07 Functions

start.py USER-EXECUTABLE – Starting point for users	
1. make_booking()	Opens Booking window
2. manage_user()	Opens Manage profile window
3. about_this_program()	View program information

user.py User functions	
1. bookings()	Bookings window
a. onlogin()	Function to be run on login
1. main()	User portal for making bookings
1. book_taxi()	Taxi booking
2. book_bus()	Bus booking
3. about_this_program()	View program information
4. logout()	Logs out and returns to start page
b. register()	Opens registration window
c. manage_profile()	Opens Manage profile window
2. manage_user_profile()	Manage profile function
a. bookings_login()	Opens bookings login page
b. onlogin()	Function to be run on login
1. manage()	Manage user window
1. delete()	Delete user profile password
2. logout()	Logs out and returns to start page
3. passwd()	Change login password
4. info()	Change personal information
1. name()	Change full name
2. contacts()	Change electronic mail and/or phone number.

bookings.py Booking functions	
1. bus() 2. taxi()	Function to book taxi or bus journeys.
1. payment()	Make payment for booking.

emp.py Corporate functions	
1. emp_main()	Bookings function
2. onlogin()	Function to be run upon login
1. admin()	Administration suite
1. logout()	Log out and return to employee login page
2. db()	Database management
3. agents()	Manage agents
4. bookings()	Manage bookings
5. users()	Manage users
6. about_this_program()	View program information
7. admins()	Manage administrators
8. passwd()	Change password for currently logged in admin
2. empbookings()	Agent portal to make and manage bookings
1. book_taxi()	Taxi booking
2. book_bus()	Bus booking
3. about_this_program()	View program information
4. logout()	Logs out and returns to employee login
5. managetaxibkgs()	Manage taxi bookings
6. managebusbkgs()	Manage bus bookings
3. about_this_program()	View program information

manage.py Administration functions	
1. manage_admin(), 2. manage_agents(), 3. manage_users(),	Function to manage administrators, agents, and/or users.
1. viewall()	View details of all profiles.
2. viewone()	View details of one profile.
3. delone()	Delete one profile.
4. passwd()	Change password for profile
5. add()	Add profile
4. manage_db()	Manage database function
1. showtb()	Show selected table
2. droptb()	Drop selected table from database
3. deltb()	Delete contents of table from database
4. exporttb()	Export table data to CSV file
5. help()	View help on DELETE FROM and DROP SQL commands.

managebkgs.py Booking functions	
1. bus() 2. taxi()	Manage bus and/or taxi booking function
1. viewall()	View details of all bookings.
2. viewone()	View details of one booking.
3. delone()	Delete a booking

sysinfo.py View program and system information	
1. about()	Program and system information
1. close()	Closes About popup

init.py Initialises the databases.	
1. initdb()	Initalises database.

08 Source code

1. start.py

```
#!/bin/python3

#import statements
#Imports libraries
import tkinter as tk
import mysql.connector as ms
from tkinter.ttk import Separator
import ctypes
import platform as pf

#Imports other Python scripts
import user
import init
import sysinfo

#definitions

#Font
fnt=('IBM Plex Mono',12,'bold italic')
fntit=('IBM Plex Mono',12,'italic')
h1fnt=('IBM Plex Sans',24)
hfnt=('IBM Plex Sans',36,'bold')
menufnt=('IBM Plex Mono',11)

#MySQL connection

con=ms.connect(host='localhost',user='root',password='123456')
cur=con.cursor()

#Initalises database
init.initdb()

#functions
def make_booking():            #to make booking
        welcome.destroy()
        user.bookings()

def manage_user():             #to manage user
        welcome.destroy()
        user.manage_user_profile()

def about_this_program():
        sysinfo.about()

#Enables DPI scaling on supported Windows versions
if pf.system()=='Windows':
        try:
                ctypes.windll.shcore.SetProcessDpiAwareness(True)
        except:
                pass

#main window
welcome=tk.Tk()
welcome.title('Start Page')

#maximises window
try:
        welcome.state('zoomed')
except:
        w,h=welcome.winfo_screenwidth(),welcome.winfo_screenheight()
        welcome.geometry(str(w)+'x'+str(h))
```

```
menubar=tk.Menu(welcome)

more=tk.Menu(menubar,tearoff=0)
menubar.add_cascade(label='Info',menu=more,font=menufnt)
more.add_command(label='About this program ... ',command=about_this_program,font=menufnt,underline=0)
welcome.config(menu=menubar)

tk.Grid.columnconfigure(welcome,0,weight=1)

#FRAME 1
tk.Grid.rowconfigure(welcome,0,weight=1)
f1=tk.Frame(welcome,bg='#283593')
f1.grid(row=0,column=0,sticky=tk.NSEW)

#frame 1 grid
tk.Grid.columnconfigure(f1,0,weight=1)
tk.Grid.rowconfigure(f1,0,weight=1)

logo_img=tk.PhotoImage(file='img/logo.png')
logo=tk.Label(f1,image=logo_img,font=hfnt,fg='white',bg='#283593')
logo.grid(column=0,row=0,padx=10,pady=10,sticky=tk.EW)
logo.image=logo_img

Separator(f1,orient='horizontal').grid(column=0,row=1,sticky=tk.EW,padx=10,pady=10,columnspan=2)

#FRAME 2
tk.Grid.rowconfigure(welcome,1,weight=1)

f2=tk.Frame(welcome)
f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

#frame 2 grid
tk.Grid.columnconfigure(f2,0,weight=1)
tk.Grid.columnconfigure(f2,1,weight=1)

#Bookings
tk.Grid.rowconfigure(f2,5,weight=1)
img6=tk.PhotoImage(file='icons/booking.png')
bkgbtn=tk.Button(f2,text='Booking',image=img6,font=fnt,command=make_booking)
bkgbtn.grid(column=0,row=5,padx=10,pady=10,sticky=tk.E)
tk.Label(f2,text='Make a
booking ... ',font=fnt,bg='#00e676').grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

#Manage Profile
tk.Grid.rowconfigure(f2,6,weight=1)
img4=tk.PhotoImage(file='icons/manage_accts.png')
passbtn=tk.Button(f2,text='Profile',image=img4,command=manage_user)
passbtn.grid(column=0,row=6,padx=10,pady=10,sticky=tk.E)
tk.Label(f2,text='Manage user profile ... ',font=fnt).grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)

#Exit
tk.Grid.rowconfigure(f2,7,weight=1)
img5=tk.PhotoImage(file='icons/close.png')
passbtn=tk.Button(f2,text='Exit',image=img5,command=welcome.destroy)
passbtn.grid(column=0,row=7,padx=10,pady=10,sticky=tk.E)
tk.Label(f2,text='Exit',font=fnt,fg='red').grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)

tk.Grid.rowconfigure(f2,8,weight=1)

welcome.mainloop()
```

