For_Label_2 += 2


    Placement_Row_For_Label_3 = 4
    for label_3 in range(len(order_service)):
        label_3 = Label(second_frame,
text=order_service[label_3][2],bg="#00FFFF",fg = "navy", font=("Century Gothic",
int(20*hf)), pady=30*hf)
        label_3.grid(row=Placement_Row_For_Label_3, column=2)
        Placement_Row_For_Label_3 += 2


    Placement_Row_For_Label_4 = 4
    for label_4 in range(len(order_service)):
        label_4 = Label(second_frame,
text=order_service[label_4][3],bg="#00FFFF",fg = "navy", font=("Century Gothic",
int(20*hf)), pady=30*hf)
        label_4.grid(row=Placement_Row_For_Label_4, column=3)
        Placement_Row_For_Label_4 += 2
        label_heading_total = Label(second_frame, text="Total",bg="#00FFFF",fg = 
"navy", font=("Century Gothic bold", int(25*hf)), pady=30*hf)

        label_total = Label(second_frame, text=str(total),bg="#00FFFF",fg = "navy", 
font=("Century Gothic", int(20*hf)), pady=30*hf)

        label_heading_total.grid(row=Placement_Row_For_Label_4, column=2)

        label_total.grid(row=Placement_Row_For_Label_4, column=3)


        label_heading_gst = Label(second_frame, text="GST (18%)",bg="#00FFFF",fg = 
"navy", font=("Century Gothic bold", int(25*hf)), pady=30*hf)

        label_gst = Label(second_frame, text=str(gst),bg="#00FFFF",fg = "navy", 
font=("Century Gothic", int(20*hf)), pady=30*hf)

        label_heading_grand_total = Label(second_frame, text="Grand 
Total",bg="#00FFFF",fg = "navy", font=("Century Gothic bold", int(25*hf)), pady=30*hf)

        label_grand_total = Label(second_frame, text=str(gtotal),bg="#00FFFF",fg = 
"navy", font=("Century Gothic", int(20*hf)), pady=30*hf)


        label_heading_gst.grid(row=Placement_Row_For_Label_4+2, column=2)

        label_gst.grid(row=Placement_Row_For_Label_4+2, column=3)

        label_heading_grand_total.grid(row=Placement_Row_For_Label_4+4, 
column=2)

        label_grand_total.grid(row=Placement_Row_For_Label_4+4, column=3)


        label_space = Label(second_frame, text='\t',bg="#00FFFF",fg = "navy", 
pady=30*hf)

        label_space.grid(row=Placement_Row_For_Label_4 + 6, column=0, rowspan=2)
        button_confirm = Button(second_frame, text="Confirm", command=lambda:
confirm_payment(id),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=("Century Gothic bold", int(25*hf)),
width=30)

        button_back_to_order_page = Button(second_frame, text="Back",
command=lambda: back_to_ordering(id),bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=("Century Gothic bold",
int(25*hf)))

        button_confirm.grid(row=Placement_Row_For_Label_4 + 8, column=1,
columnspan=3)

        button_back_to_order_page.grid(row=Placement_Row_For_Label_4 + 8,
column=0)


        Order.mainloop()


    else:

        if id == "R":

            messagebox.showerror("Resort Ivory Bliss", "Please select a dish.")

        elif id == "L":

            messagebox.showerror("Resort Ivory Bliss", "Please select the desired
garments.")

        elif id == "A":

            messagebox.showerror("Resort Ivory Bliss", "Please select an amenity.")

# --------------------------------------------------------------------------------------------------------------------

# Add Quantity Func
```

```
def qty_add(id, n, list):
    if id == "A":
        number = float(list[n].get())
        number += 0.5

    else:
        number = int(list[n].get())
        number += 1

    list[n].configure(state=NORMAL)
    list[n].delete(0, END)
    list[n].insert(0, str(number))
    list[n].configure(state=DISABLED)
# -------------------------------------------------------------------------------------------------------------------
# Subtract Quantity Func
def qty_sub(id, n, list):
    if id == "A":
        number = float(list[n].get())
        if number > 0:
            number -= 0.5
        else:
            messagebox.showerror("Resort Ivory Bliss", "Minimum quantity reached.")
    else:
        number = int(list[n].get())
        if number > 0:
            number -= 1
        else:
            messagebox.showerror("Resort Ivory Bliss", "Minimum quantity reached.")

    list[n].configure(state=NORMAL)
    list[n].delete(0, END)
    list[n].insert(0, str(number))
    list[n].configure(state=DISABLED)
# ---------------------------------------------------------------------------------------------------------------------------
# Stays Page
def stays():
    global ID, Name

    customer_id = ID
    booked_room_info = []
    booked_room_info_of_this_customer = []
    this_customer_previous_stays = []
    this_customer_upcoming_stays = []
    this_customer_current_stays = []
    con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")

    cursor_for_stays = con.cursor()

    cursor_for_stays.execute("SELECT
rl.SERIAL_NO,rl.ROOM_NO,ri.ROOM_TYPE,ri.AC_NON_AC,rl.CID,rl.CUSTOMER_NAME,rl.
CHECK_IN_DATE,rl.CHECK_OUT_DATE,rl.CHECK_IN_TIME,rl.CHECK_OUT_TIME\

                                                        FROM record_log as rl,room_info as ri\

                                                        WHERE rl.ROOM_NO = ri.ROOM_NO AND rl.STATUS
!= 'Cancelled'")

    for booking_id in cursor_for_stays.fetchall():

        booked_room_info.append(booking_id)


    con.close()


    for each_booking in booked_room_info:

        if customer_id == each_booking[4]:

            booked_room_info_of_this_customer.append(each_booking)


    def checkout_now(n):

        confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss",

                                                 "Are you sure you want to check out? This booking is
not refundable.",

                                                 parent=Your_Stays_Page)


        if confirm_yes_or_no == "yes":
```

```
        if datetime.now().hour < 10:
            req_time =  "0" + str(datetime.now().hour+1) + ":00:00"

            command = "UPDATE record_log SET CHECK_OUT_DATE = '" + str(
                datetime.now().date()) + "', CHECK_OUT_TIME ='" + req_time + "' WHERE
CID='" + ID + "' AND ROOM_NO = '" + \
                      this_customer_current_stays[n][1] + "' AND CHECK_IN_DATE= '" +
this_customer_current_stays[n][
                          6] + "'"

        elif datetime.now().hour >=10 and datetime.now().hour < 23 :
            req_time = str(datetime.now().hour + 1) + ":00:00"

            command = "UPDATE record_log SET CHECK_OUT_DATE = '" + str(
                datetime.now().date()) + "', CHECK_OUT_TIME ='" + req_time + "' WHERE
CID='" + ID + "' AND ROOM_NO = '" + \
                      this_customer_current_stays[n][1] + "' AND CHECK_IN_DATE= '" +
this_customer_current_stays[n][
                          6] + "'"

        else:
            req_time = "00:00:00"
            d = str(date.today() + timedelta(days=1))
            command = "UPDATE record_log SET CHECK_OUT_DATE = '" + d + "',
CHECK_OUT_TIME ='" + req_time + "' WHERE CID='" + ID + "' AND ROOM_NO = '" + \
                      this_customer_current_stays[n][1] + "' AND CHECK_IN_DATE= '" +
this_customer_current_stays[n][
                          6] + "'"


          con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")

          mycursor = con.cursor()


          mycursor.execute(command)

          con.commit()

          con.close()


          messagebox.showinfo("Resort Ivory Bliss",
                              "You have checked out successfully. You check out time is " +
req_time + ". Thank you for staying with us.",
                              parent=Your_Stays_Page)

          close(Your_Stays_Page)

          Main_Menu_Func()

      else:

          pass


   def display_previous_stays():
      if len(this_customer_previous_stays) == 0:
          Previous_Stays_Page = Tk()
          sw =  Previous_Stays_Page.winfo_screenwidth()
          sh =  Previous_Stays_Page.winfo_screenheight()

          wf = sw / 1920
          hf = sh / 1080

          Previous_Stays_Page.config(bg="#00FFFF")
          Previous_Stays_Page.resizable(0, 0)
          Previous_Stays_Page.state('zoomed')
          Previous_Stays_Page.title("Resort Ivory Bliss - Your Previous Stays")
          Previous_Stays_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")

          label_heading_prvious_stay = Label(Previous_Stays_Page, text="YOUR
PREVIOUS STAYS",bg="#00FFFF",fg = "navy",justify = CENTER,
                                    font=('Bookman Old Style bold', int(60*hf)))
          label_heading_prvious_stay.place(x=0*wf, y=300*hf,width=sw)
          label_no_previous_stay = Label(Previous_Stays_Page,
                                   text="Sorry " + Name + ", You Do Not Have Any Previous
Stays With Us",bg="#00FFFF",fg = "navy",justify = CENTER,
                                   font=('Century Gothic', int(25*hf)))
          label_no_previous_stay.place(x=0*wf, y=500*hf,width=sw)
```

```
            button_back = Button(Previous_Stays_Page, text="Back",
                                 command=lambda: [close(Previous_Stays_Page),
stays()],bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy",
                                 font=('Century Gothic bold', int(16*hf)))
            button_back.place(x=895*wf, y=700*hf,width=100*wf)


            Previous_Stays_Page.mainloop()


        else:
            Previous_Stays_Page = Tk()
            sw =  Previous_Stays_Page.winfo_screenwidth()
            sh =  Previous_Stays_Page.winfo_screenheight()



            wf = sw / 1920
            hf = sh / 1080


            Previous_Stays_Page.config(bg="#00FFFF")
            Previous_Stays_Page.resizable(0, 0)
            Previous_Stays_Page.state('zoomed')
            Previous_Stays_Page.title("Resort Ivory Bliss - Your Previous Stays")
            Previous_Stays_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")
            main_frame = Frame(Previous_Stays_Page, bg="#00FFFF")
            main_frame.pack(fill=BOTH, expand=1)
            my_canvas = Canvas(main_frame, bg="#00FFFF")
            my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
            my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)
            my_scrollbar.pack(side=RIGHT, fill=Y)
            my_canvas.configure(yscrollcommand=my_scrollbar.set)
            my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))
            second_frame = Frame(my_canvas,bg="#00FFFF")
            my_canvas.create_window((0, 0), window=second_frame, anchor=NW)


            label_heading_previous_stay = Label(second_frame, text="   YOUR
PREVIOUS  STAYS",bg="#00FFFF",fg = "navy",
                              font=('Bookman Old Style bold', int(50*hf)), pady=50*hf)
            label_previous_stay_room_number_heading = Label(second_frame,
text="Room Number",bg="#00FFFF",fg = "navy",
                                     font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=35*wf)
            label_previous_stay_room_type_heading = Label(second_frame, text="Room
Type",bg="#00FFFF",fg = "navy",
                                   font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=35*wf)
```

```
        label_previous_stay_room_ac_nonac_heading = Label(second_frame,
text="AC / Non AC",bg="#00FFFF",fg = "navy",
                                                font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=35*wf)

        label_previous_stay_room_check_in_date_heading = Label(second_frame,
text="Check In Date",bg="#00FFFF",fg = "navy",
                                                    font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=35*wf)

        label_previous_stay_room_check_in_time_heading = Label(second_frame,
text="Check In Time",bg="#00FFFF",fg = "navy",
                                                    font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=35*wf)

        label_previous_stay_room_check_out_date_heading = Label(second_frame,
text="Check Out Date",bg="#00FFFF",fg = "navy",
                                                     font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=35*wf)

        label_previous_stay_room_check_out_time_heading = Label(second_frame,
text="Check Out Time",bg="#00FFFF",fg = "navy",
                                                     font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=35*wf)


        label_heading_previous_stay.grid(row=0, column=0, columnspan=7,
rowspan=2)

        label_previous_stay_room_number_heading.grid(row=2, column=0,
rowspan=2)

        label_previous_stay_room_type_heading.grid(row=2, column=1, rowspan=2)
        label_previous_stay_room_ac_nonac_heading.grid(row=2, column=2,
rowspan=2)
        label_previous_stay_room_check_in_date_heading.grid(row=2, column=3,
rowspan=2)
        label_previous_stay_room_check_in_time_heading.grid(row=2, column=4,
rowspan=2)
        label_previous_stay_room_check_out_date_heading.grid(row=2, column=5,
rowspan=2)
        label_previous_stay_room_check_out_time_heading.grid(row=2, column=6,
rowspan=2)


        Placement_Row_For_Room_Number = 4
        for label_previous_stay_room_number in
range(len(this_customer_previous_stays)):
            label_previous_stay_room_number = Label(second_frame, text=

this_customer_previous_stays[label_previous_stay_room_number][1],bg="#00FFFF",fg
= "navy", font=('Century Gothic', int(16*hf)),
                                          pady=30*hf)

label_previous_stay_room_number.grid(row=Placement_Row_For_Room_Number,
column=0, rowspan=2)
            Placement_Row_For_Room_Number += 2


        Placement_Row_For_Room_Type = 4
        for label_previous_stay_room_type in
range(len(this_customer_previous_stays)):
            label_previous_stay_room_type = Label(second_frame,

text=this_customer_previous_stays[label_previous_stay_room_type][2],bg="#00FFFF",
fg = "navy",
                                        font=('Century Gothic', int(16*hf)), pady=30*hf)

label_previous_stay_room_type.grid(row=Placement_Row_For_Room_Type,
column=1 ,rowspan=2)
            Placement_Row_For_Room_Type += 2


        Placement_Row_For_Room_Type_AC_Nonac = 4
        for label_previous_stay_room_type_ac_nonac in
range(len(this_customer_previous_stays)):
            label_previous_stay_room_type_ac_nonac = Label(second_frame, text=

this_customer_previous_stays[label_previous_stay_room_type_ac_nonac][3],bg="#0
0FFFF",fg = "navy", font=('Century Gothic', int(16*hf)),
                                              pady=30*hf)

label_previous_stay_room_type_ac_nonac.grid(row=Placement_Row_For_Room_Ty
pe_AC_Nonac, column=2, rowspan=2)
            Placement_Row_For_Room_Type_AC_Nonac += 2


        Placement_Row_For_Check_In_Date = 4
        for label_previous_stay_room_check_in_date in
range(len(this_customer_previous_stays)):
            label_previous_stay_room_check_in_date = Label(second_frame, text=

this_customer_previous_stays[label_previous_stay_room_check_in_date][6],bg="#00
FFFF",fg = "navy", font=('Century Gothic',int(16*hf)),
                                                pady=30*hf)

label_previous_stay_room_check_in_date.grid(row=Placement_Row_For_Check_In_
Date, column=3, rowspan=2)
            Placement_Row_For_Check_In_Date += 2


        Placement_Row_For_Check_In_Time = 4
        for label_previous_stay_room_check_in_time in
range(len(this_customer_previous_stays)):
            label_previous_stay_room_check_in_time = Label(second_frame, text=

this_customer_previous_stays[label_previous_stay_room_check_in_time][8],bg="#00
FFFF",fg = "navy", font=('Century Gothic', int(16*hf)),
                                                pady=30*hf)

label_previous_stay_room_check_in_time.grid(row=Placement_Row_For_Check_In_
Time, column=4, rowspan=2)
            Placement_Row_For_Check_In_Time += 2


        Placement_Row_For_Check_Out_Date = 4
        for label_previous_stay_room_check_out_date in
range(len(this_customer_previous_stays)):
            label_previous_stay_room_check_out_date = Label(second_frame, text=

this_customer_previous_stays[label_previous_stay_room_check_out_date][7],bg="#
00FFFF",fg = "navy",
                                                    font=('Century Gothic', int(16*hf)), pady=30*hf)

label_previous_stay_room_check_out_date.grid(row=Placement_Row_For_Check_
Out_Date, column=5)
            Placement_Row_For_Check_Out_Date += 2


        Placement_Row_For_Check_Out_Time = 4
        for label_previous_stay_room_check_out_time in
range(len(this_customer_previous_stays)):
            label_previous_stay_room_check_out_time = Label(second_frame, text=

this_customer_previous_stays[label_previous_stay_room_check_out_time][9],bg="#0
0FFFF",fg = "navy",
                                                    font=('Century Gothic',int(16*hf)), pady=30*hf)

label_previous_stay_room_check_out_time.grid(row=Placement_Row_For_Check_O
ut_Time, column=6)
            Placement_Row_For_Check_Out_Time += 2
        label_space = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy",
pady=30*hf)

        label_space.grid(row=Placement_Row_For_Check_Out_Time, column=0,
rowspan=2)


        button_back = Button(second_frame, text="Back", command=lambda:
[close(Previous_Stays_Page), stays()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",

                                 font=('Century Gothic bold', int(20*hf)))

        button_back.grid(row=Placement_Row_For_Check_Out_Time + 2, column=0)


        Previous_Stays_Page.mainloop()


def display_upcoming_stays():

    def cancel_booking(key,ciny,cinm,cind):

        if int((date(ciny,cinm,cind) - date.today()).days) >=7:

            confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss",

                                                "Are you sure you want to cancel booking? 30%
of the booking fee will be deducted.",

                                                parent=Upcoming_Stays_Page)


            if confirm_yes_or_no == "yes":

                con = mysql.connector.connect(host="localhost", user="root",
passwd="root", database="project")

                cursor_for_stays = con.cursor()
                cursor_for_stays.execute("UPDATE record_log SET STATUS = 'Cancelled'
WHERE SERIAL_NO = " + str(key))

                con.commit()

                con.close()

                messagebox.showinfo("Resort Ivory Bliss", "Booking cancelled
successfully. 70% of the booking fee while be returned in one or two working days.",

                                    parent=Upcoming_Stays_Page)

                close(Upcoming_Stays_Page)

                stays()


            else:

                pass

        else:

            confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss",

                                                  "Are you sure you want to cancel booking?
Booking fee will not be refunded.",

                                                  parent=Upcoming_Stays_Page)


            if confirm_yes_or_no == "yes":

                con = mysql.connector.connect(host="localhost", user="root",
passwd="root", database="project")

                cursor_for_stays = con.cursor()

                cursor_for_stays.execute("UPDATE record_log SET STATUS = 'Cancelled'
WHERE SERIAL_NO = " + str(key))
```

```
                con.commit()

                con.close()

                messagebox.showinfo("Resort Ivory Bliss", "Booking cancelled
successfully.",
                                         parent=Upcoming_Stays_Page)

                close(Upcoming_Stays_Page)

                stays()


            else:

                pass


        for each_booking_of_this_customer in booked_room_info_of_this_customer:

            datetime_now = datetime.now()

            req_date_in_1 = each_booking_of_this_customer[6]

            req_date_list_1 = req_date_in_1.split("-")

            req_time_in_1 = each_booking_of_this_customer[8]

            req_time_list_1 = req_time_in_1.split(":")


            if datetime(int(req_date_list_1[0]), int(req_date_list_1[1]),
int(req_date_list_1[2]),
                          int(req_time_list_1[0]), int(req_time_list_1[1]), int(req_time_list_1[2])) >
datetime_now:

                this_customer_upcoming_stays.append(each_booking_of_this_customer)
```

```
    if len(this_customer_upcoming_stays) == 0:
        Upcoming_Stays_Page = Tk()
        sw = Upcoming_Stays_Page.winfo_screenwidth()
        sh = Upcoming_Stays_Page.winfo_screenheight()

        wf = sw / 1920
        hf = sh / 1080

        Upcoming_Stays_Page.config(bg="#00FFFF")
        Upcoming_Stays_Page.resizable(0, 0)
        Upcoming_Stays_Page.state('zoomed')
        Upcoming_Stays_Page.title("Resort Ivory Bliss - Your Upcoming Stays")
        Upcoming_Stays_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")

        label_heading_upcoming_stay = Label(Upcoming_Stays_Page, text="YOUR
UPCOMING STAYS",bg="#00FFFF",fg = "navy",justify = CENTER,
                                            font=('Bookman Old Style bold', int(60*hf)))
        label_heading_upcoming_stay.place(x=0*wf, y=300*hf,width=sw)
        label_no_upcoming_stay = Label(Upcoming_Stays_Page,
                                       text="Sorry " + Name + ", You Do Not Have Any Upcoming
Stays With Us",bg="#00FFFF",fg = "navy",justify = CENTER,
                                       font=('Century Gothic', int(25*hf)))
        label_no_upcoming_stay.place(x=0*wf, y=500*hf,width=sw)
        button_back = Button(Upcoming_Stays_Page, text="Back",
                             command=lambda: [close(Upcoming_Stays_Page),
stays()],bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground =
"navy",
                             font=('Bookman Old Style bold', int(16*hf)))
        button_back.place(x=895*wf, y=700*hf,width=100*wf)

        Upcoming_Stays_Page.mainloop()

      else:
        Upcoming_Stays_Page = Tk()
        sw = Upcoming_Stays_Page.winfo_screenwidth()
        sh = Upcoming_Stays_Page.winfo_screenheight()

        wf = sw / 1920
        hf = sh / 1080

        Upcoming_Stays_Page.config(bg="#00FFFF")
        Upcoming_Stays_Page.resizable(0, 0)
        Upcoming_Stays_Page.state('zoomed')
        Upcoming_Stays_Page.title("Resort Ivory Bliss - Your Upcoming Stays")
        Upcoming_Stays_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")

        main_frame = Frame(Upcoming_Stays_Page, bg="#00FFFF")
        main_frame.pack(fill=BOTH, expand=1)
        my_canvas = Canvas(main_frame, bg="#00FFFF")
        my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
        my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)
        my_scrollbar.pack(side=RIGHT, fill=Y)
        my_canvas.configure(yscrollcommand=my_scrollbar.set)
        my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))
        second_frame = Frame(my_canvas,bg="#00FFFF")
        my_canvas.create_window((0, 0), window=second_frame, anchor=NW)


        label_heading_upcoming_stay = Label(second_frame, text="  YOUR
UPCOMING  STAYS",
                                font=('Bookman Old Style bold',
int(50*hf)),bg="#00FFFF",fg = "navy", pady=50*hf)
        label_upcoming_stay_room_number_heading = Label(second_frame,
text="Room Number",bg="#00FFFF",fg = "navy",
                                        font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=20*wf)
        label_upcoming_stay_room_type_heading = Label(second_frame,
text="Room Type",bg="#00FFFF",fg = "navy",
                                      font=('Bookman Old Style bold',int(20*hf)),
pady=50*hf,padx=20*wf)
        label_upcoming_stay_room_ac_nonac_heading = Label(second_frame,
text="AC / Non AC",bg="#00FFFF",fg = "navy",
                                        font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=20*wf)
        label_upcoming_stay_room_check_in_date_heading = Label(second_frame,
text="Check In Date",bg="#00FFFF",fg = "navy",
                                           font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=20*wf)
        label_upcoming_stay_room_check_in_time_heading = Label(second_frame,
text="Check In Time",bg="#00FFFF",fg = "navy",
                                           font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=20*wf)
        label_upcoming_stay_room_check_out_date_heading =
Label(second_frame, text="Check Out Date",bg="#00FFFF",fg = "navy",
                                            font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=20*wf)
        label_upcoming_stay_room_check_out_time_heading =
Label(second_frame, text="Check Out Time",bg="#00FFFF",fg = "navy",
                                            font=('Bookman Old Style bold', int(20*hf)),
pady=50*hf,padx=20*wf)


        label_heading_upcoming_stay.grid(row=0, column=0, columnspan=7,
rowspan=2)
        label_upcoming_stay_room_number_heading.grid(row=2, column=0,
rowspan=2)
        label_upcoming_stay_room_type_heading.grid(row=2, column=1,
rowspan=2)
        label_upcoming_stay_room_ac_nonac_heading.grid(row=2, column=2,
rowspan=2)
```

```
        label_upcoming_stay_room_check_in_date_heading.grid(row=2, column=3,
rowspan=2)
        label_upcoming_stay_room_check_in_time_heading.grid(row=2, column=4,
rowspan=2)
        label_upcoming_stay_room_check_out_date_heading.grid(row=2,
column=5, rowspan=2)
        label_upcoming_stay_room_check_out_time_heading.grid(row=2,
column=6, rowspan=2)


        Placement_Row_For_Room_Number = 4
        for label_upcoming_stay_room_number in
range(len(this_customer_upcoming_stays)):
            label_upcoming_stay_room_number = Label(second_frame, text=

this_customer_upcoming_stays[label_upcoming_stay_room_number][1],bg="#00FFF
F",fg = "navy", font=('Century Gothic', int(16*hf)),
                                                  pady=30*hf)

label_upcoming_stay_room_number.grid(row=Placement_Row_For_Room_Number,
column=0, rowspan=2)
            Placement_Row_For_Room_Number += 2


