FINAL_PROJECTS\Program Files\index.py

```python
import subprocess
import smtplib
import os
import sys

def install(package):
    subprocess.check_call([sys.executable, "-m", "pip", "install", package])

required_packages = ["customtkinter",
                     "matplotlib",
                     "pillow",
                     "pymysql",
                     "colorama",
                     "tkcalendar"
                    ]

for package in required_packages:
    try:
        __import__(package)
        print(f"{package} Succesfully installed.")
    except ImportError:
        install(package)

# Created Modules
import Ticket_Code_Gen as TCG
from Usable_screen import ScreenGeometry as SG
from flights import major_airports
import Global_Config as GC
from Crypter import crypt

# Global Modules

from customtkinter import *
import pymysql
from tkcalendar import Calendar
import tkinter as tk
from pathlib import Path
from tkcalendar import Calendar
from datetime import datetime
from tkinter import filedialog

m_r_width, m_r_height = SG().GetUsableScreenSize()[0], SG().GetUsableScreenSize()[1]

root = CTk()
root.title("http:www.HADAirlineManagementSystem.com/")
set_appearance_mode("Dark")
GC.centreScreen(root,root, m_r_width, m_r_height)
root.state("zoomed")
```

```
49 root.minsize(m_r_width, m_r_height)
50 root.geometry(f"{m_r_width}x{m_r_height}")
51
52 con = pymysql.connect(
53     host = "localhost",
54     user = "root",
55     passwd  = "*password*11",
56                     )
57
58 cur = con.cursor()
59 #---------GLOBAL VARIABLES --------
60 _isSignedIn = False
61 is_flight_details_obtained = False
62 User = "" #vs_hari_dhejus
63 isAdmin = False ############
64 prev_page = 0
65 glb_clr_1 = "blue"
66 glb_clr_2 = "green"
67 BASE_DIR = Path(__file__).resolve().parent.parent
68 glb_clr_3 = "yellow"
69 destroy_after = None
70
71 global  PG_Get_Flight_Details
72
73 def DB_INIT_():
74     try :
75         cur.execute("CREATE DATABASE IF NOT EXISTS `airwaysms2_0` /*!40100 DEFAULT CHARACTER
   SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;")
76
77         cur.execute("""CREATE TABLE IF NOT EXISTS `user_details` (
78                         `UID` int NOT NULL AUTO_INCREMENT,
79                         `UF_name` varchar(100) DEFAULT NULL,
80                         `UL_name` varchar(100) DEFAULT NULL,
81                         `U_name` varchar(100) NOT NULL,
82                         `U_Gmail` varchar(100) NOT NULL,
83                         `U_phno` varchar(12) DEFAULT NULL,
84                         `U_password` varchar(100) NOT NULL,
85                         `U_dob` date DEFAULT NULL,
86                         `U_AGE` int DEFAULT NULL,
87                         `U_gender` varchar(5) DEFAULT NULL,
88                         `U_isActive` tinyint DEFAULT '1',
89                         PRIMARY KEY (`UID`),
90                         UNIQUE KEY `U_name_UNIQUE` (`U_name`)
91                         ) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4
   COLLATE=utf8mb4_0900_ai_ci;""")
92
93         cur.execute("""CREATE TABLE `flights` (
94                         `F_ID` int NOT NULL AUTO_INCREMENT,
95                         `F_Departure` varchar(100) DEFAULT NULL,
96                         `F_Arrival` varchar(100) DEFAULT NULL,
```

```
97                          `F_Airline` varchar(45) NOT NULL,
98                          `F_price` int DEFAULT NULL,
99                          PRIMARY KEY (`F_ID`)
100                         ) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_0900_ai_ci;
101                     """)
102
103         cur.execute("""CREATE TABLE IF NOT EXISTS `booking` (
104                         `BID` varchar(100) NOT NULL,
105                         `BU_NAME` varchar(100) DEFAULT NULL,
106                         `B_FLIGHT` int DEFAULT NULL,
107                         `IS_ACTIVE` int DEFAULT NULL,
108                         PRIMARY KEY (`BID`),
109                         KEY `B_FORIEGN_KEY_idx` (`B_FLIGHT`),
110                         CONSTRAINT `B_FORIEGN_KEY` FOREIGN KEY (`B_FLIGHT`) REFERENCES
    `flights` (`F_ID`)
111                         ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_0900_ai_ci;""")
112
113         cur.execute("""CREATE TABLE `payment` (
114                         `PID` int NOT NULL AUTO_INCREMENT,
115                         `P_UID` int DEFAULT NULL,
116                         `AMOUNT` int DEFAULT NULL,
117                         `P_STATUS` int DEFAULT NULL,
118                         `P_ACC_NUM` int DEFAULT NULL,
119                         `P_UPI_NUM` int DEFAULT NULL,
120                         `P_METHOD` varchar(45) DEFAULT NULL,
121                         PRIMARY KEY (`PID`),
122                         KEY `P_UID_idx` (`P_UID`),
123                         CONSTRAINT `P_UID` FOREIGN KEY (`P_UID`) REFERENCES `user_details`
    (`UID`)
124                         ) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8mb4
    COLLATE=utf8mb4_0900_ai_ci;
125                         """)
126
127     except Exception as e:
128         pass
129
130     finally :
131         print("Sucessfully Initialized Database.")
132         cur.execute("USE airwaysms2_0 ;")
133 DB_INIT_()
134 #--------------------- GLOBAL FUNCTIONS --------------------------------
135
136 def createRadioButton (_frame ,_text : str , _value, _variable, _command,  _xpos : int, _ypos
    : int):
137     tmpRdBtn = CTkRadioButton(_frame, text = _text , value = _value, variable =
    _variable,width = 75, command = lambda :(_command()) )
138     tmpRdBtn.place(x =_xpos, y = _ypos)
139     return tmpRdBtn
140
```

```
141 def errorLabeling(_Master, _text : str, _font = ("Bradley Hand ITC" , 18, "italic", "bold"),
    _textcolor = "red", _x = 10, _y = 10, _cooldowntime = 3000):
142     error_label = CTkLabel(_Master, text = _text, font = _font, text_color = _textcolor)
143     error_label.place(x = _x, y = _y)
144     def refresh():
145         error_label.destroy()
146     error_label.after(_cooldowntime, refresh)
147 #-----------------------------------------------------
148 
149 Main_fame = CTkFrame(root, width = m_r_width, height= m_r_height-24, border_width=2,
    border_color= "#007acc", fg_color="transparent")
150 Main_fame.place(x = 0, y = 0)
151 
152 global booking
153 def booking ():
154     Ticket_Code = TCG.Gen_Code()
155     cur.execute("SELECT * FROM flights WHERE F_ID ='%s'", Flight_ID)
156     cur.execute("SELECT COUNT(*) FROM booking")
157     F_id = cur.fetchone()[0]
158     global User
159     cur.execute("INSERT INTO booking (BID, BU_NAME, B_FLIGHT) VALUES (%s, %s, %s)",
    (Ticket_Code, User, Flight_ID))
160     con.commit()
161     file_path = filedialog.asksaveasfilename(defaultextension = ".txt",
162                                              filetypes=[("Text Files", "*.txt"),
163                                               ("All Files", "*.*")
164                                               ],
165                                              initialfile= "SELECT WHERE TO SAVE THE
    TICKET.txt")
166     temp_qry = f"SELECT F_Departure, F_Arrival, F_Airline, F_price FROM flights WHERE
    flights.F_ID = %s ;"
167     cur.execute(temp_qry, (Flight_ID,))
168     result = cur.fetchone()
169     if file_path:
170         with open(file_path, "w")as f:
171             f.write(f"Ticket Id : {Ticket_Code}, User : {User}, Departure : {result[0]},
    Arrival : {result[1]}\
172 Airline : {result[2]}, Price : {result[3]}")
173 
174     errorLabeling(form_frm, "Sucessfully Booked", _textcolor = "green", _x = 120, _y = 170)
175 
176 global PG_Payment
177 def PG_Payment():
178     root.title ("http:www.HADAirlineManagementSystem.com/Payment")
179     for i in Main_fame.winfo_children():
180         i.destroy()
181 
182     form_frm_width = 400
183     form_frm_height = 210
184     global form_frm
185     form_frm = CTkFrame(Main_fame, width=form_frm_width, height=form_frm_height)
```

```
    form_frm.place(x = (m_r_width/(2))-(form_frm_width/2), y = (m_r_height/(2)-(form_frm_height/2)))

    def go_back():
        PG_search_flight_()