2. user.py

```
def bookings():       #make bookings
        import tkinter as tk
        import random as rd
        import mysql.connector as ms
        from tkinter.ttk import Separator
        from tkinter import messagebox
        import os
        import ctypes
        import platform as pf

        import bookings
        import sysinfo

        #mysql connection
        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
        cur=con.cursor()

        #Enables DPI scaling on supported Windows versions
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass

        #init GUI
        logwin=tk.Tk()
        logwin.title('Make bookings')
        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)
        menufnt=('IBM Plex Mono',11)

        #Maximises windows
        try:
                logwin.state('zoomed')
        except:
                w,h=logwin.winfo_screenwidth(),logwin.winfo_screenheight()
                logwin.geometry(str(w)+'x'+str(h))

        #login
        def onlogin():
                #Main menu
                def main():
                        #functions
                        def book_taxi():       #Opens taxi booking window.
                                bookings.taxi()

                        def book_bus():        #Opens bus booking window
                                bookings.bus()

                        def about_this_program():
                                sysinfo.about()

                        def logout():  #Logs out and returns to the start page.
                                main_menu.destroy()
                                os.system('python3 start.py')

                        main_menu=tk.Tk()
                        main_menu.title('Main Menu')
                        try:
                                main_menu.state('zoomed')
                        except:
                                w,h=main_menu.winfo_screenwidth(),main_menu.winfo_screenheight()
                                main_menu.geometry(str(w)+'x'+str(h))

                        menubar=tk.Menu(main_menu)

                        more=tk.Menu(menubar,tearoff=0)
                        menubar.add_cascade(label='Info',menu=more,font=menufnt)
```

```
                        more.add_command(label='About this
program ... ',command=about_this_program,font=menufnt,underline=0)
                        main_menu.config(menu=menubar)

                        tk.Grid.columnconfigure(main_menu,0,weight=1)

                        #FRAME 1
                        tk.Grid.rowconfigure(main_menu,0,weight=1)
                        f1=tk.Frame(main_menu,bg='#283593')
                        f1.grid(row=0,column=0,sticky=tk.NSEW)

                        #frame 1 grid
                        tk.Grid.columnconfigure(f1,0,weight=1)

                        cur.execute('select uname,fname from users')
                        a=dict(cur.fetchall())
                        cur.execute('select uname,uuid from users')
                        uuidlist=dict(cur.fetchall())

                        tk.Grid.rowconfigure(f1,0,weight=1)
                        tk.Grid.rowconfigure(f1,1,weight=1)
                        tk.Grid.rowconfigure(f1,2,weight=1)

                        logo_img=tk.PhotoImage(file='img/logo.png')
                        logo=tk.Label(f1,image=logo_img,fg='white',bg='#283593')
                        logo.grid(column=0,row=0,padx=10,pady=10,sticky=tk.EW)
                        logo.image=logo_img

                        tk.Label(f1,text='Welcome,
'+a[uname_inp],font=h1fnt,fg='white',bg='#283593').grid(column=0,row=1)
                        tk.Label(f1,text=('ID: '+uuidlist[uname_inp]),font=('IBM Plex
Sans',12),fg='black',bg='#00e676').grid(column=0,row=2,padx=10)



        Separator(f1,orient='horizontal').grid(column=0,row=3,sticky=tk.EW,padx=10,pady=10)

                        #FRAME 2
                        tk.Grid.rowconfigure(main_menu,1,weight=1)
                        f2=tk.Frame(main_menu)
                        f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

                        #frame 2 grid
                        tk.Grid.columnconfigure(f2,0,weight=1)
                        tk.Grid.columnconfigure(f2,1,weight=1)
                        tk.Grid.columnconfigure(f2,2,weight=1)
                        tk.Grid.columnconfigure(f2,3,weight=1)


                        tk.Label(f2,text=('You
can:'),font=fntit).grid(column=1,row=2,padx=10,pady=10,sticky=tk.W)

                        tk.Grid.rowconfigure(f2,5,weight=1)
                        #Book Taxi
                        img6=tk.PhotoImage(file='icons/taxi.png')
                        bkgbtn=tk.Button(f2,text='Book taxi',image=img6,font=fnt,command=book_taxi)
                        bkgbtn.grid(column=0,row=5,padx=10,pady=10,sticky=tk.E)
                        tk.Label(f2,text='Book a
taxi.',font=fnt,bg='yellow').grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

                        #Book Bus
                        img4=tk.PhotoImage(file='icons/bus.png')
                        passbtn=tk.Button(f2,text='Book Bus',image=img4,command=book_bus)
                        passbtn.grid(column=2,row=5,padx=10,pady=10,sticky=tk.E)
                        tk.Label(f2,text='Book a
bus.',font=fnt,fg='blue').grid(column=3,row=5,padx=10,pady=10,sticky=tk.W)


                        tk.Label(f2,text=('or:'),font=fntit).grid(column=1,row=9,padx=10,sticky=tk.W)

                        tk.Grid.rowconfigure(f2,11,weight=1)
                        #Logout
                        img7=tk.PhotoImage(file='icons/logout.png')
                        logoutbtn=tk.Button(f2,text='Logout',font=fnt,image=img7,command=logout)
                        logoutbtn.grid(column=0,row=11,padx=10,pady=10,sticky=tk.E)
```

```
        tk.Label(f2,text='Logout',font=fnt).grid(column=1,row=11,padx=10,pady=10,sticky=tk.W)


                        #Logout and Exit
                        img8=tk.PhotoImage(file='icons/close.png')
                        exitbtn=tk.Button(f2,text='Logout and
exit',font=fnt,image=img8,command=main_menu.destroy)
                        exitbtn.grid(column=2,row=11,padx=10,pady=10,sticky=tk.E)
                        tk.Label(f2,text='Logout and
exit',font=fnt,fg='red').grid(column=3,row=11,padx=10,pady=10,sticky=tk.W)

                        tk.Grid.rowconfigure(f2,12,weight=1)
                        main_menu.mainloop()

                uname_inp=login_uname.get()
                passwd_inp=login_passwd.get()
                cur.execute('select uname,passwd from users')
                op=dict(cur.fetchall())

                cur.execute('select uname,fname from users')
                fnamelist=dict(cur.fetchall())

                if (not uname_inp=='' and not uname_inp.isspace()) and (not passwd_inp=='' and not
passwd_inp.isspace()):
                        if uname_inp not in op.keys():
                                messagebox.showerror('Error','Username \''+uname_inp+'\' does not
exist.')
                        else:
                                if not passwd_inp == op[uname_inp]:
                                        messagebox.showerror('Error','Invalid password entered for
'+fnamelist[uname_inp]+'.')
                                else:
                                        logwin.destroy()
                                        main()
                else:
                        messagebox.showerror('Error','Please do not leave any fields blank.')

        #Register user
        def register():
                uuid='U'+str(rd.randint(10000,99999))
                #Adds user to DB
                def reguser():

                        reg_fname_inp=reg_fname.get()
                        reg_email_inp=reg_email.get()
                        reg_num_inp=reg_num.get()
                        reg_uname_inp=reg_uname.get().lower()
                        reg_passwd_inp=reg_passwd.get()

                        cur.execute('select uname from users')
                        users=cur.fetchall()

                        b=(reg_uname_inp,)
                        if (not reg_fname_inp.isspace()==True and not reg_fname_inp=='') and (not
reg_email_inp.isspace()==True and not reg_email_inp=='') and (not reg_num_inp.isspace()==True and
not reg_num_inp=='') and (not reg_uname_inp.isspace()==True and not reg_uname_inp=='') and (not
reg_passwd_inp.isspace()==True and not reg_passwd_inp==''):         #checks if inputs are not empty
or contains spaces
                                if b not in users:
                                        if '@' in reg_email_inp and '.' in reg_email_inp:
                                                if len(reg_num_inp) == 10:
                                                        regsql='insert into users values(%s,%s,%s,%s,
%s,%s)'

        regval=(uuid,reg_fname_inp,reg_email_inp,reg_num_inp,reg_uname_inp,reg_passwd_inp)

                                                        cur.execute(regsql,regval)
                                                        con.commit()

                                                        messagebox.showinfo('','The new user
'+reg_fname_inp+'\nhas been successfully registered.',parent=regwin)
                                                        regwin.destroy()
                                                else:
```

```
                                                messagebox.showerror('Error','Invalid phone
number entered.',parent=regwin)
                                        else:
                                                messagebox.showerror('Error','Invalid electronic mail
ID entered.',parent=regwin)
                                else:
                                        messagebox.showerror('Error','Username '+reg_uname_inp+'\
nalready exists.',parent=regwin)

                        else:
                                messagebox.showerror('Error','Please do not leave any fields
blank.',parent=regwin)

                regwin=tk.Toplevel()
                regwin.title('Register')
                regwin.resizable(False, False)



tk.Label(regwin,text='Register',font=h1fnt).grid(column=0,row=0,padx=10,pady=10,columnspan=2,sticky=
tk.EW)

                tk.Label(regwin,text='ID',font=fnt).grid(column=0,row=3,sticky=tk.E,padx=10,pady=10)
                tk.Label(regwin,text=uuid,font=fnt).grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

                tk.Label(regwin,text='1. Personal
info',font=fntit).grid(column=0,row=5,sticky=tk.W,padx=10,pady=10)


        tk.Label(regwin,text='Name',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
                reg_fname=tk.Entry(regwin,font=fnt)
                reg_fname.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

                tk.Label(regwin,text='Electronic mail
ID',font=fnt).grid(column=0,row=7,sticky=tk.E,padx=10,pady=10)
                reg_email=tk.Entry(regwin,font=fnt)
                reg_email.grid(column=1,row=7,sticky=tk.EW,padx=10,pady=10)

                tk.Label(regwin,text='Phone
number',font=fnt).grid(column=0,row=8,sticky=tk.E,padx=10,pady=10)
                reg_num=tk.Entry(regwin,font=fnt)
                reg_num.grid(column=1,row=8,sticky=tk.EW,padx=10,pady=10)

                tk.Label(regwin,text='2. Login
info',font=fntit).grid(column=0,row=10,sticky=tk.W,padx=10,pady=10)


        tk.Label(regwin,text='Username',font=fnt).grid(column=0,row=11,sticky=tk.E,padx=10,pady=10)
                reg_uname=tk.Entry(regwin,font=fnt)
                reg_uname.grid(column=1,row=11,sticky=tk.EW,padx=10,pady=10)


        tk.Label(regwin,text='Password',font=fnt).grid(column=0,row=12,sticky=tk.E,padx=10,pady=10)
                reg_passwd=tk.Entry(regwin,show='*',font=fnt)
                reg_passwd.grid(column=1,row=12,sticky=tk.EW,padx=10,pady=10)

                regsubmit=tk.Button(regwin,text='Register',command=reguser,font=fntit)
                regsubmit.grid(column=1,row=14,padx=10,pady=10,sticky=tk.W)
                regwin.bind('<Return>',lambda event:reguser())
        #Opens manage profile window
        def manage_profile():
                logwin.destroy()
                manage_user_profile()

        #Window
        tk.Grid.columnconfigure(logwin,0,weight=1)

        #FRAME 1
        tk.Grid.rowconfigure(logwin,0,weight=1)
        f1=tk.Frame(logwin,bg='#283593')
        f1.grid(row=0,column=0,sticky=tk.NSEW)

        #frame 1 grid
        tk.Grid.columnconfigure(f1,0,weight=1)
```

```
        tk.Grid.rowconfigure(f1,0,weight=1)
        tk.Label(f1,text='Login',font=h1fnt,fg='white',bg='#283593').grid(column=0,row=0)
        Separator(f1,orient='horizontal').grid(row=1,column=0,sticky=tk.EW,padx=10,pady=10)

        #FRAME 2
        tk.Grid.rowconfigure(logwin,1,weight=1)
        f2=tk.Frame(logwin)
        f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

        #frame 2 grid
        tk.Grid.columnconfigure(f2,0,weight=1)
        tk.Grid.columnconfigure(f2,1,weight=1)

        tk.Label(f2,text='Username',font=fnt).grid(column=0,row=3,padx=10,pady=10,sticky=tk.E)
        login_uname=tk.Entry(f2,font=fnt)
        login_uname.grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

        tk.Label(f2,text='Password',font=fnt).grid(column=0,row=4,padx=10,pady=10,sticky=tk.E)
        login_passwd=tk.Entry(f2,show='*',font=fnt)
        login_passwd.grid(column=1,row=4,sticky=tk.W,padx=10,pady=10)

        img1=tk.PhotoImage(file='icons/login.png')
        logsubmit=tk.Button(f2,text='Login ... ',image=img1,command=onlogin)
        logsubmit.grid(column=1,row=10,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,12,weight=2)
        tk.Label(f2,text='New here?\nClick here to
register.',font=fntit,justify=tk.RIGHT,fg='#283593').grid(column=0,row=12,padx=10,pady=10,sticky=tk.
E)
        img2=tk.PhotoImage(file='icons/adduser.png')
        reg=tk.Button(f2,text='Register',image=img2,command=register)
        reg.grid(column=1,row=12,padx=10,pady=10,sticky=tk.W)

        manage=tk.Button(f2,text='Manage your profile ... ',font=fntit,command=manage_profile)
        manage.grid(column=1,row=11,padx=10,pady=10,columnspan=2,sticky=tk.W)

        logwin.bind('<Return>',lambda event:onlogin())
        logwin.mainloop()

def manage_user_profile():            #manages profile
        import tkinter as tk
        from tkinter import messagebox
        import mysql.connector as ms
        from tkinter.ttk import Separator
        import os
        import platform as pf
        import ctypes

        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
        cur=con.cursor()

        #Enables DPI scaling on supported Windows versions
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass

        #init GUI
        logwin=tk.Tk()
        logwin.title('Manage profile')


        try:
                logwin.state('zoomed')
        except:
                w,h=logwin.winfo_screenwidth(),logwin.winfo_screenheight()
                logwin.geometry(str(w)+'x'+str(h))


        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)

        def bookings_login():
```

```
                logwin.destroy()
                bookings()


        def onlogin():
                def manage(): #Manage user window

                        def delete():  #Delete function
                                def deluser(): #Delete user from DB
                                        cur.execute('select uname,passwd from users')
                                        b=cur.fetchall()
                                        upass=dict(b)
                                        p=del_passwd.get()
                                        if not p=='' and not p.isspace():
                                                if p == upass[uname_inp]:
                                                        confirm=messagebox.askyesno('','Really delete
your user profile?',parent=delwin)
                                                        if confirm==True:
                                                                sql="delete from users where uname =%s"
                                                                val=(uname_inp,)
                                                                cur.execute(sql,val)
                                                                con.commit()
                                                                messagebox.showinfo('','Username
'+uname_inp+' deleted.\nYou will be returned to the start page.',parent=delwin)
                                                                delwin.destroy()
                                                                manage_userswin.destroy()
                                                                os.system('python3 start.py')
                                                        else:
                                                                pass
                                                else:
                                                        messagebox.showerror('Error','Invalid password
entered.',parent=delwin)
                                        else:
                                                messagebox.showerror('Error','Please enter a
password.',parent=delwin)

                                delwin=tk.Toplevel()
                                delwin.title('Delete User')
                                delwin.resizable(False,False)
                                tk.Label(delwin,text='Delete
User',font=h1fnt).grid(column=0,row=0,padx=10,pady=10)

                                tk.Label(delwin,text='Please enter the
password.',font=fnt).grid(column=0,row=4,sticky=tk.W,padx=10,pady=10)


del_passwd=tk.Entry(delwin,show='*',font=fnt);del_passwd.grid(column=0,row=5,sticky=tk.EW,padx=10,pa
dy=10)

                                delsubmit=tk.Button(delwin,text='Delete
User',command=deluser,font=fntit,fg='red');delsubmit.grid(column=0,row=6,padx=10,pady=10)
                                delwin.bind('<Return>',lambda event:deluser())

                        def passwd():  #Change password function
                                def chpasswd():        #Changes passwd in DB.
                                        cur.execute('select uname,passwd from users')
                                        b=cur.fetchall()
                                        upass=dict(b)
                                        op=old_pass.get()
                                        np=new_pass.get()
                                        if (not np=='' and not np.isspace()) and (not op=='' and not
op.isspace()):
                                                if op == upass[uname_inp]:
                                                        confirm=messagebox.askyesno('','Really change
your password?',parent=passwin)
                                                        if confirm==True:
                                                                sql="update users set passwd=%s where
uname=%s"
                                                                val=(np,uname_inp)
                                                                cur.execute(sql,val)
                                                                con.commit()
                                                                messagebox.showinfo('','Password
updated.',parent=passwin)
                                                                passwin.destroy()
                                                        else:
```

```
                                                        pass
                                                else:
                                                        messagebox.showerror('Error','Invalid old
password entered.',parent=passwin)
                                        else:
                                                messagebox.showerror('Error','Please do not leave any
fields blank.',parent=passwin)

                                passwin=tk.Toplevel()
                                passwin.title('Change Password')
                                passwin.resizable(False,False)

                                tk.Label(passwin,text='Changing password for
'+fnamelist[uname_inp],font=('IBM Plex Sans',18)).grid(column=1,row=0,padx=10,pady=10)

                                tk.Label(passwin,text='Current
Password',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)


old_pass=tk.Entry(passwin,show='*',font=fnt);old_pass.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=
10)

                                tk.Label(passwin,text='New
Password',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)


new_pass=tk.Entry(passwin,show='*',font=fnt);new_pass.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=
10)

                                passsubmit=tk.Button(passwin,text='Change
password',command=chpasswd,font=fntit)
                                passsubmit.grid(column=1,row=10,padx=10,pady=10,sticky=tk.W)
                                passwin.bind('<Return>',lambda event:chpasswd())

                        def logout():  #Logs out
                                manage_userswin.destroy()
                                os.system('python3 start.py')

                        def info():            #Changes personal information.
                                chinfo_home=tk.Toplevel()
                                chinfo_home.resizable(False,False)
                                chinfo_home.title('Change personal information')
                                tk.Label(chinfo_home,text=('Change your\npersonal
information'),font=h1fnt,justify=tk.LEFT).grid(column=1,row=0,padx=10,sticky=tk.W)

                                def name():            #Change full name

                                        def chname():          #Changes name in DB
                                                new_name=en1.get()

                                                if not new_name=='' and not new_name.isspace():
                                                        sql="update users set fname=%s where uname like
%s"
                                                        val=(new_name,uname_inp)

                                                        cur.execute(sql,val)
                                                        con.commit()

                                                        messagebox.showinfo('','Name successfully
changed from '+fnamelist[uname_inp]+' to '+new_name+'.\nPlease log out for any changes to take
effect.',parent=chinfo_name)
                                                        tk.Label(chinfo_name,text='Please log out for\
nany changes to take
effect.',font=fnt,justify=tk.LEFT).grid(row=8,column=1,sticky=tk.W,padx=10,pady=10)
                                                        chinfo_name.destroy()
                                                else:
                                                        messagebox.showerror('','No new name has been
specified.',parent=chinfo_name)
                                        chinfo_name=tk.Toplevel()
                                        chinfo_name.resizable(False,False)
                                        chinfo_name.title('Change display name ... ')
                                        tk.Label(chinfo_name,text=('Change your display
name'),font=h1fnt,justify=tk.LEFT).grid(column=1,row=0,padx=10,sticky=tk.W)
```

```
                                    tk.Label(chinfo_name,text='Current
name',font=fnt).grid(row=5,column=0,sticky=tk.E,padx=10,pady=10)


tk.Label(chinfo_name,text=fnamelist[uname_inp],font=fnt).grid(row=5,column=1,sticky=tk.W,padx=10,pad
y=10)

                                    tk.Label(chinfo_name,text='New
name',font=fnt).grid(row=6,column=0,sticky=tk.E,padx=10,pady=10)
                                    en1=tk.Entry(chinfo_name,font=fnt)
                                    en1.grid(row=6,column=1,sticky=tk.W,padx=10,pady=10)

                                    btn3=tk.Button(chinfo_name,text='Make
changes',font=fntit,command=chname)
                                    btn3.grid(row=10,column=1,padx=10,pady=10,sticky=tk.W)

                                    #Binds Enter key to submit function
                                    chinfo_name.bind('<Return>',lambda event:chname())

                            def contacts():        #Change contact info
                                    def chcontacts():      #Changes email or phone number in DB
                                            new_email=en2.get()
                                            new_num=en3.get()

                                            def conf():
                                                    tk.Label(chinfo_contacts,text='Please log out
for\nany changes to take
effect.',font=fnt,justify=tk.LEFT).grid(row=10,column=1,sticky=tk.W,padx=10,pady=10)
                                                    chinfo_contacts.destroy()

                                            if (not new_num=='' and not new_num.isspace()) or (not
new_email=='' and not new_email.isspace()):
                                                    if new_num=='' or new_num.isspace():
                                                            if '@' in new_email and '.' in
new_email:
                                                                    sql='update users set email=%s
where uname like %s'
                                                                    val=(new_email,uname_inp)
                                                                    cur.execute(sql,val)
                                                                    con.commit()

        messagebox.showinfo('','Electronic mail address changed successfully to '+new_email+'.\
nPlease log out for any changes to take effect.',parent=chinfo_contacts)
                                                                    conf()

                                                            else:

        messagebox.showerror('Error','Invalid electronic mail entered',parent=chinfo_contacts)
                                                    elif new_email=='' or new_email.isspace():
                                                            if len(new_num)==10:
                                                                    sql='update users set num=%s
where uname like %s'
                                                                    val=(new_num,uname_inp)
                                                                    cur.execute(sql,val)
                                                                    con.commit()
                                                                    messagebox.showinfo('','Phone
number changed successfully to '+new_num+'.\nPlease log out for any changes to take
effect.',parent=chinfo_contacts)
                                                                    conf()

                                                            else:

        messagebox.showerror('Error','Invalid phone number entered',parent=chinfo_contacts)
                                                    elif (not new_num=='' and not
new_num.isspace()) and (not new_email=='' and not new_email.isspace()):
                                                            if ('@' in new_email and '.' in
new_email) and (len(new_num)==10):
                                                                    sql='update users set email=%s
where uname like %s'
                                                                    val=(new_email,uname_inp)
                                                                    cur.execute(sql,val)
                                                                    con.commit()


                                                                    sql='update users set num=%s
where uname like %s'
```

```
                                                                    val=(new_num,uname_inp)
                                                                    cur.execute(sql,val)
                                                                    con.commit()

        messagebox.showinfo('','Electronic mail address and phone number changed successfully to
'+new_email+' and '+new_num+', respectively.\nPlease log out for any changes to take
effect.',parent=chinfo_contacts)
                                                                    conf()

                                                                else:

        messagebox.showerror('Error','Invalid electronic mail or phone number
entered',parent=chinfo_contacts)
                                                    else:
                                                            messagebox.showerror('Error','Please fill at
least one field.',parent=chinfo_contacts)

                                        cur.execute('select uname,email from users')
                                        a=dict(cur.fetchall())

                                        cur.execute('select uname,num from users')
                                        b=dict(cur.fetchall())

                                        chinfo_contacts=tk.Toplevel()
                                        chinfo_contacts.resizable(False,False)
                                        chinfo_contacts.title('Change contact details ... ')
                                        tk.Label(chinfo_contacts,text=('Change your contact
details'),font=h1fnt,justify=tk.LEFT).grid(column=1,row=0,padx=10,sticky=tk.W)
                                        tk.Label(chinfo_contacts,text='If you do not wish to change a\
nparticular contact, then leave the\ncorresponding field
blank.',font=fnt,justify=tk.LEFT).grid(row=2,column=1,sticky=tk.W,padx=10,pady=10)
                                        tk.Label(chinfo_contacts,text='Current\nelectronic mail
address',font=fnt,justify=tk.RIGHT).grid(row=5,column=0,sticky=tk.E,padx=10,pady=10)


tk.Label(chinfo_contacts,text=a[uname_inp],font=fnt).grid(row=5,column=1,sticky=tk.W,padx=10,pady=10
)

                                        tk.Label(chinfo_contacts,text='New\nelectronic mail
address',font=fnt,justify=tk.RIGHT).grid(row=6,column=0,sticky=tk.E,padx=10,pady=10)
                                        en2=tk.Entry(chinfo_contacts,font=fnt)
                                        en2.grid(row=6,column=1,sticky=tk.EW,padx=10,pady=10)

                                        tk.Label(chinfo_contacts,text='Current phone
number',font=fnt).grid(row=7,column=0,sticky=tk.E,padx=10,pady=10)


tk.Label(chinfo_contacts,text=b[uname_inp],font=fnt).grid(row=7,column=1,sticky=tk.W,padx=10,pady=10
)

                                        tk.Label(chinfo_contacts,text='New phone
number',font=fnt).grid(row=8,column=0,sticky=tk.E,padx=10,pady=10)
                                        en3=tk.Entry(chinfo_contacts,font=fnt)
                                        en3.grid(row=8,column=1,sticky=tk.EW,padx=10,pady=10)

                                        btn3=tk.Button(chinfo_contacts,text='Make
changes',font=fntit,command=chcontacts)
                                        btn3.grid(row=15,column=1,padx=10,pady=10,sticky=tk.W)

                                        #Binds Enter key to submit function
                                        chinfo_contacts.bind('<Return>',lambda event:chcontacts())

                                img1=tk.PhotoImage(file='icons/user.png')
                                btn1=tk.Button(chinfo_home,text='Name',image=img1,command=name)
                                btn1.image=img1
                                btn1.grid(column=0,row=3,padx=10,pady=10,sticky=tk.E)
                                tk.Label(chinfo_home,text='Change your display
name',font=fnt,justify=tk.LEFT).grid(column=1,row=3,padx=10,pady=10,sticky=tk.W)

                                img2=tk.PhotoImage(file='icons/contacts-2.png')
                                btn2=tk.Button(chinfo_home,text='Contact',image=img2,command=contacts)
                                btn2.image=img2
                                btn2.grid(column=0,row=6,padx=10,pady=10,sticky=tk.E)
                                tk.Label(chinfo_home,text='Change your contact
details',font=fnt).grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)
```

```
                        cur.execute('select uname,uuid from users')
                        uuidlist=dict(cur.fetchall())
                        cur.execute('select uname,fname from users')
                        fnamelist=dict(cur.fetchall())

                        logwin.destroy()

                        manage_userswin=tk.Tk()
                        manage_userswin.title('Manage profile')
                        try:
                                manage_userswin.state('zoomed')
                        except:

        w,h=manage_userswin.winfo_screenwidth(),manage_userswin.winfo_screenheight()
                                manage_userswin.geometry(str(w)+'x'+str(h))

                        tk.Grid.columnconfigure(manage_userswin,0,weight=1)

                        #FRAME 1
                        tk.Grid.rowconfigure(manage_userswin,0,weight=1)
                        f1=tk.Frame(manage_userswin,bg='#283593')
                        f1.grid(row=0,column=0,sticky=tk.NSEW)

                        #frame 1 grid
                        tk.Grid.columnconfigure(f1,0,weight=1)

                        tk.Grid.rowconfigure(f1,0,weight=1)
                        tk.Grid.rowconfigure(f1,1,weight=1)
                        tk.Grid.rowconfigure(f1,2,weight=1)
                        tk.Grid.rowconfigure(f1,3,weight=1)

                        logo_img=tk.PhotoImage(file='img/logo.png')
                        logo=tk.Label(f1,image=logo_img,fg='white',bg='#283593')
                        logo.grid(column=0,row=0,padx=10,pady=10,sticky=tk.EW)
                        logo.image=logo_img

                        tk.Label(f1,text=('Welcome,
'+fnamelist[uname_inp]),font=h1fnt,fg='white',bg='#283593').grid(column=0,row=1,padx=10,sticky=tk.EW
)
                        tk.Label(f1,text=('ID: '+uuidlist[uname_inp]),font=('IBM Plex
Sans',12),fg='black',bg='#00e676').grid(column=0,row=2,padx=10)
                        tk.Label(f1,text=('Manage your profile'),font=('IBM Plex
Sans',12),fg='white',bg='#283593').grid(column=0,row=3,padx=10,sticky=tk.EW)



Separator(f1,orient='horizontal').grid(column=0,row=4,sticky=tk.EW,padx=10,pady=10,columnspan=2)
                        #FRAME 2
                        tk.Grid.rowconfigure(manage_userswin,1,weight=1)
                        f2=tk.Frame(manage_userswin)
                        f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

                        #frame 2 grid
                        tk.Grid.columnconfigure(f2,0,weight=1)
                        tk.Grid.columnconfigure(f2,1,weight=1)
                        tk.Grid.columnconfigure(f2,2,weight=1)
                        tk.Grid.columnconfigure(f2,3,weight=1)

                        tk.Label(f2,text=('You
can:'),font=fntit).grid(column=1,row=2,padx=10,pady=10,sticky=tk.W)

                        tk.Grid.rowconfigure(f2,5,weight=1)
                        img4=tk.PhotoImage(file='icons/passwd.png')
                        passbtn=tk.Button(f2,text='Change Password',image=img4,command=passwd)
                        passbtn.grid(column=0,row=5,padx=10,pady=10,sticky=tk.E)
                        tk.Label(f2,text='Change your
password.',font=fnt).grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

                        img9=tk.PhotoImage(file='icons/user.png')
                        delusrbtn=tk.Button(f2,text='Manage Personal Info',image=img9,command=info)
                        delusrbtn.grid(column=2,row=5,padx=10,pady=10,sticky=tk.E)
                        tk.Label(f2,text='Change your personal
information.',font=fnt,fg='green').grid(column=3,row=5,padx=10,pady=10,sticky=tk.W)
```

```
                        tk.Grid.rowconfigure(f2,6,weight=1)
                        img3=tk.PhotoImage(file='icons/ban_user.png')
                        delusrbtn=tk.Button(f2,text='Remove User',image=img3,command=delete)
                        delusrbtn.grid(column=0,row=6,padx=10,pady=10,sticky=tk.E)
                        tk.Label(f2,text='Delete your
profile.',font=fnt,fg='red').grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)

                        tk.Label(f2,text=('or:'),font=fntit).grid(column=1,row=9,padx=10,sticky=tk.W)

                        tk.Grid.rowconfigure(f2,11,weight=1)
                        img7=tk.PhotoImage(file='icons/logout.png')
                        logoutbtn=tk.Button(f2,text='Logout',font=fnt,image=img7,command=logout)
                        logoutbtn.grid(column=0,row=11,padx=10,pady=10,sticky=tk.E)

        tk.Label(f2,text='Logout',font=fnt).grid(column=1,row=11,padx=10,pady=10,sticky=tk.W)

                        img8=tk.PhotoImage(file='icons/close.png')
                        exitbtn=tk.Button(f2,text='Logout and
exit',font=fnt,image=img8,command=manage_userswin.destroy)
                        exitbtn.grid(column=2,row=11,padx=10,pady=10,sticky=tk.E)
                        tk.Label(f2,text='Logout and
exit',font=fnt,fg='red').grid(column=3,row=11,padx=10,pady=10,sticky=tk.W)

                        manage_userswin.mainloop()

                uname_inp=login_uname.get()
                passwd_inp=login_passwd.get()
                cur.execute('select uname,passwd from users')
                op=dict(cur.fetchall())

                cur.execute('select uname,fname from users')
                fnamelist=dict(cur.fetchall())

                if (not uname_inp=='' and not uname_inp.isspace()) and (not passwd_inp=='' and not
passwd_inp.isspace()):
                        if uname_inp not in op.keys():
                                messagebox.showerror('Error','Username \''+uname_inp+'\' does not
exist.')
                        else:
                                if not passwd_inp == op[uname_inp]:
                                        messagebox.showerror('Error','Invalid password entered for
'+fnamelist[uname_inp]+'.')
                                else:
                                        manage()
                else:
                        messagebox.showerror('Error','Please do not leave any fields blank.')

        tk.Grid.columnconfigure(logwin,0,weight=1)

        #FRAME 3
        tk.Grid.rowconfigure(logwin,0,weight=1)
        f3=tk.Frame(logwin,bg='#283593')
        f3.grid(row=0,column=0,sticky=tk.NSEW)

        #frame 3 grid
        tk.Grid.columnconfigure(f3,0,weight=1)

        tk.Grid.rowconfigure(f3,0,weight=1)
        tk.Label(logwin,text='Login',font=h1fnt,fg='white',bg='#283593').grid(column=0,row=0)
        Separator(f3,orient='horizontal').grid(row=1,column=0,sticky=tk.EW,padx=10,pady=10)

        #FRAME 4
        tk.Grid.rowconfigure(logwin,1,weight=1)
        f4=tk.Frame(logwin)
        f4.grid(row=1,column=0,sticky=tk.NSEW,padx=10,pady=10)

        #frame 4 grid
        tk.Grid.columnconfigure(f4,0,weight=1)
        tk.Grid.columnconfigure(f4,1,weight=1)

        tk.Label(f4,text='Username',font=fnt).grid(column=0,row=3,padx=10,pady=10,sticky=tk.E)
        login_uname=tk.Entry(f4,font=fnt)
        login_uname.grid(column=1,row=3,padx=10,pady=10,sticky=tk.W)

        tk.Label(f4,text='Password',font=fnt).grid(column=0,row=4,padx=10,pady=10,sticky=tk.E)
```

```
        login_passwd=tk.Entry(f4,show='*',font=fnt)
        login_passwd.grid(column=1,row=4,padx=10,pady=10,sticky=tk.W)

        img1=tk.PhotoImage(file='icons/login.png')
        logsubmit=tk.Button(f4,text='Login ... ',image=img1,command=onlogin)
        logsubmit.grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f4,6,weight=1)
        tk.Label(f4,text='Want to make bookings?\nClick here to
continue.',font=fntit,justify=tk.RIGHT,bg='#00e676').grid(column=0,row=6,padx=10,pady=10,sticky=tk.E
)
        img6=tk.PhotoImage(file='icons/booking.png')
        bkgbtn=tk.Button(f4,text='Booking',image=img6,font=fnt,command=bookings_login)
        bkgbtn.grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)

        #Binds enter key to login function
        logwin.bind('<Return>',lambda event:onlogin())
        logwin.mainloop()
```