        Placement_Row_For_Room_Type = 4
        for label_upcoming_stay_room_type in
range(len(this_customer_upcoming_stays)):
            label_upcoming_stay_room_type = Label(second_frame,
text=this_customer_upcoming_stays[label_upcoming_stay_room_type][2],bg="#00F
FFF",fg = "navy",
                                              font=('Century Gothic', int(16*hf)), pady=30*hf)

label_upcoming_stay_room_type.grid(row=Placement_Row_For_Room_Type,
column=1, rowspan=2)
            Placement_Row_For_Room_Type += 2


        Placement_Row_For_Room_Type_AC_Nonac = 4
        for label_upcoming_stay_room_type_ac_nonac in
range(len(this_customer_upcoming_stays)):
            label_upcoming_stay_room_type_ac_nonac = Label(second_frame, text=

this_customer_upcoming_stays[label_upcoming_stay_room_type_ac_nonac][3],bg
="#00FFFF",fg = "navy", font=('Century Gothic', int(16*hf)),
                                                     pady=30*hf)

label_upcoming_stay_room_type_ac_nonac.grid(row=Placement_Row_For_Room_
Type_AC_Nonac, column=2, rowspan=2)
            Placement_Row_For_Room_Type_AC_Nonac += 2


        Placement_Row_For_Check_In_Date = 4
        for label_upcoming_stay_room_check_in_date in
range(len(this_customer_upcoming_stays)):
            label_upcoming_stay_room_check_in_date = Label(second_frame, text=
this_customer_upcoming_stays[label_upcoming_stay_room_check_in_date][6],bg="
#00FFFF",fg = "navy", font=('Century Gothic', int(16*hf)),
                                    pady=30*hf)

label_upcoming_stay_room_check_in_date.grid(row=Placement_Row_For_Check_I
n_Date, column=3, rowspan=2)
        Placement_Row_For_Check_In_Date += 2


      Placement_Row_For_Check_In_Time = 4
      for label_upcoming_stay_room_check_in_time in
range(len(this_customer_upcoming_stays)):
        label_upcoming_stay_room_check_in_time = Label(second_frame, text=

this_customer_upcoming_stays[label_upcoming_stay_room_check_in_time][8],bg="
#00FFFF",fg = "navy", font=('Century Gothic', int(16*hf)),
                                    pady=30*hf)

label_upcoming_stay_room_check_in_time.grid(row=Placement_Row_For_Check_I
n_Time, column=4, rowspan=2)
        Placement_Row_For_Check_In_Time += 2


      Placement_Row_For_Check_Out_Date = 4
      for label_upcoming_stay_room_check_out_date in
range(len(this_customer_upcoming_stays)):
        label_upcoming_stay_room_check_out_date = Label(second_frame, text=
this_customer_upcoming_stays[label_upcoming_stay_room_check_out_date][7],bg
="#00FFFF",fg = "navy",
                                        font=('Century Gothic', int(16*hf)), pady=30*hf)

label_upcoming_stay_room_check_out_date.grid(row=Placement_Row_For_Check
_Out_Date, column=5, rowspan=2)
            Placement_Row_For_Check_Out_Date += 2


        Placement_Row_For_Check_Out_Time = 4
        for label_upcoming_stay_room_check_out_time in
range(len(this_customer_upcoming_stays)):
            label_upcoming_stay_room_check_out_time = Label(second_frame, text=

this_customer_upcoming_stays[label_upcoming_stay_room_check_out_time][9],bg
="#00FFFF",fg = "navy",
                                        font=('Century Gothic', int(16*hf)), pady=30)

label_upcoming_stay_room_check_out_time.grid(row=Placement_Row_For_Check
_Out_Time, column=6, rowspan=2)
            Placement_Row_For_Check_Out_Time += 2


        label_space = Label(second_frame, text='\t',bg="#00FFFF",fg = "navy",
pady=30*hf)
        label_space.grid(row=Placement_Row_For_Check_Out_Time, column=0,
rowspan=2)
```

```
        button_back = Button(second_frame, text="Back", command=lambda:
[close(Upcoming_Stays_Page), stays()],bg="navy",activebackground = "#00FFFF",fg
= "#00FFFF",activeforeground = "navy",

                        font=('Century Gothic bold', int(20*hf)))

        button_back.grid(row=Placement_Row_For_Check_Out_Time + 2, column=0,
rowspan=2)


        Placment_Row_For_Cancellation = 4

        for button_upcoming_stay_cancel_booking in
range(len(this_customer_upcoming_stays)):

            datetime_12_hours = datetime.now() + timedelta(hours=12)

            req_date_in_1 =
this_customer_upcoming_stays[button_upcoming_stay_cancel_booking][6]

            req_date_list_1 = req_date_in_1.split("-")

            req_time_in_1 =
this_customer_upcoming_stays[button_upcoming_stay_cancel_booking][8]

            req_time_list_1 = req_time_in_1.split(":")


            if datetime(int(req_date_list_1[0]), int(req_date_list_1[1]),
int(req_date_list_1[2]),

                    int(req_time_list_1[0]), int(req_time_list_1[1]),

                    int(req_time_list_1[2])) > datetime_12_hours:

                button_upcoming_stay_cancel_booking = Button(second_frame,
text="Cancel Booking", command=lambda
button_upcoming_stay_cancel_booking=button_upcoming_stay_cancel_booking:
cancel_booking(

this_customer_upcoming_stays[button_upcoming_stay_cancel_booking][0],int(req_
date_list_1[0]), int(req_date_list_1[1]),
int(req_date_list_1[2])),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",font=('Century Gothic bold', int(16*hf)))

button_upcoming_stay_cancel_booking.grid(row=Placment_Row_For_Cancellation
, column=7, rowspan=2)


                else:
                    label_non_cancellable = Label(second_frame, text="Going to\ncheck-
in soon",bg="#00FFFF",fg = "navy", font=('Century Gothic bold', int(16*hf)),
pady=30*hf)
                    label_non_cancellable.grid(row=Placment_Row_For_Cancellation,
column=7, rowspan=2)


                Placment_Row_For_Cancellation += 2


            Upcoming_Stays_Page.mainloop()


    for each_booking_of_this_customer in booked_room_info_of_this_customer:
        datetime_now = datetime.now()
        req_date_in_1 = each_booking_of_this_customer[6]

        req_date_list_1 = req_date_in_1.split("-")


        req_date_out_2 = each_booking_of_this_customer[7]

        req_date_list_2 = req_date_out_2.split("-")


        req_time_in_1 = each_booking_of_this_customer[8]

        req_time_list_1 = req_time_in_1.split(":")


        req_time_out_2 = each_booking_of_this_customer[9]

        req_time_list_2 = req_time_out_2.split(":")


        if datetime(int(req_date_list_1[0]), int(req_date_list_1[1]), int(req_date_list_1[2]),
int(req_time_list_1[0]),int(req_time_list_1[1]), int(req_time_list_1[2])) <= datetime_now
and datetime_now <= datetime(int(req_date_list_2[0]), int(req_date_list_2[1]),
int(req_date_list_2[2]), int(req_time_list_2[0]),int(req_time_list_2[1]),
int(req_time_list_2[2])):

            this_customer_current_stays.append(each_booking_of_this_customer)


        elif datetime_now > datetime(int(req_date_list_2[0]), int(req_date_list_2[1]),
int(req_date_list_2[2]),

                                    int(req_time_list_2[0]), int(req_time_list_2[1]),
int(req_time_list_2[2])):

            this_customer_previous_stays.append(each_booking_of_this_customer)
```

```
    Your_Stays_Page = Tk()

    sw = Your_Stays_Page.winfo_screenwidth()

    sh = Your_Stays_Page.winfo_screenheight()


    wf = sw / 1920

    hf = sh / 1080


    Your_Stays_Page.config(bg="#00FFFF")

    Your_Stays_Page.resizable(0, 0)

    Your_Stays_Page.state('zoomed')

    Your_Stays_Page.title("Resort Ivory Bliss - Your Stays")

    Your_Stays_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")


    if len(this_customer_current_stays):

        row = 2

        sno = 0

        main_frame = Frame(Your_Stays_Page, bg="#00FFFF")

        main_frame.pack(fill=BOTH, expand=1)

        my_canvas = Canvas(main_frame, bg="#00FFFF")

        my_canvas.pack(side=LEFT, fill=BOTH, expand=1)

        my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)

        my_scrollbar.pack(side=RIGHT, fill=Y)
        my_canvas.configure(yscrollcommand=my_scrollbar.set)

        my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))

        second_frame = Frame(my_canvas,bg="#00FFFF")

        my_canvas.create_window((0, 0), window=second_frame, anchor=NW)


        Placement_Label_Current_Stay = 2

        for label_current_stay in range(len(this_customer_current_stays)):

            label_current_stay = Label(second_frame,

                                    text=str(

                                        sno + 1) + "                Your current stay from " +
this_customer_current_stays[label_current_stay][6] + "(" +
this_customer_current_stays[label_current_stay][8] + ") to " +
this_customer_current_stays[label_current_stay][7] + "(" +
this_customer_current_stays[label_current_stay][9] + ") \n\n" \

                                        + "Room Number : " +
this_customer_current_stays[label_current_stay][

                                            1] + "\n\n" + " Room Type : " +
this_customer_current_stays[label_current_stay][

                                            2] + "\n\n" + " AC / Non AC : " +
this_customer_current_stays[label_current_stay][3],bg="#00FFFF",fg = "navy",

                                    font=('Century Gothic', int(16*hf)), padx=65*wf, pady=50*hf)

            label_current_stay.grid(row=Placement_Label_Current_Stay, column=0)


            Placement_Label_Current_Stay += 2
            row += 2

            sno += 1

        Placement_Button_Order = 2

        for button_order in range(len(this_customer_current_stays)):

            button_order = Button(second_frame, text="Order Services",

                                  command=lambda: [close(Your_Stays_Page),
Services_Page_Func()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",

                                  font=('Century Gothic bold', int(16*hf)))

            button_order.grid(row=Placement_Button_Order, column=2)

            label_spacing_1 = Label(second_frame, text="\t\t",bg="#00FFFF",fg = "navy")

            label_spacing_1.grid(row=Placement_Button_Order, column=1)

            Placement_Button_Order += 2


        Placement_Button_Checkout = 2

        for button_checkout in range(len(this_customer_current_stays)):

            datetime_now = datetime.now()

            req_date_in_1 = this_customer_current_stays[button_checkout][7]

            req_date_list_1 = req_date_in_1.split("-")

            req_time_in_1 = this_customer_current_stays[button_checkout][9]

            req_time_list_1 = req_time_in_1.split(":")


            label_spacing_2 = Label(second_frame, text="\t\t\t\t",bg="#00FFFF",fg =
"navy")
```

```
        label_spacing_2.grid(row=Placement_Button_Checkout, column=3)


        if datetime(int(req_date_list_1[0]), int(req_date_list_1[1]),
int(req_date_list_1[2]),int(req_time_list_1[0]),int(req_time_list_1[1]),
int(req_time_list_1[2])) > (datetime_now+timedelta(hours=1)) :
            button_checkout = Button(second_frame, text="Checkout",
                                     command=lambda button_checkout=button_checkout:
checkout_now(button_checkout),
                                     bg="navy", activebackground="#00FFFF", fg="#00FFFF",
activeforeground="navy",
                                     font=('Century Gothic bold', int(16*hf)))
            button_checkout.grid(row=Placement_Button_Checkout, column=4)


        else:
            label_checkout_soon = Label(second_frame, text="Going to\n check-out
soon", bg="#00FFFF", fg="navy",
                                        font=('Century Gothic bold', int(16*hf)))
            label_checkout_soon.grid(row=Placement_Button_Checkout, column=4)


        Placement_Button_Checkout += 2


    label_heading = Label(second_frame, text="Your Stays", font=('Bookman Old
Style bold', int(60*hf)),bg="#00FFFF",fg = "navy", justify=CENTER,
                          pady=50*hf)
    label_heading.grid(row=0, column=0, columnspan=5)
    button_previous_stays = Button(second_frame, text="Previous Stays", width=25,
                                   command=lambda: [close(Your_Stays_Page),
display_previous_stays()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",
                                   font=('Century Gothic bold', int(18*hf)))
    button_upcoming_stays = Button(second_frame, text="Upcoming Stays",
width=25,
                                   command=lambda: [close(Your_Stays_Page),
display_upcoming_stays()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",
                                   font=('Century Gothic bold', int(18*hf)))
    button_back_to_main_menu = Button(second_frame, text="Back", width=10,
                                      command=lambda: [close(Your_Stays_Page),
Main_Menu_Func()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",
                                      font=('Century Gothic bold', int(16*hf)))
    label_spacing_3 = Label(second_frame, text=" ",bg="#00FFFF",fg = "navy",
pady=30*hf)
    label_spacing_4 = Label(second_frame, text=" ",bg="#00FFFF",fg = "navy")


    button_previous_stays.grid(row=row + 2, column=0)
    button_upcoming_stays.grid(row=row + 2, column=2)
    button_back_to_main_menu.grid(row=row + 4, column=0)
    label_spacing_3.grid(row=row + 3, column=0, columnspan=5)
    label_spacing_4.grid(row=row + 1, column=0, columnspan=5)
else:
    Your_Stays_Page.config(bg="#00FFFF")
    label_heading = Label(Your_Stays_Page, text="Your Stays",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold', int(60*hf)))
    label_heading.place(x=745*wf, y=200*hf)
    button_previous_stays = Button(Your_Stays_Page, text="Previous Stays",
                                   command=lambda: [close(Your_Stays_Page),
display_previous_stays()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",
                                   font=('Century Gothic bold', int(25*hf)))
    button_previous_stays.place(x=500*wf, y=485*hf, width=400*wf)
    button_upcoming_stays = Button(Your_Stays_Page, text="Upcoming Stays",
                                   command=lambda: [close(Your_Stays_Page),
display_upcoming_stays()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",
                                   font=('Century Gothic bold', int(25*hf)))
    button_upcoming_stays.place(x=1025*wf, y=485*hf, width=400*wf)
    button_back_to_main_menu = Button(Your_Stays_Page, text="Back", width=10,
                                      command=lambda: [close(Your_Stays_Page),
Main_Menu_Func()],bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",
                                      font=('Century Gothic bold', int(20*hf)))
    button_back_to_main_menu.place(x=830*wf, y=700*hf, width=250*wf)
Your_Stays_Page.mainloop()
# -------------------------------------------------------------------------------------------------------------------
# Services Page
def Services_Page_Func():
services_page = Tk()
sw = services_page.winfo_screenwidth()
sh = services_page.winfo_screenheight()

wf = sw / 1920
hf = sh / 1080

services_page.config(bg="#00FFFF")
services_page.resizable(0, 0)
services_page.state('zoomed')
services_page.title("Resort Ivory Bliss - Services")
services_page.iconbitmap("D:\\pythonProject\\HOTEL.ico")

label_heading = Label(services_page,text="HOTEL IVORY BLISS -
SERVICES",bg="#00FFFF",fg = "navy",justify=CENTER, font=('Bookman Old Style bold',
int(60*hf)))
button_order_food = Button(services_page,text="Order
Food",width=20,command= lambda :
[close(services_page),Restaurant_Menu_Page_Func()],bg="navy",activebackgroun
d = "#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Century Gothic bold',
int(25*hf)))
```

```
    button_order_laundry_services = Button(services_page,text = "Order
Laundry",width=20,command = lambda :
[close(services_page),Laundry_Service_Page_Func()],bg="navy",activebackground
= "#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Century Gothic bold',
int(25*hf)))

    button_order_other_amenities = Button(services_page,text = "Book
Amenities",width=20,command= lambda :
[close(services_page),Amenities_Page_Func()],bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Century Gothic bold',
int(25*hf)))

    button_order_room_service = Button(services_page, text="Order Room Service",
width=20, command=lambda : messagebox.showinfo("Resort Ivory Bliss", "Room
Service booked successfully! Our room decor staff will be there shortly."),
bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",

                                activeforeground = "navy", font=('Century Gothic bold',
int(25*hf)))

    button_back_to_stays_page = Button(services_page,text="Back",command =
lambda : [close(services_page),stays()],bg="navy",activebackground = "#00FFFF",fg
= "#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(25*hf)))



    label_heading.place(x=0*wf,y=100*hf,width=1920*wf)

    button_order_food.place(x=335*wf,y=350*hf)

    button_order_laundry_services.place(x=1135*wf,y=350*hf)

    button_order_other_amenities.place(x=335*wf,y=600*hf)

    button_order_room_service.place(x=1135*wf, y=600*hf)

    button_back_to_stays_page.place(x=885*wf,y=825*hf)
    services_page.mainloop()
# ------------------------------------------------------------------------------------------------------------------
# ADMIN FUNCTIONS
# Save Changes Func
def save_changes(list_data, list_new_data, list_items, list_new_items, list_delete,
list_options, list_name, list_price, list_status, mode):

    confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss", "Do you wish to
confirm the changes?")
    if confirm_yes_or_no == "yes":
        con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")
        mycursor_services = con.cursor()

        new_data_empty = 0
        new_data_valid = 0
        item_found = 0

        list_new_items.clear()

        for data in list_new_data:
            if data[1].get() == "" or data[2].get() == "" or data[3].get() == "":
                new_data_empty = 1
                messagebox.showerror("Resort Ivory Bliss", "Empty fields present.")
                error_index = list_new_data.index(data)

                break


            elif not(data[2].get().isdigit()):

                new_data_valid = 1

                messagebox.showerror("Resort Ivory Bliss", "Invalid data present.")

                error_index = list_new_data.index(data)

                break


            elif data[1].get().upper() in list_items or data[1].get().upper() in
list_new_items:

                item_found = 1

                messagebox.showerror("Resort Ivory Bliss", "Item already exists.")

                error_index = list_new_data.index(data)

                break


            else:

                data_index = list_new_data.index(data)


                command = "INSERT INTO services VALUES ('" + data[0]["text"] + "','" +
data[1].get().upper() + "','" + \

                     data[2].get() + "','" + data[3].get() + "')"


                mycursor_services.execute(command)
                con.commit()

                name = data[1].get().upper()
                data[1].delete(0, END)
                data[1].insert(0, name)
                data[1].configure(state=DISABLED, bd=0)

                list_delete[data_index].grid_forget()

                list_items.append(data[1].get().upper())
                list_name.append(data[1])
                list_price.append(data[2])
                list_status.append(data[3])

                mycursor_services.execute("SELECT * FROM services WHERE ID LIKE '" +
mode + "%'")
                new_list_data = mycursor_services.fetchall()
                list_data.clear()
                list_data.extend(new_list_data)

                if data[1].get().upper() not in list_new_items:
                   list_new_items.append(data[1].get().upper())
```

```
if new_data_empty == 1 or new_data_valid == 1 or item_found == 1:

    new_list = list_new_data[error_index::]

    list_new_data.clear()

    list_new_data.extend(new_list)

    delete_buttons = list_delete[error_index::]

    list_delete.clear()

    list_delete.extend(delete_buttons)

    options = list_options[error_index::]

    list_options.clear()

    list_options.extend(options)


else:

    list_new_data.clear()

    list_delete.clear()

    list_options.clear()


new_name_list = []

new_price_list = []

new_status_list = []


for entry_name_index in range(len(list_name)):

    name = list_name[entry_name_index].get().upper()

    if name != list_data[entry_name_index][1]:
            list_name[entry_name_index].delete(0, END)

            list_name[entry_name_index].insert(0, name)

            new_name_list.append([list_data[entry_name_index][0], name])


    for entry_price_index in range(len(list_price)):

        price = list_price[entry_price_index].get()


        if price != list_data[entry_price_index][2] :

            if price.isdigit():

                new_price_list.append([list_data[entry_price_index][0], price])


            else:

                id_string = list_data[entry_price_index][0]

                messagebox.showerror("Resort Ivory Bliss", "Invalid data for item number "
+ id_string)

                new_price_list.append([list_data[entry_price_index][0],
list_data[entry_price_index][2]])


    for status_variable_index in range(len(list_status)):

        status = list_status[status_variable_index].get()

        if status != list_data[status_variable_index][3]:

            new_status_list.append([list_data[status_variable_index][0], status])


    for name_data in new_name_list:
        table_id = name_data[0]
        new_name = name_data[1]

        command = "UPDATE services SET ITEM='" + new_name + "' WHERE ID='" +
table_id + "'"
        mycursor_services.execute(command)
        con.commit()

    for price_data in new_price_list:
        table_id = price_data[0]
        new_price = price_data[1]

        command = "UPDATE services SET PRICE='" + new_price + "' WHERE ID='" +
table_id + "'"
        mycursor_services.execute(command)
        con.commit()

    for status_data in new_status_list:
        table_id = status_data[0]
        new_status = status_data[1]

        command = "UPDATE services SET STATUS='" + new_status + "' WHERE ID='" +
table_id + "'"
        mycursor_services.execute(command)
        con.commit()


    if new_data_empty == 0 and new_data_valid == 0 and item_found == 0:
        messagebox.showinfo("Resort Ivory Bliss", "Successfully updated.")


    mycursor_services.execute("SELECT * FROM services WHERE ID LIKE '" + mode +
"%'")
    new_list_data = mycursor_services.fetchall()
    list_data.clear()
    list_data.extend(new_list_data)


    con.close()
  else:
    pass
# ------------------------------------------------------------------------------------------------------------------
# Add Item Func
def add_item(page, mode, list_data, list_new_data,list_items, list_new_items,
list_delete, list_options,  placement_row, label_1, label_2,
          button_add, button_save, button_back, window, state, geometry, HF):


  new_data_empty = 0
  new_data_valid = 0
  item_found = 0
```

```
list_new_items.clear()


if list_new_data != []:
    for i in list_new_data:
        if i[1].get() == "" or i[2].get() == "" or i[3].get() == 0:
            new_data_empty = 1
            messagebox.showerror("Resort Ivory Bliss", "Please fill in the new fields first.")
            break


        elif not(i[2].get().isdigit()) :
            new_data_valid = 1
            messagebox.showerror("Resort Ivory Bliss", "Invalid data.")
            break


        elif i[1].get().upper() in list_items or i[1].get().upper() in list_new_items:
            item_found = 1
            messagebox.showerror("Resort Ivory Bliss", "Item already exists.")
            break


        if i[1].get().upper() not in list_new_items:
            list_new_items.append(i[1].get().upper())
    if list_new_data == [] or (new_data_empty == 0 and new_data_valid == 0 and
item_found == 0):
        label_1.grid_forget()
        button_add.grid_forget()
        label_2.grid_forget()
        button_save.grid_forget()
        button_back.grid_forget()

        key = len(list_data) + 1

        if key < 10:
            table_id = mode + "0" + str(key)
        else:
            table_id = mode + str(key)

        label_id = Label(page, text=table_id,bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(20*HF)), pady=30)
        label_id.grid(row=placement_row, column=0, rowspan=2)

        entry_item = Entry(page, font=('Century Gothic', int(20*HF)),fg =
"navy",justify=CENTER, disabledbackground="#00FFFF",disabledforeground = "navy")
        entry_item.grid(row=placement_row, column=2, rowspan=2)
        entry_price = Entry(page, font=('Century Gothic', int(20*HF)),fg =
"navy",justify=CENTER)

        entry_price.grid(row=placement_row, column=4, rowspan=2)


        status_variable = StringVar()

        status_option = OptionMenu(page, status_variable, "Available", "Not
Available")

        status_option.grid(row=placement_row, column=6, rowspan=2)

        status_option.configure(font=('Century Gothic',
int(20*HF)),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy",width=16)


        list_data.append([label_id["text"], entry_item.get(), entry_price.get(),
status_variable.get()])

        list_new_data.append([label_id, entry_item, entry_price, status_variable])

        list_options.append(status_option)

        list_new_items.append(entry_item.get().upper())


        button_delete = Button(page, text="X", font=('Century Gothic', int(12*HF)),

                            command=lambda: click_delete(list_data, list_new_data,
list_delete, list_options, list_new_items,