    dummy_back_btn = CTkButton(Main_fame, text="Back", command = go_back)
    dummy_back_btn.place(x =10, y = 10)

    Payment_Method_label = CTkLabel(form_frm, text= "Select Payment Method")
    Payment_Method_label.place(x = 25, y = 10)

    CBF_width = 290
    CBF_height = 28
    Centre_btn_frame = CTkFrame(form_frm, height=CBF_height, width= CBF_width, fg_color="transparent")
    Centre_btn_frame.place(x = (form_frm_width/(2))-(CBF_width/2),
                                  y = (form_frm_height/(2)-(CBF_height/2)) )

    def on_UPI_btn_click():
        root.title ("http:www.HADAirlineManagementSystem.com/Payment/UPI")
        for i in form_frm.winfo_children():
            i.destroy()

        def go_back():
            PG_Payment()

        dummy_back_btn = CTkButton(Main_fame, text="Back", command = go_back)
        dummy_back_btn.place(x =10, y = 10)
        Temp_Entry_Width = 350

        UPI_Method_label = CTkLabel(form_frm, text= "UPI -")
        UPI_Method_label.place(x = 25, y = 10)

        UPI_Number_Entry = CTkEntry(form_frm, placeholder_text= "UPI Number",width= Temp_Entry_Width)
        UPI_Number_Entry.place(x =25, y = 50)

        cur.execute(f"SELECT F_Price FROM flights WHERE F_ID = '{Flight_ID}'")
        Amount = str(cur.fetchone()[0])

        Amount_LBL = CTkLabel(form_frm,text= f"Amount : {Amount}",width= Temp_Entry_Width)
        Amount_LBL.place(x =25, y = 90)

        def _on_UPI_pay_btn_click():
            UPI_Number = UPI_Number_Entry.get()
            cur.execute(f"SELECT F_Price FROM flights WHERE F_ID = '{Flight_ID}'")
            Amount = str(cur.fetchone()[0])

            if UPI_Number == "" or Amount == "":
                errorLabeling(form_frm, "Feilds Can Not Be Empty", _x = 90, _y = 170)
```

```
234
235             elif UPI_Number.isdigit() == False or Amount.isdigit() == False:
236                 is_flight_details_obtained
237             else:
238                 Amount = int(Amount)
239                 UPI_Number = int(UPI_Number)
240                 tempqry = "SELECT COUNT(*) FROM payment"
241                 cur.execute(tempqry)
242                 p_id = int(cur.fetchone()[0])+1
243                 global User
244                 tempqry = f"SELECT UID FROM user_details WHERE U_name = '{User}'"
245                 cur.execute(tempqry)
246                 Uid = int(cur.fetchone()[0])
247                 tempqry = "INSERT INTO payment (PID, AMOUNT, P_STATUS, P_UPI_NUM, P_METHOD)
    VALUES (%s, %s, %s, %s, %s)"
248                 cur.execute(tempqry, (p_id,Amount,400,UPI_Number, "UPI"))
249                 errorLabeling(form_frm, "Payment Sucessful", _textcolor = "green", _x = 110,
    _y = 170)
250                 def delayed_lbl():
251                     errorLabeling(form_frm, "Booking Ticket ...", _textcolor = "green", _x =
    1100, _y = 170)
252                     def delayed():
253                         booking()
254                         PG_Get_Flight_Details()
255                     root.after(3000, delayed)
256                 con.commit()
257                 root.after(3000, delayed_lbl)
258
259         Pay_btn = CTkButton(form_frm, width= Temp_Entry_Width, text = "Pay", corner_radius=
    100, command= _on_UPI_pay_btn_click)
260         Pay_btn.place(x = 25, y = 130)
261
262     UPI_btn = CTkButton(Centre_btn_frame, text = "UPI", command= on_UPI_btn_click)
263     UPI_btn.place(x = 0, y = 0)
264
265     def on_NET_btn_click():
266         root.title ("http:www.HADAirlineManagementSystem.com/Payment/UPI")
267         for i in form_frm.winfo_children():
268             i.destroy()
269
270         def go_back():
271             PG_Payment()
272
273         dummy_back_btn = CTkButton(Main_fame, text="Back", command = go_back)
274         dummy_back_btn.place(x =10, y = 10)
275
276         Temp_Entry_Width = 350
277
278         NETB_Method_label = CTkLabel(form_frm, text= "NET BANKING -")
279         NETB_Method_label.place(x = 25, y = 10)
280
```

```
281         NETB_Number_Entry = CTkEntry(form_frm, placeholder_text= "Account Number",width=
    Temp_Entry_Width)
282         NETB_Number_Entry.place(x =25, y = 50)
283 
284         cur.execute(f"SELECT F_Price FROM flights WHERE F_ID = '{Flight_ID}'")
285         Amount = str(cur.fetchone()[0])
286 
287         Amount_LBL = CTkLabel(form_frm,text= f"Amount : {Amount}",width= Temp_Entry_Width)
288         Amount_LBL.place(x =25, y = 90)
289 
290         def _on_NETB_pay_btn_click():
291             ACC_Number = NETB_Number_Entry.get()
292             cur.execute(f"SELECT F_Price FROM flights WHERE F_ID = '{Flight_ID}'")
293             Amount = str(cur.fetchone()[0])
294 
295             if ACC_Number == "" or Amount == "":
296                 errorLabeling(form_frm, "Feilds Can Not Be Empty", _x = 90, _y = 170)
297 
298             elif ACC_Number.isdigit() == False or Amount.isdigit() == False:
299                 is_flight_details_obtained
300             else:
301                 Amount = int(Amount)
302                 ACC_Number = int(ACC_Number)
303                 tempqry = "SELECT COUNT(*) FROM payment"
304                 cur.execute(tempqry)
305                 p_id = int(cur.fetchone()[0])+1
306                 global User
307                 tempqry = f"SELECT UID FROM user_details WHERE U_name = '{User}'"
308                 cur.execute(tempqry)
309                 Uid = int(cur.fetchone()[0])
310                 tempqry = "INSERT INTO payment (PID, AMOUNT, P_STATUS, P_UPI_NUM, P_METHOD)
    VALUES (%s, %s, %s, %s, %s)"
311                 cur.execute(tempqry, (p_id,Amount,400,ACC_Number, "NET"))
312                 con.commit()
313                 errorLabeling(form_frm, "Payment Sucessful", _textcolor = "green", _x = 110,
    _y = 170)
314                 def delayed_lbl():
315                     errorLabeling(form_frm, "Booking Ticket ...", _textcolor = "green", _x =
    1100, _y = 170)
316                     def delayed():
317                         booking()
318                         PG_Get_Flight_Details()
319                     root.after(3000, delayed)
320                 con.commit()
321                 root.after(3000, delayed_lbl)
322 
323         Pay_btn = CTkButton(form_frm, width= Temp_Entry_Width, text = "Pay", corner_radius=
    100, command= _on_NETB_pay_btn_click)
324         Pay_btn.place(x = 25, y = 130)
325 
```

```
    NetBanking_btn = CTkButton(Centre_btn_frame, text = "Net Banking", command=
on_NET_btn_click)
    NetBanking_btn.place(x = 150, y = 0)
    Main_frm_Authentication_Btns()

#=> --------Sign Up --------------------
global PG_Sign_Up
def PG_Sign_Up() :
    root.title ("http:www.HADAirlineManagementSystem.com/SignUp")
    for i in Main_fame.winfo_children():
        i.destroy()

    form_frm_width = 400
    form_frm_height = 600
    form_frm = CTkFrame(Main_fame, width=form_frm_width, height=form_frm_height)
    form_frm.place(x = (m_r_width/(2))-(form_frm_width/2),
                                  y = (m_r_height/(2)-(form_frm_height/2))
                                  )

    def go_back():
        for i in Main_fame.winfo_children():
            i.destroy()
        PG_Sign_in()

    def NullCheck():
        global DOB_selected_date, cal
        F_name = F_name_Entry.get()
        L_name = L_name_Entry.get()
        U_name = U_name_Entry.get()
        _Gender = Gender.get()
        if "cal" in globals():
            dob = cal.get_date()
            dob_dt = datetime.strptime(dob, "%Y-%m-%d")
            today = datetime.today()
            Age = today.year - dob_dt.year
            if (today.month, today.day) > (dob_dt.month, dob_dt.day):
                Age +=1
        else :
            dob = ""
        Gmail = gmail_Entry.get()
        _pass = crypt(pass_Entry.get()).encrypt()
        _re_pass = crypt(re_pass_Entry.get()).encrypt()
        phonenumber = phonnumber_Entry.get()
        tmp_qry =f"SELECT U_name FROM user_details WHERE U_name= '{U_name}'"
        cur.execute(tmp_qry)
        row = cur.fetchone()
        if F_name == "" or L_name == "" or U_name == "" or dob == "" or  Gmail == "" or _pass
== "" or _re_pass == "" or phonenumber == "" or _Gender == "other" :