3. bookings.py

```
def bus():
        #import statements
        import mysql.connector as ms
        import random as rd
        import tkinter as tk
        from tkinter import ttk
        from tkinter import scrolledtext
        from tkinter.ttk import Separator
        from tkinter import messagebox
        from datetime import datetime,timedelta
        import platform as pf
        import ctypes

        #Enables DPI scaling on supported Windows versions
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass

        #definitions

        id='B'+str(rd.randint(10000,99999))   #random number for ID

locations=['Blackcastle','Westerwitch','Ironlyn','Wellsummer','Meadowynne','Aldcourt','Butterhaven',
'Winterglass','Northcrest','Mallowdell']     #defines locations
        ctype=['','Standard','Express','Premium']    #defines coach type

        #mysql connection
        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
        cur=con.cursor()

        #GUI
        window=tk.Toplevel()
        #fonts for GUI
        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)
        #Main Window parameters
        window.title('Bus Booking')
        window.resizable(False, False)

        def payment(): #Payment window

                #Taking of inputs
                start_inp=start.get().capitalize()
                end_inp=end.get().capitalize()
                date_inp=date.get()
                time_inp=time.get()
                bustype_inp=n.get()
                passno=q.get()

                format='%Y-%m-%d %H:%M'        #datetime format
                current_ts=datetime.now()+timedelta(minutes=45)      #timestamp for reference - 45
min from current time

                ts_str=current_ts.strftime(format)    #Converts datetime to string in specific time
format (YYYY-MM-DD HH:MM; MySQL datetime format)

                ts=datetime.strptime(ts_str,format)          #Converts string back to datetime object
for comparision


                y=date_inp+' '+time_inp               #Combines date and time inputs into correct
format for comparision purpose


                d_res=True
                try:
                        d_res=bool(datetime.strptime(y,format))  #Is date and time inputted in format?
```

```
                except ValueError:
                        d_res=False

                if d_res==True:
                        x=datetime.strptime(y,format) #Converts input to datetime for comparision

                        if x ≥ ts:            #Is y not before minimum 45min from now?
                                isNotPast=True
                        else:
                                isNotPast=False

                        if x ≤ ts+timedelta(days=1096):             # 3-year limit on dates entered
                                isNotDistFuture=True
                        else:
                                isNotDistFuture=False

                #Checking of inputs before proceeding to payment
                if (not start_inp=='' and not start_inp.isspace()) and (not end_inp=='' and not
end_inp.isspace()) and (not date_inp=='' and not date_inp.isspace()) and (not time_inp=='' and not
time_inp.isspace()) and (not bustype_inp=='' and not bustype_inp.isspace()):
                        if start_inp in locations and end_inp in locations:
                                if not start_inp == end_inp:
                                        if d_res==True and len(date_inp)==10 and len(time_inp)==5:
                                                if isNotPast==True and isNotDistFuture==True:
                                                        pay_win=tk.Toplevel()
                                                        pay_win.title('Payment Gateway')
                                                        pay_win.resizable(False,False)

                                                        def make_payment():           #Payment function

                                                                #Takes inputs of payment details
                                                                paytype_inp=m.get()
                                                                cardno_inp=card_no.get()
                                                                cardname_inp=card_name.get()
                                                                expyear_inp=exp_year.get()
                                                                expmonth_inp=exp_month.get()
                                                                cvv_inp=cvv_no.get()

                                                                #Gets current month and year for expiry
date checking
                                                                x=datetime.now()
                                                                cmonth=x.month
                                                                cyear=x.year


                                                                def pay():
                                                                        def submit():  #Adds booking to
DB
                                                                                #timestamp to mark
bookings
                                                                                t=datetime.now()
                                                                                today=t.strftime('%Y-%m-
%d %H:%M:%S')  #Converts ts to string in MySQL datetime format for insertion into db - YYYY-MM-DD
HH:MM

                                                                                sql='insert into bus_bkgs
values (%s,%s,%s,%s,%s,%s,%s,%s)'

        val=(id,today,passno,start_inp,end_inp,date_inp,time_inp,bustype_inp)
                                                                                cur.execute(sql,val)
                                                                                con.commit()

                                                                                #Confirmation message

        submit_message=tk.Toplevel()

        submit_message.resizable(False,False)

        submit_message.title('Booking successful')


        tk.Label(submit_message,text='The booking has been\nsuccessfully
made.',font=h1fnt,justify=tk.LEFT).grid(row=0,column=0,sticky=tk.W,padx=10,pady=10)

                                                                                cardtype=''
                                                                                if cardno_inp[0] == '3':
```

```
                                                                        cardtype='AMEX'
                                                                elif cardno_inp[0] ==
'4':
                                                                        cardtype='VISA'
                                                                elif cardno_inp[0] ==
'5':
                                                                        cardtype='MASTER'
                                                                elif cardno_inp[0] ==
'6':

        cardtype='DISCOVER'


        booking_summary=scrolledtext.ScrolledText(submit_message,font=fnt,width=30,height=8)

        booking_summary.grid(column=0,row=3,sticky=tk.EW,padx=10,pady=10,columnspan=2)

                                                                text2='Bus Booking\
n——————————\n\nBooking ID: '+id+'\nBooking Timestamp: \n'+today+'\n\nFrom: '+o+'\nTo: '+d+'\nType:
'+n.get()+'\n\nDate: '+date_inp+'\nTime: '+time_inp+'\n\nNumber of passengers: '+str(passno)+'\n\
nTotal fare: $'+str(total_fare)+'\n\nPayment\n———————\n\n'+'Payment ID: '+payment_id+'\nPaid by:
'+m.get()+'\nCardholder name: '+card_name.get()+'\nCard number: XXXX-XXXX-XXXX-'+card_no.get()[-
4:]+'\nCard type: '+cardtype+'\nAmount paid: $'+str(total_fare)+'\n\n——————————————————'+'\nPAYMENT
SUCCESSFUL'+'\n——————————————————'

        booking_summary.insert(tk.INSERT,text2)

        booking_summary.configure(state='disabled')

                                                                def clipboard():

        submit_message.clipboard_clear()

        submit_message.clipboard_append(text2)

        btn1.configure(fg='green',text='Copied!')


        btn1=tk.Button(submit_message,text='Copy to
clipboard',font=fnt,command=clipboard,justify=tk.CENTER)

        btn1.grid(row=5,column=0,padx=10,pady=10)


        tk.Label(submit_message,text='The e-receipt will also be sent to\nyour registered electronic
mail\naddress.',font=fnt,justify=tk.LEFT).grid(row=6,column=0,padx=10,pady=10,sticky=tk.W)

                                                                def exit():

        submit_message.destroy()
                                                                        pay_win.destroy()
                                                                        window.destroy()


        btn2=tk.Button(submit_message,text='OK',font=fnt,command=exit,justify=tk.CENTER)

        btn2.grid(row=8,column=0,padx=10,pady=10)

        submit_message.bind('<Return>',lambda event:exit())

                                                                #timestamp to mark bookings
                                                                t=datetime.now()
                                                                today=t.strftime('%Y-%m-%d %H:
%M:%S') #Converts ts to string in MySQL datetime format for insertion into db - YYYY-MM-DD HH:MM

                                                                sql=('insert into
payment_details values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)')


val=(payment_id,today,id,total_fare,paytype_inp,cardno_inp,cardname_inp,cvv_inp,expmonth_inp,expyear
_inp)
                                                                cur.execute(sql,val)
                                                                con.commit()

                                                                submit()
```

```
                                                        #Payment details input checking
                                                        if (not paytype_inp=='' and not
paytype_inp.isspace()) and (not cardno_inp=='' and not cardno_inp.isspace()) and (not
cardname_inp=='' and not cardname_inp.isspace()) and (not expyear_inp=='' and not
expyear_inp.isspace()) and (not expmonth_inp=='' and not expmonth_inp.isspace()) and (not
cvv_inp=='' and not cvv_inp.isspace()):
                                                                if len(cardno_inp) == 16 and
cardno_inp[0] in '3456':
                                                                        if len(expyear_inp) == 4
and int(expyear_inp) ≥ cyear:
                                                                                if
int(expyear_inp) == cyear:
                                                                                        if
(len(expmonth_inp) == 2) and (int(expmonth_inp)≥ 1 and int(expmonth_inp) ≤ 12) and
(int(expmonth_inp) > cmonth):
                                                                                                if
len(cvv_inp)==3:

        pay()

        else:

        messagebox.showerror('Error','CVV must be a 3-digit number.',parent=pay_win)
                                                                                        else:

        messagebox.showerror('Error','Invalid expiry month.',parent=pay_win)
                                                                                elif
int(expyear_inp) > cyear:
                                                                                        if
(len(expmonth_inp) == 2) and (int(expmonth_inp)≥ 1 and int(expmonth_inp) ≤ 12):
                                                                                                if
len(cvv_inp)==3:

        pay()

        else:

        messagebox.showerror('Error','CVV must be a 3-digit number.',parent=pay_win)
                                                                                        else:

        messagebox.showerror('Error','Invalid expiry month.',parent=pay_win)
                                                                                else:
                                                                                        pass
                                                                        else:

        messagebox.showerror('Error','Invalid expiry year.',parent=pay_win)
                                                                else:

        messagebox.showerror('Error','Invalid card number.',parent=pay_win)
                                                        else:

        messagebox.showerror('Error','Please enter all required\npayment details.',parent=pay_win)

                                                f3=tk.Frame(pay_win)
                                                f3.grid(row=0,column=0)

                                                img1=tk.PhotoImage(file='icons/make-
payment.png')
                                                img=tk.Label(f3,image=img1,font=h1fnt)
                                                img.grid(column=0,row=0,padx=10,pady=10)
                                                img.image=img1


tk.Label(f3,text='Payment',font=h1fnt,justify=tk.LEFT).grid(column=1,row=0,padx=10,pady=10,sticky=tk
.W)

                                                f4=tk.Frame(pay_win)
                                                f4.grid(row=1,column=0)


        payment_summary=scrolledtext.ScrolledText(f4,font=fnt,width=25,height=5)
```

```
        payment_summary.grid(column=1,row=2,sticky=tk.EW,padx=10,pady=10)

                                                    if n.get()=='Standard':
                                                           rate=5
                                                    elif n.get()=='Express':
                                                           rate=10
                                                    elif n.get()=='Premium':
                                                           rate=15

                                                    o=start.get()
                                                    d=end.get()
                                                    distance=abs((locations.index(d))-
(locations.index(o)))*4       #distance between locations - 4 km.
                                                    total_fare=(rate*distance)*passno

                                                    text='Booking ID: '+id+'\nFrom: '+o+'\nTo:
'+d+'\nType: '+n.get()+'\n\nDate: '+date_inp+'\nTime: '+time_inp+'\n\nRate: $'+str(rate)+' per km\
nDistance: '+str(distance)+' km\nNumber of passengers: '+str(passno)+'\n\nTotal fare:
$'+str(total_fare)
                                                    payment_summary.insert(tk.INSERT,text)
                                                    payment_summary.configure(state='disabled')

                                                    tk.Label(f4,text='Payment
ID',font=fnt).grid(column=0,row=3,sticky=tk.E,padx=10,pady=10)
                                                    payment_id='P'+str(rd.randint(10000,99999))
                                                    payid=tk.Label(f4,text=payment_id,font=fnt)

        payid.grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

                                                    m=tk.StringVar()
                                                    tk.Label(f4,text='Pay
by',font=fnt).grid(column=0,row=4,sticky=tk.E,padx=10,pady=10)
                                                    card=('','Debit card','Credit card')
                                                    pay_type=ttk.OptionMenu(f4,m,*card)

        pay_type.grid(column=1,row=4,sticky=tk.W,padx=10,pady=10)

                                                    tk.Label(f4,text='Accepted
cards',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)

                                                    img2=tk.PhotoImage(file='img/cards.png')
                                                    card_image=tk.Label(f4,image=img2,font=fnt)

        card_image.grid(column=1,row=5,sticky=tk.W,padx=10,pady=10)
                                                    card_image.image=img2

                                                    tk.Label(f4,text='Card
number',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
                                                    card_no=tk.Entry(f4,font=fnt)

        card_no.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

                                                    tk.Label(f4,text='Cardholder
name',font=fnt).grid(column=0,row=7,sticky=tk.E,padx=10,pady=10)
                                                    card_name=tk.Entry(f4,font=fnt)

        card_name.grid(column=1,row=7,sticky=tk.EW,padx=10,pady=10)

                                                    tk.Label(f4,text='Expiry Year and Month\n[YYYY-
MM]',font=fnt,justify=tk.RIGHT).grid(column=0,row=8,sticky=tk.E,padx=10,pady=10)
                                                    exp_year=tk.Entry(f4,font=fnt,width=10)

        exp_year.grid(column=1,row=8,sticky=tk.EW,padx=10,pady=10)


        tk.Label(f4,text='-',font=fnt).grid(column=2,row=8,sticky=tk.EW,padx=10,pady=10)
                                                    exp_month=tk.Entry(f4,font=fnt,width=10)

        exp_month.grid(column=3,row=8,sticky=tk.W,padx=10,pady=10)

                                                    tk.Label(f4,text='CVV
number',font=fnt).grid(column=0,row=9,sticky=tk.E,padx=10,pady=10)
                                                    cvv_no=tk.Entry(f4,font=fnt)
```

```
        cvv_no.grid(column=1,row=9,sticky=tk.EW,padx=10,pady=10)


btn=tk.Button(f4,font=fntit,text='Pay',command=make_payment,fg='green');btn.grid(column=1,row=10,pad
x=10,pady=10,sticky=tk.W)

                                                retimg=tk.PhotoImage(file='icons/return.png')

        btn4=tk.Button(f4,font=fnt,image=retimg,command=pay_win.destroy)

        btn4.grid(column=0,row=15,padx=10,pady=10,sticky=tk.SW)
                                                btn4.img=retimg

                                                pay_win.bind('<Return>',lambda
event:make_payment())

                                            else:
                                                messagebox.showerror('Error','Invalid timing
entered.',parent=window)
                                    else:
                                                messagebox.showerror('Error','Invalid date or time
format entered.',parent=window)
                                else:
                                        messagebox.showerror('Error','The origin and destination are
the same.',parent=window)
                        else:
                                messagebox.showerror('Error','Invalid origin or
destination.',parent=window)
                    else:
                        messagebox.showerror('Error','Please do not leave any fields
blank.',parent=window)
        #FRAME 1
        f1=tk.Frame(window)
        f1.grid(row=0,column=0)

        tk.Label(f1,text='BUS BOOKING',font=h1fnt,fg='blue').grid(column=1,row=0,padx=10,pady=10)

        #FRAME 2
        f2=tk.Frame(window)
        f2.grid(row=1,column=0)

        #Booking ID
        tk.Label(f2,text='ID',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)
        bkgid=tk.Label(f2,text=id,font=fnt)
        bkgid.grid(column=1,row=5,sticky=tk.W,padx=10,pady=10)

        #Input fields
        tk.Label(f2,text='Number of
passengers',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
        q=tk.IntVar()
        pass_no=tk.Scale(f2,from_=1,to=100,orient='horizontal',variable=q,font=fnt)
        pass_no.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

        n=tk.StringVar()
        tk.Label(f2,text='Bus Type',font=fnt).grid(column=0,row=7,sticky=tk.E,padx=10,pady=10)
        bustype=ttk.OptionMenu(f2,n,*ctype)
        bustype.grid(column=1,row=7,sticky=tk.W,padx=10,pady=10)

        tk.Label(f2,text='From',font=fnt).grid(column=0,row=8,sticky=tk.E,padx=10,pady=10)
        l=tk.StringVar()
        start=ttk.Combobox(f2,textvariable=l,font=fnt,width=19,state='readonly')
        start.grid(column=1,row=8,sticky=tk.EW,padx=10,pady=10)
        start['values']=locations

        tk.Label(f2,text='To',font=fnt).grid(column=0,row=9,sticky=tk.E,padx=10,pady=10)
        m=tk.StringVar()
        end=ttk.Combobox(f2,textvariable=m,font=fnt,width=19,state='readonly')
        end.grid(column=1,row=9,sticky=tk.EW,padx=10,pady=10)
        end['values']=locations

        tk.Label(f2,text='Date',font=fnt).grid(column=0,row=10,sticky=tk.E,padx=10,pady=10)
        date=tk.Entry(f2,font=fnt)
        date.grid(column=1,row=10,sticky=tk.EW,padx=10,pady=10)
```

```
        tk.Label(f2,text='[YYYY-MM-DD]',font=fnt).grid(column=2,row=10,padx=10,pady=10)

        tk.Label(f2,text='Time',font=fnt).grid(column=0,row=11,sticky=tk.E,padx=10,pady=10)
        time=tk.Entry(f2,font=fnt)
        time.grid(column=1,row=11,sticky=tk.EW,padx=10,pady=10)
        tk.Label(f2,text='24h [HH:MM]',font=fnt).grid(column=2,row=11,padx=10,pady=10)

        Separator(f2,orient='horizontal').grid(row=14,column=0,columnspan=3,sticky=tk.EW)

        tk.Label(f2,text='Proceed to
checkout',font=fnt,justify=tk.RIGHT).grid(column=0,row=15,sticky=tk.E,padx=10,pady=10)
        subimg=tk.PhotoImage(file='icons/checkout.png')
        btn=tk.Button(f2,font=fnt,text='Continue to Payment',image=subimg,command=payment)
        btn.grid(column=1,row=15,padx=10,pady=10,sticky=tk.W)
        btn.image=subimg

        #Binds enter key to submit function
        window.bind('<Return>',lambda event:payment())

def taxi():