                                                button_delete, mode, window,
geometry),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy")

        list_delete.append(button_delete)
```

```
button_delete.grid(row=placement_row, column=7, rowspan=2)


if mode == "F":
   Restaurant_Menu_Page_Admin_Func.Placement_Row_For_New_Item += 2
   if state == 'normal':
      Restaurant_Menu_Page_Admin_Func.state = 'zoomed'
   else:
      Restaurant_Menu_Page_Admin_Func.state = 'normal'
elif mode == "L":
   Laundry_Service_Page_Admin_Func.Placement_Row_For_New_Item += 2
   if state == 'normal' :
      Laundry_Service_Page_Admin_Func.state = 'zoomed'
   else:
      Laundry_Service_Page_Admin_Func.state = 'normal'
elif mode == "A":
   Amenities_Page_Admin_Func.Placement_Row_For_New_Item += 2
   if state == 'normal' :
      Amenities_Page_Admin_Func.state = 'zoomed'
   else:
      Amenities_Page_Admin_Func.state = 'normal'


label_1.grid(row=placement_row + 2, column=0, columnspan=7, rowspan=2)
```

```
        button_add.grid(row=placement_row + 4, column=0, columnspan=7,
rowspan=2)
        label_2.grid(row=placement_row + 6, column=0, columnspan=7, rowspan=2)
        button_save.grid(row=placement_row + 8, column=2, columnspan=5,
rowspan=2)
        button_back.grid(row=placement_row + 8, column=0, columnspan=2,
rowspan=2)

        if state == 'normal' :
            window.state('zoomed')

        else:
            window.state('normal')
            window.geometry(geometry)
# ------------------------------------------------------------------------------------------------------------
# Click Delete Func
def click_delete(list_data, list_new_data, list_delete, list_options, list_new_items,
button, mode, window, geometry):
    index = list_delete.index(button)

    if index != len(list_delete)-1 :
        messagebox.showerror("Resort Ivory Bliss", "Please delete the last item first.")

    else:
        delete_data = list_new_data[index][0]["text"]
        for data in list_data:
            if data[0] == delete_data:
                list_data.remove(data)

        for widget in [list_new_data[index][0], list_new_data[index][1],
list_new_data[index][2]]:
            widget.grid_forget()

        list_options[index].grid_forget()

        del list_new_data[index]
        del list_options[index]
        button.grid_forget()
        list_delete.remove(button)

        if list_new_items != []:
            del list_new_items[-1]

        if mode == "F":
            if Restaurant_Menu_Page_Admin_Func.state == 'normal':
                window.state('zoomed')
                Restaurant_Menu_Page_Admin_Func.state = 'zoomed'
```

```
    else:
        window.state('normal')
        window.geometry(geometry)
        Restaurant_Menu_Page_Admin_Func.state = 'normal'
elif mode == "L":
    if Laundry_Service_Page_Admin_Func.state == 'normal':
        window.state('zoomed')
        Laundry_Service_Page_Admin_Func.state = 'zoomed'

    else:
        window.state('normal')
        window.geometry(geometry)
        Laundry_Service_Page_Admin_Func.state = 'normal'
elif mode == "A":
    if Amenities_Page_Admin_Func.state == 'normal':
        window.state('zoomed')
        Amenities_Page_Admin_Func.state = 'zoomed'

    else:
        window.state('normal')
        window.geometry(geometry)
        Amenities_Page_Admin_Func.state = 'normal'
```

```
#---------------------------------------------------------------------------------------------------------------------

# Back Button To Main Menu

def back_to_main_menu_admin(page):

   confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss",

                                    "Are you sure you want to return to main menu? All
unsave changes will be lost.")

   if confirm_yes_or_no == "yes":

      close(page)

      Main_Menu_Func()

   else:

      pass

# --------------------------------------------------------------------------------------------------------------------
--

# Edit Restaurant Menu

def Restaurant_Menu_Page_Admin_Func():

   con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")

   mycursor_services = con.cursor()

   mycursor_services.execute("SELECT * FROM services WHERE ID LIKE 'F%'")

   con.close()


   Restaurant_Menu_Page_Admin_Func.restaurant_menu =
mycursor_services.fetchall()
```

```
Restaurant_Menu_Page_Admin_Func.list_for_new_data = []

Restaurant_Menu_Page_Admin_Func.list_delete_button = []

Restaurant_Menu_Page_Admin_Func.list_for_new_options = []


Restaurant_Menu_Page_Admin_Func.list_for_dish_names = []

Restaurant_Menu_Page_Admin_Func.list_for_new_dish_names = []



Restaurant_Menu_Page_Admin_Func.Entry_Dish = []

Restaurant_Menu_Page_Admin_Func.Entry_Price = []

Restaurant_Menu_Page_Admin_Func.List_Option_Change_Status = []


Restaurant_Menu_Page = Tk()


sw = Restaurant_Menu_Page.winfo_screenwidth()

sh = Restaurant_Menu_Page.winfo_screenheight()


gwf = 1915/1920

ghf = 1050/1080


wf = sw/1920 * gwf

hf = sh/1080 * ghf
```

```
    g = str(int(sw*gwf)) + "x" + str(int(sh*ghf))


    Restaurant_Menu_Page.config(bg="#00FFFF")
    Restaurant_Menu_Page.geometry(g)
    Restaurant_Menu_Page.title("Resort Ivory Bliss - Edit Restaurant Menu (Admin)")
    Restaurant_Menu_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")
    Restaurant_Menu_Page.resizable(0,0)


    Restaurant_Menu_Page_Admin_Func.state = 'normal'


    main_frame = Frame(Restaurant_Menu_Page, bg="#00FFFF")
    main_frame.pack(fill=BOTH, expand=1)
    my_canvas = Canvas(main_frame, bg="#00FFFF")
    my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
    my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)
    my_scrollbar.pack(side=RIGHT, fill=Y)
    my_canvas.configure(yscrollcommand=my_scrollbar.set)
    my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))
    second_frame = Frame(my_canvas,bg="#00FFFF")
    my_canvas.create_window((0, 0), window=second_frame, anchor=NW)
    label_main_heading = Label(second_frame, text="EDIT  RESTAURANT 
MENU",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(40*hf)), 
pady=50*hf)
    label_main_heading.grid(row=0, column=0, columnspan=9, rowspan=2)


    label_heading_id = Label(second_frame, text="Id",bg="#00FFFF",fg = "navy", 
font=('Bookman Old Style bold', int(40*hf)), pady=50*hf, padx=140*wf)
    label_heading_dish = Label(second_frame, text="Dish",bg="#00FFFF",fg = "navy", 
font=('Bookman Old Style bold', int(40*hf)), pady=50*hf, padx=150*wf)
    label_heading_price = Label(second_frame, text="Price in Rs.",bg="#00FFFF",fg = 
"navy", font=('Bookman Old Style bold', int(40*hf)), pady=50*hf, padx=150*wf)
    label_heading_status = Label(second_frame, text="Status",bg="#00FFFF",fg = 
"navy", font=('Bookman Old Style bold', int(40*hf)), pady=50*hf, padx=150*wf)


    label_heading_id.grid(row=2, column=0, rowspan=2)
    label_heading_dish.grid(row=2, column=2, rowspan=2)
    label_heading_price.grid(row=2, column=4, rowspan=2)
    label_heading_status.grid(row=2, column=6, rowspan=2)


    Placement_Row_For_Label_Id = 4
    for label_id in range(len(Restaurant_Menu_Page_Admin_Func.restaurant_menu)):
        i = label_id
        label_id = Label(second_frame, 
text=Restaurant_Menu_Page_Admin_Func.restaurant_menu[i][0],bg="#00FFFF",fg = 
"navy", font=('Century Gothic', int(20*hf)), pady=30*hf)
```

```
        label_id.grid(row=Placement_Row_For_Label_Id, column=0, rowspan=2)

        Placement_Row_For_Label_Id += 2


    Placement_Row_For_Entry_Dish = 4

    for entry_dish in
range(len(Restaurant_Menu_Page_Admin_Func.restaurant_menu)):

        i = entry_dish

        entry_dish = Entry(second_frame, font=('Century Gothic', int(20*hf)),fg = "navy",
justify=CENTER, disabledbackground="#00FFFF",disabledforeground = "navy")

        entry_dish.grid(row=Placement_Row_For_Entry_Dish, column=2, rowspan=2)

        entry_dish.insert(0, Restaurant_Menu_Page_Admin_Func.restaurant_menu[i][1])

        entry_dish.configure(state=DISABLED, bd=0)

        Restaurant_Menu_Page_Admin_Func.Entry_Dish.append(entry_dish)


Restaurant_Menu_Page_Admin_Func.list_for_dish_names.append(Restaurant_Menu
_Page_Admin_Func.restaurant_menu[i][1])

        Placement_Row_For_Entry_Dish += 2


    Placement_Row_For_Entry_Price = 4

    for entry_price in
range(len(Restaurant_Menu_Page_Admin_Func.restaurant_menu)):

        i = entry_price

        entry_price = Entry(second_frame, font=('Century Gothic', int(20*hf)),fg =
"navy", justify=CENTER)

        entry_price.grid(row=Placement_Row_For_Entry_Price, column=4, rowspan=2)
        entry_price.insert(0,
Restaurant_Menu_Page_Admin_Func.restaurant_menu[i][2])
        Restaurant_Menu_Page_Admin_Func.Entry_Price.append(entry_price)
        Placement_Row_For_Entry_Price += 2


    for opt_variable in
range(len(Restaurant_Menu_Page_Admin_Func.restaurant_menu)):
        i = opt_variable
        opt_variable = StringVar()
        opt_variable.set(Restaurant_Menu_Page_Admin_Func.restaurant_menu[i][3])

Restaurant_Menu_Page_Admin_Func.List_Option_Change_Status.append(opt_vari
able)


    Placement_Row_For_Option_Change_Status = 4
    for opt_status_menu in
range(len(Restaurant_Menu_Page_Admin_Func.restaurant_menu)):
        i = opt_status_menu
        opt_status_menu = OptionMenu(second_frame,

Restaurant_Menu_Page_Admin_Func.List_Option_Change_Status[i],
                                "Available", "Not Available")
        opt_status_menu.grid(row=Placement_Row_For_Option_Change_Status,
column=6, rowspan=2)
```

```
        opt_status_menu.configure(font=('Century Gothic', 
int(20*hf)),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground 
= "navy", width=16)

        Placement_Row_For_Option_Change_Status += 2


    Restaurant_Menu_Page_Admin_Func.Placement_Row_For_New_Item = 
Placement_Row_For_Label_Id


    label_space_1 = Label(second_frame, text="\t",bg="#00FFFF",fg = 
"navy",pady=30*hf)


label_space_1.grid(row=Restaurant_Menu_Page_Admin_Func.Placement_Row_For
_New_Item, column=0, columnspan=7, rowspan=2)


    label_space_2 = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy", 
pady=30*hf)


label_space_2.grid(row=Restaurant_Menu_Page_Admin_Func.Placement_Row_For
_New_Item + 4, column=0, columnspan=7, rowspan=2)


    button_save_changes = Button(second_frame, text="Save Changes", width=80, 
font=('Century Gothic bold', int(20*hf)),bg="navy",activebackground = "#00FFFF",fg 
= "#00FFFF",activeforeground = "navy",
                                command=lambda: save_changes(
                                    Restaurant_Menu_Page_Admin_Func.restaurant_menu,
                                    Restaurant_Menu_Page_Admin_Func.list_for_new_data,
                    Restaurant_Menu_Page_Admin_Func.list_for_dish_names,
                    Restaurant_Menu_Page_Admin_Func.list_for_new_dish_names,
                    Restaurant_Menu_Page_Admin_Func.list_delete_button,
                    Restaurant_Menu_Page_Admin_Func.list_for_new_options,
                    Restaurant_Menu_Page_Admin_Func.Entry_Dish,
                    Restaurant_Menu_Page_Admin_Func.Entry_Price,

Restaurant_Menu_Page_Admin_Func.List_Option_Change_Status, "F"))


button_save_changes.grid(row=Restaurant_Menu_Page_Admin_Func.Placement_R
ow_For_New_Item + 6, column=2, columnspan=5, rowspan=2)


    button_add_item = Button(second_frame, text="Add Item", width=80,
font=('Century Gothic bold', int(20*hf)),bg="navy",activebackground = "#00FFFF",fg
= "#00FFFF",activeforeground = "navy", justify=CENTER)
    button_add_item.configure(command=lambda: add_item(
        second_frame,
        "F",
        Restaurant_Menu_Page_Admin_Func.restaurant_menu,
        Restaurant_Menu_Page_Admin_Func.list_for_new_data,
        Restaurant_Menu_Page_Admin_Func.list_for_dish_names,
        Restaurant_Menu_Page_Admin_Func.list_for_new_dish_names,
        Restaurant_Menu_Page_Admin_Func.list_delete_button,
        Restaurant_Menu_Page_Admin_Func.list_for_new_options,
    Restaurant_Menu_Page_Admin_Func.Placement_Row_For_New_Item,

    label_space_1,

    label_space_2,

    button_add_item,

    button_save_changes,

    button_back_to_main_menu_admin,

    Restaurant_Menu_Page, Restaurant_Menu_Page_Admin_Func.state, g, hf))



button_add_item.grid(row=Restaurant_Menu_Page_Admin_Func.Placement_Row_F
or_New_Item + 2, column=0, columnspan=7, rowspan=2)


  button_back_to_main_menu_admin = Button(second_frame, text="Back",
font=('Century Gothic bold', int(20*hf)),bg="navy",activebackground = "#00FFFF",fg
= "#00FFFF",activeforeground = "navy",

                        command=lambda:
back_to_main_menu_admin(Restaurant_Menu_Page))


button_back_to_main_menu_admin.grid(row=Restaurant_Menu_Page_Admin_Func
.Placement_Row_For_New_Item + 6, column=0,

                      columnspan=2, rowspan=2)


  Restaurant_Menu_Page.mainloop()
# ------------------------------------------------------------------------------------------------------------

# Edit Laundry Services Menu
```

```
def Laundry_Service_Page_Admin_Func():

    con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")

    mycursor_services = con.cursor()

    mycursor_services.execute("SELECT * FROM services WHERE ID LIKE 'L%'")

    con.close()


    Laundry_Service_Page_Admin_Func.laundry_menu = mycursor_services.fetchall()


    Laundry_Service_Page_Admin_Func.list_for_new_data = []

    Laundry_Service_Page_Admin_Func.list_delete_button = []

    Laundry_Service_Page_Admin_Func.list_for_new_options = []


    Laundry_Service_Page_Admin_Func.list_for_type_names = []

    Laundry_Service_Page_Admin_Func.list_for_new_type_names = []


    Laundry_Service_Page_Admin_Func.Entry_Type = []

    Laundry_Service_Page_Admin_Func.Entry_Price = []

    Laundry_Service_Page_Admin_Func.List_Option_Change_Status = []


    Laundry_Menu_Page = Tk()


    sw = Laundry_Menu_Page.winfo_screenwidth()
```

```
sh = Laundry_Menu_Page.winfo_screenheight()

gwf = 1915 / 1920
ghf = 1050 / 1080

wf = sw / 1920 * gwf
hf = sh / 1080 * ghf

g = str(int(sw * gwf)) + "x" + str(int(sh * ghf))

Laundry_Menu_Page.config(bg="#00FFFF")
Laundry_Menu_Page.geometry(g)
Laundry_Menu_Page.title("Resort Ivory Bliss - Edit Laundry Service Menu (Admin)")
Laundry_Menu_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")
Laundry_Menu_Page.resizable(0,0)

Laundry_Service_Page_Admin_Func.state = 'normal'

main_frame = Frame(Laundry_Menu_Page, bg="#00FFFF")
main_frame.pack(fill=BOTH, expand=1)
my_canvas = Canvas(main_frame, bg="#00FFFF")
my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
```

```
    my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)
    my_scrollbar.pack(side=RIGHT, fill=Y)
    my_canvas.configure(yscrollcommand=my_scrollbar.set)
    my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))
    second_frame = Frame(my_canvas,bg="#00FFFF")
    my_canvas.create_window((0, 0), window=second_frame, anchor=NW)

    label_main_heading = Label(second_frame, text="EDIT  LAUNDRY
MENU",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(40*hf)),
pady=50)
    label_main_heading.grid(row=0, column=0, columnspan=9, rowspan=2)

    label_heading_id = Label(second_frame, text="Id",bg="#00FFFF",fg = "navy",
font=('Bookman Old Style bold', int(30*hf)), pady=50*hf, padx=140*wf)
    label_heading_dish = Label(second_frame, text="Type",bg="#00FFFF",fg = "navy",
font=('Bookman Old Style bold', int(30*hf)), pady=50*hf, padx=150*wf)
    label_heading_price = Label(second_frame, text="Price per clothing in
Rs.",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(30*hf)),
pady=50*hf, padx=130*wf)
    label_heading_status = Label(second_frame, text="Status",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold', int(30*hf)), pady=50*hf, padx=120*wf)

    label_heading_id.grid(row=2, column=0, rowspan=2)
    label_heading_dish.grid(row=2, column=2, rowspan=2)
    label_heading_price.grid(row=2, column=4, rowspan=2)
    label_heading_status.grid(row=2, column=6, rowspan=2)

    Placement_Row_For_Label_Id = 4
    for label_id in range(len(Laundry_Service_Page_Admin_Func.laundry_menu)):
        i = label_id
        label_id = Label(second_frame,
text=Laundry_Service_Page_Admin_Func.laundry_menu[i][0],bg="#00FFFF",fg =
"navy", font=('Century Gothic', int(20*hf)), pady=30*hf)
        label_id.grid(row=Placement_Row_For_Label_Id, column=0, rowspan=2)
        Placement_Row_For_Label_Id += 2

    Placement_Row_For_Entry_Dish = 4
    for entry_dish in range(len(Laundry_Service_Page_Admin_Func.laundry_menu)):
        i = entry_dish
        entry_dish = Entry(second_frame, font=('Century Gothic', int(20*hf)),fg = "navy",
justify=CENTER, disabledbackground="#00FFFF",disabledforeground = "navy")
        entry_dish.grid(row=Placement_Row_For_Entry_Dish, column=2, rowspan=2)
        entry_dish.insert(0, Laundry_Service_Page_Admin_Func.laundry_menu[i][1])
        entry_dish.configure(state=DISABLED, bd=0)
        Laundry_Service_Page_Admin_Func.Entry_Type.append(entry_dish)

Laundry_Service_Page_Admin_Func.list_for_type_names.append(Laundry_Service_
Page_Admin_Func.laundry_menu[i][1])
        Placement_Row_For_Entry_Dish += 2
```

```
   Placement_Row_For_Entry_Price = 4
   for entry_price in range(len(Laundry_Service_Page_Admin_Func.laundry_menu)):
      i = entry_price
      entry_price = Entry(second_frame, font=('Century Gothic', int(20*hf)),fg =
"navy", justify=CENTER)
      entry_price.grid(row=Placement_Row_For_Entry_Price, column=4, rowspan=2)
      entry_price.insert(0, Laundry_Service_Page_Admin_Func.laundry_menu[i][2])
      Laundry_Service_Page_Admin_Func.Entry_Price.append(entry_price)
      Placement_Row_For_Entry_Price += 2


   for opt_variable in
range(len(Laundry_Service_Page_Admin_Func.laundry_menu)):
      i = opt_variable
      opt_variable = StringVar()
      opt_variable.set(Laundry_Service_Page_Admin_Func.laundry_menu[i][3])

Laundry_Service_Page_Admin_Func.List_Option_Change_Status.append(opt_varia
ble)


   Placement_Row_For_Option_Change_Status = 4
   for opt_status_menu in
range(len(Laundry_Service_Page_Admin_Func.laundry_menu)):
      i = opt_status_menu
      opt_status_menu = OptionMenu(second_frame,
Laundry_Service_Page_Admin_Func.List_Option_Change_Status[i],
                              "Available", "Not Available")
        opt_status_menu.grid(row=Placement_Row_For_Option_Change_Status,
column=6, rowspan=2)
        opt_status_menu.configure(font=('Century Gothic',
int(20*hf)),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground
= "navy", width=16)
        Placement_Row_For_Option_Change_Status += 2

    Laundry_Service_Page_Admin_Func.Placement_Row_For_New_Item =
Placement_Row_For_Label_Id

    label_space_1 = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy",
pady=30*hf)

label_space_1.grid(row=Laundry_Service_Page_Admin_Func.Placement_Row_For_
New_Item, column=0, columnspan=7, rowspan=2)

    label_space_2 = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy", pady
=30*hf)

label_space_2.grid(row=Laundry_Service_Page_Admin_Func.Placement_Row_For_
New_Item + 4, column=0, columnspan=7, rowspan=2)
```

```
    button_save_changes = Button(second_frame, font=('Century Gothic bold', int(25*hf)),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground = "navy",
                              text="Save Changes", width=60,
                              command=lambda: save_changes(
                                  Laundry_Service_Page_Admin_Func.laundry_menu,
                                  Laundry_Service_Page_Admin_Func.list_for_new_data,
                                  Laundry_Service_Page_Admin_Func.list_for_type_names,
                                  Laundry_Service_Page_Admin_Func.list_for_new_type_names,
                                  Laundry_Service_Page_Admin_Func.list_delete_button,
                                  Laundry_Service_Page_Admin_Func.list_for_new_options,
                                  Laundry_Service_Page_Admin_Func.Entry_Type,
                                  Laundry_Service_Page_Admin_Func.Entry_Price,

Laundry_Service_Page_Admin_Func.List_Option_Change_Status, "L"))

button_save_changes.grid(row=Laundry_Service_Page_Admin_Func.Placement_Row_For_New_Item + 6, column=2, columnspan=5, rowspan=2)


    button_add_item = Button(second_frame, text="Add Type of Clothing", width=60,bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(25*hf)))
    button_add_item.configure(command=lambda: add_item(
        second_frame,
        "L",
        Laundry_Service_Page_Admin_Func.laundry_menu,

        Laundry_Service_Page_Admin_Func.list_for_new_data,

        Laundry_Service_Page_Admin_Func.list_for_type_names,

        Laundry_Service_Page_Admin_Func.list_for_new_type_names,

        Laundry_Service_Page_Admin_Func.list_delete_button,

        Laundry_Service_Page_Admin_Func.list_for_new_options,

        Laundry_Service_Page_Admin_Func.Placement_Row_For_New_Item,

        label_space_1,

        label_space_2,

        button_add_item,

        button_save_changes,

        button_back_to_main_menu_admin,

        Laundry_Menu_Page, Laundry_Service_Page_Admin_Func.state, g, hf))



button_add_item.grid(row=Laundry_Service_Page_Admin_Func.Placement_Row_F
or_New_Item + 2, column=0, columnspan=7, rowspan=2)

   button_back_to_main_menu_admin = Button(second_frame,
text="Back",bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(25*hf)),

                              command=lambda:
back_to_main_menu_admin(Laundry_Menu_Page))


button_back_to_main_menu_admin.grid(row=Laundry_Service_Page_Admin_Func.
Placement_Row_For_New_Item + 6, column=0,
                             columnspan=2, rowspan=2)

   Laundry_Menu_Page.mainloop()
# ----------------------------------------------------------------------------------------------------------------
# Edit Amenities Menu
def Amenities_Page_Admin_Func():
   con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")
   mycursor_services = con.cursor()
   mycursor_services.execute("SELECT * FROM services WHERE ID LIKE 'A%'")
   con.close()

   Amenities_Page_Admin_Func.amenities_menu = mycursor_services.fetchall()

   Amenities_Page_Admin_Func.list_for_new_data = []
   Amenities_Page_Admin_Func.list_delete_button = []
   Amenities_Page_Admin_Func.list_for_new_options = []

   Amenities_Page_Admin_Func.list_for_amenities = []
   Amenities_Page_Admin_Func.list_for_new_amenities = []

   Amenities_Page_Admin_Func.Entry_Amenity = []
   Amenities_Page_Admin_Func.Entry_Price = []
   Amenities_Page_Admin_Func.List_Option_Change_Status = []


   Amenities_Menu_Page = Tk()


   sw = Amenities_Menu_Page.winfo_screenwidth()

   sh = Amenities_Menu_Page.winfo_screenheight()


   gwf = 1915 / 1920

   ghf = 1050 / 1080


   wf = sw / 1920 * gwf

   hf = sh / 1080 * ghf


   g = str(int(sw * gwf)) + "x" + str(int(sh * ghf))


   Amenities_Menu_Page.config(bg="#00FFFF")

   Amenities_Menu_Page.geometry(g)

   Amenities_Menu_Page.title("Resort Ivory Bliss - Edit Amenities Menu (Admin)")

   Amenities_Menu_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")

   Amenities_Menu_Page.resizable(0,0)