            errorLabeling(form_frm, "Feilds Cannot Be Null", _x = 110, _y = 410)

```

```
374         if _pass != _re_pass:
375             errorLabeling(form_frm, "Passwords Don't Match", _x = 110, _y = 410)
376 
377         elif row :
378             errorLabeling(form_frm, "Username Already Exist", _x = 110, _y = 410)
379 
380         else:
381             cur.execute("SELECT Count(*) FROM user_details")
382             ans = cur.fetchone()
383 
384             new_uid = ans[0] + 1
385 
386             tmp_qry = """
387                 INSERT INTO user_details (UID, UF_name, UL_name, U_name, U_Gmail, U_phno,
    U_password, U_dob, U_gender, U_AGE)
388                 VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
389             """
390             cur.execute(tmp_qry, (new_uid, F_name, L_name, U_name, Gmail, phonenumber, _pass,
    dob, _Gender, Age))
391             con.commit()
392 
393             cur.execute("SELECT Count(*) FROM user_details")
394             ans2 = cur.fetchone()
395 
396             con.commit()
397             if ans2 > ans :
398                 errorLabeling(form_frm, "Succesfully Added", _textcolor = "green", _x = 110,
    _y = 410)
399                 def _fun():
400                     global _isSignedIn, User
401                     _isSignedIn = True
402                     User = U_name
403                     PG_Get_Flight_Details()
404                 errorLabeling(form_frm, "Succesfully Added", _textcolor = "green", _x = 110,
    _y = 410)
405                 fun_lbl = CTkLabel(form_frm)
406                 fun_lbl.after(4000, _fun)
407             else:
408                 errorLabeling(form_frm, "Error Occured While Inserting Try Restarting", _x =
    5, _y = 410)
409 
410     dummy_back_btn = CTkButton(Main_fame,text="Back", command = go_back)
411     dummy_back_btn.place(x =10, y = 10)
412 
413     F_name_Entry = CTkEntry(form_frm, width = 350, placeholder_text="First Name")
414     F_name_Entry.place(x = 25, y = 10)
415 
416     L_name_Entry = CTkEntry(form_frm, width = 350, placeholder_text="Last Name")
417     L_name_Entry.place(x = 25, y = 50)
418 
419     def Func_radio_btn():
```

```
420         global _Gender
421         _Gender = Gender.get()
422     global Gender
423     Gender = StringVar(value = "other")
424 
425     rd_btn_y_pos = 90
426 
427     male_radio_btn = createRadioButton(form_frm, "Male","M",Gender,Func_radio_btn,25,
    rd_btn_y_pos)
428 
429     female_radio_btn = createRadioButton(form_frm,"Female", "F", Gender,Func_radio_btn,
    25+130, rd_btn_y_pos)
430 
431     other_radio_btn = createRadioButton(form_frm,"Other", "O", Gender,Func_radio_btn, 25+130 +
    130, rd_btn_y_pos)
432     global cal
433 
434     def DOB_open_date_picker():
435 
436         top = CTkToplevel(form_frm)
437         top.title("Select a Date")
438         top.attributes("-topmost", True)
439         global cal
440         cal = Calendar(top, selectmode='day', date_pattern = "yyyy-mm-dd")
441         cal.pack(pady=10)
442         def select_date():
443             global DOB_selected_date
444             DOB_selected_date = cal.get_date()
445             DOB_Date_label.configure(text=f"Selected Date : {DOB_selected_date}")
446             top.destroy()
447 
448         select_button = CTkButton(top, text="Select Date", command=select_date)
449         select_button.pack(pady=10)
450 
451     DOB_Date_Btn = CTkButton(form_frm, text="Date Of Birth", command=DOB_open_date_picker,
    corner_radius=100)
452     DOB_Date_Btn.place(x=25, y =rd_btn_y_pos+40)
453 
454     DOB_Date_label = CTkLabel(form_frm, text= "Select Date")
455     DOB_Date_label.place(x = 25+170, y = 90+40)
456 
457     U_name_Entry = CTkEntry(form_frm, width = 350, placeholder_text="Username")
458     U_name_Entry.place(x = 25, y = 130+40)
459 
460 
461     gmail_Entry = CTkEntry(form_frm, width = 350, placeholder_text="Gmail")
462     gmail_Entry.place(x = 25, y = 170+40)
463 
464 
465     def Show_pass():
466         if pass_Entry.cget('show') == '*':
```

```
            pass_Entry.configure(show='')
            show_btn.configure(text=" Hide ")
        else:
            pass_Entry.configure(show='*')
            show_btn.configure(text="Show")

    def Re_Show_pass():
        if re_pass_Entry.cget('show') == '*':
            re_pass_Entry.configure(show='')
            re_show_btn.configure(text=" Hide ")
        else:
            re_pass_Entry.configure(show='*')
            re_show_btn.configure(text="Show")

    pass_Entry = CTkEntry(form_frm, width = 350, placeholder_text="Password", show = "*")
    show_btn = CTkButton(pass_Entry, width = 22, height=28,
text="Show",border_color="#565b5e",border_width=2, fg_color="transparent", command=Show_pass)
    show_btn.place(x = 304, y=0)
    pass_Entry.place(x = 25, y = 170+80)

    re_pass_Entry = CTkEntry(form_frm, width = 350, placeholder_text="Re-Password", show =
"*")
    re_show_btn = CTkButton(re_pass_Entry, width = 22, height=28,
text="Show",border_color="#565b5e",border_width=2, fg_color="transparent",
command=Re_Show_pass)
    re_show_btn.place(x = 304, y=0)
    re_pass_Entry.place(x = 25, y = 210+80)

    phonnumber_Entry = CTkEntry(form_frm, 350, placeholder_text="Phone Number")
    phonnumber_Entry.place(x = 25, y =250+80)

    Create_acc_btn = CTkButton(form_frm, width = 350, text="Create Account",
corner_radius=100, command = NullCheck)
    Create_acc_btn.place(x = 25, y = 290+80)

    tempxpos = 100
    tempypos = 410
    Dnt_hv_acc_lbl = CTkLabel(form_frm, text="Already have an account ? ")
    Dnt_hv_acc_lbl.place(x = tempxpos, y = tempypos)
    Sign_up_lbl = CTkLabel(form_frm, text="Sign In", font = ("Arial" , 12, "italic",
"underline"))
    Sign_up_lbl.place(x = tempxpos + 150, y = tempypos)
#=>3------Sign In Page -------------------------------------
global PG_Sign_in
def PG_Sign_in():
    root.title ("http:www.HADAirlineManagementSystem.com/Sign_In")
    for i in Main_fame.winfo_children():
        i.destroy()

    form_frm_width = 400
    form_frm_height = 340
```

```
    form_frm = CTkFrame(Main_fame, width=form_frm_width, height=form_frm_height)
    form_frm.place(x = (m_r_width/(2))-(form_frm_width/2),
                                  y = (m_r_height/(2)-(form_frm_height/2))
                                  )

    def go_back():
        global _isSignedIn
        if _isSignedIn == True :
            for i in Main_fame.winfo_children():
                i.destroy()
            Main_frm_Authentication_Btns()
            PG_search_flight_()
        else :
            for i in Main_fame.winfo_children():
                i.destroy()
            Main_frm_Authentication_Btns()
            PG_Get_Flight_Details()

    dummy_back_btn = CTkButton(Main_fame,text="Back", command = go_back)
    dummy_back_btn.place(x =10, y = 10)

    login_lb = CTkLabel(form_frm, text = "LOGIN")
    login_lb.place(x = 10, y = 10)

    def Login_authentication(_username, _pass):
        global _isSignedIn, User
        _isSignedIn = True
        _username=  _username.get()
        _pass = crypt(_pass.get()).encrypt()
        if (_username == "" or _pass == "" ) :
            errorLabeling(form_frm, "Feilds Cannot Be Empty", _x = 90, _y = 150)
        else:
            temp_qry = f"SELECT U_name, U_Gmail, U_password, U_isAdmin FROM user_details WHERE
U_name = '{_username}' or U_Gmail = '{_username}' and U_password = '{_pass}'"
            cur.execute(temp_qry)
            qry_result = cur.fetchone()

            if qry_result == None :
                errorLabeling(form_frm, "Username Or Password Is Incorrect", _x = 50, _y =
150)

            else:
                global User, isAdmin
                User = qry_result[0]
                _isSignedIn = True
                isAdmin = qry_result[-1]
                if is_flight_details_obtained == True :
                    PG_Payment()
                else:
                    PG_Get_Flight_Details()
```

```
560
561     user_Entry = CTkEntry(form_frm, width = 350, placeholder_text= "Username/Gmail")
562     user_Entry.place(x = 25, y = 40)
563
564     def Show_pass():