        #import statements
        import mysql.connector as ms
        import random as rd
        from tkinter.ttk import Separator
        import tkinter as tk
        from tkinter import ttk
        from tkinter import scrolledtext
        from tkinter import messagebox
        from datetime import datetime,timedelta
        import ctypes
        import platform as pf

        #Enables DPI scaling on supported Windows versions
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass

        #definitions
        id='T'+str(rd.randint(10000,99999))   #random number for ID

locations=['Blackcastle','Westerwitch','Ironlyn','Wellsummer','Meadowynne','Aldcourt','Butterhaven',
'Winterglass','Northcrest','Mallowdell']     #defines locations
        ctype=['','Standard','XL','Luxury']   #defines coach type

        #timestamp to mark bookings
        t=datetime.now()

        #mysql connection
        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
        cur=con.cursor()


        #GUI
        window=tk.Toplevel()
        #fonts for GUI
        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)
        #Main Window parameters
        window.title('Taxi Booking')
        window.resizable(False, False)

        def payment(): #Payment function

                bkgdate=''
                if 'Today' in p.get():
                        bkgdate=a.strftime('%Y-%m-%d')
                elif 'Tomorrow' in p.get():
                        bkgdate=b.strftime('%Y-%m-%d')
                else:
                        pass
```

```
                start_inp=start.get().capitalize()
                end_inp=end.get().capitalize()
                date_inp=bkgdate
                time_inp=time.get()
                taxitype_inp=n.get()

                format='%Y-%m-%d %H:%M'        #datetime format
                current_ts=datetime.now()+timedelta(minutes=10)      #timestamp for reference - 10
min from current time

                ts_str=current_ts.strftime(format)    #Converts datetime to string in specific time
format (YYYY-MM-DD HH:MM; MySQL datetime format)

                ts=datetime.strptime(ts_str,format)          #Converts string back to datetime object
for comparision


                y=date_inp+' '+time_inp


                d_res=True
                try:
                        d_res=bool(datetime.strptime(y,format))              #Is date and time
inputted in correct format?
                except ValueError:
                        d_res=False

                if d_res==True:
                        x=datetime.strptime(y,format)        #Converts inputs to datetime format

                        if x ≥ ts:                                                  #Is input in the
past?
                                isNotPast=True
                        else:
                                isNotPast=False

                        if x ≤ ts+timedelta(days=2):         #Is input greater than 2 days?
                                isNotDistFuture=True
                        else:
                                isNotDistFuture=False

                if (not start_inp=='' and not start_inp.isspace()) and (not end_inp=='' and not
end_inp.isspace()) and (not date_inp=='' and not date_inp.isspace()) and (not time_inp=='' and not
time_inp.isspace()) and (not taxitype_inp=='' and not taxitype_inp.isspace()):
                        if start_inp in locations and end_inp in locations:
                                if not start_inp == end_inp:
                                        if d_res==True and len(date_inp)==10 and len(time_inp)==5:
                                                if isNotPast==True and isNotDistFuture==True:
                                                        pay_win=tk.Toplevel()
                                                        pay_win.title('Payment Gateway')
                                                        pay_win.resizable(False,False)

                                                        def make_payment():            #Payment window
                                                                paytype_inp=m.get()
                                                                cardno_inp=card_no.get()
                                                                cardname_inp=card_name.get()
                                                                expyear_inp=exp_year.get()
                                                                expmonth_inp=exp_month.get()
                                                                cvv_inp=cvv_no.get()

                                                                x=datetime.now()
                                                                cmonth=x.month
                                                                cyear=x.year

                                                                def pay():             #Makes payment

                                                                        def submit():          #Adds
booking to DB

                                                                                #timestamp to mark
bookings
                                                                                t=datetime.now()
                                                                                today=t.strftime('%Y-%m-
%d %H:%M:%S')  #Converts ts to string in MySQL datetime format for insertion into db - YYYY-MM-DD
HH:MM
```

```
                                                                        sql='insert into
taxi_bkgs values (%s,%s,%s,%s,%s,%s,%s)'

        val=(id,today,start_inp,end_inp,date_inp,time_inp,taxitype_inp)
                                                                        cur.execute(sql,val)
                                                                        con.commit()


        submit_message=tk.Toplevel()

        submit_message.resizable(False,False)

        submit_message.title('Booking successful')


        tk.Label(submit_message,text='The booking has been\nsuccessfully
made.',font=h1fnt,justify=tk.LEFT).grid(row=0,column=0,sticky=tk.W,padx=10,pady=10)

                                                                        if cardno_inp[0] == '3':
                                                                                cardtype='AMEX'
                                                                        elif cardno_inp[0] ==
'4':
                                                                                cardtype='VISA'
                                                                        elif cardno_inp[0] ==
'5':
                                                                                cardtype='MASTER'
                                                                        elif cardno_inp[0] ==
'6':

        cardtype='DISCOVER'


        booking_summary=scrolledtext.ScrolledText(submit_message,font=fnt,width=30,height=8)

        booking_summary.grid(column=0,row=3,sticky=tk.EW,padx=10,pady=10,columnspan=2)

                                                                        text2='Taxi Booking\
n————————————\n\nBooking ID: '+id+'\nBooking Timestamp: \n'+today+'\n\nFrom: '+o+'\nTo: '+d+'\nType:
'+n.get()+'\n\nDate: '+date_inp+'\nTime: '+time_inp+'\n\nTotal fare: $'+str(total_fare)+'\n\
nPayment\n———————\n\n'+'Payment ID: '+payment_id+'\nPaid by: '+m.get()+'\nCardholder name:
'+card_name.get()+'\nCard number: XXXX-XXXX-XXXX-'+card_no.get()[-4:]+'\nCard type: '+cardtype+'\
nAmount paid: $'+str(total_fare)+'\n\n——————————————————'+'\nPAYMENT SUCCESSFUL'+'\
n——————————————————'

        booking_summary.insert(tk.INSERT,text2)

        booking_summary.configure(state='disabled')

                                                                        def clipboard():

        submit_message.clipboard_clear()

        submit_message.clipboard_append(text2)

        btn1.configure(fg='green',text='Copied!')


        btn1=tk.Button(submit_message,text='Copy to
clipboard',font=fnt,command=clipboard,justify=tk.CENTER)

        btn1.grid(row=5,column=0,padx=10,pady=10)


        tk.Label(submit_message,text='The e-receipt will also be sent to\nyour registered electronic
mail\naddress.',font=fnt,justify=tk.LEFT).grid(row=6,column=0,padx=10,pady=10,sticky=tk.W)

                                                                        def exit():

        submit_message.destroy()
                                                                                pay_win.destroy()
                                                                                window.destroy()
```

```
        btn2=tk.Button(submit_message,text='OK',font=fnt,command=exit,justify=tk.CENTER)

        btn2.grid(row=8,column=0,padx=10,pady=10)

        submit_message.bind('<Return>',lambda event:exit())


                                                                #timestamp to mark bookings
                                                                t=datetime.now()
                                                                today=t.strftime('%Y-%m-%d %H:
%M:%S') #Converts ts to string in MySQL datetime format for insertion into db - YYYY-MM-DD HH:MM

                                                                sql=('insert into
payment_details values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)')


val=(payment_id,today,id,total_fare,paytype_inp,cardno_inp,cardname_inp,cvv_inp,expmonth_inp,expyear
_inp)
                                                                cur.execute(sql,val)
                                                                con.commit()

                                                                submit()

                                                            if (not paytype_inp=='' and not
paytype_inp.isspace()) and (not cardno_inp=='' and not cardno_inp.isspace()) and (not
cardname_inp=='' and not cardname_inp.isspace()) and (not expyear_inp=='' and not
expyear_inp.isspace()) and (not expmonth_inp=='' and not expmonth_inp.isspace()) and (not
cvv_inp=='' and not cvv_inp.isspace()):
                                                                if len(cardno_inp) == 16 and
cardno_inp[0] in '3456':
                                                                        if len(expyear_inp) == 4
and int(expyear_inp) ≥ cyear:
                                                                                if
int(expyear_inp) == cyear:
                                                                                        if
(len(expmonth_inp) == 2) and (int(expmonth_inp)≥ 1 and int(expmonth_inp) ≤ 12) and
(int(expmonth_inp) > cmonth):
                                                                                                if
len(cvv_inp)==3:

        pay()

        else:

        messagebox.showerror('Error','CVV must be a 3-digit number.',parent=pay_win)
                                                                                        else:

        messagebox.showerror('Error','Invalid expiry month.',parent=pay_win)
                                                                                elif
int(expyear_inp) > cyear:
                                                                                        if
(len(expmonth_inp) == 2) and (int(expmonth_inp)≥ 1 and int(expmonth_inp) ≤ 12):
                                                                                                if
len(cvv_inp)==3:

        pay()

        else:

        messagebox.showerror('Error','CVV must be a 3-digit number.',parent=pay_win)
                                                                                        else:

        messagebox.showerror('Error','Invalid expiry month..',parent=pay_win)
                                                                                else:
                                                                                        pass
                                                                        else:

        messagebox.showerror('Error','Invalid expiry year.',parent=pay_win)
                                                                else:

        messagebox.showerror('Error','Invalid card number.',parent=pay_win)
                                                            else:

        messagebox.showerror('Error','Please enter all required\npayment details.',parent=pay_win)
```

```
                                                f3=tk.Frame(pay_win)
                                                f3.grid(row=0,column=0)

                                                img1=tk.PhotoImage(file='icons/make-
payment.png')
                                                img=tk.Label(f3,image=img1,font=h1fnt)
                                                img.grid(column=0,row=0,padx=10,pady=10)
                                                img.image=img1


tk.Label(f3,text='Payment',font=h1fnt,justify=tk.LEFT).grid(column=1,row=0,padx=10,pady=10,sticky=tk
.W)

                                                f4=tk.Frame(pay_win)
                                                f4.grid(row=1,column=0)


        payment_summary=scrolledtext.ScrolledText(f4,font=fnt,width=25,height=5)

        payment_summary.grid(column=1,row=2,sticky=tk.EW,padx=10,pady=10)

                                                if n.get()=='Standard':
                                                        base_rate=15
                                                elif n.get()=='XL':
                                                        base_rate=25
                                                elif n.get()=='Luxury':
                                                        base_rate=40

                                                if n.get()=='Standard':
                                                        rate=3
                                                elif n.get()=='XL':
                                                        rate=5
                                                elif n.get()=='Luxury':
                                                        rate=10

                                                o=start.get()
                                                d=end.get()
                                                distance=abs((locations.index(d))-
(locations.index(o)))*4       #distance between locations - 4 km.
                                                if distance > 5:
                                                        total_fare=(base_rate+(rate*(distance-
5)))
                                                else:
                                                        total_fare=(base_rate+0)

                                                text='Booking ID: '+id+'\nFrom: '+o+'\nTo:
'+d+'\nType: '+n.get()+'\n\nDate: '+date_inp+'\nTime: '+time_inp+'\n\nBase rate: $'+str(base_rate)+'
for first 5 km\n$'+str(rate)+' per additional km\nDistance: '+str(distance)+' km'+'\n\nTotal fare:
$'+str(total_fare)
                                                payment_summary.insert(tk.INSERT,text)
                                                payment_summary.configure(state='disabled')


                                                tk.Label(f4,text='Payment
ID',font=fnt).grid(column=0,row=3,sticky=tk.E,padx=10,pady=10)
                                                payment_id='P'+str(rd.randint(10000,99999))
                                                payid=tk.Label(f4,text=payment_id,font=fnt)

        payid.grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

                                                m=tk.StringVar()
                                                tk.Label(f4,text='Pay
by',font=fnt).grid(column=0,row=4,sticky=tk.E,padx=10,pady=10)
                                                card=('','Debit card','Credit card')
                                                pay_type=ttk.OptionMenu(f4,m,*card)

        pay_type.grid(column=1,row=4,sticky=tk.W,padx=10,pady=10)

                                                tk.Label(f4,text='Accepted
cards',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)

                                                img2=tk.PhotoImage(file='img/cards.png')
                                                card_image=tk.Label(f4,image=img2,font=fnt)
```

```
        card_image.grid(column=1,row=5,sticky=tk.W,padx=10,pady=10)
                                                card_image.image=img2

                                                tk.Label(f4,text='Card
number',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
                                                card_no=tk.Entry(f4,font=fnt)

        card_no.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

                                                tk.Label(f4,text='Cardholder
name',font=fnt).grid(column=0,row=7,sticky=tk.E,padx=10,pady=10)
                                                card_name=tk.Entry(f4,font=fnt)

        card_name.grid(column=1,row=7,sticky=tk.EW,padx=10,pady=10)

                                                tk.Label(f4,text='Expiry Year and Month\n[YYYY-
MM]',font=fnt,justify=tk.RIGHT).grid(column=0,row=8,sticky=tk.E,padx=10,pady=10)
                                                exp_year=tk.Entry(f4,font=fnt,width=10)

        exp_year.grid(column=1,row=8,sticky=tk.EW,padx=10,pady=10)


        tk.Label(f4,text='-',font=fnt).grid(column=2,row=8,sticky=tk.EW,padx=10,pady=10)
                                                exp_month=tk.Entry(f4,font=fnt,width=10)

        exp_month.grid(column=3,row=8,sticky=tk.W,padx=10,pady=10)

                                                tk.Label(f4,text='CVV
number',font=fnt).grid(column=0,row=9,sticky=tk.E,padx=10,pady=10)
                                                cvv_no=tk.Entry(f4,font=fnt)

        cvv_no.grid(column=1,row=9,sticky=tk.EW,padx=10,pady=10)



btn=tk.Button(f4,font=fntit,text='Pay',command=make_payment,fg='green');btn.grid(column=1,row=10,pad
x=10,pady=10,sticky=tk.W)

                                                retimg=tk.PhotoImage(file='icons/return.png')

        btn4=tk.Button(f4,font=fnt,image=retimg,command=pay_win.destroy)

        btn4.grid(column=0,row=15,padx=10,pady=10,sticky=tk.SW)
                                                btn4.img=retimg

                                                pay_win.bind('<Return>',lambda
event:make_payment())

                                            else:
                                                messagebox.showerror('Error','Invalid timing
entered.',parent=window)
                                    else:
                                                messagebox.showerror('Error','Invalid time format
entered.',parent=window)
                                else:
                                                messagebox.showerror('Error','The origin and destination are
the same.',parent=window)
                            else:
                                messagebox.showerror('Error','Invalid origin or
destination.',parent=window)
                        else:
                                messagebox.showerror('Error','Please do not leave any fields
blank.',parent=window)

        #FRAME 1
        f1=tk.Frame(window)
        f1.grid(row=0,column=0)

        tk.Label(f1,text='TAXI BOOKING',font=h1fnt,bg='yellow').grid(column=1,row=0,padx=10,pady=10)


        #FRAME 2
        f2=tk.Frame(window)
        f2.grid(row=1,column=0)
```

```
        #Input fields
        tk.Label(f2,text='ID',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)
        bkgid=tk.Label(f2,text=id,font=fnt)
        bkgid.grid(column=1,row=5,sticky=tk.W,padx=10,pady=10)

        n=tk.StringVar()
        tk.Label(f2,text='Taxi Type',font=fnt).grid(column=0,row=7,sticky=tk.E,padx=10,pady=10)
        taxitype=ttk.OptionMenu(f2,n,*ctype)
        taxitype.grid(column=1,row=7,sticky=tk.W,padx=10,pady=10)

        tk.Label(f2,text='From',font=fnt).grid(column=0,row=8,sticky=tk.E,padx=10,pady=10)
        l=tk.StringVar()
        start=ttk.Combobox(f2,textvariable=l,font=fnt,width=19,state='readonly')
        start.grid(column=1,row=8,sticky=tk.EW,padx=10,pady=10)
        start['values']=locations

        tk.Label(f2,text='To',font=fnt).grid(column=0,row=9,sticky=tk.E,padx=10,pady=10)
        m=tk.StringVar()
        end=ttk.Combobox(f2,textvariable=m,font=fnt,width=19,state='readonly')
        end.grid(column=1,row=9,sticky=tk.EW,padx=10,pady=10)
        end['values']=locations

        a=(t+timedelta(minutes=10))           #today
        b=(t+timedelta(days=1,minutes=10))           #tomorrow
        datetype=('','Today '+a.strftime('%Y-%m-%d'),'Tomorrow '+b.strftime('%Y-%m-%d'))

        p=tk.StringVar()
        tk.Label(f2,text='Date',font=fnt).grid(column=0,row=10,sticky=tk.E,padx=10,pady=10)
        date=ttk.OptionMenu(f2,p,*datetype)
        date.grid(column=1,row=10,sticky=tk.W,padx=10,pady=10)

        tk.Label(f2,text='Time',font=fnt).grid(column=0,row=11,sticky=tk.E,padx=10,pady=10)
        time=tk.Entry(f2,font=fnt)
        time.grid(column=1,row=11,sticky=tk.EW,padx=10,pady=10)
        tk.Label(f2,text='24h [HH:MM]',font=fnt).grid(column=2,row=11,padx=10,pady=10)

        Separator(f2,orient='horizontal').grid(row=14,column=0,columnspan=3,sticky=tk.EW)

        tk.Label(f2,text='Proceed to
checkout',font=fnt,justify=tk.RIGHT).grid(column=0,row=15,sticky=tk.E,padx=10,pady=10)
        subimg=tk.PhotoImage(file='icons/checkout.png')
        btn=tk.Button(f2,font=fnt,text='Continue to Payment',image=subimg,command=payment)
        btn.grid(column=1,row=15,padx=10,pady=10,sticky=tk.W)
        btn.image=subimg

        window.bind('<Return>',lambda event:payment())
```