   Amenities_Page_Admin_Func.state = 'normal'
```

```
    main_frame = Frame(Amenities_Menu_Page, bg="#00FFFF")

    main_frame.pack(fill=BOTH, expand=1)

    my_canvas = Canvas(main_frame, bg="#00FFFF")

    my_canvas.pack(side=LEFT, fill=BOTH, expand=1)

    my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)

    my_scrollbar.pack(side=RIGHT, fill=Y)

    my_canvas.configure(yscrollcommand=my_scrollbar.set)

    my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))

    second_frame = Frame(my_canvas,bg="#00FFFF")

    my_canvas.create_window((0, 0), window=second_frame, anchor=NW)


    label_main_heading = Label(second_frame, text="EDIT  AMENITIES
MENU",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(40*hf)),
pady=50*hf)

    label_main_heading.grid(row=0, column=0, columnspan=9, rowspan=2)


    label_heading_id = Label(second_frame, text="Id",bg="#00FFFF",fg = "navy",
font=('Bookman Old Style bold', int(30*hf)), pady=50*hf, padx=120*wf)

    label_heading_dish = Label(second_frame, text="Amenity",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold', int(30*hf)), pady=50*hf, padx=150*wf)

    label_heading_price = Label(second_frame, text="Price per hour in
Rs.",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(30*hf)),
pady=50*hf, padx=150*wf)
```

```
    label_heading_status = Label(second_frame, text="Status",bg="#00FFFF",fg = 
"navy", font=('Bookman Old Style bold', int(30*hf)), pady=50*hf, padx=120*wf)

    label_heading_id.grid(row=2, column=0, rowspan=2)
    label_heading_dish.grid(row=2, column=2, rowspan=2)
    label_heading_price.grid(row=2, column=4, rowspan=2)
    label_heading_status.grid(row=2, column=6, rowspan=2)

    Placement_Row_For_Label_Id = 4
    for label_id in range(len(Amenities_Page_Admin_Func.amenities_menu)):
        i = label_id
        label_id = Label(second_frame, 
text=Amenities_Page_Admin_Func.amenities_menu[i][0],bg="#00FFFF",fg = "navy", 
font=('Century Gothic', int(20*hf)), pady=30*hf)
        label_id.grid(row=Placement_Row_For_Label_Id, column=0, rowspan=2)
        Placement_Row_For_Label_Id += 2

    Placement_Row_For_Entry_Dish = 4
    for entry_dish in range(len(Amenities_Page_Admin_Func.amenities_menu)):
        i = entry_dish
        entry_dish = Entry(second_frame, font=('Century Gothic', int(20*hf)),fg = "navy", 
justify=CENTER, disabledbackground="#00FFFF",disabledforeground = "navy")
        entry_dish.grid(row=Placement_Row_For_Entry_Dish, column=2, rowspan=2)
        entry_dish.insert(0, Amenities_Page_Admin_Func.amenities_menu[i][1])
        entry_dish.configure(state=DISABLED, bd=0)
        Amenities_Page_Admin_Func.Entry_Amenity.append(entry_dish)

Amenities_Page_Admin_Func.list_for_amenities.append(Amenities_Page_Admin_Fu
nc.amenities_menu[i][1])
        Placement_Row_For_Entry_Dish += 2


    Placement_Row_For_Entry_Price = 4
    for entry_price in range(len(Amenities_Page_Admin_Func.amenities_menu)):
        i = entry_price
        entry_price = Entry(second_frame, font=('Century Gothic', int(20*hf)),fg =
"navy", justify=CENTER)
        entry_price.grid(row=Placement_Row_For_Entry_Price, column=4, rowspan=2)
        entry_price.insert(0, Amenities_Page_Admin_Func.amenities_menu[i][2])
        Amenities_Page_Admin_Func.Entry_Price.append(entry_price)
        Placement_Row_For_Entry_Price += 2


    for opt_variable in range(len(Amenities_Page_Admin_Func.amenities_menu)):
        i = opt_variable
        opt_variable = StringVar()
        opt_variable.set(Amenities_Page_Admin_Func.amenities_menu[i][3])

Amenities_Page_Admin_Func.List_Option_Change_Status.append(opt_variable)
```

```
    Placement_Row_For_Option_Change_Status = 4

    for opt_status_menu in
range(len(Amenities_Page_Admin_Func.amenities_menu)):

        i = opt_status_menu

        opt_status_menu = OptionMenu(second_frame,

                             Amenities_Page_Admin_Func.List_Option_Change_Status[i],

                             "Available", "Not Available")

        opt_status_menu.grid(row=Placement_Row_For_Option_Change_Status,
column=6, rowspan=2)

        opt_status_menu.configure(font=('Century Gothic',
int(20*hf)),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground
= "navy", width=16)

        Placement_Row_For_Option_Change_Status += 2



    Amenities_Page_Admin_Func.Placement_Row_For_New_Item =
Placement_Row_For_Label_Id



    label_space_1 = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy",
pady=30*hf)


label_space_1.grid(row=Amenities_Page_Admin_Func.Placement_Row_For_New_It
em, column=0, columnspan=7)


    label_space_2 = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy",
pady=30*hf)
```

```
label_space_2.grid(row=Amenities_Page_Admin_Func.Placement_Row_For_New_It
em + 4, column=0, columnspan=7)


    button_save_changes = Button(second_frame,bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Century Gothic bold',
int(25*hf)),
                                text="Save Changes", width=60,
                                command=lambda: save_changes(
                                    Amenities_Page_Admin_Func.amenities_menu,
                                    Amenities_Page_Admin_Func.list_for_new_data,
                                    Amenities_Page_Admin_Func.list_for_amenities,
                                    Amenities_Page_Admin_Func.list_for_new_amenities,
                                    Amenities_Page_Admin_Func.list_delete_button,
                                    Amenities_Page_Admin_Func.list_for_new_options,
                                    Amenities_Page_Admin_Func.Entry_Amenity,
                                    Amenities_Page_Admin_Func.Entry_Price,
                                    Amenities_Page_Admin_Func.List_Option_Change_Status,
"A"))

button_save_changes.grid(row=Amenities_Page_Admin_Func.Placement_Row_For
_New_Item + 6, column=2, columnspan=7)


    button_add_item = Button(second_frame, text="Add Amenity",
width=60,bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground
= "navy", font=('Century Gothic bold', int(25*hf)))
```

```
    button_add_item.configure(command=lambda: add_item(
        second_frame,
        "A",
        Amenities_Page_Admin_Func.amenities_menu,
        Amenities_Page_Admin_Func.list_for_new_data,
        Amenities_Page_Admin_Func.list_for_amenities,
        Amenities_Page_Admin_Func.list_for_new_amenities,
        Amenities_Page_Admin_Func.list_delete_button,
        Amenities_Page_Admin_Func.list_for_new_options,
        Amenities_Page_Admin_Func.Placement_Row_For_New_Item,
        label_space_1,
        label_space_2,
        button_add_item,
        button_save_changes,
        button_back_to_main_menu_admin,
        Amenities_Menu_Page, Amenities_Page_Admin_Func.state, g, hf))


button_add_item.grid(row=Amenities_Page_Admin_Func.Placement_Row_For_New
_Item + 2, column=0, columnspan=7)


    button_back_to_main_menu_admin = Button(second_frame,
text="Back",bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(25*hf)),
                                  command=lambda:
back_to_main_menu_admin(Amenities_Menu_Page))


button_back_to_main_menu_admin.grid(row=Amenities_Page_Admin_Func.Place
ment_Row_For_New_Item + 6, column=0,
                               columnspan=2)


    Amenities_Menu_Page.mainloop()
# ------------------------------------------------------------------------------------------------------------
# Edit Rooms Info
def Room_Info_Page_Admin_Func():
    # --------------------------------------------------------------------------------------------------------
    def save_room_changes(list_data, list_new_data, list_room_numbers,
list_new_room_numbers, list_delete, list_options, list_room_number,
                          list_room_type, list_ac_nonaac, list_price, list_status, HF):


        confirm_yes_or_no = messagebox.askquestion("Resort Ivory Bliss", "Do you wish
to confirm the changes?")
        if confirm_yes_or_no == "yes":
            con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")
            mycursor_services = con.cursor()


            new_data_empty = 0
            new_data_valid = 0
        room_found = 0


        list_new_room_numbers.clear()


        for data in list_new_data:
            if data[0].get() == "" or data[1].get() == "" or data[2].get() == "" or
data[3].get() == "" or data[
                4].get() == "":
                new_data_empty = 1
                error_index = list_new_data.index(data)
                messagebox.showerror("Resort Ivory Bliss", "Empty fields present.")
                break

            elif not(data[0].get().isdigit()) or not(data[3].get().isdigit()):
                new_data_valid = 1
                error_index = list_new_data.index(data)
                messagebox.showerror("Resort Ivory Bliss", "Invalid data present.")
                break

            elif data[0].get() in list_room_numbers or data[0].get() in
list_new_room_numbers:
                room_found = 1
                error_index = list_new_data.index(data)
                messagebox.showerror("Resort Ivory Bliss", "Room already exists.")
                break


            else:
                data_index = list_new_data.index(data)


                command = "INSERT INTO room_info VALUES ('" + data[0].get() + "','" +
data[
                    1].get().upper() + "','" + data[2].get() + "','" + data[3].get() + "','" +
data[4].get() + "')"
                mycursor_services.execute(command)
                con.commit()


                data[0].configure(state=DISABLED,
disabledbackground="#00FFFF",disabledforeground="navy",font=('Century Gothic
bold', int(15*HF)), bd=0)


                list_delete[data_index].grid_forget()


                list_room_number.append(data[0])
                list_room_numbers.append(data[0].get())
                list_room_type.append(data[1])
                list_ac_nonaac.append(data[2])
                list_price.append(data[3])
                list_status.append(data[4])
```

```
        mycursor_services.execute("SELECT * FROM room_info")
        updated_list = mycursor_services.fetchall()
        list_data.clear()
        list_data.extend(updated_list)

        if data[0].get() not in list_new_room_numbers:
            list_new_room_numbers.append(data[0].get())

if new_data_empty == 1 or new_data_valid == 1 or room_found == 1:
    new_list = list_new_data[error_index::]
    list_new_data.clear()
    list_new_data.extend(new_list)
    delete_buttons = list_delete[error_index::]
    list_delete.clear()
    list_delete.extend(delete_buttons)
    options = list_options[error_index::]
    list_options.clear()
    list_options.extend(options)

else:
    list_new_data.clear()
    list_delete.clear()
    list_options.clear()

new_room_type_list = []
new_price_list = []
new_ac_nonac_list = []
new_status_list = []

for room_type_variable_index in range(len(list_room_type)):
    room_type = list_room_type[room_type_variable_index].get()
    if room_type != list_data[room_type_variable_index][1]:
        new_room_type_list.append([list_data[room_type_variable_index][0],
room_type])

for ac_nonac_variable_index in range(len(list_ac_nonaac)):
    ac_nonac = list_ac_nonaac[ac_nonac_variable_index].get()
    if ac_nonac != list_data[ac_nonac_variable_index][2]:
        new_ac_nonac_list.append([list_data[ac_nonac_variable_index][0],
ac_nonac])

for entry_price_index in range(len(list_price)):
    price = list_price[entry_price_index].get()

    if price != str(list_data[entry_price_index][3]):
        if price.isdigit():
```

```
                    new_price_list.append([list_data[entry_price_index][0], price])


                else:

                    room_no_string = list_data[entry_price_index][0]

                    messagebox.showerror("Resort Ivory Bliss", "Invalid data for room
number " + room_no_string)

                    new_price_list.append([list_data[entry_price_index][0],
str(list_data[entry_price_index][3])])




        for status_variable_index in range(len(list_status)):

            status = list_status[status_variable_index].get()

            if status != list_data[status_variable_index][4]:

                new_status_list.append([list_data[status_variable_index][0], status])


        for name_data in new_room_type_list:

            table_id = name_data[0]

            new_name = name_data[1]


            command = "UPDATE room_info SET ROOM_TYPE='" + new_name + "'
WHERE ROOM_NO='" + table_id + "'"

            mycursor_services.execute(command)

            con.commit()
        for ac_nonac_data in new_ac_nonac_list:
            table_id = ac_nonac_data[0]
            new_ac_nonac = ac_nonac_data[1]

            command = "UPDATE room_info SET AC_NON_AC='" + new_ac_nonac + "'
WHERE ROOM_NO='" + table_id + "'"
            mycursor_services.execute(command)
            con.commit()

        for price_data in new_price_list:
            table_id = price_data[0]
            new_price = price_data[1]

            command = "UPDATE room_info SET PRICE=" + new_price + " WHERE
ROOM_NO='" + table_id + "'"
            mycursor_services.execute(command)
            con.commit()

        for status_data in new_status_list:
            table_id = status_data[0]
            new_status = status_data[1]

            command = "UPDATE room_info SET STATUS='" + new_status + "' WHERE
ROOM_NO='" + table_id + "'"
```

```
            mycursor_services.execute(command)

            con.commit()


         if new_data_empty == 0 and room_found == 0 and new_data_valid == 0:

            messagebox.showinfo("Resort Ivory Bliss", "Successfully updated.")


         mycursor_services.execute("SELECT * FROM room_info")

         updated_list = mycursor_services.fetchall()

         list_data.clear()

         list_data.extend(updated_list)


         con.close()


      else:

         pass

   # ------------------------------------------------------------------------------------------------------------------------------

   def add_room(page, list_data, list_new_data, list_room_numbers,
list_new_room_numbers, list_delete, list_options, placement_row, label_1,

            label_2, button_add, button_save, button_back,window, state, geometry,
HF):


      new_data_empty = 0

      room_found = 0

      new_data_valid = 0
```

```
        list_new_room_numbers.clear()


        if list_new_data != []:
            for i in list_new_data:
                if i[0].get() == "" or i[1].get() == "" or i[2].get() == "" or i[3].get() == "" or
i[4].get() == "":
                    new_data_empty = 1
                    messagebox.showerror("Resort Ivory Bliss", "Please fill in the new fields
first.")
                    break


                elif not(i[0].get().isdigit()) or not(i[3].get().isdigit()):
                    new_data_valid = 1
                    messagebox.showerror("Resort Ivory Bliss", "Invalid data.")
                    break


                elif i[0].get() in list_room_numbers or i[0].get() in list_new_room_numbers:
                    room_found = 1
                    messagebox.showerror("Resort Ivory Bliss", "Room already exists.")
                    break


                if i[0].get() not in list_new_room_numbers:
                    list_new_room_numbers.append(i[0].get())
```

```
    if list_new_data == [] or (new_data_empty == 0 and room_found == 0 and
new_data_valid == 0):
        label_1.grid_forget()
        button_add.grid_forget()
        label_2.grid_forget()
        button_save.grid_forget()
        button_back.grid_forget()

        entry_room_number = Entry(page, justify=CENTER, font=('Century Gothic',
int(15*HF)),fg = "navy")
        entry_room_number.grid(row=placement_row, column=0, rowspan=2,
pady=30)

        list_of_room_types = ["CLASSIC", "DELUXE", "SUPER DELUXE", "ELITE", "COTTAGE"]

        room_type_variable = StringVar()
        room_type_option = OptionMenu(page, room_type_variable,
*list_of_room_types)
        room_type_option.grid(row=placement_row, column=2, rowspan=2)
        room_type_option.configure(font=('Century Gothic',
int(15*HF)),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", width=15)
```

```
        ac_nonac_variable = StringVar()
        ac_nonac_option = OptionMenu(page, ac_nonac_variable, "AC", "NON
AC")
        ac_nonac_option.grid(row=placement_row, column=4, rowspan=2)
        ac_nonac_option.configure(font=('Century Gothic',
int(15*HF)),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", width=15)

        entry_price = Entry(page, font=('Century Gothic', int(15*HF)),fg = "navy",
justify=CENTER)
        entry_price.grid(row=placement_row, column=6, rowspan=2)

        status_variable = StringVar()
        status_option = OptionMenu(page, status_variable, "Available", "Not
Available")
        status_option.grid(row=placement_row, column=8, rowspan=2)
        status_option.configure(font=('Century Gothic',
int(15*HF)),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", width=15)

        list_data.append(
            [entry_room_number.get(), room_type_variable.get(),
ac_nonac_variable.get(), entry_price.get(),
             status_variable.get()]
        )
        list_new_data.append(
            [entry_room_number, room_type_variable, ac_nonac_variable,
entry_price, status_variable])

        list_new_room_numbers.append(entry_room_number.get())

        list_options.append([room_type_option, ac_nonac_option, status_option])

        button_delete = Button(page, text="X",
                               command=lambda: click_room_delete(list_data,
list_new_data, list_delete,
                                                                 list_options, list_new_room_numbers,
button_delete, window, geometry),bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Century Gothic', int(12*HF)))
        list_delete.append(button_delete)
        button_delete.grid(row=placement_row, column=9, rowspan=2)

        Room_Info_Page_Admin_Func.Placement_Row_For_New_Room += 2

        label_1.grid(row=placement_row + 2, column=0, columnspan=9)
        button_add.grid(row=placement_row + 3, column=0, columnspan=9)
        label_2.grid(row=placement_row + 4, column=0, columnspan=9)
        button_save.grid(row=placement_row + 5, column=3, columnspan=6)
        button_back.grid(row=placement_row + 5, column=0, columnspan=3)


        if state == 'normal':
            window.state('zoomed')
            Room_Info_Page_Admin_Func.state = 'zoomed'


        else:
            window.state('normal')
            window.geometry(geometry)
            Room_Info_Page_Admin_Func.state = 'normal'


    # ------------------------------------------------------------------------------------------------------------------
    def click_room_delete(list_data, list_new_data, list_delete, list_options,
list_new_room_numbers,  button, window, geometry):


        index = list_delete.index(button)


        if index != len(list_delete)-1 :
            messagebox.showerror("Resort Ivory Bliss", "Please delete the last room first.")


        else:
            delete_data = list_new_data[index][0]["text"]
            for data in list_data:
```

```
    if data[0] == delete_data:
        list_data.remove(data)

for widget in [list_new_data[index][0], list_new_data[index][3]]:
    widget.grid_forget()

list_options[index][0].grid_forget()
list_options[index][1].grid_forget()
list_options[index][2].grid_forget()

del list_new_data[index]
del list_options[index]
button.grid_forget()
list_delete.remove(button)

if list_new_room_numbers != []:
    del list_new_room_numbers[-1]

if Room_Info_Page_Admin_Func.state == 'normal':
    window.state('zoomed')
    Room_Info_Page_Admin_Func.state = 'zoomed'

else:
```

```
            window.state('normal')

            window.geometry(geometry)

            Room_Info_Page_Admin_Func.state = 'normal'

    # -----------------------------------------------------------------------------------------------------

    con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")

    mycursor_services = con.cursor()

    mycursor_services.execute("SELECT * FROM room_info")

    con.close()


    Room_Info_Page_Admin_Func.rooms_data = mycursor_services.fetchall()


    Room_Info_Page_Admin_Func.list_for_new_data = []

    Room_Info_Page_Admin_Func.list_delete_button = []

    Room_Info_Page_Admin_Func.list_for_new_options = []


    Room_Info_Page_Admin_Func.List_Entry_Room_Number = []

    Room_Info_Page_Admin_Func.List_Entry_Room_Number_Data = []

    Room_Info_Page_Admin_Func.List_Room_Type_Var = []

    Room_Info_Page_Admin_Func.List_Room_Type_Option = []

    Room_Info_Page_Admin_Func.List_AC_NonAC_Option = []

    Room_Info_Page_Admin_Func.List_AC_Non_AC_Var = []

    Room_Info_Page_Admin_Func.List_Entry_Price = []
Room_Info_Page_Admin_Func.List_Status_Option = []

Room_Info_Page_Admin_Func.List_Status_Var = []


Room_Info_Page_Admin_Func.List_Entry_New_Room_Number_Data = []


Room_Info_Page = Tk()


sw = Room_Info_Page.winfo_screenwidth()

sh = Room_Info_Page.winfo_screenheight()


gwf = 1915 / 1920

ghf = 1050 / 1080


wf = sw / 1920 * gwf

hf = sh / 1080 * ghf


g = str(int(sw * gwf)) + "x" + str(int(sh * ghf))


Room_Info_Page.config(bg="#00FFFF")

Room_Info_Page.geometry(g)

Room_Info_Page.title("Resort Ivory Bliss - Edit Rooms Info (Admin)")

Room_Info_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")

Room_Info_Page.resizable(0,0)
```

```
    Room_Info_Page_Admin_Func.state = 'normal'


    main_frame = Frame(Room_Info_Page, bg="#00FFFF")
    main_frame.pack(fill=BOTH, expand=1)
    my_canvas = Canvas(main_frame, bg="#00FFFF")
    my_canvas.pack(side=LEFT, fill=BOTH, expand=1)
    my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)
    my_scrollbar.pack(side=RIGHT, fill=Y)
    my_canvas.configure(yscrollcommand=my_scrollbar.set)
    my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))
    second_frame = Frame(my_canvas,bg="#00FFFF")
    my_canvas.create_window((0, 0), window=second_frame, anchor=NW)


    label_main_heading = Label(second_frame, text="EDIT  ROOMS
INFO",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(35*hf)),
pady=50*hf)
    label_main_heading.grid(row=0, column=0, columnspan=9, rowspan=2)


    label_heading_room_no = Label(second_frame, text="Room
Number",bg="#00FFFF",fg = "navy" ,font=('Bookman Old Style bold', int(25*hf)),
pady=50*hf, padx=70*wf)
```

```
    label_heading_room_type = Label(second_frame, text="Room Type",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(25*hf)), pady=50*hf, padx=70*wf)

    label_heading_ac_nonac = Label(second_frame, text="AC / Non-AC",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(25*hf)), pady=50*hf, padx=70*wf)

    label_heading_price = Label(second_frame, text="Price per day in Rs.",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(25*hf)), pady=50*hf, padx=70*wf)

    label_heading_status = Label(second_frame, text="Status",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(25*hf)), pady=50*hf, padx=70*wf)


    label_heading_room_no.grid(row=2, column=0, rowspan=2)

    label_heading_room_type.grid(row=2, column=2, rowspan=2)

    label_heading_ac_nonac.grid(row=2, column=4, rowspan=2)

    label_heading_price.grid(row=2, column=6, rowspan=2)

    label_heading_status.grid(row=2, column=8, rowspan=2)


    Placement_Row_For_Entry_Room_No = 4

    for entry_room_no in range(len(Room_Info_Page_Admin_Func.rooms_data)):

        i = entry_room_no

        entry_room_no = Entry(second_frame, justify=CENTER,disabledbackground="#00FFFF",disabledforeground = "navy", font=('Century Gothic bold', int(15*hf)))

        entry_room_no.grid(row=Placement_Row_For_Entry_Room_No, column=0, rowspan=2, pady=30*hf)
        entry_room_no.insert(0, Room_Info_Page_Admin_Func.rooms_data[i][0])
        entry_room_no.configure(state=DISABLED, bd=0)

Room_Info_Page_Admin_Func.List_Entry_Room_Number.append(entry_room_no)

Room_Info_Page_Admin_Func.List_Entry_Room_Number_Data.append(Room_Info_
Page_Admin_Func.rooms_data[i][0])
        Placement_Row_For_Entry_Room_No += 2


    for opt_room_type_variable in
range(len(Room_Info_Page_Admin_Func.rooms_data)):
        i = opt_room_type_variable
        opt_room_type_variable = StringVar()
        opt_room_type_variable.set(Room_Info_Page_Admin_Func.rooms_data[i][1])

Room_Info_Page_Admin_Func.List_Room_Type_Var.append(opt_room_type_variab
le)


    list_of_room_types = ["CLASSIC", "DELUXE", "SUPER DELUXE", "ELITE", "COTTAGE"]


    Placement_Row_For_Option_Room_Type = 4
    for opt_room_type in range(len(Room_Info_Page_Admin_Func.rooms_data)):
        i = opt_room_type
        opt_room_type = OptionMenu(second_frame,
                            Room_Info_Page_Admin_Func.List_Room_Type_Var[i],
                    *list_of_room_types)
        opt_room_type.grid(row=Placement_Row_For_Option_Room_Type, column=2,
rowspan=2)
        opt_room_type.configure(font=('Century Gothic',
int(15*hf)),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground
= "navy", width=15)
        Placement_Row_For_Option_Room_Type += 2


    for opt_ac_nonac_variable in
range(len(Room_Info_Page_Admin_Func.rooms_data)):
        i = opt_ac_nonac_variable
        opt_ac_nonac_variable = StringVar()
        opt_ac_nonac_variable.set(Room_Info_Page_Admin_Func.rooms_data[i][2])