565
566         if pass_Entry.cget('show') == '*':
567             pass_Entry.configure(show='')
568             show_btn.configure(text=" Hide ")
569         else:
570             pass_Entry.configure(show='*')
571             show_btn.configure(text="Show")
572
573     pass_Entry = CTkEntry(form_frm, width = 350, placeholder_text="Password", show = "*")
574     show_btn = CTkButton(pass_Entry, width = 22, height=28,
text="Show",border_color="#565b5e",border_width=2, fg_color="transparent", command=Show_pass)
575     show_btn.place(x = 304, y=0)
576     pass_Entry.place(x = 25, y = 80)
577
578     Login_btn = CTkButton(form_frm, text= "LOGIN", width= 350, corner_radius=100,
command=lambda :(Login_authentication(user_Entry, pass_Entry)))
579     Login_btn.place(x = 25, y = 120)
580
581     No_of_hyphen = 41
582     line_lbl = CTkLabel(form_frm, text = f"{'-'*No_of_hyphen} OR {'-'*No_of_hyphen}")
583     line_lbl.place(x = 25, y = 170)
584
585     def loginWithGoogle():
586         temp_TL = CTkToplevel(root)
587         temp_TL.title("http:www.HADAirlineManagementSystem.com/Error?loginWithGoogle")
588         temp_TL.attributes("-topmost", True)
589         temp_TL.geometry("400x300")
590         errorLabeling(temp_TL, _text="""
591 We are Extremly Sorry for the Inconvenience:
592 We at HAD are currently working to
593 bring in that feature.""",_textcolor = "#007acc", _cooldowntime = None)
594
595     Login_btn2 = CTkButton(form_frm, text= "LOGIN with Google", width= 350, corner_radius=100,
command=loginWithGoogle)
596     Login_btn2.place(x = 25, y = 220)
597
598     tempxpos = 100
599     tempypos = 270
600     Dnt_hv_acc_lbl = CTkLabel(form_frm, text="Don't have an account ? ")
601     Dnt_hv_acc_lbl.place(x = tempxpos, y = tempypos)
602     Sign_up_lbl = CTkLabel(form_frm, text="Sign Up", font = ("Arial" , 12, "italic",
"underline"))
603     Sign_up_lbl.place(x = tempxpos + 140, y = tempypos)
604
605
606     Sign_up_lbl.bind("<Button-1>", lambda event, : PG_Sign_Up())
```

```
607     Sign_up_lbl.bind("<Enter>", lambda event, lbl = Sign_up_lbl: lbl.configure(text_color =
    "#007acc"))
608     Sign_up_lbl.bind("<Leave>", lambda event, lbl = Sign_up_lbl: lbl.configure(text_color =
    "Light Gray"))
609 # =>2------Show Flights details -------------------------------------
610 global PG_search_flight_
611 def PG_search_flight_():
612     prev_page = 2
613     root.title ("http:www.HADAirlineManagementSystem.com/Search_flights")
614     if div_frame.winfo_exists():
615         for i in div_frame.winfo_children():
616             i.destroy()
617         div_frame.destroy()
618 
619     temp_frm_width = 900
620     temp_frm_height = 550
621     global fligt_search_result_frm
622     fligt_search_result_frm = CTkScrollableFrame (Main_fame, width=temp_frm_width, height =
    temp_frm_height, scrollbar_button_color= None)
623     fligt_search_result_frm.place(x = (m_r_width/(2))-(temp_frm_width/2),
624                                   y = (m_r_height/(2)-(temp_frm_height/2))
625                                   )
626 
627     PG_Heading = CTkLabel(fligt_search_result_frm, text = "AVAILABLE FLIGHTS")
628     PG_Heading.pack(padx = 25, pady = 10, anchor = "w")
629     def go_back():
630         for i in Main_fame.winfo_children():
631             i.destroy()
632         Main_frm_Authentication_Btns()
633         PG_Get_Flight_Details()
634 
635     dummy_back_btn = CTkButton(Main_fame,text="Back", command = go_back)
636     dummy_back_btn.place(x =10, y = 10)
637 
638     btns = []
639 
640     global radio_val, dest_airport, origin_airport
641     temp_qry = "SELECT F_Departure, F_Arrival, F_Airline, F_price, F_ID FROM flights WHERE
    F_Departure = '" + origin_airport+ "' AND F_Arrival = '" + dest_airport+ "';"
642     cur.execute(temp_qry)
643     result = cur.fetchall()
644     btn_data= {}
645     FLIGHT_ID_dict = {}
646     key = 1
647     for i in result :
648         btn_data[key] = f"{i[0]} -----> {i[1]} : {i[2]} : {i[3]}"
649         FLIGHT_ID_dict[key] = i[4]
650         key += 1
651 
652     def on_btn_click(index):
653         global Flight_ID
```

```
654         Flight_ID = FLIGHT_ID_dict[index]
655         if _isSignedIn == True:
656             PG_Payment()
657         else :
658             PG_Sign_in()
659             pass
660     for id, label in btn_data.items() :
661 
662         btn = CTkButton(fligt_search_result_frm, text = label,bg_color="transparent", command
    = lambda idx=id: on_btn_click(idx)).pack(pady = 10, padx = 10, anchor = "w")
663         btns.append(btn)
664     con.commit()
665     Main_frm_Authentication_Btns()
666 
667 # =>1-----Get Flight details-----------------------------------------------------------------
    ---
668 def PG_Get_Flight_Details():
669     global div_frame, _isSignedIn, User
670     for i in Main_fame.winfo_children():
671         i.destroy()
672 
673     prev_page = 1
674     temp_frm_width = 500
675     temp_frm_height = 320
676     div_frame = CTkFrame(Main_fame,width=temp_frm_width, height = temp_frm_height)
677     div_frm_xpos = 75
678     din_frm_widget_width = 350
679     div_frame.place(x = (m_r_width/(2))-(temp_frm_width/2),
680                                 y = (m_r_height/(2)-(temp_frm_height/2)))
681 
682     def Func_radio_btn():
683         global radio_val
684         radio_val = book_a_fligt_radio_val.get()
685         if radio_val == "ReturnRadio":
686             Dest_Date_Btn.configure(state = tk.NORMAL)
687         else:
688             Dest_Date_Btn.configure(state = tk.DISABLED)
689 
690     global book_a_fligt_radio_val
691     book_a_fligt_radio_val = StringVar(value = "other")
692     rd_btn_y_pos = 20
693     return_radio_btn =
    createRadioButton(div_frame,"Return","ReturnRadio",book_a_fligt_radio_v↩
    al,Func_radio_btn,div_frm_xpos, rd_btn_y_pos)
694 
695     one_way_radio_btn = createRadioButton(div_frame,"One Way", "OnewayRadio",
    book_a_fligt_radio_val,Func_radio_btn, div_frm_xpos+130, rd_btn_y_pos)
696 
697 
698 
699     def Combo_get_origin_val(origin_combo_value):
```

```
        global origin_airport
        origin_airport = origin_combo_value

    departure_place = StringVar(value="dep_combo_other")
    departure_place.set("Departure")

    Origin_Airport = CTkComboBox(div_frame,width=din_frm_widget_width,
                                 values=major_airports,
                                     variable= departure_place, command = Combo_get_origin_val)
    Origin_Airport.place (x = div_frm_xpos, y = rd_btn_y_pos+50)

    def Combo_get_dest_val(dest_combo_value):
        global dest_airport
        dest_airport = dest_combo_value

    global arrival_place
    arrival_place = StringVar(value="des_combo_other")
    arrival_place.set("Destination")
    Dest_Airport = CTkComboBox(div_frame,width=din_frm_widget_width,
                               values=major_airports,variable= arrival_place, command=
Combo_get_dest_val)

    Dest_Airport.place (x = div_frm_xpos, y = rd_btn_y_pos+82)

    arrival_place = arrival_place.get()

    def Dept_open_date_picker():

        top = CTkToplevel(root)
        top.title("Select a Date")
        top.attributes("-topmost", True)
        cal = Calendar(top, selectmode='day', date_pattern = "yyyy-mm-dd")
        cal.pack(pady=10)
        def select_date():
            global Dept_selected_date
            Dept_selected_date = cal.get_date()
            Dept_date_label.configure(text=f"Selected Date : {Dept_selected_date}")
            top.destroy()

        select_button = CTkButton(top, text="Select Date", command=select_date)
        select_button.pack(pady=10)

    Dept_Date_Btn = CTkButton(div_frame, text="Departure Date", command=Dept_open_date_picker,
corner_radius=100)
    Dept_Date_Btn.place(x=div_frm_xpos, y =rd_btn_y_pos+114)

    Dept_date_label = CTkLabel(div_frame, text= "Select Date")
    Dept_date_label.place(x = div_frm_xpos+170, y = rd_btn_y_pos+114)
    def Dest_open_date_picker():

```

```
748         top = CTkToplevel(root)
749         top.title("Select a Date")
750         top.attributes("-topmost", True)