4. init.py

```
def initdb():
	import mysql.connector as ms

	con=ms.connect(host='localhost',user='root',password='123456')
	cur=con.cursor()

	#initial creation of db and tables if not existing in MySQL database'
	cur.execute('create database if not exists taxi')
	cur.execute('use taxi')
	cur.execute('create table if not exists taxi_bkgs(bkgid varchar(6) primary key,bkgtime
datetime,start varchar(50),end varchar(50),jdate date,jtime time,taxitype varchar(50))')
	cur.execute('create table if not exists bus_bkgs(bkgid varchar(6) primary key,bkgtime
datetime,pass_no int,start varchar(50),end varchar(50),jdate date,jtime time,bustype varchar(50))')
	cur.execute('create table if not exists users(uuid varchar(6) primary key,fname
varchar(50),email varchar(50),num varchar(10),uname varchar(50),passwd varchar(50))')
	cur.execute('create table if not exists payment_details(pay_id varchar(6) primary key,paytime
datetime,bkgid varchar(6),amt int,payment_type varchar(20),cardno varchar(16),cardname
varchar(50),cvv int(3),exp_month int(2),exp_year int(4))')
	cur.execute('create table if not exists employees(emp_id varchar(5) primary key,emp_uname
varchar(50),emp_name varchar(50),emp_passwd varchar(50))')
	cur.execute('create table if not exists admin(admin_id varchar(5) primary key,admin_uname
varchar(50),admin_name varchar(50),admin_passwd varchar(50))')

	try:	#creates root and demo users IF NOT EXISTS
		cur.execute("insert into admin values('A0001','root','System
Administrator','123456')")
		cur.execute("insert into employees values('E0001','demoagent','Demonstration
Agent','demoagent')")
		cur.execute("insert into users values('U00001','Demonstration
User','demo@abc.com','1234567890','demo','demo')")
	except:
		pass

	con.commit()
```