Room_Info_Page_Admin_Func.List_AC_Non_AC_Var.append(opt_ac_nonac_varia
ble)


    Placement_Row_For_Option_AC_NonAC = 4
    for opt_ac_nonac in range(len(Room_Info_Page_Admin_Func.rooms_data)):
        i = opt_ac_nonac
        opt_ac_nonac = OptionMenu(second_frame,
                    Room_Info_Page_Admin_Func.List_AC_Non_AC_Var[i],
                    "AC", "NON AC")
        opt_ac_nonac.grid(row=Placement_Row_For_Option_AC_NonAC, column=4,
rowspan=2)
```

```
        opt_ac_nonac.configure(font=('Century Gothic', 
int(15*hf)),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground 
= "navy", width=15)
        Placement_Row_For_Option_AC_NonAC += 2

    Placement_Row_For_Entry_Price = 4
    for entry_price in range(len(Room_Info_Page_Admin_Func.rooms_data)):
        i = entry_price
        entry_price = Entry(second_frame,fg = "navy", font=('Century Gothic', 
int(15*hf)), justify=CENTER)
        entry_price.grid(row=Placement_Row_For_Entry_Price, column=6, rowspan=2)
        entry_price.insert(0, Room_Info_Page_Admin_Func.rooms_data[i][3])
        Room_Info_Page_Admin_Func.List_Entry_Price.append(entry_price)
        Placement_Row_For_Entry_Price += 2

    for opt_status_variable in range(len(Room_Info_Page_Admin_Func.rooms_data)):
        i = opt_status_variable
        opt_status_variable = StringVar()
        opt_status_variable.set(Room_Info_Page_Admin_Func.rooms_data[i][4])
        Room_Info_Page_Admin_Func.List_Status_Var.append(opt_status_variable)

    Placement_Row_For_Option_Status = 4
    for opt_status in range(len(Room_Info_Page_Admin_Func.rooms_data)):
        i = opt_status
        opt_status = OptionMenu(second_frame,
                                Room_Info_Page_Admin_Func.List_Status_Var[i],
                                "Available", "Not Available")
        opt_status.grid(row=Placement_Row_For_Option_Status, column=8,
rowspan=2)
        opt_status.configure(font=('Century Gothic',
int(15*hf)),bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground
= "navy", width=15)
        Placement_Row_For_Option_Status += 2


    Room_Info_Page_Admin_Func.Placement_Row_For_New_Room =
Placement_Row_For_Entry_Room_No


    label_space_1 = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy",
pady=30*hf)

label_space_1.grid(row=Room_Info_Page_Admin_Func.Placement_Row_For_New_
Room, column=0, columnspan=9)


    label_space_2 = Label(second_frame, text="\t",bg="#00FFFF",fg = "navy",
pady=30*hf)

label_space_2.grid(row=Room_Info_Page_Admin_Func.Placement_Row_For_New_
Room + 4, column=0, columnspan=9)
```

```
    button_save_changes = Button(second_frame,bg="navy",activebackground =
"#00FFFF",fg = "#00FFFF",activeforeground = "navy", font=('Century Gothic bold',
int(20*hf)),

                                text="Save Changes", width=60,

                                command=lambda: save_room_changes(

                                    Room_Info_Page_Admin_Func.rooms_data,

                                    Room_Info_Page_Admin_Func.list_for_new_data,

                                    Room_Info_Page_Admin_Func.List_Entry_Room_Number_Data,


Room_Info_Page_Admin_Func.List_Entry_New_Room_Number_Data,

                                    Room_Info_Page_Admin_Func.list_delete_button,

                                    Room_Info_Page_Admin_Func.list_for_new_options,

                                    Room_Info_Page_Admin_Func.List_Entry_Room_Number,

                                    Room_Info_Page_Admin_Func.List_Room_Type_Var,

                                    Room_Info_Page_Admin_Func.List_AC_Non_AC_Var,

                                    Room_Info_Page_Admin_Func.List_Entry_Price,

                                    Room_Info_Page_Admin_Func.List_Status_Var, hf))


button_save_changes.grid(row=Room_Info_Page_Admin_Func.Placement_Row_For
_New_Room + 6, column=3, columnspan=6)



    button_add_item = Button(second_frame, text="Add Room",
width=60,bg="navy",activebackground = "#00FFFF",fg = "#00FFFF",activeforeground
= "navy", font=('Century Gothic bold', int(20*hf)))

    button_add_item.configure(command=lambda: add_room(
        second_frame,
        Room_Info_Page_Admin_Func.rooms_data,
        Room_Info_Page_Admin_Func.list_for_new_data,
        Room_Info_Page_Admin_Func.List_Entry_Room_Number_Data,
        Room_Info_Page_Admin_Func.List_Entry_New_Room_Number_Data,
        Room_Info_Page_Admin_Func.list_delete_button,
        Room_Info_Page_Admin_Func.list_for_new_options,
        Room_Info_Page_Admin_Func.Placement_Row_For_New_Room,
        label_space_1,
        label_space_2,
        button_add_item,
        button_save_changes,
        button_back_to_main_menu_admin,
        Room_Info_Page, Room_Info_Page_Admin_Func.state, g, hf))

button_add_item.grid(row=Room_Info_Page_Admin_Func.Placement_Row_For_Ne
w_Room + 2, column=0, columnspan=9)


    button_back_to_main_menu_admin = Button(second_frame,
text="Back",bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(20*hf)),
                                        command=lambda:
back_to_main_menu_admin(Room_Info_Page))
```

```
button_back_to_main_menu_admin.grid(row=Room_Info_Page_Admin_Func.Place
ment_Row_For_New_Room + 6, column=0,
                                    columnspan=3)

    Room_Info_Page.mainloop()
# ----------------------------------------------------------------------------------------------------------------
# View Record Log
def Admin_View_Record_Log():
    con = mysql.connector.connect(host="localhost", user="root", passwd="root",
database="project")
    mycursor_record_log = con.cursor()

    command = '''SELECT rl.SERIAL_NO, rl.ROOM_NO,ri.ROOM_TYPE, ri.AC_NON_AC,
rl.CID, rl.CUSTOMER_NAME, rl.CHECK_IN_DATE, rl.CHECK_IN_TIME,
rl.CHECK_OUT_DATE, rl.CHECK_OUT_TIME, rl.STATUS
                                              FROM record_log rl, room_info ri
                                              WHERE rl.ROOM_NO = ri.ROOM_NO'''
    mycursor_record_log.execute(command)
    record_log_data = mycursor_record_log.fetchall()
    con.close()

    Record_Log_Page = Tk()

    sw = Record_Log_Page.winfo_screenwidth()
    sh = Record_Log_Page.winfo_screenheight()


    wf = sw / 1920
    hf = sh / 1080


    Record_Log_Page.config(bg="#00FFFF")
    Record_Log_Page.title("Resort Ivory Bliss - Record Log")
    Record_Log_Page.iconbitmap("D:\\pythonProject\\HOTEL.ico")
    Record_Log_Page.resizable(0,0)
    Record_Log_Page.state('zoomed')


    if len(record_log_data) == 0:
        label_main_heading = Label(Record_Log_Page, text="RECORD
LOG",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold',int(40*hf)))
        label_empty_1 = Label(Record_Log_Page, text="Sorry",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style', int(35*hf)))
        label_empty_2 = Label(Record_Log_Page, text="No bookings
yet",bg="#00FFFF",fg = "navy", font=('Bookman Old Style', int(35*hf)))
        button_back_to_main_menu = Button(Record_Log_Page,
text="Back",bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(15*hf)),
                                          command=lambda: [close(Record_Log_Page),
Main_Menu_Func()])
        label_main_heading.place(x=0*wf, y=200*hf,width=sw)
        label_empty_1.place(x=0*wf, y=400*hf,width=sw)
        label_empty_2.place(x=0*wf, y=500*hf,width=sw)

        button_back_to_main_menu.place(x=910*wf,y=700*hf,width=100*wf)

        Record_Log_Page.mainloop()


    else:

        main_frame = Frame(Record_Log_Page, bg="#00FFFF")

        main_frame.pack(fill=BOTH, expand=1)

        my_canvas = Canvas(main_frame, bg="#00FFFF")

        my_canvas.pack(side=LEFT, fill=BOTH, expand=1)

        my_scrollbar = ttk.Scrollbar(main_frame, orient=VERTICAL,
command=my_canvas.yview)

        my_scrollbar.pack(side=RIGHT, fill=Y)

        my_canvas.configure(yscrollcommand=my_scrollbar.set)

        my_canvas.bind('<Configure>', lambda e:
my_canvas.configure(scrollregion=my_canvas.bbox("all")))

        second_frame = Frame(my_canvas,bg="#00FFFF")

        my_canvas.create_window((0, 0), window=second_frame, anchor=NW)


        label_main_heading = Label(second_frame, text='RECORD
LOG',bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(30*hf)),
pady=30*hf)

        label_main_heading.grid(row=0, column=0, columnspan=11, rowspan=2)


        label_heading_sno = Label(second_frame, text="Serial No.",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold', int(15*hf)), pady=30*hf,
                        padx=15*wf)

    label_heading_room_no = Label(second_frame, text="Room 
Number",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(15*hf)), 
pady=30*hf,

                          padx=15*wf)

    label_heading_room_type = Label(second_frame, text="Room 
Type",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(15*hf)), 
pady=30*hf,

                           padx=15*wf)

    label_heading_ac_nonac = Label(second_frame, text="AC / Non-
AC",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(15*hf)), 
pady=30*hf,

                           padx=15*wf)

    label_heading_cus_id = Label(second_frame, text="Customer 
ID",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(15*hf)), pady=30*hf,

                          padx=15*wf)

    label_heading_cus_name = Label(second_frame, text="Customer 
Name",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(15*hf)), 
pady=30*hf,

                           padx=15*wf)

    label_heading_checkin_date = Label(second_frame, text="Check-In 
Date",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(15*hf)),

                             pady=30*hf, padx=15*wf)

    label_heading_checkin_time = Label(second_frame, text="Check-In 
Time",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(15*hf)),

                             pady=30*hf, padx=15*wf)
```

```
        label_heading_checkout_date = Label(second_frame, text="Check-Out
Date",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(15*hf)),
                                    pady=30*hf, padx=15*wf)
        label_heading_checkout_time = Label(second_frame, text="Check-Out
Time",bg="#00FFFF",fg = "navy", font=('Bookman Old Style bold', int(15*hf)),
                                    pady=30*hf, padx=15*wf)
        label_heading_status = Label(second_frame, text="Status",bg="#00FFFF",fg =
"navy", font=('Bookman Old Style bold', int(15*hf)), pady=30*hf,
                              padx=20*wf)

        label_heading_sno.grid(row=2, column=0, rowspan=2)
        label_heading_room_no.grid(row=2, column=1, rowspan=2)
        label_heading_room_type.grid(row=2, column=2, rowspan=2)
        label_heading_ac_nonac.grid(row=2, column=3, rowspan=2)
        label_heading_cus_id.grid(row=2, column=4, rowspan=2)
        label_heading_cus_name.grid(row=2, column=5, rowspan=2)
        label_heading_checkin_date.grid(row=2, column=6, rowspan=2)
        label_heading_checkin_time.grid(row=2, column=7, rowspan=2)
        label_heading_checkout_date.grid(row=2, column=8, rowspan=2)
        label_heading_checkout_time.grid(row=2, column=9, rowspan=2)
        label_heading_status.grid(row=2, column=10, rowspan=2)

        Placement_Row_For_Label_SNo = 4
        for label_sno in range(len(record_log_data)):
```

```
        label_index = label_sno
        label_sno = Label(second_frame,
text=record_log_data[label_index][0],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)), pady=15*hf)
        label_sno.grid(row=Placement_Row_For_Label_SNo, column=0, rowspan=2)
        Placement_Row_For_Label_SNo += 2


    Placement_Row_For_Label_Room_No = 4
    for label_room_no in range(len(record_log_data)):
        label_index = label_room_no
        label_room_no = Label(second_frame,
text=record_log_data[label_index][1],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)),
                              pady=15*hf)
        label_room_no.grid(row=Placement_Row_For_Label_Room_No, column=1,
rowspan=2)
        Placement_Row_For_Label_Room_No += 2


    Placement_Row_For_Label_Room_Type = 4
    for label_room_type in range(len(record_log_data)):
        label_index = label_room_type
        label_room_type = Label(second_frame,
text=record_log_data[label_index][2],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)),
                               pady=15*hf)
```

```
            label_room_type.grid(row=Placement_Row_For_Label_Room_Type,
column=2, rowspan=2)
            Placement_Row_For_Label_Room_Type += 2


        Placement_Row_For_Label_AC_NonAC = 4
        for label_ac_nonac in range(len(record_log_data)):
            label_index = label_ac_nonac
            label_ac_nonac = Label(second_frame,
text=record_log_data[label_index][3],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)),
                                   pady=15*hf)
            label_ac_nonac.grid(row=Placement_Row_For_Label_AC_NonAC,
column=3, rowspan=2)
            Placement_Row_For_Label_AC_NonAC += 2


        Placement_Row_For_Label_Cus_ID = 4
        for label_cus_id in range(len(record_log_data)):
            label_index = label_cus_id
            label_cus_id = Label(second_frame,
text=record_log_data[label_index][4],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)),
                                 pady=15*hf)
            label_cus_id.grid(row=Placement_Row_For_Label_Cus_ID, column=4,
rowspan=2)
            Placement_Row_For_Label_Cus_ID += 2
```

```
    Placement_Row_For_Label_Cus_Name = 4
    for label_cus_name in range(len(record_log_data)):
        label_index = label_cus_name
        label_cus_name = Label(second_frame,
text=record_log_data[label_index][5],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)),
                          pady=15*hf)
        label_cus_name.grid(row=Placement_Row_For_Label_Cus_Name,
column=5, rowspan=2)
        Placement_Row_For_Label_Cus_Name += 2

    Placement_Row_For_Label_Checkin_Date = 4
    for label_checkin_date in range(len(record_log_data)):
        label_index = label_checkin_date
        label_checkin_date = Label(second_frame,
text=record_log_data[label_index][6],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)),
                            pady=15*hf)
        label_checkin_date.grid(row=Placement_Row_For_Label_Checkin_Date,
column=6, rowspan=2)
        Placement_Row_For_Label_Checkin_Date += 2

    Placement_Row_For_Label_Checkin_Time = 4
    for label_checkin_time in range(len(record_log_data)):
         label_index = label_checkin_time
         label_checkin_time = Label(second_frame,
text=record_log_data[label_index][7],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)),
                                      pady=15*hf)
         label_checkin_time.grid(row=Placement_Row_For_Label_Checkin_Time,
column=7, rowspan=2)
         Placement_Row_For_Label_Checkin_Time += 2

      Placement_Row_For_Label_Checkout_Date = 4
      for label_checkout_date in range(len(record_log_data)):
         label_index = label_checkout_date
         label_checkout_date = Label(second_frame,
text=record_log_data[label_index][8],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)),
                                      pady=15*hf)
         label_checkout_date.grid(row=Placement_Row_For_Label_Checkout_Date,
column=8, rowspan=2)
         Placement_Row_For_Label_Checkout_Date += 2

      Placement_Row_For_Label_Checkout_Time = 4
      for label_checkout_time in range(len(record_log_data)):
         label_index = label_checkout_time
```

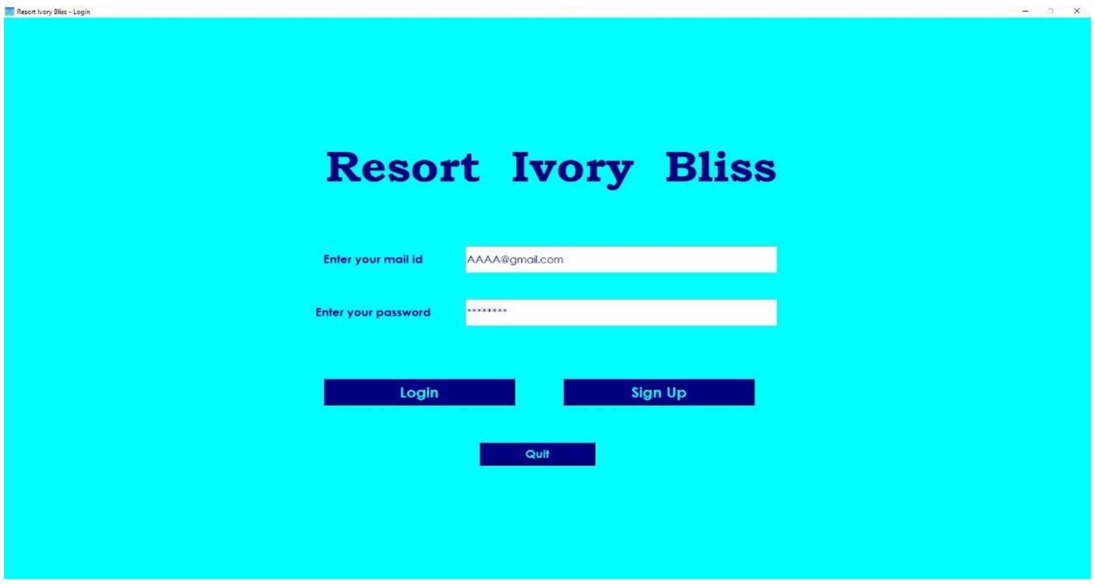
```
            label_checkout_time = Label(second_frame,
text=record_log_data[label_index][9],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)),
                                      pady=15*hf)
            label_checkout_time.grid(row=Placement_Row_For_Label_Checkout_Time,
column=9, rowspan=2)
            Placement_Row_For_Label_Checkout_Time += 2


        Placement_Row_For_Status = 4
        for label_status in range(len(record_log_data)):
            label_index = label_status
            label_status = Label(second_frame,
text=record_log_data[label_index][10],bg="#00FFFF",fg = "navy", font=('Century
Gothic', int(12*hf)),
                                 pady=15*hf)
            label_status.grid(row=Placement_Row_For_Status, column=10, rowspan=2)
            Placement_Row_For_Status += 2


        label_space = Label(second_frame, text='\t',bg="#00FFFF",fg = "navy",
pady=15*hf)
        label_space.grid(row=Placement_Row_For_Status, column=0, rowspan=2)


        button_back_to_main_menu = Button(second_frame,
text="Back",bg="navy",activebackground = "#00FFFF",fg =
"#00FFFF",activeforeground = "navy", font=('Century Gothic bold', int(15*hf)),
                    command=lambda: [close(Record_Log_Page),
Main_Menu_Func()])

    button_back_to_main_menu.grid(row=Placement_Row_For_Status + 2,
column=0)

    Record_Log_Page.mainloop()

# -----------------------------------------------------------------------------------------------------------------------------