751         cal = Calendar(top, selectmode='day', date_pattern = "yyyy-mm-dd")
752         cal.pack(pady=10)
753         def select_date():
754             global Dest_selected_date
755             Dest_selected_date = cal.get_date()
756             Dest_date_label.configure(text=f"Selected Date : {Dest_selected_date}")
757             top.destroy()
758 
759         select_button = CTkButton(top, text="Select Date", command=select_date)
760         select_button.pack(pady=10)
761 
762 
763     Dest_Date_Btn = CTkButton(div_frame, text="Arrival Date", command=Dest_open_date_picker,
corner_radius=100, state=tk.DISABLED)
764     Dest_Date_Btn.place(x=div_frm_xpos, y =rd_btn_y_pos+147)
765 
766     Dest_date_label = CTkLabel(div_frame, text= "Select Date")
767     Dest_date_label.place(x = div_frm_xpos+170, y = rd_btn_y_pos+147)
768 
769     def Null_Check():
770         error_name = ""
771         try :
772             global radio_val, dest_airport, origin_airport
773             radio_val
774             dest_airport
775             origin_airport
776             Dept_selected_date
777             if radio_val == "ReturnRadio" :
778                 Dest_selected_date
779 
780             if dest_airport not in globals() or origin_airport not in globals() or radio_val
not in globals():
781                 pass
782 
783         except NameError :
784             error_name = "NameError"
785             error_label = CTkLabel(div_frame, text = "Feilds Cannot Be Empty", font =
("Bradley Hand ITC" , 18, "italic", "bold"), text_color = "red")
786             error_label.place(x = 140, y = 238)
787             def refresh():
788                 error_label.destroy()
789             error_label.after(3000, refresh)
790 
791         finally:
792             if error_name != "NameError":
793                 global is_flight_details_obtained
794                 is_flight_details_obtained = True
795                 PG_search_flight_()
```

```
796
797     Search_Fligths_btn = CTkButton(div_frame, text = "Search Fligths", width =
    din_frm_widget_width, corner_radius=75, command=lambda :(Null_Check()))
798     Search_Fligths_btn.place(x = div_frm_xpos, y =rd_btn_y_pos + 180)
799     con.commit()
800     Main_frm_Authentication_Btns()
801
802 # =>-----MAIN FRAME-------------------------------------------------------------------
803 global Main_frm_Authentication_Btns
804 def Main_frm_Authentication_Btns():
805     if _isSignedIn == True:
806         global User
807         User_Label = CTkLabel(Main_fame, text = f"{User}", text_color= "#007acc", font=
    ("Bradley Hand ITC" , 18, "italic", "bold"), width = 20)
808         User_Label.place(x = m_r_width-200, y= 10)
809         User_Label.update_idletasks()
810         lbl_width = User_Label.winfo_width()
811         User_Label.place(x = (m_r_width/(1.2))-(lbl_width/2),y = 10)
812
813         def on_select(option):
814             global _isSignedIn, User , isAdmin
815
816             if option ==  "Cancel Flights":
817                 Setting_btn.set("Settings")
818                 tmp_root = CTkToplevel(root)
819                 tmp_root_width , tmp_root_height = 400, 300
820                 tmp_root.geometry(f"{tmp_root_width}x{tmp_root_height}")
821                 tmp_root.attributes("-topmost", True)
822
823                 temp_frame = CTkFrame(tmp_root, width= tmp_root_width, height =
    tmp_root_height, border_color= "#007acc", border_width= 2)
824                 temp_frame.pack()
825
826                 Flight_ID_entry = CTkEntry(temp_frame, placeholder_text= "Ticket ID",width=
    350)
827                 Flight_ID_entry.place(x = (tmp_root_width/(2))-(350/2),
828                                       y = (tmp_root_height/(2)-(28/2))-38)
829
830                 def Cancel_tickets():
831                     Flight_id = Flight_ID_entry.get()
832
833                     cur.execute("SELECT BID, BU_NAME FROM booking WHERE BID = %s AND IS_ACTIVE
    = 1", Flight_id)
834                     row = cur.fetchone()
835                     global User
836                     if row and row[1] == User :
837                         cur.execute("UPDATE booking SET IS_ACTIVE = 0 WHERE BID = %s AND
    IS_ACTIVE = 1", Flight_id)
838                         lbl = CTkLabel(temp_frame, text ="Successfully Canceled",
    text_color="green", font = ("Bradley Hand ITC", 18, "italic", "bold"))
```

```
    #errorLabeling(temp_frame, "Successfully Canceled", _textcolor = "green", _x = 110, _y =
    (tmp_root_height/(2)-(28/2)+38) )
839                         lbl.place(x = 110, y = (tmp_root_height/(2)-(28/2)+38))
840                         def w8():
841                             tmp_root.destroy()
842                             con.commit()
843                         lbl.after(3000,w8)
844                     else:
845                         errorLabeling(temp_frame, "Flight ID is Incorect", _x = 110, _y =
    (tmp_root_height/(2)-(28/2)+38))
846
847                 Cancel_btn = CTkButton(temp_frame, text="Cancel Flight", width = 350 ,
    corner_radius= 100, command= Cancel_tickets)
848                 Cancel_btn.place(x = (tmp_root_width/(2))-(350/2),
849                                  y = (tmp_root_height/(2)-(28/2)))
850
851             if option == "Booking History" :
852                 Setting_btn.set("Settings")
853                 tmp_root = CTkToplevel(root)
854                 tmp_root_width , tmp_root_height = 900, 300
855                 tmp_root.geometry(f"{tmp_root_width}x{tmp_root_height}")
856                 tmp_root.attributes("-topmost", True)
857                 tmp_root.resizable(False, False)
858
859                 temp_frame = CTkScrollableFrame(tmp_root, width= tmp_root_width, height =
    tmp_root_height, border_color= "#007acc", border_width= 2)
860                 temp_frame.pack()
861                 global User
862                 cur.execute("SELECT BU_NAME, BID, F_Departure, F_Arrival, F_Airline, F_price
    FROM  booking B, flights F WHERE B.BU_NAME = %s AND B.B_FLIGHT = F.F_ID AND B.IS_ACTIVE = 1",
    User)
863                 row = cur.fetchall()
864                 for i in row :
865                     CTkLabel(temp_frame, text = i).pack(padx = 20, pady = 5, anchor = "w")
866
867             if option == "Edit Account" :
868                 Setting_btn.set("Settings")
869                 tmp_root = CTkToplevel(root)
870                 tmp_root_width , tmp_root_height = 400, 600
871                 tmp_root.geometry(f"{tmp_root_width}x{tmp_root_height}")
872                 tmp_root.attributes("-topmost", True)
873                 tmp_root.resizable(False, False)
874
875                 form_frm_width = 400
876                 form_frm_height = 600
877                 form_frm = CTkFrame(tmp_root, width=form_frm_width, height=form_frm_height)
878                 form_frm.place(x = 0, y = 0)
879
880                 def go_back():
881                     for i in Main_fame.winfo_children():
882                         i.destroy()
```

```
883                     PG_Sign_in()
884
885                 def NullCheck():
886                     global DOB_selected_date, cal
887                     F_name = F_name_Entry.get()
888                     L_name = L_name_Entry.get()
889                     U_name = U_name_Entry.get()
890                     _Gender = Gender.get()
891                     if "cal" in globals():
892                         dob = cal.get_date()
893                         dob_dt = datetime.strptime(dob, "%Y-%m-%d")
894                         today = datetime.today()
895                         global Age
896                         Age = today.year - dob_dt.year
897                         if (today.month, today.day) > (dob_dt.month, dob_dt.day):
898                             Age +=1
899                     else :
900                         dob = ""
901                     Gmail = gmail_Entry.get()
902                     _pass = crypt(pass_Entry.get()).encrypt()
903                     _re_pass = crypt(re_pass_Entry.get()).encrypt()
904                     phonenumber = phonnumber_Entry.get()
905
906                     if _pass != _re_pass:
907                         errorLabeling(form_frm, "Passwords Don't Match", _x = 110, _y = 410)
908
909                     if F_name != "" :
910                         cur.execute("UPDATE user_details set UF_name = %s", F_name)
911
912                     if L_name != "" :
913                         cur.execute("UPDATE user_details set UL_name = %s", L_name)
914
915                     if U_name != "" :
916                         cur.execute("UPDATE user_details set U_name = %s", U_name)
917
918                     if _Gender != "other"  :
919                         cur.execute("UPDATE user_details set U_gender = %s", _Gender)
920
921                     if phonenumber != "" :
922                         cur.execute("UPDATE user_details set U_phno = %s", phonenumber)
923
924                     if _pass != "" :
925                         cur.execute("UPDATE user_details set U_password = %s", _pass)
926
927                     if Gmail != "" :
928                         cur.execute("UPDATE user_details set U_Gmail = %s", Gmail)
929
930                     if dob != "" :
931                         cur.execute("UPDATE user_details set U_dob = %s", dob)
932                         cur.execute("UPDATE user_details set U_AGE = %s", Age)
```

```
933
934                     if F_name == "" and L_name == "" and U_name == "" and dob == "" and  Gmail
    == "" and _pass == "" and _re_pass == "" and phonenumber == "" and _Gender == "other" :
935                         errorLabeling(form_frm, "Feilds Cannot Be Null", _x = 110, _y = 410)
936
937                     else :
938                         lbl = CTkLabel(form_frm, text= "Succesfully Updated", text_color=
    "green", font=("Bradley Hand ITC" , 18, "italic", "bold"))
939                         lbl.place(x = 110, y = 410)
940                         # errorLabeling(form_frm, "Succesfully Updated", _textcolor = "green",
    _x = 110, _y = 410)
941                         def w8():
942                             tmp_root.destroy()
943                         lbl.after(3000, w8)
944                     con.commit()
945
946                 cur.execute("SELECT UF_name, UL_name, U_name, U_Gmail, U_phno, U_password,
    U_dob, U_gender, U_AGE FROM user_details WHERE U_name = %s", User)
947                 result = cur.fetchone()
948
949                 F_name_Entry = CTkEntry(form_frm, width = 350, placeholder_text=result[0])
950                 F_name_Entry.place(x = 25, y = 10)
951
952                 L_name_Entry = CTkEntry(form_frm, width = 350, placeholder_text=result[1])
953                 L_name_Entry.place(x = 25, y = 50)
954
955                 def Func_radio_btn():
956                     global _Gender
957                     _Gender = Gender.get()
958
959                 global Gender
960                 Gender = StringVar(value = "other")
961
962                 rd_btn_y_pos = 90
963
964                 male_radio_btn = createRadioButton(form_frm,
    "Male","M",Gender,Func_radio_btn,25, rd_btn_y_pos)
965
966                 female_radio_btn = createRadioButton(form_frm,"Female", "F",
    Gender,Func_radio_btn, 25+130, rd_btn_y_pos)
967
968                 other_radio_btn = createRadioButton(form_frm,"Other", "O",
    Gender,Func_radio_btn, 25+130 + 130, rd_btn_y_pos)
969                 global cal
970
971                 def DOB_open_date_picker():
972
973                     top = CTkToplevel(form_frm)
974                     top.title("Select a Date")
975                     top.attributes("-topmost", True)
976                     global cal
```

```
977                     cal = Calendar(top, selectmode='day', date_pattern = "yyyy-mm-dd")
978                     cal.pack(pady=10)
979                     def select_date():
980                         global DOB_selected_date
981                         DOB_selected_date = cal.get_date()
982                         DOB_Date_label.configure(text=f"Selected Date : {DOB_selected_date}")
983                         top.destroy()
984 
985                     select_button = CTkButton(top, text="Select Date", command=select_date)
986                     select_button.pack(pady=10)
987 
988                 DOB_Date_Btn = CTkButton(form_frm, text="Date Of Birth",
    command=DOB_open_date_picker, corner_radius=100)
989                 DOB_Date_Btn.place(x=25, y =rd_btn_y_pos+40)
990 
991                 DOB_Date_label = CTkLabel(form_frm, text= "Select Date")
992                 DOB_Date_label.place(x = 25+170, y = 90+40)
993 
994                 U_name_Entry = CTkEntry(form_frm, width = 350, placeholder_text=result[2])
995                 U_name_Entry.place(x = 25, y = 130+40)
996 
997 
998                 gmail_Entry = CTkEntry(form_frm, width = 350, placeholder_text=result[3])
999                 gmail_Entry.place(x = 25, y = 170+40)
1000 
1001 
1002                 def Show_pass():
1003                     if pass_Entry.cget('show') == '*':
1004                         pass_Entry.configure(show='')
1005                         show_btn.configure(text=" Hide ")
1006                     else:
1007                         pass_Entry.configure(show='*')
1008                         show_btn.configure(text="Show")
1009 
1010                 def Re_Show_pass():
1011                     if re_pass_Entry.cget('show') == '*':
1012                         re_pass_Entry.configure(show='')
1013                         re_show_btn.configure(text=" Hide ")
1014                     else:
1015                         re_pass_Entry.configure(show='*')
1016                         re_show_btn.configure(text="Show")
1017 
1018                 pass_Entry = CTkEntry(form_frm, width = 350, placeholder_text="Password", show
    = "*")
1019                 show_btn = CTkButton(pass_Entry, width = 22, height=28,
    text="Show",border_color="#565b5e",border_width=2, fg_color="transparent", command=Show_pass)
1020                 show_btn.place(x = 304, y=0)
1021                 pass_Entry.place(x = 25, y = 170+80)
1022 
1023                 re_pass_Entry = CTkEntry(form_frm, width = 350, placeholder_text="Re-
    Password", show = "*")
```

```
1024                re_show_btn = CTkButton(re_pass_Entry, width = 22, height=28,
    text="Show",border_color="#565b5e",border_width=2, fg_color="transparent",
    command=Re_Show_pass)
1025                re_show_btn.place(x = 304, y=0)
1026                re_pass_Entry.place(x = 25, y = 210+80)
1027
1028                phonnumber_Entry = CTkEntry(form_frm, 350, placeholder_text=result[4])
1029                phonnumber_Entry.place(x = 25, y =250+80)
1030
1031                Create_acc_btn = CTkButton(form_frm, width = 350, text="Alter Account",
    corner_radius=100, command = NullCheck)
1032                Create_acc_btn.place(x = 25, y = 290+80)
1033
1034                temp_frame2 = CTkFrame(form_frm, width = 100, height= 200, border_color=
    "light Grey", border_width= 2,fg_color= "transparent")
1035                lbl = CTkLabel(temp_frame2, text="Enter The Feilds That You Want to
    Change",fg_color= "transparent")
1036                lbl.pack()
1037                def onhover(event):
1038                    temp_frame2.place(x = 25, y =410+40)
1039
1040                info_Btn = CTkButton(form_frm, text = "i", width= 10, height = 5,
    corner_radius= 100, hover= "on_hover")
1041                info_Btn.place(x = 25, y = 370+40)
1042                info_Btn.bind("<Enter>", onhover)
1043                info_Btn.bind("<Leave>", lambda event : temp_frame2.place_forget())
1044
1045            if  option == "Logout":
1046                _isSignedIn = False
1047                User =  ""
1048                isAdmin = False
1049                PG_Get_Flight_Details()
1050
1051            if  option == "Adiministrate":
1052                temp_TL = CTkToplevel(root)
1053                temp_TL.title("http:www.HADAirlineManagementSystem.com/Admin")
1054                temp_TL.attributes("-topmost", True)
1055                temp_TL.geometry("400x300")
1056
1057                def addAdmin():
1058                    temp_TL.destroy()
1059                    temp_TL2 = CTkToplevel(root)