5. emp.py

```
#!/bin/python3

def emp_main():
	#import statements

	#Imports libraries
	import mysql.connector as ms
	import tkinter as tk
	from tkinter import ttk
	from tkinter import messagebox
	import platform as pf
	import ctypes
	from tkinter.ttk import Separator

	#Imports other Python scripts
	import manage
	import managebkgs
	import sysinfo
	import bookings
	import init

	#Enables DPI scaling on supported Windows versions
	if pf.system()=='Windows':
		try:
			ctypes.windll.shcore.SetProcessDpiAwareness(True)
		except:
			pass

	#Definitions
```

```
        #mysql connection
        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
        cur=con.cursor()

        init.initdb()

        #fonts
        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)
        menufnt=('IBM Plex Mono',11)

        #main window
        emp_login_win=tk.Tk()
        emp_login_win.title('Employee login')

        #maximises window
        try:
                emp_login_win.state('zoomed')
        except:
                w,h=emp_login_win.winfo_screenwidth(),emp_login_win.winfo_screenheight()
                emp_login_win.geometry(str(w)+'x'+str(h))

        #functions
        def onlogin(): #action on login

                def admin():   #Admin menu

                        root=tk.Tk()
                        root.title('Admin menu')

                        try:
                                root.state('zoomed')
                        except:
                                w,h=root.winfo_screenwidth(),root.winfo_screenheight()
                                root.geometry(str(w)+'x'+str(h))

                        def logout():
                                root.destroy()
                                emp_main()

                        def db():
                                manage.manage_db()

                        def agents():
                                manage.manage_agents()

                        def bookings():
                                empbookings()

                        def users():
                                manage.manage_users()

                        def about_this_program():
                                sysinfo.about()

                        def admins():
                                manage.manage_admin()

                        def passwd():
                                passwd_win=tk.Toplevel()
                                passwd_win.resizable(False,False)
                                passwd_win.title('Change administrator password')

                                def change_admin_passwd():
                                        if not npass.get()=='' and not npass.get().isspace():

                                                confirm=messagebox.askyesno('','Do you wish to change
the administrator password for '+a[emp_uname_inp]+' ?',parent=passwd_win)
                                                if confirm == True:
                                                        sql="update admin set admin_passwd=%s where
admin_uname=%s"
                                                        val=(npass.get(),emp_uname_inp)
                                                        cur.execute(sql,val)
```

```
                                                con.commit()
                                                messagebox.showinfo('','Administrator password
changed for '+a[emp_uname_inp]+'.',parent=passwd_win)
                                                passwd_win.destroy()
                                        else:
                                                messagebox.showinfo('','Administrator password
has not been changed.',parent=passwd_win)

                                    else:
                                                messagebox.showerror('','Please enter a
password.',parent=passwd_win)

                                img14=tk.PhotoImage(file='icons/passwd.png')
                                img=tk.Label(passwd_win,image=img14,font=h1fnt)
                                img.grid(column=0,row=0,padx=10,pady=10)
                                img.image=img14

                                tk.Label(passwd_win,text='Changing the administrator\npassword for
'+a[emp_uname_inp],font=h1fnt,justify=tk.LEFT).grid(column=1,row=0,padx=10,pady=10,sticky=tk.W)

                                tk.Label(passwd_win,text='New
password',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
                                npass=tk.Entry(passwd_win,font=fnt,show='*')
                                npass.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

                                subbtn=tk.Button(passwd_win,text='Make
changes',font=fntit,command=change_admin_passwd)
                                subbtn.grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)

                        tk.Grid.columnconfigure(root,0,weight=1)

                        menubar=tk.Menu(root)

                        user=tk.Menu(menubar,tearoff=0)
                        menubar.add_cascade(label='User',menu=user,font=menufnt)

                        user.add_command(label='Change the administrator
password ... ',command=passwd,font=menufnt,underline=0)
                        user.add_separator()
                        user.add_command(label='Logout',command=logout,font=menufnt,underline=0)
                        user.add_command(label='Logout and
Exit',command=root.destroy,font=menufnt,underline=11)

                        more=tk.Menu(menubar,tearoff=0)
                        menubar.add_cascade(label='Info',menu=more,font=menufnt)

                        more.add_command(label='About this
program ... ',command=about_this_program,font=menufnt,underline=0)
                        root.config(menu=menubar)

                        #FRAME 1
                        tk.Grid.rowconfigure(root,0,weight=1)
                        f1=tk.Frame(root,bg='#283593')
                        f1.grid(row=0,column=0,sticky=tk.NSEW)

                        #frame 1 grid
                        tk.Grid.columnconfigure(f1,0,weight=1)

                        cur.execute('select admin_uname,admin_name from admin')
                        a=dict(cur.fetchall())

                        cur.execute('select admin_uname,admin_id from admin')
                        uuidlist=dict(cur.fetchall())
                        tk.Grid.rowconfigure(f1,0,weight=1)
                        tk.Grid.rowconfigure(f1,1,weight=1)
                        tk.Grid.rowconfigure(f1,2,weight=1)
                        tk.Grid.rowconfigure(f1,3,weight=1)

                        logo_img=tk.PhotoImage(file='img/logo.png')
                        logo=tk.Label(f1,image=logo_img,font=h1fnt,fg='white',bg='#283593')
                        logo.grid(column=0,row=0,padx=10,pady=10,sticky=tk.EW)
                        logo.image=logo_img
```

```
                    tk.Label(f1,text='Welcome,
'+a[emp_uname_inp],font=h1fnt,justify=tk.CENTER,fg='white',bg='#283593').grid(column=0,row=1,padx=10
)

                    tk.Label(f1,text=('User ID: '+uuidlist[emp_uname_inp]),font=('IBM Plex
Sans',12),fg='black',bg='#00e676').grid(column=0,row=2,padx=10)

                    tk.Label(f1,text='Administrator\'s Toolbox',font=('IBM Plex
Sans',12),justify=tk.CENTER,fg='white',bg='#283593').grid(column=0,row=3,padx=10)


        Separator(f1,orient='horizontal').grid(column=0,row=4,sticky=tk.EW,padx=10,pady=10)

                    #FRAME 2
                    tk.Grid.rowconfigure(root,1,weight=1)
                    f2=tk.Frame(root)
                    f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

                    #frame 2 grid
                    tk.Grid.columnconfigure(f2,0,weight=1)
                    tk.Grid.columnconfigure(f2,1,weight=1)
                    tk.Grid.columnconfigure(f2,2,weight=1)
                    tk.Grid.columnconfigure(f2,3,weight=1)

                    tk.Label(f2,text='You
can:',font=fntit).grid(column=1,row=2,sticky=tk.W,padx=10,pady=10)

                    tk.Grid.rowconfigure(f2,5,weight=1)
                    img6=tk.PhotoImage(file='icons/dataset.png')
                    btn1=tk.Button(f2,text='View the
database',image=img6,font=fnt,command=db,width=48,height=48)
                    btn1.grid(column=0,row=5,padx=10,pady=10,sticky=tk.E)
                    btn1.image=img6
                    tk.Label(f2,text='Manage the
databases.',font=fnt,fg='blue').grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

                    img9=tk.PhotoImage(file='icons/employee.png')
                    btn2=tk.Button(f2,text='Manage agents',image=img9,font=fnt,command=agents)
                    btn2.grid(column=2,row=5,padx=10,pady=10,sticky=tk.E)
                    btn2.image=img9
                    tk.Label(f2,text='Manage the
agents.',font=fnt,fg='green').grid(column=3,row=5,padx=10,pady=10,sticky=tk.W)

                    tk.Grid.rowconfigure(f2,6,weight=1)
                    img12=tk.PhotoImage(file='icons/supervisor.png')
                    btn5=tk.Button(f2,text='Manage
administrators',image=img12,font=fnt,command=admins)
                    btn5.grid(column=0,row=6,padx=10,pady=10,sticky=tk.E)
                    btn5.image=img12
                    tk.Label(f2,text='Manage the
administrators.',font=fnt,fg='red').grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)

                    img11=tk.PhotoImage(file='icons/people.png')
                    btn4=tk.Button(f2,text='Manage users',image=img11,font=fnt,command=users)
                    btn4.grid(column=2,row=6,padx=10,pady=10,sticky=tk.E)
                    btn4.image=img11
                    tk.Label(f2,text='Manage the
users.',font=fnt,fg='purple').grid(column=3,row=6,padx=10,pady=10,sticky=tk.W)

                    tk.Grid.rowconfigure(f2,8,weight=1)

                    img10=tk.PhotoImage(file='icons/booking.png')
                    btn3=tk.Button(f2,text='Bookings',image=img10,font=fnt,command=bookings)
                    btn3.grid(column=0,row=8,padx=10,pady=10,sticky=tk.E)
                    btn3.image=img10
                    tk.Label(f2,text='Make and manage
bookings.',font=fnt).grid(column=1,row=8,padx=10,pady=10,sticky=tk.W)

                    tk.Grid.rowconfigure(f2,9,weight=1)

                    root.mainloop()

            def empbookings():            #Agent booking menu

                    #functions
```

```
                        def book_taxi():      #Opens taxi booking window.
                                bookings.taxi()

                        def book_bus():       #Opens bus booking window
                                bookings.bus()

                        def about_this_program():
                                sysinfo.about()

                        def logout():
                                main_menu.destroy()
                                emp_main()

                        def managetaxibkgs():
                                managebkgs.taxi()

                        def managebusbkgs():
                                managebkgs.bus()

                        if emptype_inp=='Agent':
                                main_menu=tk.Tk()
                        elif emptype_inp=='Administrator':
                                main_menu=tk.Toplevel()

                        main_menu.title('Booking Portal')

                        if emptype_inp=='Agent':
                                try:
                                        main_menu.state('zoomed')
                                except:

        w,h=main_menu.winfo_screenwidth(),main_menu.winfo_screenheight()
                                        main_menu.geometry(str(w)+'x'+str(h))

                        elif emptype_inp=='Administrator':
                                main_menu.geometry('960x540')

                        if emptype_inp=='Agent':
                                menubar=tk.Menu(main_menu)

                                user=tk.Menu(menubar,tearoff=0)
                                menubar.add_cascade(label='User',menu=user,font=menufnt)

        user.add_command(label='Logout',command=logout,font=menufnt,underline=0)
                                user.add_command(label='Logout and
exit',command=main_menu.destroy,font=menufnt,underline=11)
                                main_menu.config(menu=menubar)

                                more=tk.Menu(menubar,tearoff=0)
                                menubar.add_cascade(label='Info',menu=more,font=menufnt)
                                more.add_command(label='About this
program ... ',command=about_this_program,font=menufnt,underline=0)
                                main_menu.config(menu=menubar)

                        tk.Grid.columnconfigure(main_menu,0,weight=1)


                        #FRAME 1
                        tk.Grid.rowconfigure(main_menu,0,weight=1)
                        f1=tk.Frame(main_menu,bg='#283593')
                        f1.grid(row=0,column=0,sticky=tk.NSEW)

                        tk.Grid.columnconfigure(f1,0,weight=1)

                        tk.Grid.rowconfigure(f1,0,weight=1)
                        tk.Grid.rowconfigure(f1,1,weight=1)
                        tk.Grid.rowconfigure(f1,2,weight=1)
                        tk.Grid.rowconfigure(f1,2,weight=1)

                        cur.execute('select emp_uname,emp_name from employees')
                        b=dict(cur.fetchall())

                        cur.execute('select emp_uname,emp_id from employees')
                        uuidlist=dict(cur.fetchall())
```

```
                        if emptype_inp=='Agent':
                                tk.Grid.rowconfigure(f1,0,weight=1)

                                logo_img=tk.PhotoImage(file='img/logo.png')
                                logo=tk.Label(f1,image=logo_img,font=h1fnt,fg='white',bg='#283593')
                                logo.grid(column=0,row=0,padx=10,pady=10,sticky=tk.EW)
                                logo.image=logo_img

                                txt='Welcome, '+b[emp_uname_inp]
                                tk.Label(f1,text=('User ID: '+uuidlist[emp_uname_inp]),font=('IBM Plex
Sans',12),fg='black',bg='#00e676').grid(column=0,row=2,padx=10)
                                tk.Label(f1,text='Make and manage
bookings',fg='white',bg='#283593',font=('IBM Plex
Sans',12),justify=tk.CENTER).grid(column=0,row=3,padx=10,pady=10)
                        elif emptype_inp=='Administrator':
                                txt='Make and manage bookings'



tk.Label(f1,text=txt,fg='white',bg='#283593',font=h1fnt,justify=tk.CENTER).grid(column=0,row=1,padx=
10,pady=10)


        Separator(f1,orient='horizontal').grid(column=0,row=4,sticky=tk.EW,padx=10,pady=10)
                        #FRAME 2
                        tk.Grid.rowconfigure(main_menu,1,weight=1)
                        f2=tk.Frame(main_menu)
                        f2.grid(row=1,column=0,sticky=tk.NSEW)

                        tk.Grid.columnconfigure(f2,0,weight=1)
                        tk.Grid.columnconfigure(f2,1,weight=1)
                        tk.Grid.columnconfigure(f2,2,weight=1)
                        tk.Grid.columnconfigure(f2,3,weight=1)

                        tk.Label(f2,text=('You
can:'),font=fntit).grid(column=1,row=2,padx=10,pady=10,sticky=tk.W)

                        tk.Grid.rowconfigure(f2,5,weight=1)
                        img6=tk.PhotoImage(file='icons/taxi.png')
                        bkgbtn=tk.Button(f2,text='Book taxi',image=img6,font=fnt,command=book_taxi)
                        bkgbtn.grid(column=0,row=5,padx=10,pady=1,sticky=tk.E)
                        bkgbtn.image=img6
                        tk.Label(f2,text='Book a
taxi.',font=fnt,bg='yellow').grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

                        img4=tk.PhotoImage(file='icons/bus.png')
                        passbtn=tk.Button(f2,text='Book Bus',image=img4,command=book_bus)
                        passbtn.grid(column=2,row=5,padx=10,pady=10,sticky=tk.E)
                        passbtn.image=img4
                        tk.Label(f2,text='Book a
bus.',font=fnt,fg='blue').grid(column=3,row=5,padx=5,pady=10,sticky=tk.W)


        tk.Label(f2,text=('or:'),font=fntit).grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)

                        tk.Grid.rowconfigure(f2,7,weight=1)

                        btn5=tk.Button(f2,text='Manage taxi
bookings',font=fntit,command=managetaxibkgs)
                        btn5.grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)

                        btn6=tk.Button(f2,text='Manage bus bookings',font=fntit,command=managebusbkgs)
                        btn6.grid(column=3,row=7,padx=10,pady=10,sticky=tk.W)

                        tk.Grid.rowconfigure(f2,10,weight=1)

                        if emptype_inp=='Agent':
                                main_menu.mainloop()

                #Converts inputs to strings
                emp_uname_inp=emp_uname.get().lower()
                emptype_inp=n.get()
                emp_passwd_inp=emp_passwd.get()

                #Checking for validity in inputs
```

```
                if emptype_inp == 'Agent':

                        cur.execute('select emp_uname,emp_passwd from employees')           #list of
agent usernames and passwords
                        e=dict(cur.fetchall())

                        cur.execute('select emp_uname,emp_name from employees')
        #list of agent usernames and names
                        f=dict(cur.fetchall())

                        if not emp_uname_inp=='' or emp_uname_inp.isspace():
                                if emp_uname_inp in e.keys():
                                        if emp_passwd_inp==e[emp_uname_inp]:
                                                emp_login_win.destroy()
                                                empbookings()
                                        else:
                                                messagebox.showerror('Error','Invalid password for
agent '+f[emp_uname_inp]+'.')
                                else:
                                        messagebox.showerror('Error','Agent '+emp_uname_inp+' does not
exist.')
                        else:
                                messagebox.showerror('Error','Do not leave any fields empty.')

                elif emptype_inp == 'Administrator':

                        cur.execute('select admin_uname,admin_passwd from admin')   #list of admin
usernames and passwords
                        a=dict(cur.fetchall())

                        cur.execute('select admin_uname,admin_name from admin')             #list of
admin usernames and names
                        b=dict(cur.fetchall())

                        if not emp_uname_inp=='' or emp_uname_inp.isspace():
                                if emp_uname_inp in a.keys():
                                        if emp_passwd_inp==a[emp_uname_inp]:
                                                emp_login_win.destroy()
                                                admin()
                                        else:
                                                messagebox.showerror('Error','Invalid password for
administrator '+b[emp_uname_inp]+'.')
                                else:
                                        messagebox.showerror('Error','Administrator '+emp_uname_inp+'
does not exist.')
                        else:
                                messagebox.showerror('Error','Do not leave any fields empty.')
                else:
                        messagebox.showerror('Error','Please select login type.')

        def about_this_program():
                sysinfo.about()

        menubar=tk.Menu(emp_login_win)

        more=tk.Menu(menubar,tearoff=0)
        menubar.add_cascade(label='Info',menu=more,font=menufnt)
        more.add_command(label='About this
program ... ',command=about_this_program,font=menufnt,underline=0)
        emp_login_win.config(menu=menubar)

        tk.Grid.columnconfigure(emp_login_win,0,weight=1)

        #FRAME 1
        tk.Grid.rowconfigure(emp_login_win,0,weight=1)
        f1=tk.Frame(emp_login_win,bg='#283593')
        f1.grid(row=0,column=0,sticky=tk.NSEW)

        #frame 1 grid
        tk.Grid.columnconfigure(f1,0,weight=1)
        tk.Grid.rowconfigure(f1,0,weight=1)
        tk.Grid.rowconfigure(f1,1,weight=1)

        logo_img=tk.PhotoImage(file='img/logo.png')
        logo=tk.Label(f1,image=logo_img,font=h1fnt,fg='white',bg='#283593')
```

```
        logo.grid(column=0,row=0,sticky=tk.EW,padx=10,pady=10)
        logo.image=logo_img

        tk.Label(f1,text='Employee
login',font=h1fnt,fg='white',bg='#283593').grid(column=0,row=1,padx=10,pady=10,sticky=tk.EW)

        ttk.Separator(f1,orient='horizontal').grid(row=2,column=0,sticky=tk.EW,pady=10,columnspan=2)

        #FRAME 2
        tk.Grid.rowconfigure(emp_login_win,1,weight=1)
        f2=tk.Frame(emp_login_win)
        f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

        #frame 2 grid
        tk.Grid.columnconfigure(f2,0,weight=1)
        tk.Grid.columnconfigure(f2,1,weight=1)

        #Login type
        tk.Label(f2,text='Login as',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)
        n=tk.StringVar()
        values=('','Agent','Administrator')
        emptype=ttk.OptionMenu(f2,n,*values);emptype.grid(column=1,row=5,sticky=tk.W,padx=10,pady=10)

        #uname
        tk.Label(f2,text='Username',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
        emp_uname=tk.Entry(f2,font=fnt)
        emp_uname.grid(column=1,row=6,sticky=tk.W,padx=10,pady=10)

        #passwd
        tk.Label(f2,text='Password',font=fnt).grid(column=0,row=7,sticky=tk.E,padx=10,pady=10)
        emp_passwd=tk.Entry(f2,show='*',font=fnt)
        emp_passwd.grid(column=1,row=7,sticky=tk.W,padx=10,pady=10)

        #Login button
        img1=tk.PhotoImage(file='icons/login.png')
        logsubmit=tk.Button(f2,text='Login',image=img1,command=onlogin)
        logsubmit.grid(column=1,row=8,padx=10,pady=10,sticky=tk.W)

        emp_login_win.bind('<Return>',lambda event:onlogin())

        emp_login_win.mainloop()

emp_main()
```