# MAIN PROGRAM

Login_Page_Func()
```

END OF SOURCE CODE

Project Screenshots:

➢ Customer Portal:

1. Login Page :



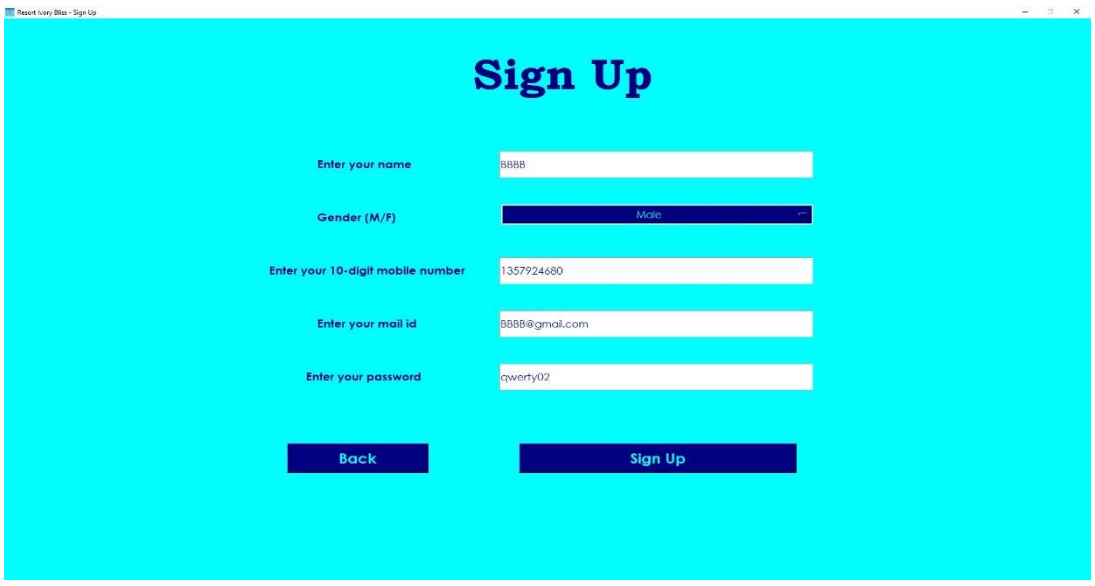
Resort Ivory Bliss - Login

Resort Ivory Bliss

Enter your mail id AAAA@gmail.com

Enter your password ********

2. Sign-Up Page :


Resort Ivory Bliss - Sign Up

Sign Up

Enter your name: BBBB

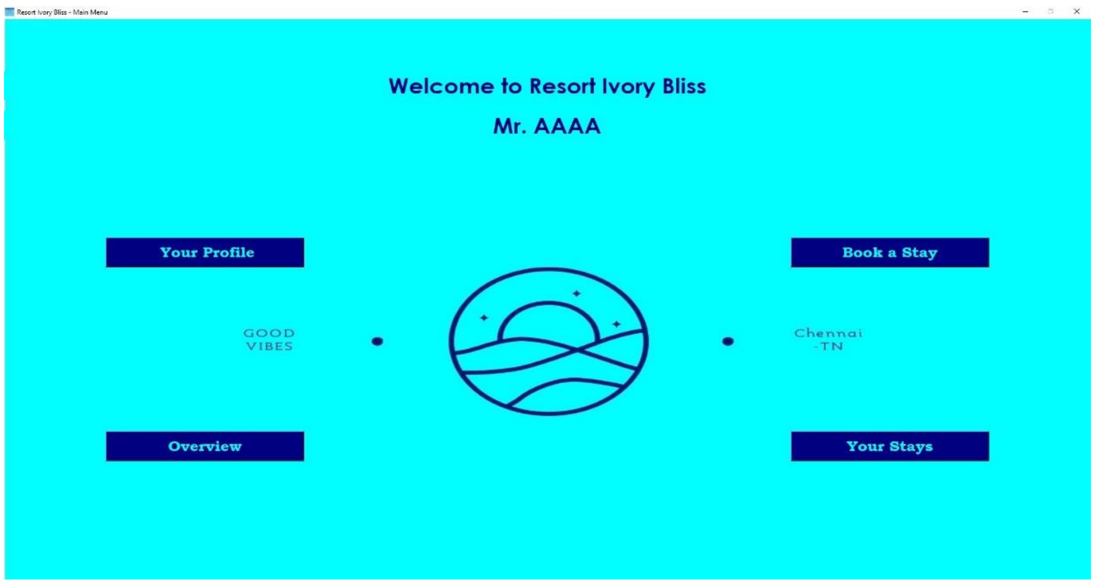
Gender (M/F): Male

Enter your 10-digit mobile number: 1357924680

Enter your mail id: BBBB@gmail.com

Enter your password: qwerty02

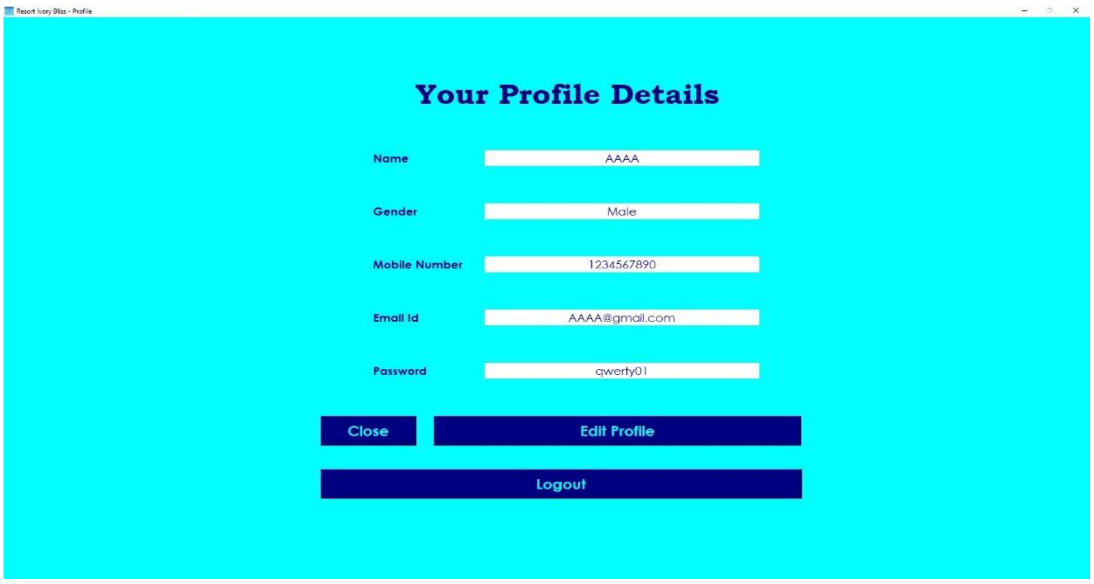
3. Main Menu :


Resort Ivory Bliss - Main Menu

Welcome to Resort Ivory Bliss

Mr. AAAA

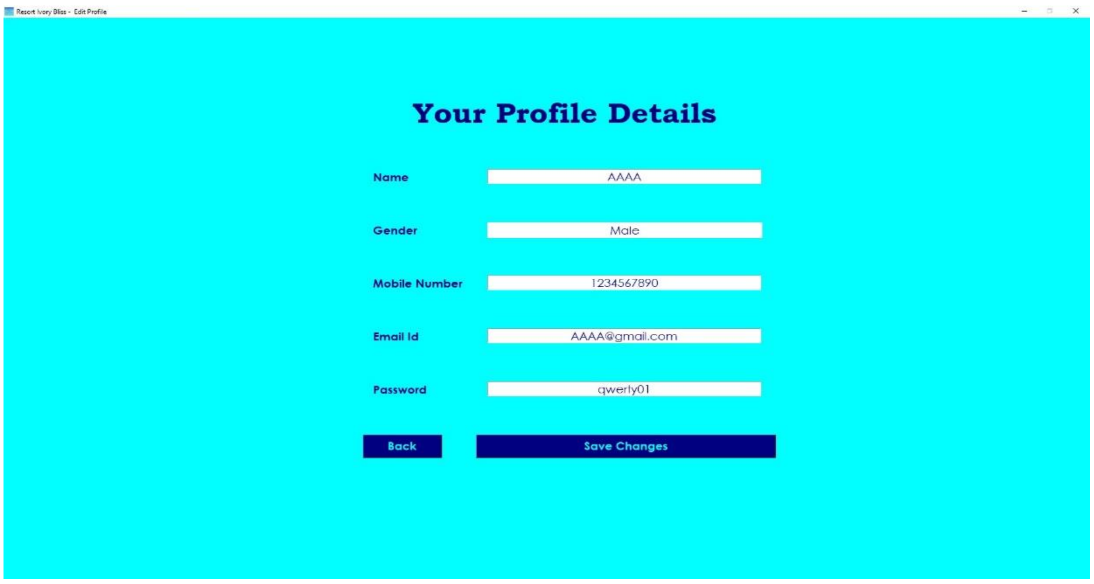
4. Profile Page :



Your Profile Details

Name	AAAA
Gender	Male
Mobile Number	1234567890
Email Id	AAAA@gmail.com
Password	qwerty01

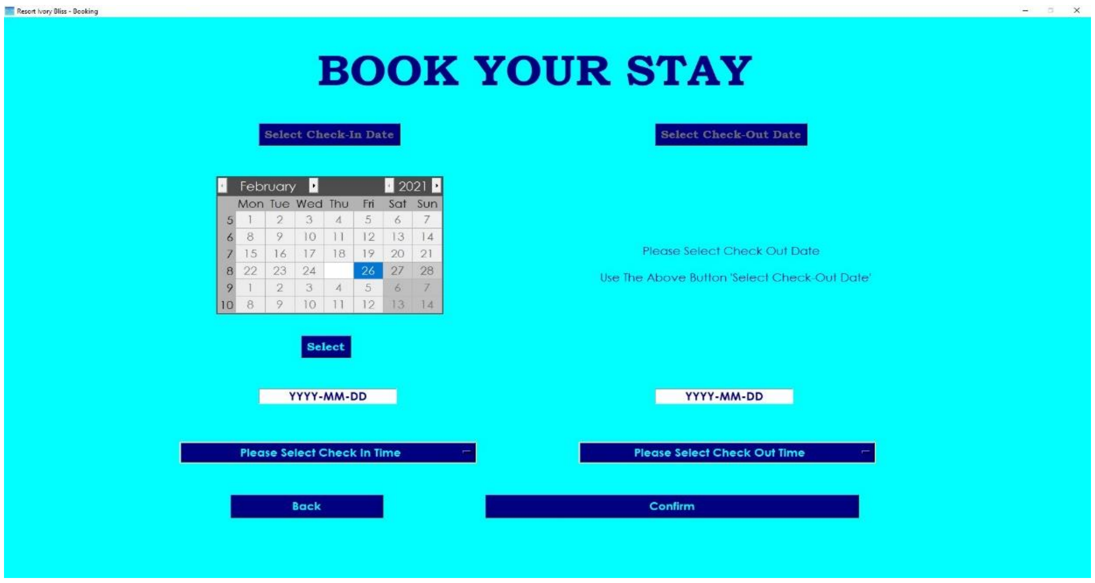
5. Edit Profile Page :



Your Profile Details

Name	AAAA
Gender	Male
Mobile Number	1234567890
Email Id	AAAA@gmail.com
Password	qwerty01

6. Booking Page :


Resort Ivory Bliss - Booking
BOOK YOUR STAY
Select Check-In Date
Select Check-Out Date
Select
Please Select Check In Time
Please Select Check Out Time
Back
Confirm

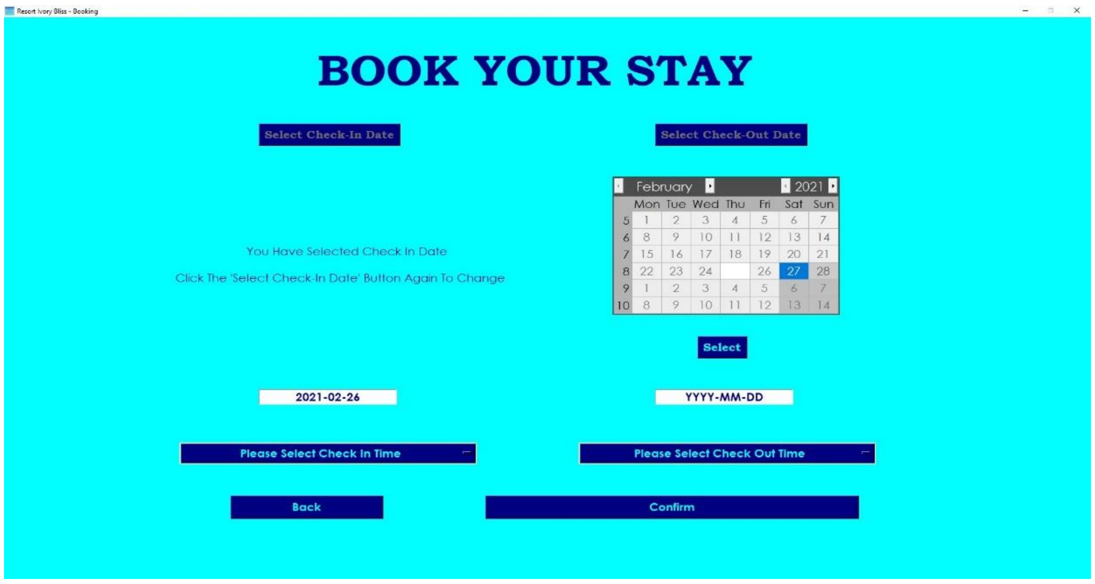
	February				2021		
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
5	1	2	3	4	5	6	7
6	8	9	10	11	12	13	14
7	15	16	17	18	19	20	21
8	22	23	24		26	27	28
9	1	2	3	4	5	6	7
10	8	9	10	11	12	13	14

Please Select Check Out Date

Use The Above Button 'Select Check-Out Date'

YYYY-MM-DD

YYYY-MM-DD


Resort Ivory Bliss - Booking
BOOK YOUR STAY
Select Check-In Date
Select Check-Out Date
Select
Please Select Check In Time
Please Select Check Out Time
Back
Confirm

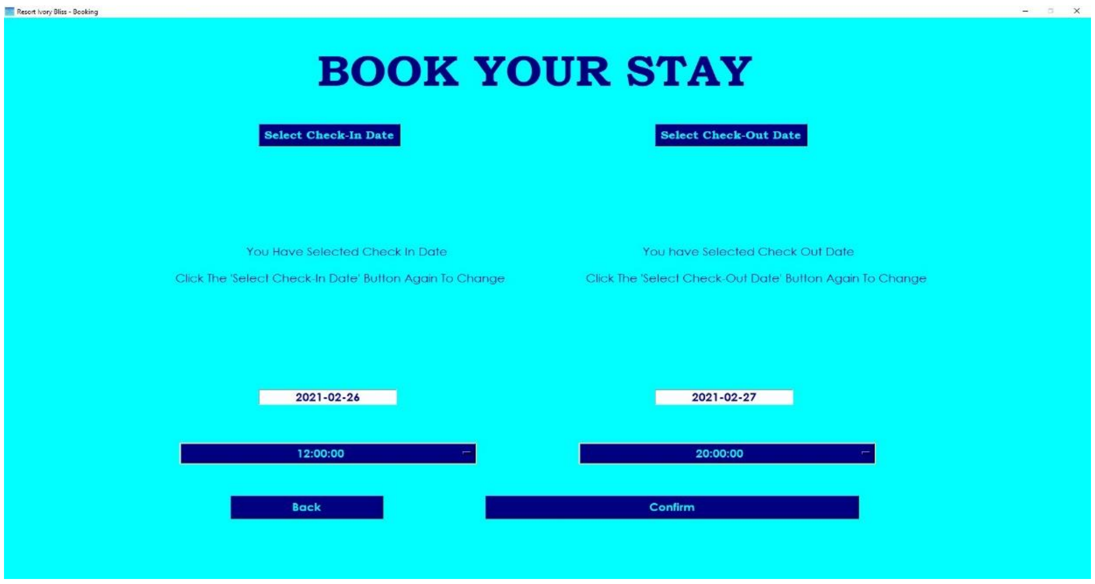
You Have Selected Check In Date

Click The 'Select Check-In Date' Button Again To Change

	February				2021		
	Mon	Tue	Wed	Thu	Fri	Sat	Sun
5	1	2	3	4	5	6	7
6	8	9	10	11	12	13	14
7	15	16	17	18	19	20	21
8	22	23	24		26	27	28
9	1	2	3	4	5	6	7
10	8	9	10	11	12	13	14

2021-02-26

YYYY-MM-DD



You Have Selected Check In Date

Click The 'Select Check-In Date' Button Again To Change

You have Selected Check Out Date

Click The 'Select Check-Out Date' Button Again To Change

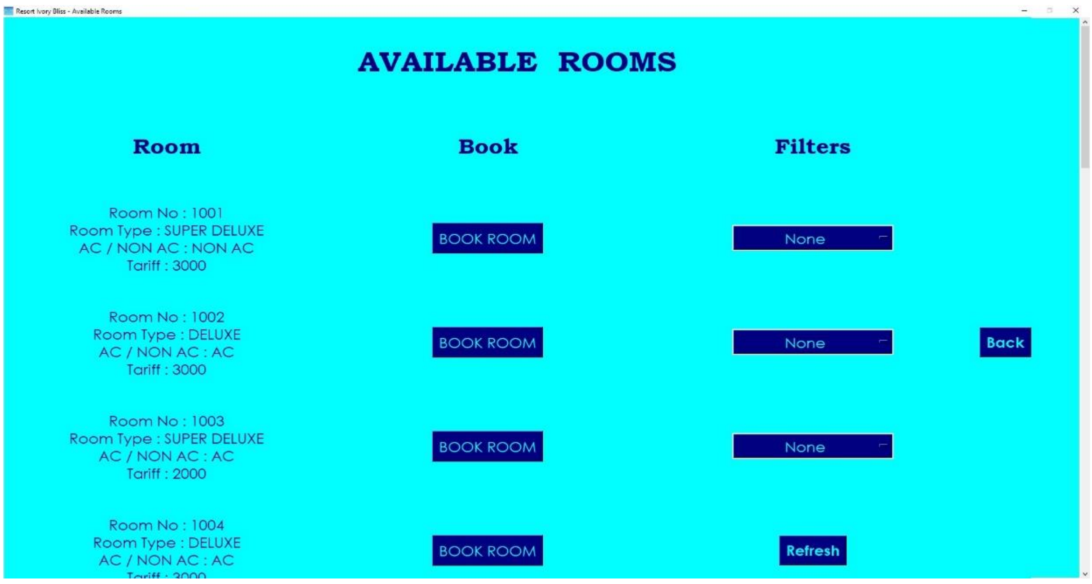
2021-02-26

2021-02-27

12:00:00

20:00:00

7. Available Rooms Page :



AVAILABLE ROOMS

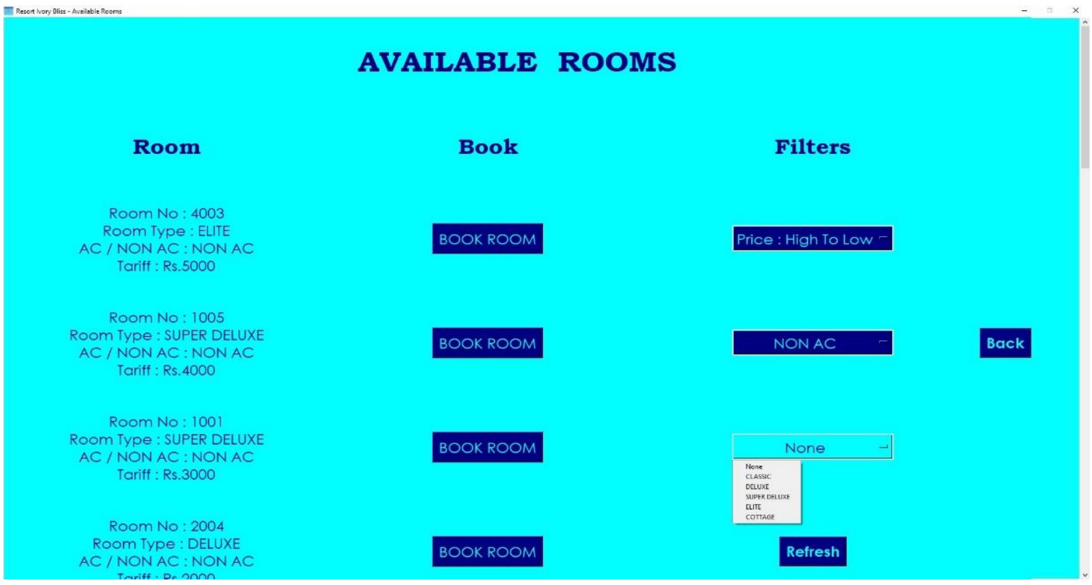
Room | Book | Filters

Room No : 1001
Room Type : SUPER DELUXE
AC / NON AC : NON AC
Tariff : 3000

Room No : 1002
Room Type : DELUXE
AC / NON AC : AC
Tariff : 3000

Room No : 1003
Room Type : SUPER DELUXE
AC / NON AC : AC
Tariff : 2000

Room No : 1004
Room Type : DELUXE
AC / NON AC : AC



AVAILABLE ROOMS

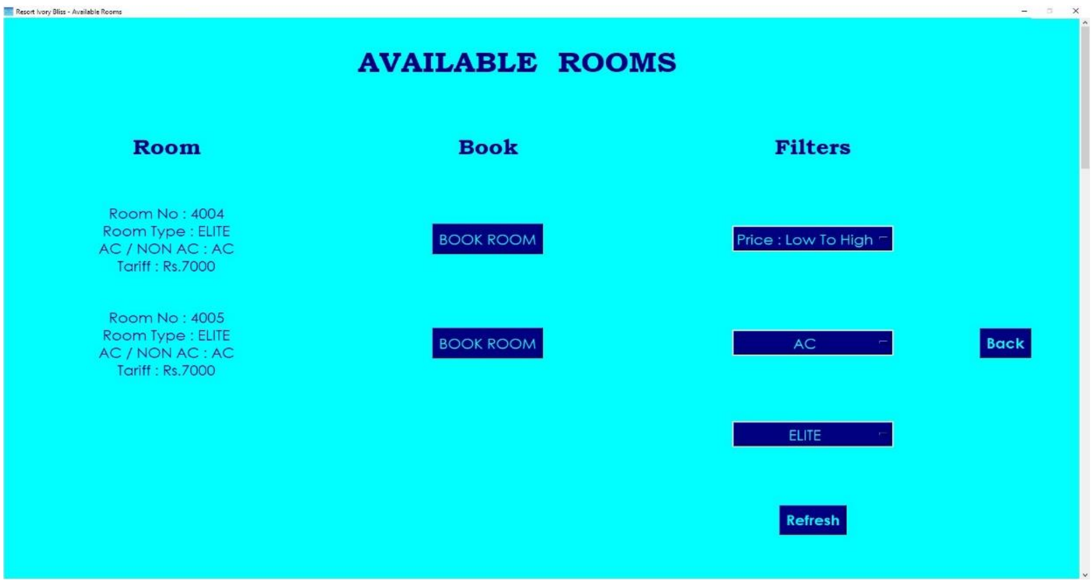
Room | Book | Filters

Room No : 4003
Room Type : ELITE
AC / NON AC : NON AC
Tariff : Rs.5000

Room No : 1005
Room Type : SUPER DELUXE
AC / NON AC : NON AC
Tariff : Rs.4000

Room No : 1001
Room Type : SUPER DELUXE
AC / NON AC : NON AC
Tariff : Rs.3000

Room No : 2004
Room Type : DELUXE
AC / NON AC : NON AC



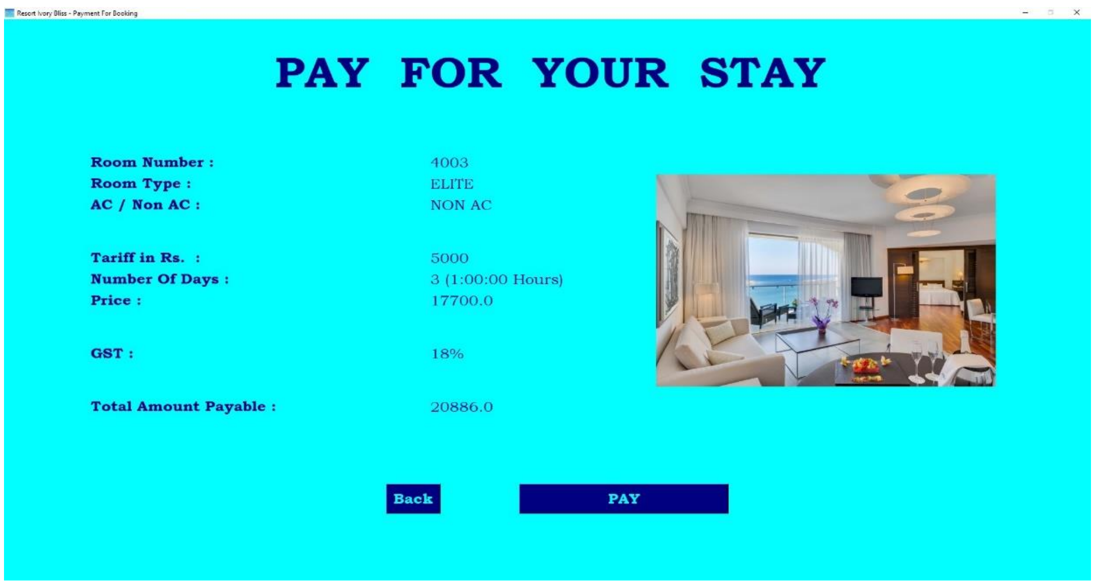
AVAILABLE ROOMS

Room | Book | Filters

Room No : 4004
Room Type : ELITE
AC / NON AC : AC
Tariff : Rs.7000

Room No : 4005
Room Type : ELITE
AC / NON AC : AC
Tariff : Rs.7000