1060                    temp_TL2.title("http:www.HADAirlineManagementSystem.com/Admin")
1061                    temp_TL2.attributes("-topmost", True)
1062                    temp_TL2.geometry("400x300")
1063
1064                    def add():
1065                        user = user_Entry.get()
1066                        cur.execute("UPDATE user_details SET U_isAdmin = 1 WHERE U_name = %s",
    user)
1067                        con.commit()
```

```
1068                        if cur.execute("SELECT U_isAdmin from user_details WHERE U_name = %s",
    user)==1:
1069                            label = CTkLabel(temp_TL2, text= "Successfully Updated", font=
    ("Bradley Hand ITC" , 18, "italic", "bold"), text_color= "green")
1070                            label.place(x = 110, y = 178)
1071                            def dest():
1072                                label.destroy()
1073                                temp_TL2.destroy()
1074                            label.after(3000, dest)
1075                        else:
1076                            label = CTkLabel(temp_TL2, text= "Username Not Found Or Server
    Down ", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text_color= "red")
1077                            label.place(x = 45, y = 178)
1078                            def dest():
1079                                label.destroy()
1080                            label.after(3000, dest)
1081
1082
1083
1084                    user_Entry = CTkEntry(temp_TL2, placeholder_text= "UserName To Change As
    Admin", width=240)
1085                    user_Entry.place(x = 80, y = 100)
1086
1087                    user_Entry_btn = CTkButton(temp_TL2, text= " Change", command=add)
1088                    user_Entry_btn.place(x = 130, y = 138)
1089
1090                def delAdmin():
1091                    temp_TL.destroy()
1092                    temp_TL2 = CTkToplevel(root)
1093                    temp_TL2.title("http:www.HADAirlineManagementSystem.com/Admin")
1094                    temp_TL2.attributes("-topmost", True)
1095                    temp_TL2.geometry("400x300")
1096
1097                    def _del():
1098                        user = user_Entry.get()
1099                        cur.execute("UPDATE user_details SET U_isActive = 0 WHERE U_name =
    %s", user)
1100                        con.commit()
1101                        if cur.execute("SELECT U_isActive from user_details WHERE U_name =
    %s", user)==0:
1102                            label = CTkLabel(temp_TL2, text= "Username Not Found Or Server
    Down ", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text_color= "red")
1103                            label.place(x = 45, y = 178)
1104                            def dest():
1105                                label.destroy()
1106                            label.after(3000, dest)
1107
1108                        else:
1109                            label = CTkLabel(temp_TL2, text= "Successfully Deleted", font=
    ("Bradley Hand ITC" , 18, "italic", "bold"), text_color= "green")
1110                            label.place(x = 110, y = 178)
```

```
                            def dest():
                                label.destroy()
                                temp_TL2.destroy()
                            label.after(3000, dest)


                    user_Entry = CTkEntry(temp_TL2, placeholder_text= "UserName To Delete From Admin", width=240)
                    user_Entry.place(x = 80, y = 100)

                    user_Entry_btn = CTkButton(temp_TL2, text= " Change", command=_del)
                    user_Entry_btn.place(x = 130, y = 138)

                def addFlights():
                    temp_TL.destroy()
                    temp_TL2 = CTkToplevel(root)
                    temp_TL2.title("http:www.HADAirlineManagementSystem.com/Admin")
                    temp_TL2.attributes("-topmost", True)
                    temp_TL2.geometry("400x300")

                    def add():
                        dept = Dept_Entry.get()
                        arr = Arr_Entry.get()
                        Air = Air_Entry.get()
                        price = price_Entry.get()
                        cur.execute("SELECT COUNT(*) FROM flights")
                        tempCount = cur.fetchone()[0]
                        if dept =="" or arr =="" or Air =="" or price =="":
                            errorLabeling(temp_TL2, "Fields Cannot Be Empty", _x = 30, _y = 190+35)
                        elif not price.isdigit():
                           errorLabeling(temp_TL2, "Check The Price", _x = 30, _y = 190+35)
                        else:
                            cur.execute("INSERT INTO flights(F_Departure,F_Arrival,F_Airline, F_price ) VALUES (%s, %s,%s,%s)", (dept, arr, Air, int(price)))
                            con.commit()
                        cur.execute("SELECT COUNT(*) FROM flights")
                        check_Count = cur.fetchone()[0]
                        if check_Count ==tempCount+1:
                            label = CTkLabel(temp_TL2, text= "Successfully Added", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text_color= "green")
                            label.place(x = 110, y = 190+35)
                            def dest():
                                label.destroy()
                                temp_TL2.destroy()
                            label.after(3000, dest)

                        else:
                            label = CTkLabel(temp_TL2, text= " Error Occured While Inserting ", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text_color= "red")
                            label.place(x = 45, y = 190+35)
```

```
                            def dest():
                                label.destroy()
                            label.after(3000, dest)

                    tempy = 50
                    Dept_Entry = CTkEntry(temp_TL2, placeholder_text= "Depature Place", width=240)
                    Dept_Entry.place(x = 80, y = tempy)

                    Arr_Entry = CTkEntry(temp_TL2, placeholder_text= "Arrival Place", width=240)
                    Arr_Entry.place(x = 80, y = tempy+35)

                    Air_Entry = CTkEntry(temp_TL2, placeholder_text= "Airline Name", width=240)
                    Air_Entry.place(x = 80, y = tempy+35+35)

                    price_Entry = CTkEntry(temp_TL2, placeholder_text= "Price", width=240)
                    price_Entry.place(x = 80, y = tempy+35+35+35)

                    add_Flight_btn = CTkButton(temp_TL2, text="Add", command=add)
                    add_Flight_btn.place(x = 125, y = tempy+35+35+35+35)

                def delflights():
                    temp_TL.destroy()
                    temp_TL2 = CTkToplevel(root)
                    temp_TL2.title("http:www.HADAirlineManagementSystem.com/Admin")
                    temp_TL2.attributes("-topmost", True)
                    temp_TL2.geometry("760x300")

                    def _del():
                        flightId = FlightIDEntry.get()
                        if flightId =="":
                            errorLabeling(temp_TL2, "Feild Empty", _x = 345, _y = 1)
                        elif flightId.isalpha():
                            errorLabeling(temp_TL2, "Invalid ID", _x = 346, _y = 1)
                        else:
                            cur.execute("SELECT F_ID FROM flights WHERE F_IsActive = 1")
                            result = cur.fetchall()
                            if (int(flightId),) not in result:
                                errorLabeling(temp_TL2, "ID Not Found", _x = 344, _y = 1)
                            else:
                                cur.execute("UPDATE flights SET F_IsActive = 0 WHERE F_ID = %s", flightId)
                                #errorLabeling(temp_TL2, "Deleted", _x = 347, _y = 1)
                                con.commit()
                                label = CTkLabel(temp_TL2, text= "Successfully Deleted", font= ("Bradley Hand ITC" , 18, "italic", "bold"), text_color= "green")
                                label.place(x = 336, y = 1)