6. manage.py

```
def manage_admin():    #Manage admins
        import mysql.connector as ms
        import tkinter as tk
        import platform as pf
        import ctypes
        from tkinter import ttk
        from tkinter import messagebox
        import random as rd

        #Enables DPI scaling on supported versions of Windows
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass

        #MySQL connection
        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
        cur=con.cursor()

        #Fonts
        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)

        #Creating Toplevel window
        manageadminwin=tk.Toplevel()
        manageadminwin.title('Administrator Manager')

        def viewall(): #Show all Administrators
                viewall_win=tk.Toplevel()
                viewall_win.title('All administrators')
                viewall_win.resizable(False,False)

                header=('Admin ID','Admin Username','Admin Name','Admin Password')

                sql2=str('select * from admin')                      #getting data from table
                cur.execute(sql2)
                data=[header]+cur.fetchall()                                                #appending header
to data

                rows=len(data)
                cols=len(data[0])

                for i in range(rows):                                                       #drawing the
table in GUI
                        for j in range(cols):
                                entry =
tk.Label(viewall_win,borderwidth=1,relief='solid',padx=10,height=2,font=fnt)
                                entry.grid(row=i, column=j,padx=2,pady=2,sticky=tk.EW)
                                entry.configure(text=data[i][j])
                                if i==0:
                                        entry.configure(fg='red',font=fntit)  #colors and italicises
header

        def viewone(): #View details of administrator
                def getadmninfo():     #Gets data from DB

                        if not uname.get()=='' and not uname.get().isspace():
                                if uname.get() in admin_list:
                                        sql='select * from admin where admin_uname=%s'
                                        val=(uname.get(),)
                                        cur.execute(sql,val)
                                        c=cur.fetchall()
                                        admin_id=c[0][0]
                                        admin_uname=c[0][1]
                                        admin_name=c[0][2]
                                        admin_passwd=c[0][3]
```

```
                                        data=[('Administrator ID',admin_id),('Administrator
Username',admin_uname),('Administrator Full Name',admin_name),('Administrator
Password',admin_passwd)]

                                        rows=len(data)
                                        cols=len(data[0])

        tk.Label(frame3,font=fntit,text='Data').grid(row=0,column=0,sticky=tk.W)
                                        for i in range(rows):
        #drawing the table in GUI
                                                for j in range(cols):
                                                        entry =
tk.Label(frame2,borderwidth=1,relief='solid',padx=10,width=30,height=2,font=fnt)

        entry.grid(row=i,column=j,padx=2,pady=2,sticky=tk.EW)
                                                        entry.configure(text=data[i][j])
                                                        if j==0:
                                                                entry.configure(fg='red',font=fntit)
#colors and italicises header
                                else:
                                        messagebox.showerror('Error','Username \''+uname.get()+'\'
does not exist.',parent=viewone_win)
                        else:
                                messagebox.showerror('Error','Please enter the administrator
username.',parent=viewone_win)

                #Creating Toplevel window
                viewone_win=tk.Toplevel()
                viewone_win.title('View admin details')
                viewone_win.resizable(False,False)

                #Dividing window into frames
                frame1=tk.Frame(viewone_win)
                frame1.grid(row=0,column=0,padx=10,pady=10,sticky=tk.EW)

                frame2=tk.Frame(viewone_win)
                frame2.grid(row=2,column=0,padx=10,pady=10,sticky=tk.EW)

                frame3=tk.Frame(viewone_win)
                frame3.grid(row=1,column=0,padx=10,pady=10,sticky=tk.W)

                #Creates list of admins for dropdown
                cur.execute('select admin_uname from admin')
                a=cur.fetchall()
                admin_list=[]
                for i in a:
                        admin_list.append(i[0])

                img14=tk.PhotoImage(file='icons/searchusr.png')
                img=tk.Label(frame1,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(frame1,font=h1fnt,text='View administrator
details').grid(row=0,column=1,padx=10,pady=10,sticky=tk.W)

                tk.Label(frame1,font=fnt,text='Enter username of
administrator.').grid(row=4,column=1,padx=10,pady=10,sticky=tk.W)
                n=tk.StringVar()
                uname=ttk.Combobox(frame1,textvariable=n,font=fnt)
                uname.grid(row=5,column=1,padx=10,pady=10,sticky=tk.EW)
                uname['values']=admin_list

                submit=tk.Button(frame1,font=fntit,text='Submit',command=getadmninfo)
                submit.grid(row=5,column=2,padx=10,pady=10)

                #Binds Enter key to submit function
                viewone_win.bind('<Return>',lambda event:getadmninfo())

        def delone():  #Deletes an administrator.

                delone_win=tk.Toplevel()
                delone_win.resizable(False,False)
                delone_win.title('Delete adminstrator')
```

```
                #Creates list of admins and respective full names.
                cur.execute('select admin_uname,admin_name from admin')
                a=cur.fetchall()
                admin_namelist=dict(a)

                def delete_admin():           #Delets from DB.
                        if not uname.get()=='' and not uname.get().isspace():
                                if uname.get() in admin_list:
                                        messagebox.showwarning('','This operation will delete\nthe
username of the administrator permanently.\nContinue?',parent=delone_win)
                                        confirm=messagebox.askyesno('','Do you wish to delete the
administrator '+admin_namelist[uname.get()]+'?',parent=delone_win)
                                        if confirm == True:
                                                sql='delete from admin where admin_uname =%s'
                                                val=(uname.get(),)
                                                cur.execute(sql,val)
                                                con.commit()
                                                messagebox.showinfo('','Administrator
'+admin_namelist[uname.get()]+' deleted.',parent=delone_win)
                                                delone_win.destroy()
                                        else:
                                                messagebox.showinfo('','Administrator
'+admin_namelist[uname.get()]+' not deleted.\nThe database has not been
modified.',parent=delone_win)
                                else:
                                        messagebox.showerror('Error','Username \''+uname.get()+'\'
does not exist.',parent=delone_win)
                        else:
                                messagebox.showerror('','Please enter the administrator
username.',parent=delone_win)

                img14=tk.PhotoImage(file='icons/ban_user.png')
                img=tk.Label(delone_win,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(delone_win,text='Delete an
administrator ... ',font=h1fnt).grid(column=1,row=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select admin_uname from admin')
                d=cur.fetchall()
                admin_list=[]
                for i in d:
                        admin_list.append(i[0])

                tk.Label(delone_win,text='Select an
administrator.',font=fntit).grid(column=1,row=4,padx=10,pady=10,sticky=tk.W)

                n=tk.StringVar()
                uname=ttk.Combobox(delone_win,textvariable=n,font=fnt,width=19)
                uname.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=10)
                uname['values']=admin_list

                delbtn=tk.Button(delone_win,text='Delete',font=fntit,command=delete_admin,fg='red')
                delbtn.grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)

                #Binds Enter key to submit function.
                delone_win.bind('<Return>',lambda event:delete_admin())

        def passwd():  #Change password for administrator currently logged in

                passwd_win=tk.Toplevel()
                passwd_win.resizable(False,False)
                passwd_win.title('Change password for administrator')

                cur.execute('select admin_uname,admin_name from admin')
                a=cur.fetchall()
                admin_namelist=dict(a)

                def change_admin_passwd():    #Changes admin password in DB
                        if (not uname.get()=='' and not uname.get().isspace()) and (not
npass.get()=='' and not npass.get().isspace()):
                                if uname.get() in admin_list:
```

```
                                        confirm=messagebox.askyesno('','Do you wish to change the
password of '+admin_namelist[uname.get()]+'?',parent=passwd_win)
                                        if confirm == True:
                                                sql='update admin set admin_passwd=%s where
admin_uname=%s'
                                                val=(npass.get(),uname.get())
                                                cur.execute(sql,val)
                                                con.commit()
                                                messagebox.showinfo('','Password for
'+admin_namelist[uname.get()]+'\nchanged.',parent=passwd_win)
                                                passwd_win.destroy()
                                        else:
                                                messagebox.showinfo('','Password for
'+admin_namelist[uname.get()]+' has not been changed..\nThe databasehas not\nbeen
modified.',parent=passwd_win)
                                else:
                                        messagebox.showerror('Error','Username \''+uname.get()+'\'
does not exist.',parent=passwd_win)
                        else:
                                messagebox.showerror('','Do not leave any fields
blank.',parent=passwd_win)

                img14=tk.PhotoImage(file='icons/passwd.png')
                img=tk.Label(passwd_win,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(passwd_win,text='Change password\nfor
administrator ... ',font=h1fnt,justify=tk.LEFT).grid(column=1,row=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select admin_uname from admin')
                d=cur.fetchall()
                admin_list=[]
                for i in d:
                        admin_list.append(i[0])

                n=tk.StringVar()


tk.Label(passwd_win,text='Username',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)
                uname=ttk.Combobox(passwd_win,textvariable=n,font=fnt,width=19)
                uname.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=10)
                uname['values']=admin_list
                uname.current(0)

                tk.Label(passwd_win,text='New
password',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
                npass=tk.Entry(passwd_win,font=fnt,show='*')
                npass.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

                subbtn=tk.Button(passwd_win,text='Make
changes',font=fntit,command=change_admin_passwd)
                subbtn.grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)
                passwd_win.bind('<Return>',lambda event:change_admin_passwd())

        def add():      #Register a new administrator.

                add_win=tk.Toplevel()
                add_win.resizable(False,False)
                add_win.title('Add administrator')

                def add_admin():        #Adds admin in DB
                        uname_inp=uname.get().lower()
                        fname_inp=fname.get()
                        passwd_inp=passwd.get()

                        cur.execute('select admin_uname from admin')
                        a=cur.fetchall()
                        admin_list=[]
                        for i in a:
                                admin_list.append(i[0])

                        if (not uname_inp=='' and not uname_inp.isspace()) and (not fname_inp=='' and
not fname_inp.isspace()) and (not passwd_inp=='' and not passwd_inp.isspace()):
                                if uname_inp not in admin_list:
```

```
                                    sql='insert into admin values (%s,%s,%s,%s)'
                                    val=(id,uname_inp,fname_inp,passwd_inp)
                                    cur.execute(sql,val)
                                    con.commit()
                                    messagebox.showinfo('','Administrator '+fname_inp+' registered
successfully.',parent=add_win)
                                    add_win.destroy()
                                else:
                                    messagebox.showerror('Error','Username \''+uname_inp+'\'\
nalready exists.',parent=add_win)
                            else:
                                    messagebox.showerror('Error','Please do not leave any fields
blank.',parent=add_win)

                id='A'+str(rd.randint(1000,9999))

                img14=tk.PhotoImage(file='icons/adduser.png')
                img=tk.Label(add_win,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(add_win,text='Register
administrator ... ',font=h1fnt).grid(column=1,row=0,sticky=tk.W,padx=10,pady=10)


        tk.Label(add_win,text='UID',font=fnt).grid(column=0,row=3,sticky=tk.E,padx=10,pady=10)
                bkgid=tk.Label(add_win,text=id,font=fnt)
                bkgid.grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

                #Input fields
                tk.Label(add_win,text='Full
Name',font=fnt).grid(column=0,row=4,sticky=tk.E,padx=10,pady=10)
                fname=tk.Entry(add_win,font=fnt)
                fname.grid(column=1,row=4,sticky=tk.EW,padx=10,pady=10)


        tk.Label(add_win,text='Username',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)
                uname=tk.Entry(add_win,font=fnt)
                uname.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=10)


        tk.Label(add_win,text='Password',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
                passwd=tk.Entry(add_win,font=fnt,show='*')
                passwd.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

                subbtn=tk.Button(add_win,font=fntit,text='Register',command=add_admin)
                subbtn.grid(column=1,row=12,padx=10,pady=10,sticky=tk.W)

                #Binds Enter to submit function.
                add_win.bind('<Return>',lambda event:add_admin())

        tk.Grid.columnconfigure(manageadminwin,0,weight=1)

        #FRAME 1
        tk.Grid.rowconfigure(manageadminwin,0,weight=1)
        f1=tk.Frame(manageadminwin)
        f1.grid(row=0,column=0,sticky=tk.NSEW)

        #frame 1 grid
        tk.Grid.columnconfigure(f1,0,weight=1)
        tk.Grid.columnconfigure(f1,1,weight=1)

        tk.Grid.rowconfigure(f1,0,weight=1)
        img6=tk.PhotoImage(file='icons/supervisor.png')
        himg=tk.Label(f1,image=img6)
        himg.grid(column=0,row=0,sticky=tk.E,padx=10,pady=10)
        himg.image=img6
        tk.Label(f1,text=('Manage the
administrators ... '),font=h1fnt).grid(column=1,row=0,sticky=tk.W,padx=10,pady=10)

        tk.Label(f1,text=('Connected to database: '+con.database),font=('IBM Plex
Sans',12),justify=tk.LEFT,fg='green').grid(column=1,row=1,sticky=tk.W,padx=10,pady=10)

ttk.Separator(f1,orient='horizontal').grid(column=0,row=2,sticky=tk.EW,padx=10,pady=10,columnspan=2)
        #FRAME 2
```

```
        tk.Grid.rowconfigure(manageadminwin,1,weight=1)
        f2=tk.Frame(manageadminwin)
        f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

        #frame 2 grid
        tk.Grid.columnconfigure(f2,0,weight=1)
        tk.Grid.columnconfigure(f2,1,weight=1)
        tk.Grid.columnconfigure(f2,2,weight=1)
        tk.Grid.columnconfigure(f2,3,weight=1)

        tk.Label(f2,text='You
can:',font=fntit,justify=tk.LEFT).grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

        tk.Grid.rowconfigure(f2,5,weight=1)
        img8=tk.PhotoImage(file='icons/preview.png')
        tbviewbtn=tk.Button(f2,text='view all',image=img8,font=fnt,command=viewall)
        tbviewbtn.grid(column=0,row=5,padx=10,pady=10,sticky=tk.E)
        tbviewbtn.image=img8
        tk.Label(f2,text='View all administrator
details.',font=fnt,fg='blue').grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

        img10=tk.PhotoImage(file='icons/searchusr.png')
        viewbtn=tk.Button(f2,text='viewone',image=img10,font=fnt,command=viewone)
        viewbtn.grid(column=2,row=5,padx=10,pady=10,sticky=tk.E)
        viewbtn.image=img10
        tk.Label(f2,text='View a single admin\'s
details.',font=fnt).grid(column=3,row=5,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,6,weight=1)
        img7=tk.PhotoImage(file='icons/adduser.png')
        tbviewbtn=tk.Button(f2,text='add',image=img7,font=fnt,command=add)
        tbviewbtn.grid(column=0,row=6,padx=10,pady=10,sticky=tk.E)
        tbviewbtn.image=img7
        tk.Label(f2,text='Register an
administrator.',font=fnt,fg='green').grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)

        img11=tk.PhotoImage(file='icons/passwd.png')
        passbtn=tk.Button(f2,text='passwd',image=img11,font=fnt,command=passwd)
        passbtn.grid(column=2,row=6,padx=10,pady=10,sticky=tk.E)
        passbtn.image=img11
        tk.Label(f2,text='Change the password for an
administrator.',font=fnt).grid(column=3,row=6,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,7,weight=1)
        img12=tk.PhotoImage(file='icons/deluser.png')
        delbtn=tk.Button(f2,text='del',image=img12,font=fnt,command=delone)
        delbtn.grid(column=0,row=7,padx=10,pady=10,sticky=tk.E)
        delbtn.image=img12
        tk.Label(f2,text='Delete an
administrator.',font=fnt,fg='red').grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)
        tk.Grid.rowconfigure(f2,8,weight=1)
        tk.Message(f2,text='WARNING: This will delete\nan admin\'s profile\nfrom the system
permanently.',width=500,font=fnt,fg='white',bg='red').grid(column=1,row=8,padx=10,pady=10,sticky=tk.
NW)

        tk.Grid.rowconfigure(f2,16,weight=1)

def manage_agents():   #Manage agents (employees)
        import mysql.connector as ms
        import tkinter as tk
        import platform as pf
        import ctypes
        from tkinter import ttk
        from tkinter import messagebox
        import random as rd

        #Enables DPI scaling on supported versions of Windows
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass

        #MySQL connection
        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
```

```
        cur=con.cursor()

        #Fonts for GUI
        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)

        #Creating Toplevel window
        manage_agentwin=tk.Toplevel()
        manage_agentwin.title('Agent Manager')

        def viewall(): #View all Agent

                viewall_win=tk.Toplevel()
                viewall_win.title('All Agents')
                viewall_win.resizable(False,False)

                #Headers for table
                header=('Agent ID','Agent Username','Agent Name','Agent Password')

                sql2=str('select * from employees')
                cur.execute(sql2)
                data=[header]+cur.fetchall()                                    #appending header
to data

                rows=len(data)
                cols=len(data[0])

                for i in range(rows):                                           #drawing the
table in GUI
                        for j in range(cols):
                                entry =
tk.Label(viewall_win,borderwidth=1,relief='solid',padx=10,height=2,font=fnt)
                                entry.grid(row=i, column=j,padx=2,pady=2,sticky=tk.EW)
                                entry.configure(text=data[i][j])
                                if i==0:
                                        entry.configure(fg='red',font=fntit)  #colors and italicises
header

        def viewone(): #Show an agent's info
                def getagentinfo():    #Gets data from DB

                        if not uname.get()=='' and not uname.get().isspace():
                                if uname.get() in agent_list:
                                        sql='select * from employees where emp_uname=%s'
                                        val=(uname.get(),)
                                        cur.execute(sql,val)
                                        c=cur.fetchall()
                                        agent_id=c[0][0]
                                        agent_uname=c[0][1]
                                        agent_name=c[0][2]
                                        agent_passwd=c[0][3]

                                        e=[('Agent ID',agent_id),('Agent Username',agent_uname),
('Agent Full Name',agent_name),('Agent Password',agent_passwd)]


                                        rows=len(e)
                                        cols=len(e[0])

        tk.Label(frame3,font=fntit,text='Data').grid(row=0,column=0,sticky=tk.W)
                                        for i in range(rows):
        #drawing the table in GUI
                                                for j in range(cols):
                                                        entry =
tk.Label(frame2,borderwidth=1,relief='solid',padx=10,width=30,height=2,font=fnt)

        entry.grid(row=i,column=j,padx=2,pady=2,sticky=tk.EW)
                                                        entry.configure(text=e[i][j])
                                                        if j==0:

        entry.configure(fg='red',font=fntit,width=20) #colors and italicises header
                                else:
                                        messagebox.showerror('Error','Username \''+uname.get()+'\'
does not exist.',parent=viewone_win)
```

```
                        else:
                                messagebox.showerror('Error','Please enter the agent
username.',parent=viewone_win)
                viewone_win=tk.Toplevel()
                viewone_win.title('View agent details')
                viewone_win.resizable(False,False)

                frame1=tk.Frame(viewone_win)
                frame1.grid(row=0,column=0,padx=10,pady=10,sticky=tk.EW)

                frame2=tk.Frame(viewone_win)