8. Pay For Your Stay :

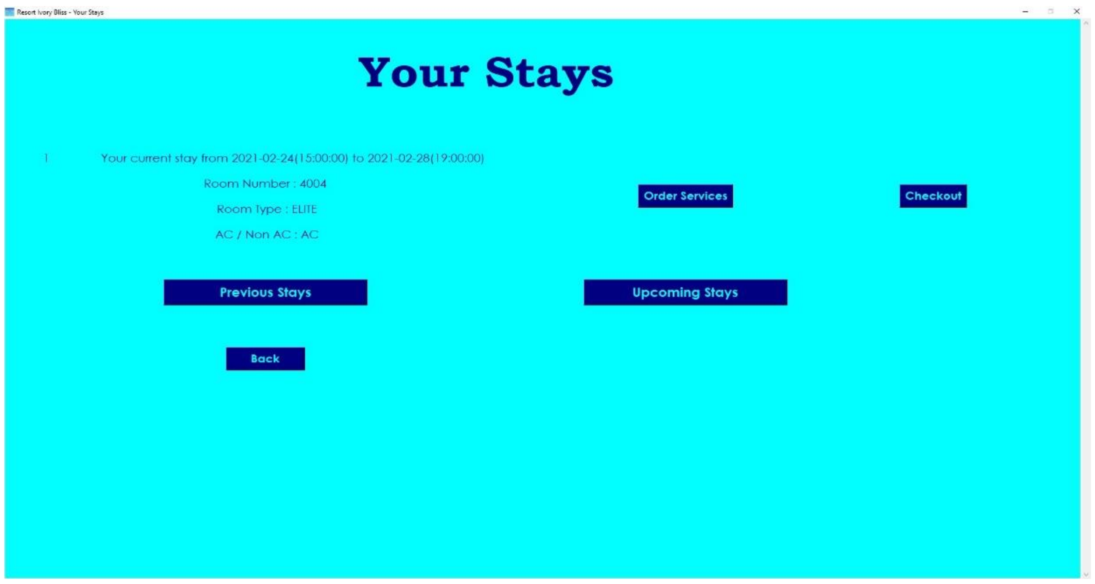


Resort Ivory Bliss - Payment For Booking

PAY FOR YOUR STAY

Room Number :	4003
Room Type :	ELITE
AC / Non AC :	NON AC
Tariff in Rs. :	5000
Number Of Days :	3 (1:00:00 Hours)
Price :	17700.0
GST :	18%
Total Amount Payable :	20886.0

9. Stays :



Resort Ivory Bliss - Your Stays

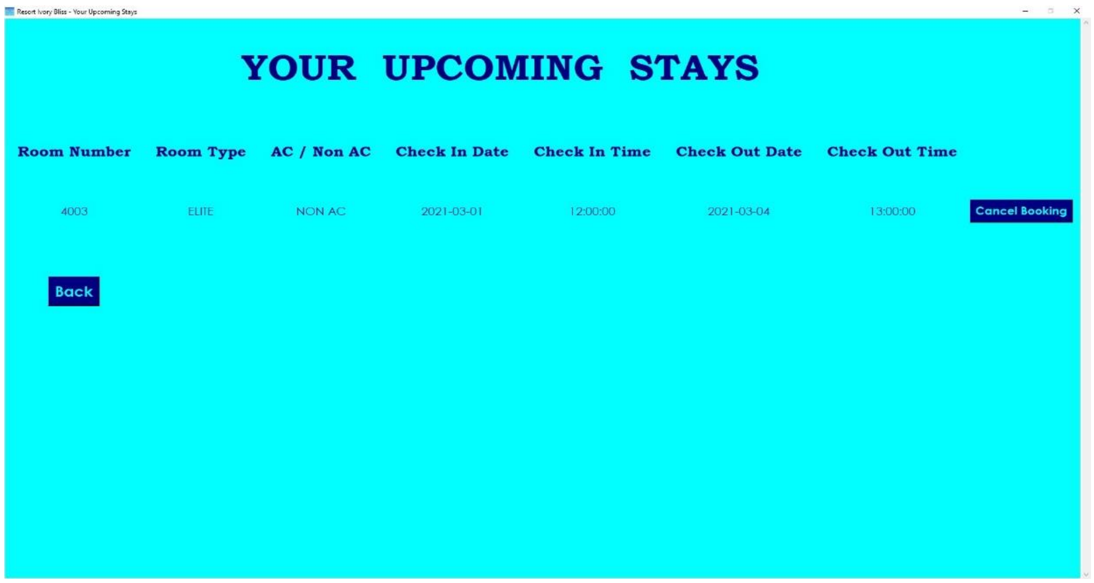
Your Stays

1 Your current stay from 2021-02-24(15:00:00) to 2021-02-28(19:00:00)

Room Number : 4004

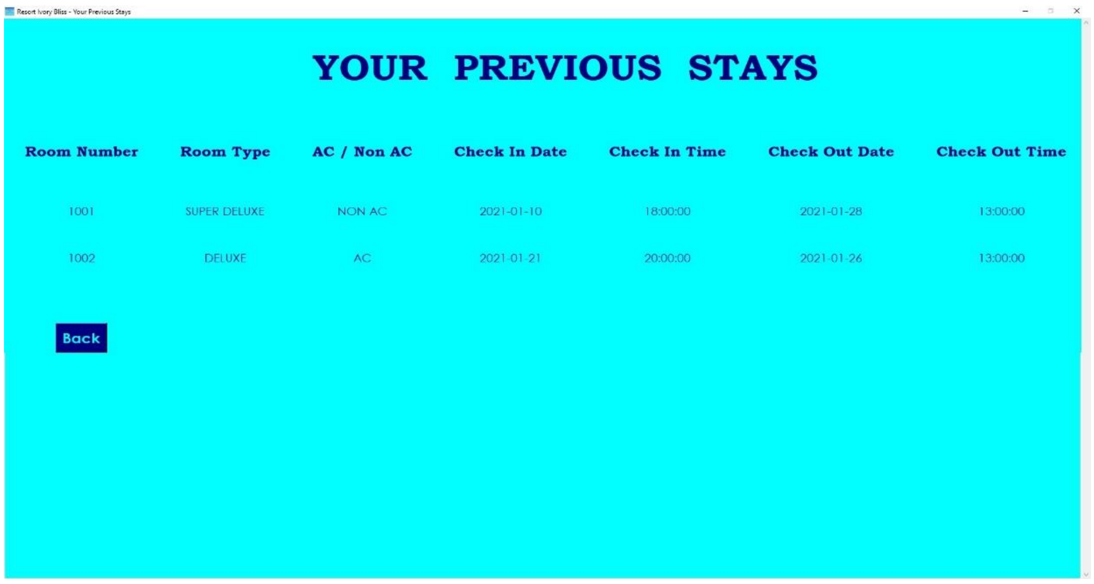
Room Type : ELITE

AC / Non AC : AC



YOUR UPCOMING STAYS

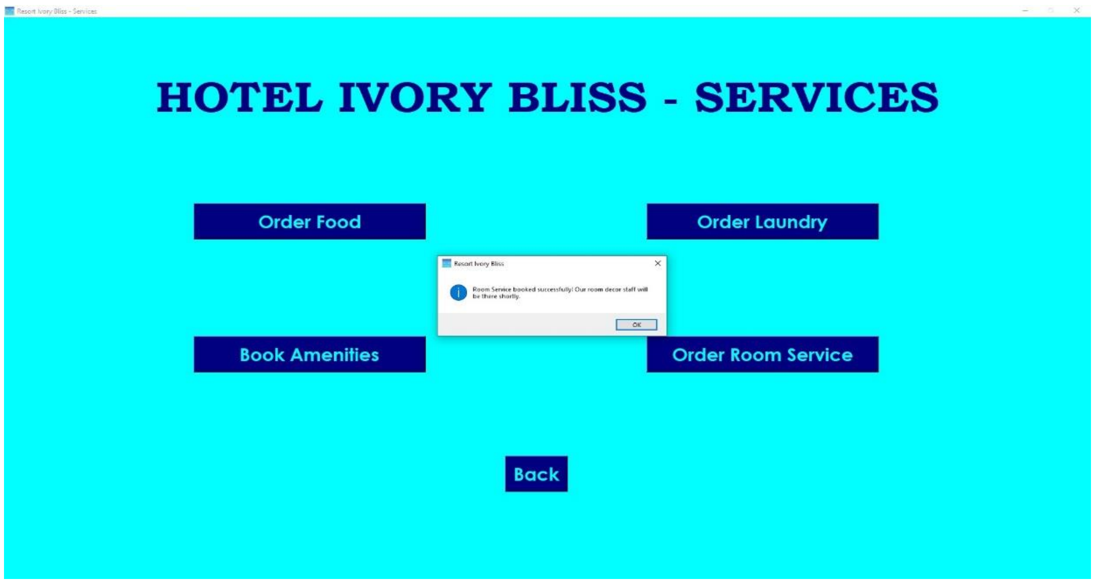
Room Number	Room Type	AC / Non AC	Check In Date	Check In Time	Check Out Date	Check Out Time
4003	ELITE	NON AC	2021-03-01	12:00:00	2021-03-04	13:00:00



YOUR PREVIOUS STAYS

Room Number	Room Type	AC / Non AC	Check In Date	Check In Time	Check Out Date	Check Out Time
1001	SUPER DELUXE	NON AC	2021-01-10	18:00:00	2021-01-28	13:00:00
1002	DELUXE	AC	2021-01-21	20:00:00	2021-01-26	13:00:00

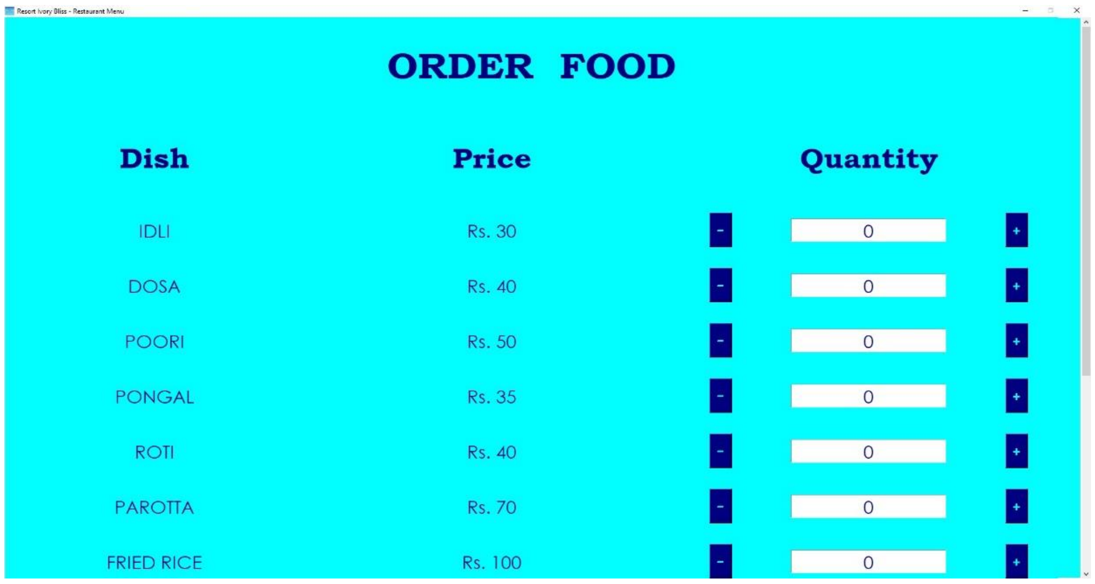
10. Order Services :



HOTEL IVORY BLISS - SERVICES

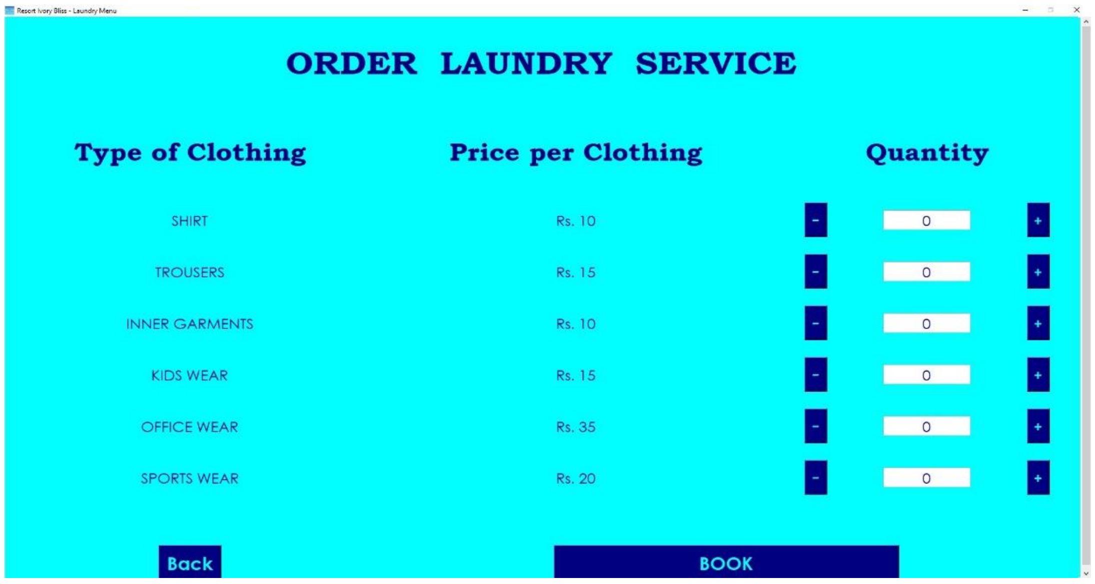
Resort Ivory Bliss

Room Service booked successfully! Our room decor staff will be there shortly.



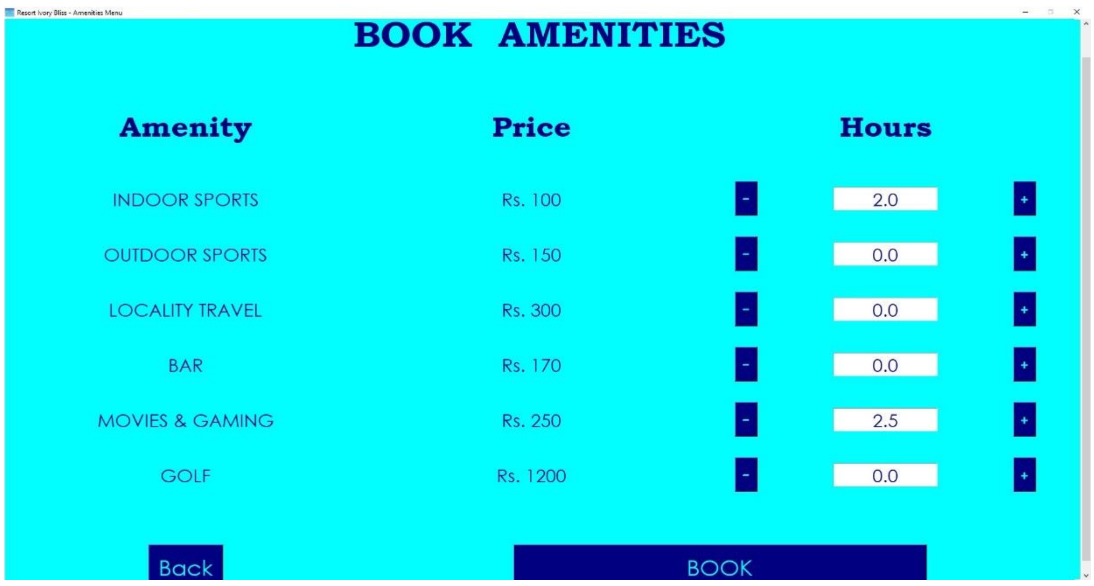
ORDER FOOD

Dish	Price	Quantity
IDLI	Rs. 30	0
DOSA	Rs. 40	0
POORI	Rs. 50	0
PONGAL	Rs. 35	0
ROTI	Rs. 40	0
PAROTTA	Rs. 70	0
FRIED RICE	Rs. 100	0



ORDER LAUNDRY SERVICE

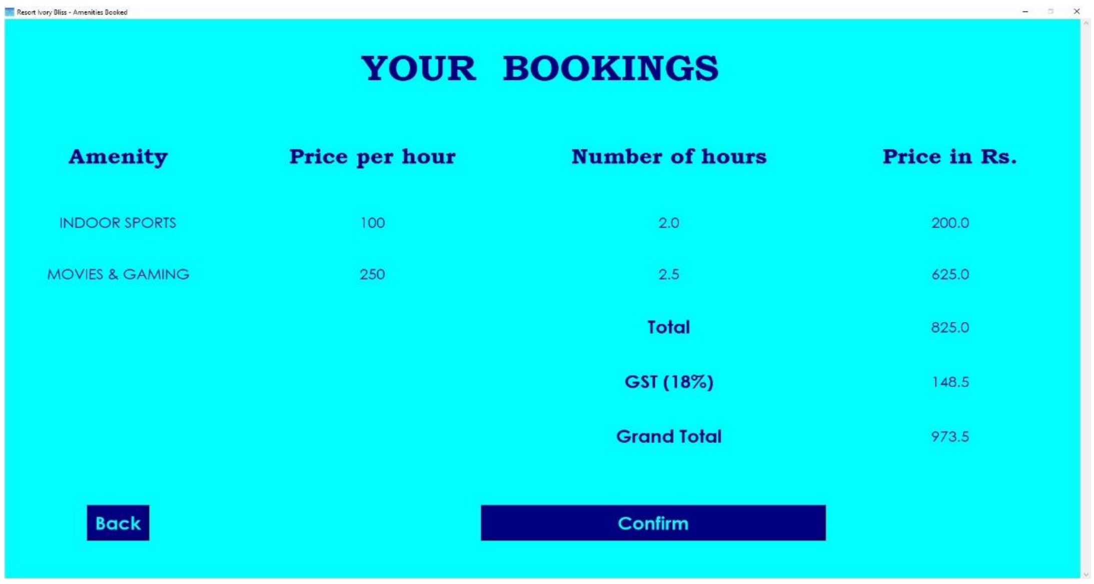
Type of Clothing	Price per Clothing	Quantity
SHIRT	Rs. 10	0
TROUSERS	Rs. 15	0
INNER GARMENTS	Rs. 10	0
KIDS WEAR	Rs. 15	0
OFFICE WEAR	Rs. 35	0
SPORTS WEAR	Rs. 20	0



Resort Ivory Bliss - Amenities Menu

BOOK AMENITIES

Amenity	Price		Hours	
INDOOR SPORTS	Rs. 100	-	2.0	+
OUTDOOR SPORTS	Rs. 150	-	0.0	+
LOCALITY TRAVEL	Rs. 300	-	0.0	+
BAR	Rs. 170	-	0.0	+
MOVIES & GAMING	Rs. 250	-	2.5	+
GOLF	Rs. 1200	-	0.0	+



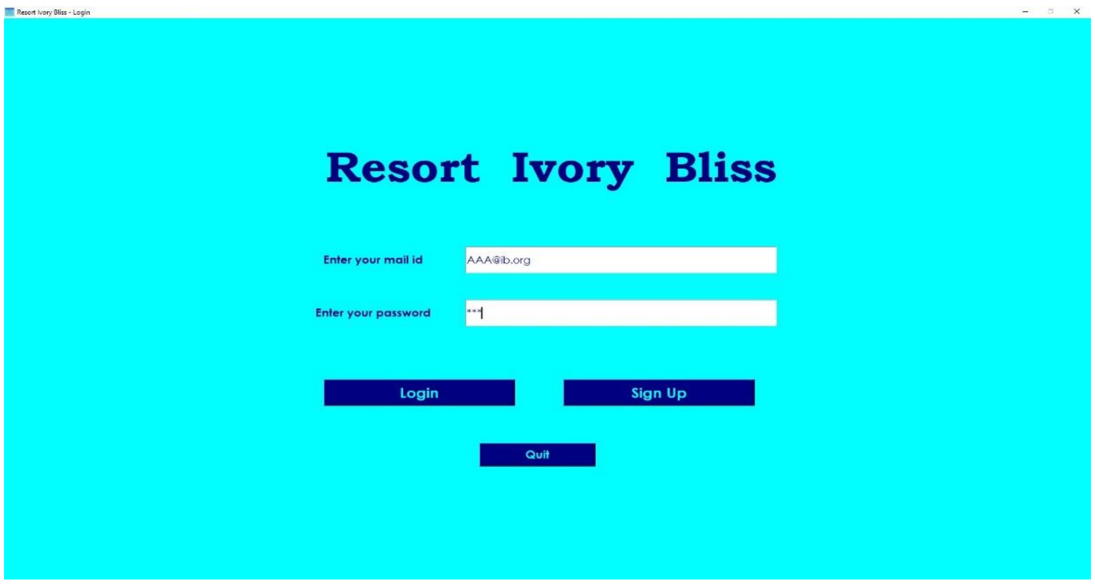
Resort Ivory Bliss - Amenities Booked

YOUR BOOKINGS

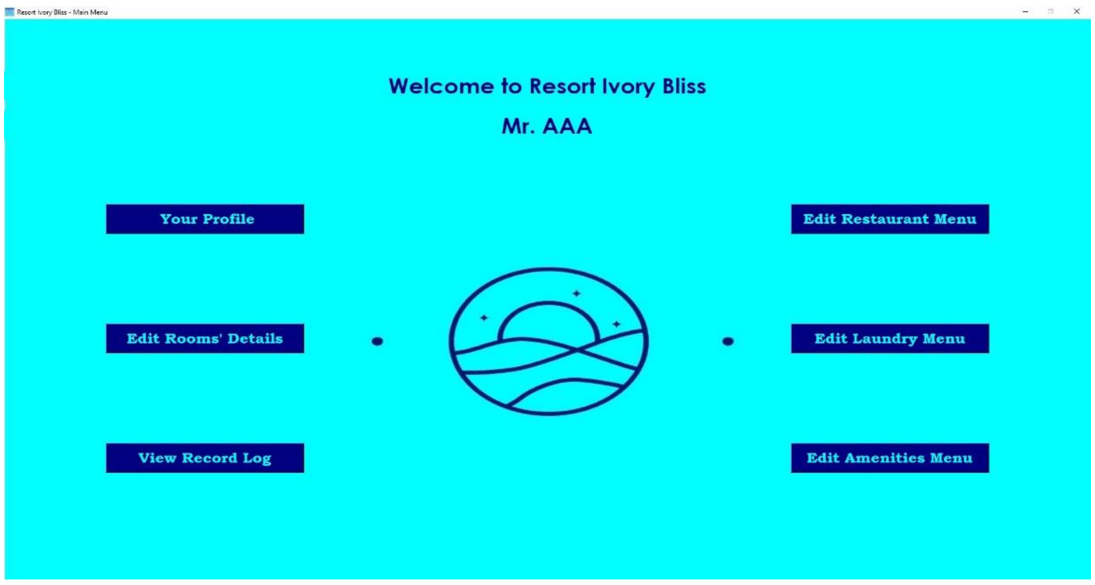
Amenity	Price per hour	Number of hours	Price in Rs.
INDOOR SPORTS	100	2.0	200.0
MOVIES & GAMING	250	2.5	625.0
		Total	825.0
		GST (18%)	148.5
		Grand Total	973.5

➢ Admin Portal :

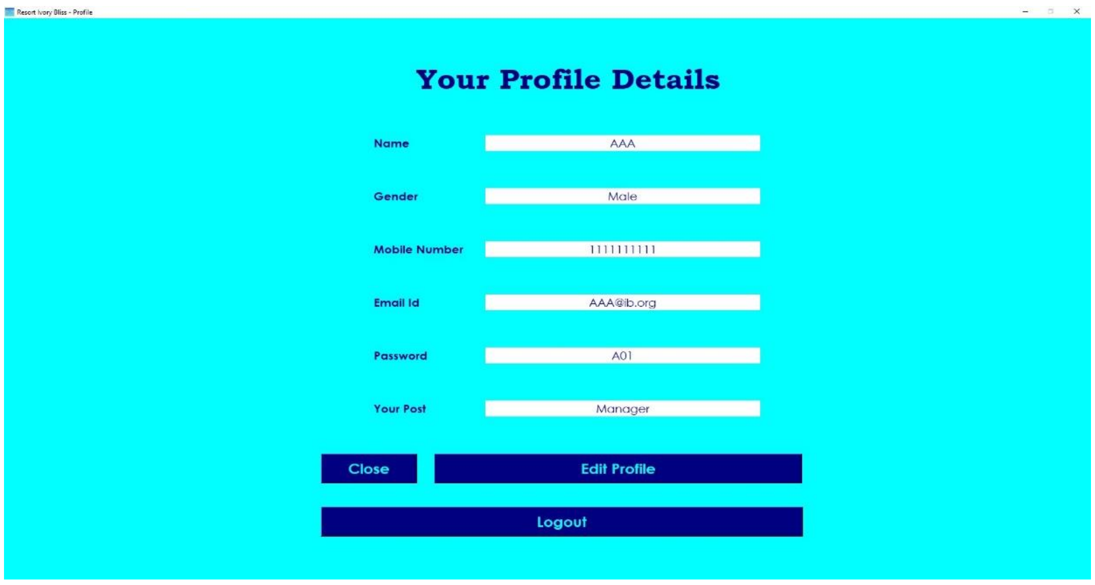
1. Login Page :



2. Main Menu (Manager) Page :



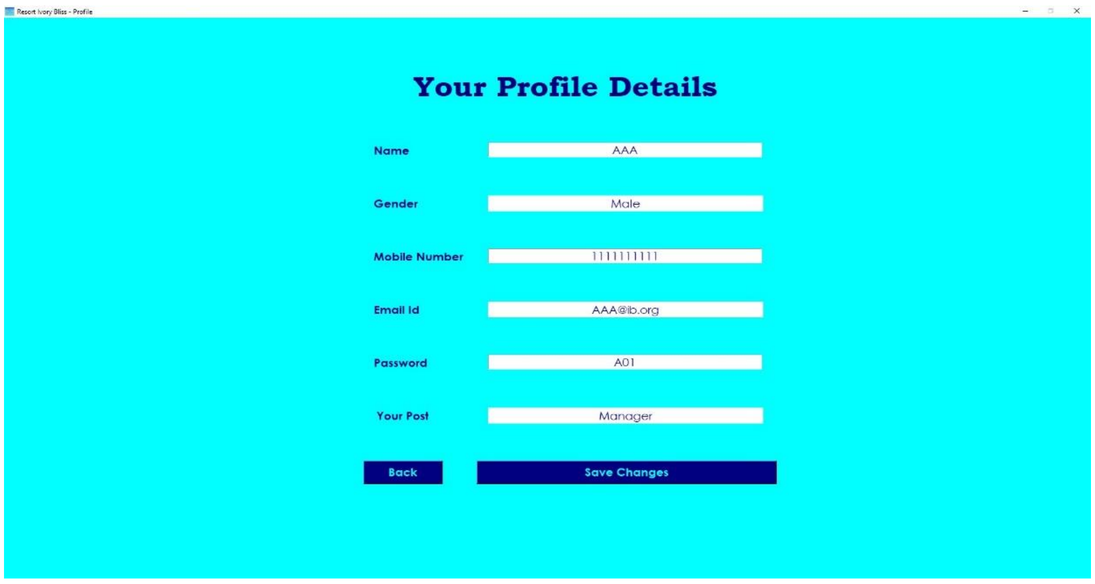
3. Profile Page :



Your Profile Details

Name	AAA
Gender	Male
Mobile Number	1111111111
Email Id	AAA@ib.org
Password	A01
Your Post	Manager

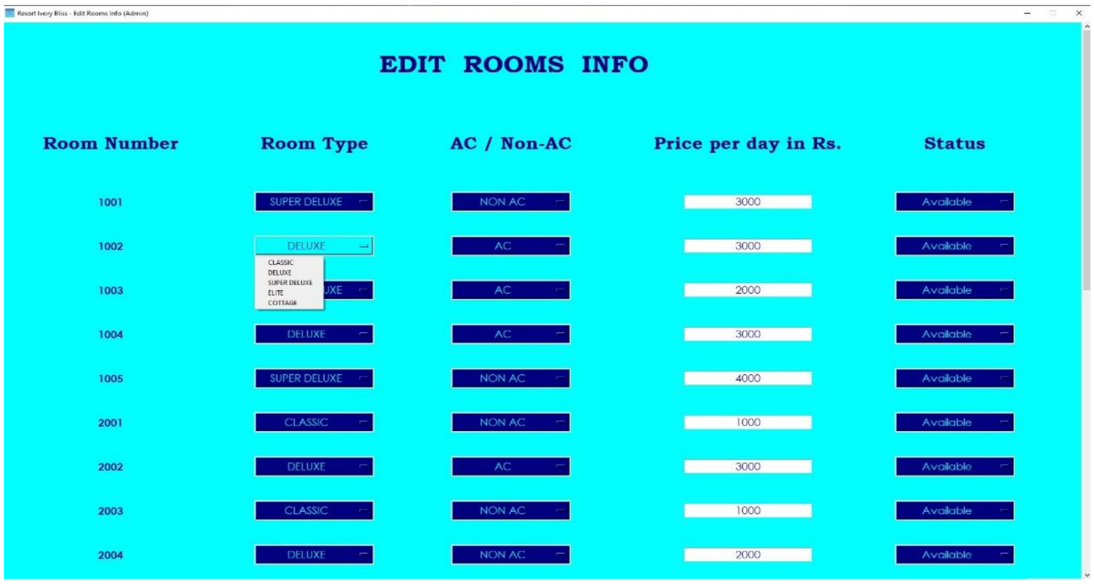
4. Edit Profile Page :



Your Profile Details

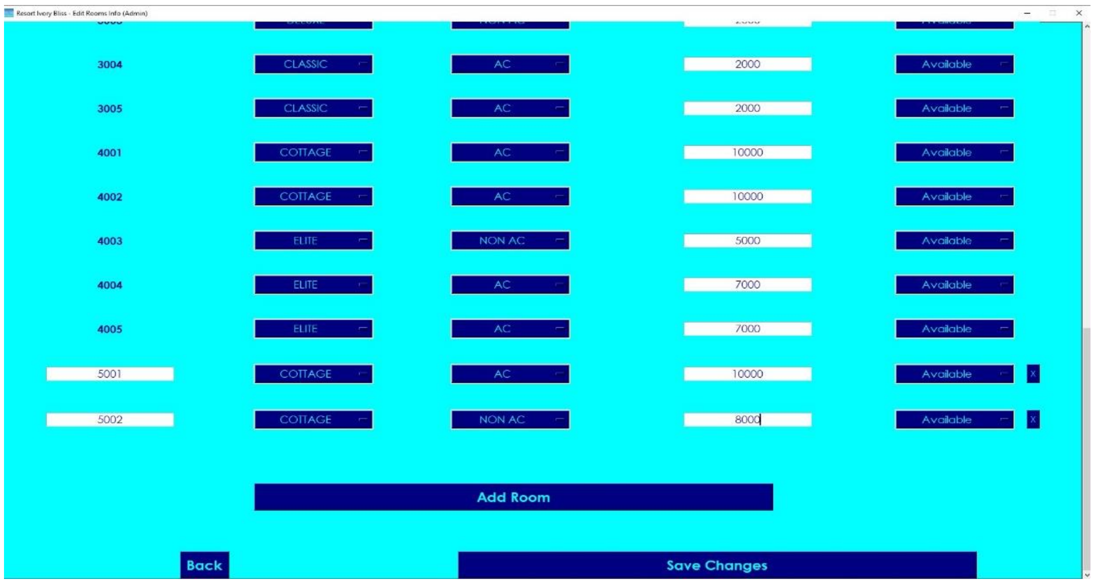
Name	AAA
Gender	Male
Mobile Number	1111111111
Email Id	AAA@ib.org
Password	A01
Your Post	Manager

5. Edit Rooms' Details :

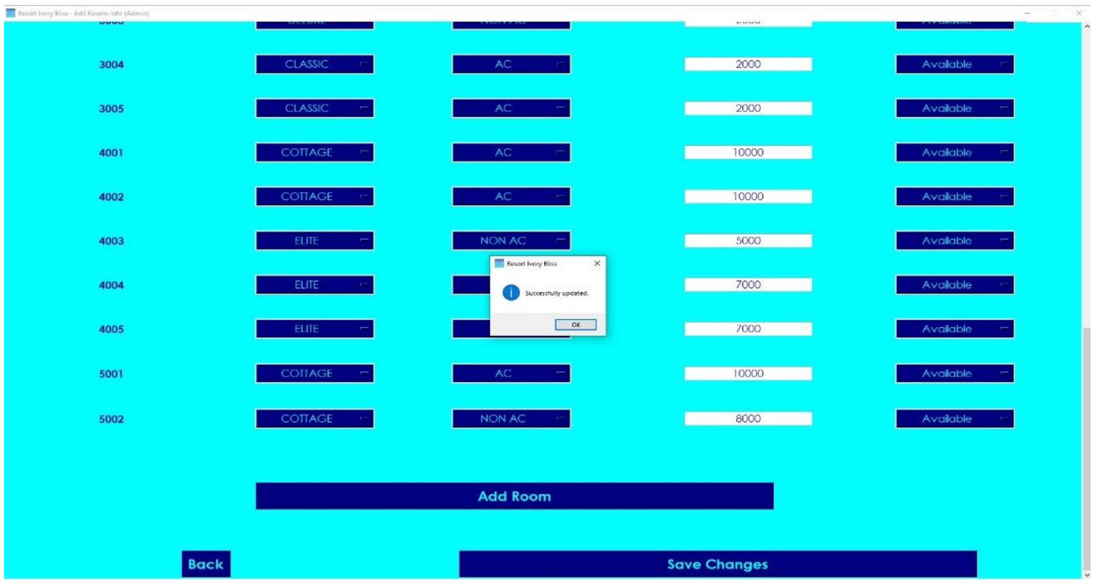

Resort Ivory Bliss - Edit Rooms Info (Admin)

EDIT ROOMS INFO

Room Number	Room Type	AC / Non-AC	Price per day in Rs.	Status
1001	SUPER DELUXE	NON AC	3000	Available
1002	DELUXE	AC	3000	Available
1003	UXE	AC	2000	Available
1004	DELUXE	AC	3000	Available
1005	SUPER DELUXE	NON AC	4000	Available
2001	CLASSIC	NON AC	1000	Available
2002	DELUXE	AC	3000	Available
2003	CLASSIC	NON AC	1000	Available
2004	DELUXE	NON AC	2000	Available