                                def dest():
                                    label.destroy()
```

```
                                    temp_TL2.destroy()
                                label.after(3000, dest)


                    tempx, tempy = 10, 30
                    FlightIDEntry = CTkEntry(temp_TL2, placeholder_text="Flight ID")
                    FlightIDEntry.place(x = tempx+240, y = tempy)

                    mod_btn = CTkButton(temp_TL2, text="Modify", command=_del)
                    mod_btn.place(x = tempx+390, y = tempy)

                    sckrlble_frame = CTkScrollableFrame(temp_TL2, width = 720, height=220)
                    sckrlble_frame.place(x = tempx, y = tempy+35)
                    cur.execute("SELECT * FROM Flights WHERE F_IsActive = 1")
                    result = cur.fetchall()
                    for i in result:
                        lbl = CTkLabel(sckrlble_frame, text=f"{i[0]}  {i[1]}  {i[2]}  {i[3]}
{i[4]}", text_color= "Light Gray")
                        lbl.pack(padx = 50,pady = 10, anchor = "w")

                def ModFlights():
                    temp_TL.destroy()
                    temp_TL2 = CTkToplevel(root)
                    temp_TL2.title("http:www.HADAirlineManagementSystem.com/Admin")
                    temp_TL2.attributes("-topmost", True)
                    temp_TL2.geometry("760x300")

                    def mod():
                        flightId = FlightIDEntry.get()
                        if flightId =="":
                            errorLabeling(temp_TL2, "Feild Empty", _x = 345, _y = 1)
                        elif flightId.isalpha():
                            errorLabeling(temp_TL2, "Invalid ID", _x = 346, _y = 1)
                        else:
                            cur.execute("SELECT F_ID FROM flights WHERE F_IsActive = 1")
                            result = cur.fetchall()
                            if (int(flightId),) not in result:
                                errorLabeling(temp_TL2, "ID Not Found", _x = 344, _y = 1)
                            else:
                                temp_TL2.destroy()
                                temp_TL3 = CTkToplevel(root)
                                temp_TL3.title("http:www.HADAirlineManagement↩
System.com/Admin")
                                temp_TL3.attributes("-topmost", True)
                                temp_TL3.geometry("400x300")

                                tempx, tempy = 10 , 30
                                cur.execute("SELECT * FROM flights WHERE F_ID = %s", flightId)
                                result = cur.fetchone()

```

```
1251                                def Check():
1252                                    Dept = Dept_Entry.get()
1253                                    Arr = Arr_Entry.get()
1254                                    Air = Air_Entry.get()
1255                                    price = price_Entry.get()
1256
1257                                    if Dept != "":
1258                                        cur.execute("UPDATE flights SET F_Departure = %s WHERE
    F_ID =%s", (Dept,flightId))
1259
1260                                    if Arr != "":
1261                                        cur.execute("UPDATE flights SET F_Arrival = %s WHERE
    F_ID =%s", (Arr,flightId) )
1262
1263                                    if Air != "":
1264                                        cur.execute("UPDATE flights SET F_Airline = %s WHERE
    F_ID =%s", (Air,flightId) )
1265
1266                                    if price != "":
1267                                        cur.execute("UPDATE flights SET F_price = %s WHERE
    F_ID =%s", (price,flightId))
1268
1269                                    con.commit()
1270                                    label = CTkLabel(temp_TL3, text= "Successfully Updated",
    font= ("Bradley Hand ITC" , 18, "italic", "bold"), text_color= "green")
1271                                    label.pack(pady = 5)
1272                                    def dest():
1273                                        label.destroy()
1274                                        temp_TL3.destroy()
1275                                    label.after(3000, dest)
1276
1277                                Dept_Entry = CTkEntry(temp_TL3,
    placeholder_text=result[1],width = 240)
1278                                Dept_Entry.pack(pady = 5)
1279                                Arr_Entry = CTkEntry(temp_TL3,
    placeholder_text=result[2],width = 240)
1280                                Arr_Entry.pack(pady = 5)
1281                                Air_Entry = CTkEntry(temp_TL3,
    placeholder_text=result[3],width = 240)
1282                                Air_Entry.pack(pady = 5)
1283                                price_Entry = CTkEntry(temp_TL3,
    placeholder_text=result[4],width = 240)
1284                                price_Entry.pack(pady = 5)
1285                                Upd_BTN = CTkButton(temp_TL3, text="Update",width = 240,
    command=Check)
1286                                Upd_BTN.pack(pady = 5)
1287
1288                    tempx, tempy = 10, 30
1289                    FlightIDEntry = CTkEntry(temp_TL2, placeholder_text="Flight ID")
1290                    FlightIDEntry.place(x = tempx+240, y = tempy)
1291
```

```
1292                    mod_btn = CTkButton(temp_TL2, text="Modify", command=mod)
1293                    mod_btn.place(x = tempx+390, y = tempy)
1294
1295                    sckrlble_frame = CTkScrollableFrame(temp_TL2, width = 720, height=220)
1296                    sckrlble_frame.place(x = tempx, y = tempy+35)
1297                    cur.execute("SELECT * FROM Flights WHERE F_IsActive = 1")
1298                    result = cur.fetchall()
1299                    for i in result:
1300                        lbl = CTkLabel(sckrlble_frame, text=f"{i[0]}  {i[1]}  {i[2]}  {i[3]}
    {i[4]}", text_color= "Light Gray")
1301                        lbl.pack(padx = 50,pady = 10, anchor = "w")
1302
1303                tempx = 55
1304                tempy = 150
1305                Add_Admin_btn = CTkButton(temp_TL, text="Add Admin", command=addAdmin)
1306                Add_Admin_btn.place(x = tempx, y = tempy-30)
1307
1308                Add_Flight_btn = CTkButton(temp_TL, text="Add Flights", command=addFlights)
1309                Add_Flight_btn.place(x = tempx+150, y = tempy-30)
1310
1311                Rem_Flight_btn = CTkButton(temp_TL, text="Delete Flights", command=
    delflights)
1312                Rem_Flight_btn.place(x = tempx+150, y = tempy+10)
1313
1314                Rem_Admin_btn = CTkButton(temp_TL, text="Delete Admin", command=delAdmin)
1315                Rem_Admin_btn.place(x = tempx, y = tempy+10)
1316
1317                Mod_Flight_btn = CTkButton(temp_TL, text="Modify Flights", command=
    ModFlights)
1318                Mod_Flight_btn.place(x = 130, y = tempy+50)
1319
1320
1321
1322      ######################## ADMINISTRATE
1323            if isAdmin==1:
1324                option = ["Cancel Flights", "Booking History", "Edit Account","Adiministrate",
    "Logout"]
1325            else:
1326                option = ["Cancel Flights", "Booking History", "Edit Account", "Logout"]
1327            Setting_btn = CTkOptionMenu(Main_fame,  values= option, command= on_select)
1328            Setting_btn.place(x = ((m_r_width/(1.2))-(lbl_width/2)) +lbl_width+ 10, y = 10)
1329            Setting_btn.set("Settings")
1330        else:
1331            temp_xpos = m_r_width-300
1332            Sign_in_btn = CTkButton(Main_fame, text = "Sign In", command= PG_Sign_in)
1333            Sign_in_btn.place(x = temp_xpos, y = 10)
1334
1335            Sign_up_btn = CTkButton(Main_fame, text = "Sign Up", fg_color="transparent",
    border_color= "grey", border_width=2, command=PG_Sign_Up)
1336            Sign_up_btn.place(x = temp_xpos+150, y = 10)
1337
```

```
1338 Main_frm_Authentication_Btns()
1339
1340 PG_Get_Flight_Details()
1341
1342 #--------------------------------------------------------------------------------
1343
1344 root.mainloop()
1345 cur.close()
1346 con.commit()
1347
```