Resort Ivory Bliss - Edit Rooms Info (Admin)

3004	CLASSIC	AC	2000	Available	
3005	CLASSIC	AC	2000	Available	
4001	COTTAGE	AC	10000	Available	
4002	COTTAGE	AC	10000	Available	
4003	ELITE	NON AC	5000	Available	
4004	ELITE	AC	7000	Available	
4005	ELITE	AC	7000	Available	
5001	COTTAGE	AC	10000	Available	x
5002	COTTAGE	NON AC	8000	Available	x

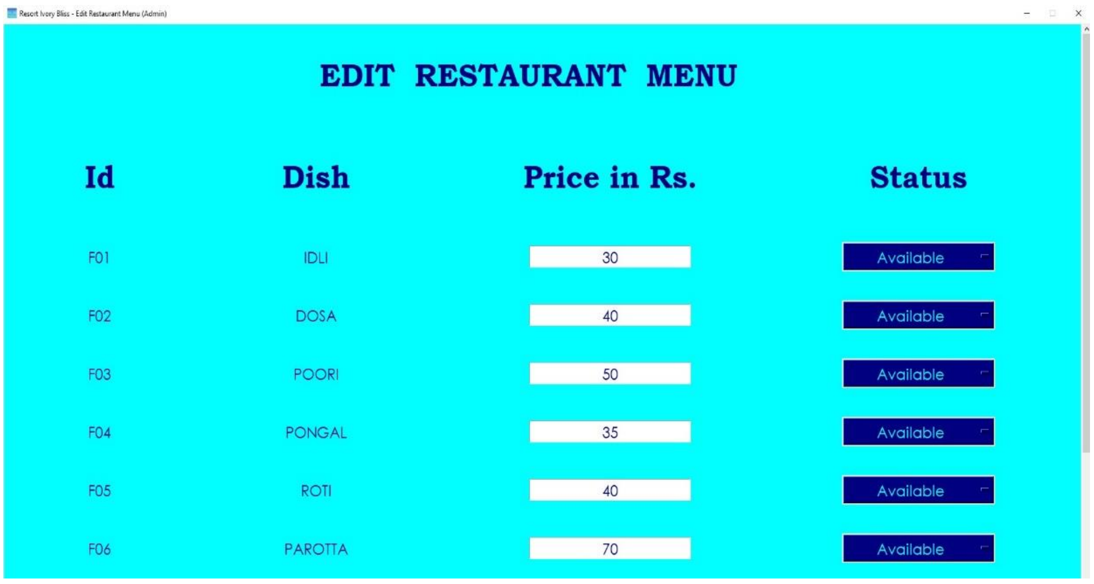

Resort Ivory Bliss - Edit Rooms Info (Admin)

3004	CLASSIC	AC	2000	Available
3005	CLASSIC	AC	2000	Available
4001	COTTAGE	AC	10000	Available
4002	COTTAGE	AC	10000	Available
4003	ELITE	NON AC	5000	Available
4004	ELITE		7000	Available
4005	ELITE		7000	Available
5001	COTTAGE	AC	10000	Available
5002	COTTAGE	NON AC	8000	Available

Resort Ivory Bliss

Successfully updated.

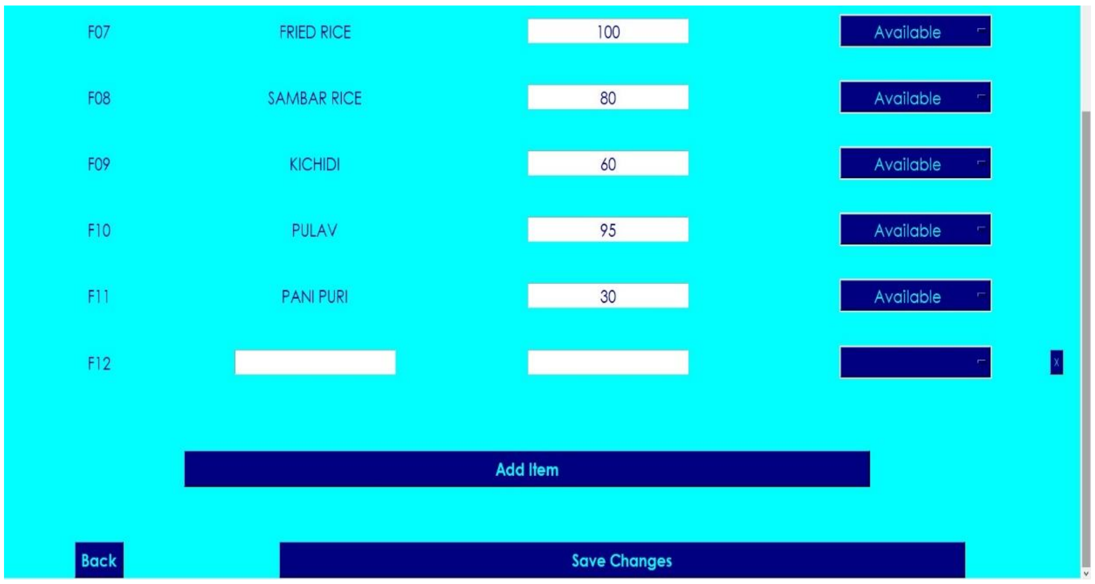
6. Edit Restaurant Menu :



Resort Ivory Bliss - Edit Restaurant Menu (Admin)

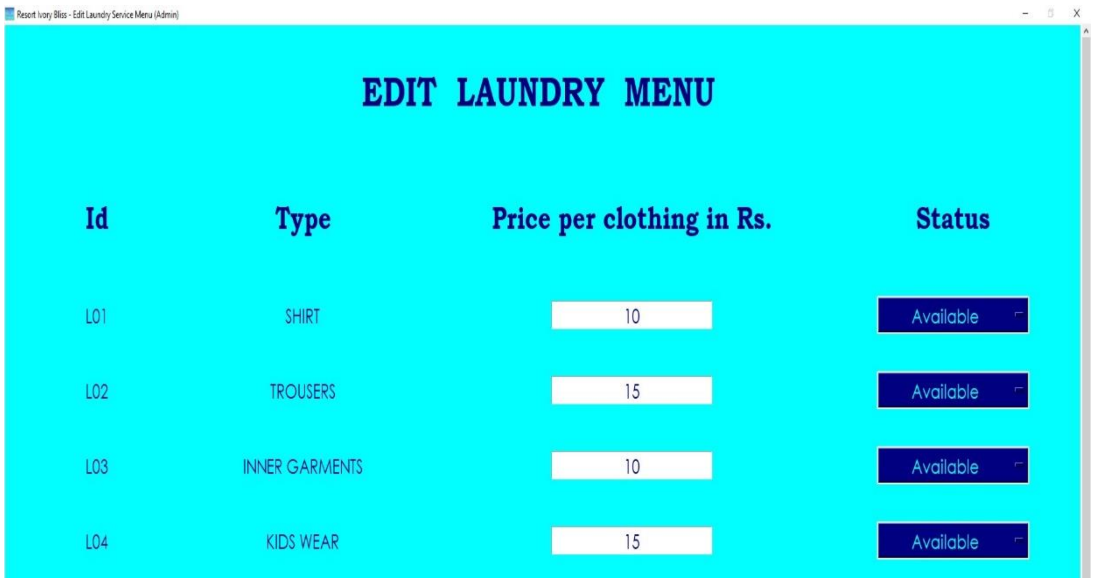
EDIT RESTAURANT MENU

Id	Dish	Price in Rs.	Status
F01	IDLI	30	Available
F02	DOSA	40	Available
F03	POORI	50	Available
F04	PONGAL	35	Available
F05	ROTI	40	Available
F06	PAROTTA	70	Available



F07	FRIED RICE	100	Available
F08	SAMBAR RICE	80	Available
F09	KICHIDI	60	Available
F10	PULAV	95	Available
F11	PANI PURI	30	Available
F12			

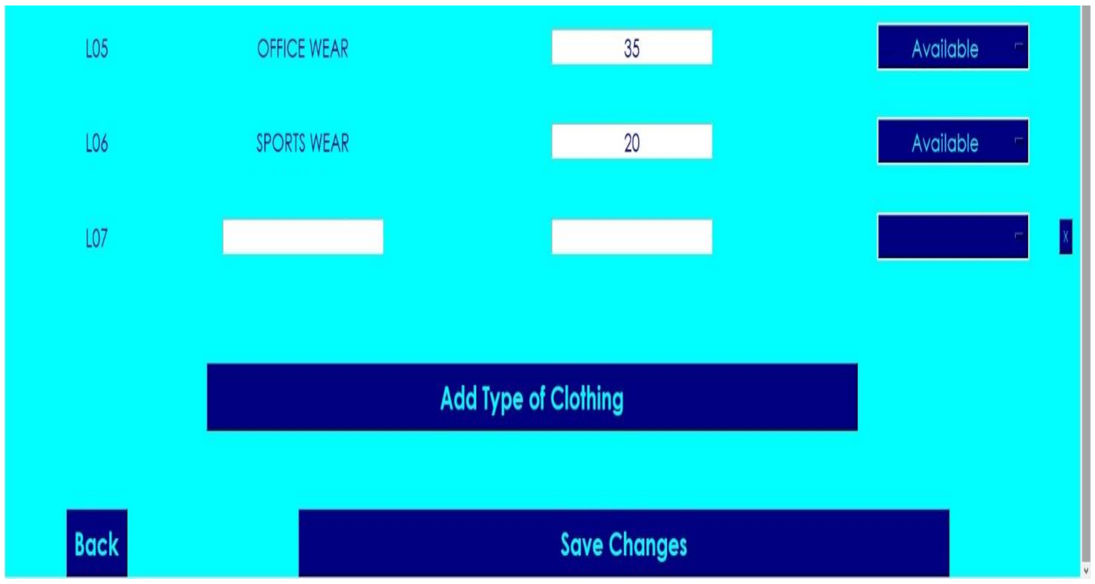
7. Edit Laundry Menu :



Resort Ivory Bliss - Edit Laundry Service Menu (Admin)

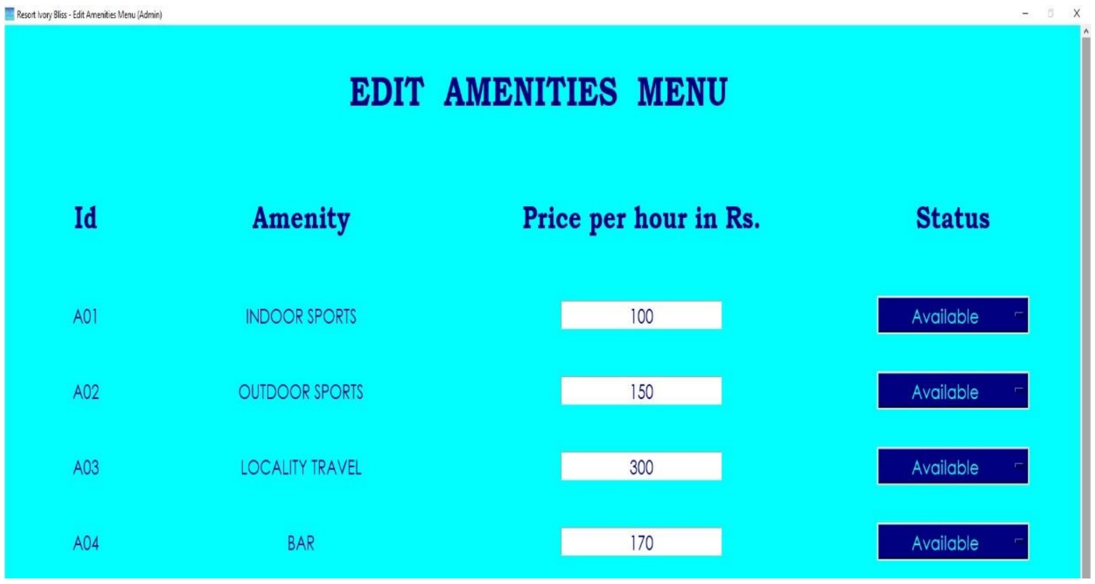
EDIT LAUNDRY MENU

Id	Type	Price per clothing in Rs.	Status
L01	SHIRT	10	Available
L02	TROUSERS	15	Available
L03	INNER GARMENTS	10	Available
L04	KIDS WEAR	15	Available



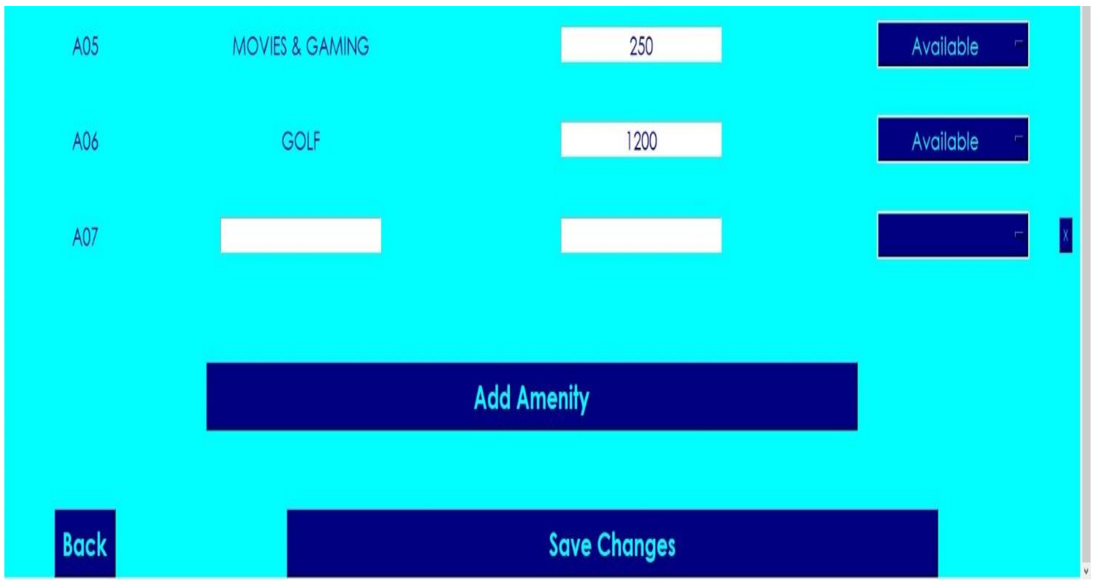
L05	OFFICE WEAR	35	Available
L06	SPORTS WEAR	20	Available
L07			

8. Edit Amenities Menu :


Resort Ivory Bliss - Edit Amenities Menu (Admin)

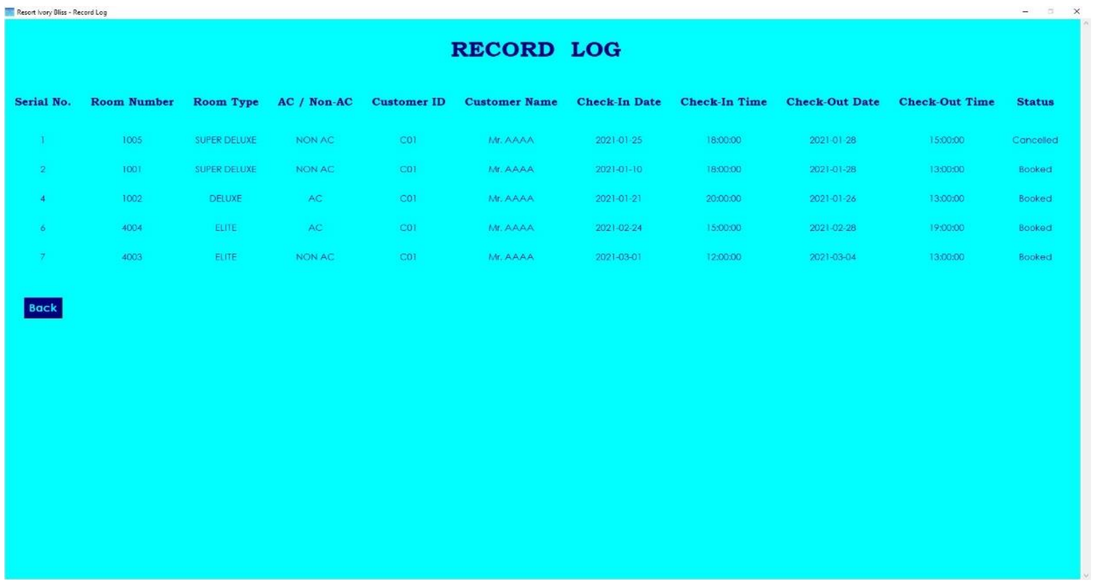
EDIT AMENITIES MENU

Id	Amenity	Price per hour in Rs.	Status
A01	INDOOR SPORTS	100	Available
A02	OUTDOOR SPORTS	150	Available
A03	LOCALITY TRAVEL	300	Available
A04	BAR	170	Available



A05	MOVIES & GAMING	250	Available
A06	GOLF	1200	Available
A07			

9. Record Log :



RECORD LOG

Serial No.	Room Number	Room Type	AC / Non-AC	Customer ID	Customer Name	Check-In Date	Check-In Time	Check-Out Date	Check-Out Time	Status
1	1005	SUPER DELUXE	NON AC	C01	Mr. AAAA	2021-01-25	18:00:00	2021-01-28	15:00:00	Cancelled
2	1001	SUPER DELUXE	NON AC	C01	Mr. AAAA	2021-01-10	18:00:00	2021-01-28	13:00:00	Booked
4	1002	DELUXE	AC	C01	Mr. AAAA	2021-01-21	20:00:00	2021-01-26	13:00:00	Booked
6	4004	ELITE	AC	C01	Mr. AAAA	2021-02-24	15:00:00	2021-02-28	19:00:00	Booked
7	4003	ELITE	NON AC	C01	Mr. AAAA	2021-03-01	12:00:00	2021-03-04	13:00:00	Booked

Limitations :

- The application is not connected to any payment APIs.
- The design of the application (i.e., the placement of widgets) is done with respect to any screen which maintains the same aspect ration as 1920x1080 and running of application in any other aspect ratio may result in a messed up UI.

References :

- Computer Science with python – Sumita Arora
- YouTube – Codemy.com

- www.tutorialspoint.com

- www.geeksforgeeks.com

Conclusion :

The project titled **Resort Management System** done by **Nithin.R.C.Mouli** for the academic year 2020 - 2021, has been completed and compiled, tested and executed successfully.