                frame2.grid(row=2,column=0,padx=10,pady=10,sticky=tk.EW)

                frame3=tk.Frame(viewone_win)
                frame3.grid(row=1,column=0,padx=10,pady=10,sticky=tk.W)

                #Creating list of agent
                cur.execute('select emp_uname from employees')
                a=cur.fetchall()
                agent_list=[]
                for i in a:
                        agent_list.append(i[0])

                img14=tk.PhotoImage(file='icons/searchusr.png')
                img=tk.Label(frame1,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(frame1,font=h1fnt,text='View agent
details').grid(row=0,column=1,padx=10,pady=10,sticky=tk.W)

                tk.Label(frame1,font=fnt,text='Enter username of
agent.').grid(row=4,column=1,padx=10,pady=10,sticky=tk.W)
                n=tk.StringVar()
                uname=ttk.Combobox(frame1,textvariable=n,font=fnt)
                uname.grid(row=5,column=1,padx=10,pady=10,sticky=tk.EW)
                uname['values']=agent_list

                submit=tk.Button(frame1,font=fntit,text='Submit',command=getagentinfo)
                submit.grid(row=5,column=2,padx=10,pady=10)

                #Binds Enter to submit function.
                viewone_win.bind('<Return>',lambda event:getagentinfo())

        def delone():  #Delete an agent
                delone_win=tk.Toplevel()
                delone_win.resizable(False,False)
                delone_win.title('Delete agent')
                cur.execute('select emp_uname,emp_name from employees')
                a=cur.fetchall()
                agent_namelist=dict(a)
                def delete_agent():    #Delete agent from db
                        if not uname.get()=='' and not uname.get().isspace():
                                if uname.get() in agent_list:
                                        messagebox.showwarning('','This operation will delete\nthe
profile of the agent permanently.\nContinue?',parent=delone_win)
                                        confirm=messagebox.askyesno('','Do you wish to delete the
agent '+agent_namelist[uname.get()]+'?',parent=delone_win)
                                        if confirm == True:
                                                sql='delete from employees where emp_uname =%s'
                                                val=(uname.get(),)
                                                cur.execute(sql,val)
                                                con.commit()
                                                messagebox.showinfo('','Agent
'+agent_namelist[uname.get()]+' deleted.',parent=delone_win)
                                                delone_win.destroy()
                                        else:
                                                messagebox.showinfo('','Agent
'+agent_namelist[uname.get()]+' not deleted.\nThe database has not been
modified.',parent=delone_win)
                                else:
                                        messagebox.showerror('Error','Username \''+uname.get()+'\'
does not exist.',parent=delone_win)
                        else:
```

```
                                messagebox.showerror('','Please enter the agent
username.',parent=delone_win)

                img14=tk.PhotoImage(file='icons/ban_user.png')
                img=tk.Label(delone_win,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(delone_win,text='Delete an
agent ... ',font=h1fnt).grid(column=1,row=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select emp_uname from employees')
                d=cur.fetchall()
                agent_list=[]
                for i in d:
                        agent_list.append(i[0])

                tk.Label(delone_win,text='Select an
agent.',font=fntit).grid(column=1,row=4,padx=10,pady=10,sticky=tk.W)

                n=tk.StringVar()
                uname=ttk.Combobox(delone_win,textvariable=n,font=fnt,width=19)
                uname.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=10)
                uname['values']=agent_list

                delbtn=tk.Button(delone_win,text='Delete',font=fntit,command=delete_agent,fg='red')
                delbtn.grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)
                delone_win.bind('<Return>',lambda event:delete_agent())

        def passwd():  #Change password for agent
                passwd_win=tk.Toplevel()
                passwd_win.resizable(False,False)
                passwd_win.title('Change password for employee')

                cur.execute('select emp_uname,emp_name from employees')
                a=cur.fetchall()
                agent_namelist=dict(a)
                def change_emp_passwd():      #Changes agent passwd in DB
                        if (not uname.get()=='' and not uname.get().isspace()) and (not
npass.get()=='' and not npass.get().isspace()):
                                if uname.get() in agent_list:

                                        confirm=messagebox.askyesno('','Do you wish to change the
password of '+agent_namelist[uname.get()]+'?',parent=passwd_win)
                                        if confirm == True:
                                                sql='update employees set emp_passwd=%s where
emp_uname=%s'
                                                val=(npass.get(),uname.get())
                                                cur.execute(sql,val)
                                                con.commit()
                                                messagebox.showinfo('','Password for
'+agent_namelist[uname.get()]+'\nchanged.',parent=passwd_win)
                                                passwd_win.destroy()
                                        else:
                                                messagebox.showinfo('','Password for
'+agent_namelist[uname.get()]+' has not been changed..\nThe databasehas not\nbeen
modified.',parent=passwd_win)
                                else:
                                        messagebox.showerror('Error','Username \''+uname.get()+'\'
does not exist.',parent=passwd_win)
                        else:
                                messagebox.showerror('','Do not leave any fields
blank.',parent=passwd_win)

                img14=tk.PhotoImage(file='icons/passwd.png')
                img=tk.Label(passwd_win,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(passwd_win,text='Change password\nfor
agent ... ',font=h1fnt,justify=tk.LEFT).grid(column=1,row=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select emp_uname from employees')
                d=cur.fetchall()
                agent_list=[]
```

```
                for i in d:
                        agent_list.append(i[0])

                n=tk.StringVar()


tk.Label(passwd_win,text='Username',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)
                uname=ttk.Combobox(passwd_win,textvariable=n,font=fnt,width=19)
                uname.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=10)
                uname['values']=agent_list
                uname.current(0)

                tk.Label(passwd_win,text='New
password',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
                npass=tk.Entry(passwd_win,font=fnt,show='*')
                npass.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

                subbtn=tk.Button(passwd_win,text='Make changes',font=fntit,command=change_emp_passwd)
                subbtn.grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)
                passwd_win.bind('<Return>',lambda event:change_emp_passwd())

        def add():      #Add an agent.
                add_win=tk.Toplevel()
                add_win.resizable(False,False)
                add_win.title('Add agent')

                def add_agent():        #Adds agent to DB.
                        uname_inp=uname.get().lower()
                        fname_inp=fname.get()
                        passwd_inp=passwd.get()

                        #Creates list of employees
                        cur.execute('select emp_uname from employees')
                        a=cur.fetchall()
                        agent_list=[]
                        for i in a:
                                agent_list.append(i[0])

                        if (not uname_inp=='' and not uname_inp.isspace()) and (not fname_inp=='' and
not fname_inp.isspace()) and (not passwd_inp=='' and not passwd_inp.isspace()):
                                if uname_inp not in agent_list:
                                        sql='insert into employees values (%s,%s,%s,%s)'
                                        val=(id,uname_inp,fname_inp,passwd_inp)
                                        cur.execute(sql,val)
                                        con.commit()
                                        messagebox.showinfo('','Agent '+fname_inp+' registered
successfully.',parent=add_win)
                                        add_win.destroy()
                                else:
                                        messagebox.showerror('Error','Username \''+uname_inp+'\'\
nalready exists.',parent=add_win)
                        else:
                                messagebox.showerror('Error','Please do not leave any fields
blank.',parent=add_win)

                id='E'+str(rd.randint(1000,9999))

                img14=tk.PhotoImage(file='icons/adduser.png')
                img=tk.Label(add_win,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(add_win,text='Register
agent ... ',font=h1fnt).grid(column=1,row=0,sticky=tk.W,padx=10,pady=10)


        tk.Label(add_win,text='UID',font=fnt).grid(column=0,row=3,sticky=tk.E,padx=10,pady=10)
                bkgid=tk.Label(add_win,text=id,font=fnt)
                bkgid.grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

                tk.Label(add_win,text='Full
Name',font=fnt).grid(column=0,row=4,sticky=tk.E,padx=10,pady=10)
                fname=tk.Entry(add_win,font=fnt)
                fname.grid(column=1,row=4,sticky=tk.EW,padx=10,pady=10)
```

```
        tk.Label(add_win,text='Username',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)
                uname=tk.Entry(add_win,font=fnt)
                uname.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=10)


        tk.Label(add_win,text='Password',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
                passwd=tk.Entry(add_win,font=fnt,show='*')
                passwd.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

                subbtn=tk.Button(add_win,font=fntit,text='Register',command=add_agent)
                subbtn.grid(column=1,row=12,padx=10,pady=10,sticky=tk.W)

                add_win.bind('<Return>',lambda event:add_agent())


        tk.Grid.columnconfigure(manage_agentwin,0,weight=1)

        #FRAME 1
        tk.Grid.rowconfigure(manage_agentwin,0,weight=1)
        f1=tk.Frame(manage_agentwin)
        f1.grid(row=0,column=0,sticky=tk.NSEW)

        #frame 1 grid
        tk.Grid.columnconfigure(f1,0,weight=1)
        tk.Grid.columnconfigure(f1,1,weight=1)

        tk.Grid.rowconfigure(f1,0,weight=1)
        img6=tk.PhotoImage(file='icons/employee.png')
        himg=tk.Label(f1,image=img6)
        himg.grid(column=0,row=0,sticky=tk.E,padx=10,pady=10)
        himg.image=img6
        tk.Label(f1,text=('Manage the
agents ... '),font=h1fnt).grid(column=1,row=0,sticky=tk.W,padx=10,pady=10)

        tk.Label(f1,text=('Connected to database: '+con.database),font=('IBM Plex
Sans',12),justify=tk.LEFT,fg='green').grid(column=1,row=1,sticky=tk.W,padx=10,pady=10)

ttk.Separator(f1,orient='horizontal').grid(column=0,row=2,sticky=tk.EW,padx=10,pady=10,columnspan=2)
        #FRAME 2
        tk.Grid.rowconfigure(manage_agentwin,1,weight=1)
        f2=tk.Frame(manage_agentwin)
        f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

        #frame 2 grid
        tk.Grid.columnconfigure(f2,0,weight=1)
        tk.Grid.columnconfigure(f2,1,weight=1)
        tk.Grid.columnconfigure(f2,2,weight=1)
        tk.Grid.columnconfigure(f2,3,weight=1)

        tk.Label(f2,text='You
can:',font=fntit,justify=tk.LEFT).grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

        tk.Grid.rowconfigure(f2,5,weight=1)
        img8=tk.PhotoImage(file='icons/preview.png')
        tbviewbtn=tk.Button(f2,text='view all',image=img8,font=fnt,command=viewall)
        tbviewbtn.grid(column=0,row=5,padx=10,pady=10,sticky=tk.E)
        tbviewbtn.image=img8
        tk.Label(f2,text='View all agent
details.',font=fnt,fg='blue').grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

        img10=tk.PhotoImage(file='icons/searchusr.png')
        viewbtn=tk.Button(f2,text='viewone',image=img10,font=fnt,command=viewone)
        viewbtn.grid(column=2,row=5,padx=10,pady=10,sticky=tk.E)
        viewbtn.image=img10
        tk.Label(f2,text='View a single agent\'s
details.',font=fnt).grid(column=3,row=5,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,6,weight=1)
        img7=tk.PhotoImage(file='icons/adduser.png')
        tbviewbtn=tk.Button(f2,text='add',image=img7,font=fnt,command=add)
        tbviewbtn.grid(column=0,row=6,padx=10,pady=10,sticky=tk.E)
        tbviewbtn.image=img7
        tk.Label(f2,text='Register an
agent.',font=fnt,fg='green').grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)
```

```
        img11=tk.PhotoImage(file='icons/passwd.png')
        passbtn=tk.Button(f2,text='passwd',image=img11,font=fnt,command=passwd)
        passbtn.grid(column=2,row=6,padx=10,pady=10,sticky=tk.E)
        passbtn.image=img11
        tk.Label(f2,text='Change the password for an
agent.',font=fnt).grid(column=3,row=6,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,7,weight=1)
        img12=tk.PhotoImage(file='icons/deluser.png')
        delbtn=tk.Button(f2,text='del',image=img12,font=fnt,command=delone)
        delbtn.grid(column=0,row=7,padx=10,pady=10,sticky=tk.E)
        delbtn.image=img12
        tk.Label(f2,text='Delete an
agent.',font=fnt,fg='red').grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)
        tk.Grid.rowconfigure(f2,8,weight=1)
        tk.Message(f2,text='WARNING: This will delete\nan agent\'s profile\nfrom the system
permanently.',width=500,font=fnt,fg='white',bg='red').grid(column=1,row=8,padx=10,pady=10,sticky=tk.
NW)

        tk.Grid.rowconfigure(f2,16,weight=1)

def manage_users():    #Manage users
        import mysql.connector as ms
        import tkinter as tk
        import platform as pf
        import ctypes
        from tkinter import ttk
        from tkinter import messagebox
        import random as rd

        #Enables DPI scaling on supported versions of Windows
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass

        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
        cur=con.cursor()

        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)

        manageuserwin=tk.Toplevel()
        manageuserwin.title('User Manager')


        def viewall():         #View all users
                viewall_win=tk.Toplevel()
                viewall_win.title('All users')
                viewall_win.resizable(False,False)

                header=('User ID','Full Name','Electronic Mail','Number','Username','Password')

                sql2=str('select * from users')

                cur.execute(sql2)
                data=[header]+cur.fetchall()                                    #appending header
to data

                rows=len(data)
                cols=len(data[0])

                for i in range(rows):                                           #drawing the
table in GUI
                        for j in range(cols):
                                entry =
tk.Label(viewall_win,borderwidth=1,relief='solid',padx=10,height=2,font=fnt)
                                entry.grid(row=i,column=j,padx=2,pady=2,sticky=tk.EW)
                                entry.configure(text=data[i][j])
                                if i==0:
                                        entry.configure(fg='red',font=fntit)  #colors and italicises
header
```

```
        def viewone(): #view single user

                def getuserinfo():     #gets user info from DB

                        if not uname.get()=='' and not uname.get().isspace():
                                if uname.get() in user_list:
                                        sql='select * from users where uname=%s'
                                        val=(uname.get(),)
                                        cur.execute(sql,val)
                                        c=cur.fetchall()
                                        user_id=c[0][0]
                                        user_fname=c[0][1]
                                        user_email=c[0][2]
                                        user_num=c[0][3]
                                        user_uname=c[0][4]
                                        user_passwd=c[0][5]

                                        data=[('User ID',user_id),('Full Name',user_fname),
('Electronic Mail',user_email),('Phone Number',user_num),('Username',user_uname),
('Password',user_passwd)]

                                        rows=len(data)
                                        cols=len(data[0])

        tk.Label(frame3,font=fntit,text='Data').grid(row=0,column=0,sticky=tk.W)
                                        for i in range(rows):
        #drawing the table in GUI
                                                for j in range(cols):
                                                        entry =
tk.Label(frame2,borderwidth=1,relief='solid',padx=10,height=2,width=25,font=fnt)

        entry.grid(row=i,column=j,padx=2,pady=2,sticky=tk.EW)
                                                        entry.configure(text=data[i][j])
                                                        if j==0:

        entry.configure(fg='red',font=fntit,width=15) #colors and italicises header
                                else:
                                        messagebox.showerror('Error','Username \''+uname.get()+'\'
does not exist.',parent=viewone_win)
                        else:
                                messagebox.showerror('Error','Please enter the
username.',parent=viewone_win)

                viewone_win=tk.Toplevel()
                viewone_win.title('View user details')
                viewone_win.resizable(False,False)

                frame1=tk.Frame(viewone_win)
                frame1.grid(row=0,column=0,padx=10,pady=10,sticky=tk.EW)

                frame2=tk.Frame(viewone_win)
                frame2.grid(row=2,column=0,padx=10,pady=10,sticky=tk.EW)

                frame3=tk.Frame(viewone_win)
                frame3.grid(row=1,column=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select uname from users')
                a=cur.fetchall()
                user_list=[]
                for i in a:
                        user_list.append(i[0])

                img14=tk.PhotoImage(file='icons/searchusr.png')
                img=tk.Label(frame1,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(frame1,font=h1fnt,text='View user
details').grid(row=0,column=1,padx=10,pady=10,sticky=tk.W)

                tk.Label(frame1,font=fnt,text='Enter
username.').grid(row=4,column=1,padx=10,pady=10,sticky=tk.W)
                n=tk.StringVar()
                uname=ttk.Combobox(frame1,textvariable=n,font=fnt)
```

```
                uname.grid(row=5,column=1,padx=10,pady=10,sticky=tk.EW)
                uname['values']=user_list

                submit=tk.Button(frame1,font=fntit,text='Submit',command=getuserinfo)
                submit.grid(row=5,column=2,padx=10,pady=10)

                #Binds Enter to submit function.
                viewone_win.bind('<Return>',lambda event:getuserinfo())

        def delone():  #delete user
                delone_win=tk.Toplevel()
                delone_win.resizable(False,False)
                delone_win.title('Delete user')

                def delete_user(): #deletes user from DB.

                        if not uname.get()=='' and not uname.get().isspace():
                                if uname.get() in users_list:
                                        messagebox.showwarning('','This operation will delete\nthe
user permanently.\nContinue?',parent=delone_win)
                                        confirm=messagebox.askyesno('','Do you wish to delete the user
'+uname.get()+'?',parent=delone_win)
                                        if confirm == True:
                                                sql='delete from users where uname =%s'
                                                val=(uname.get(),)
                                                cur.execute(sql,val)
                                                con.commit()
                                                messagebox.showinfo('','User '+uname.get()+'
deleted.',parent=delone_win)
                                                delone_win.destroy()
                                        else:
                                                messagebox.showinfo('','User '+uname.get()+' not
deleted.\nThe database has not been modified.',parent=delone_win)
                                else:
                                        messagebox.showerror('Error','Username \''+uname.get()+'\'
does not exist.',parent=delone_win)
                        else:
                                messagebox.showerror('','Please enter the
username.',parent=delone_win)

                img14=tk.PhotoImage(file='icons/ban_user.png')
                img=tk.Label(delone_win,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(delone_win,text='Delete a
user.',font=h1fnt).grid(column=1,row=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select uname from users')
                d=cur.fetchall()
                users_list=[]
                for i in d:
                        users_list.append(i[0])

                tk.Label(delone_win,text='Select a
user.',font=fntit).grid(column=1,row=4,padx=10,pady=10,sticky=tk.W)

                n=tk.StringVar()
                uname=ttk.Combobox(delone_win,textvariable=n,font=fnt,width=19)
                uname.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=10)
                uname['values']=users_list

                delbtn=tk.Button(delone_win,text='Delete',font=fntit,command=delete_user,fg='red')
                delbtn.grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)
                delone_win.bind('<Return>',lambda event:delete_user())

        def passwd():  #changes password for user
                passwd_win=tk.Toplevel()
                passwd_win.resizable(False,False)
                passwd_win.title('Change password for user')

                def ch_user_passwd():  #changes password in db
                        if (not uname.get()=='' and not uname.get().isspace()) and (not
npass.get()=='' and not npass.get().isspace()):
                                if uname.get() in users_list:
```

```
                                        confirm=messagebox.askyesno('','Do you wish to change the
password of '+uname.get()+'?',parent=passwd_win)
                                        if confirm == True:
                                                sql='update users set passwd=%s where uname=%s'
                                                val=(npass.get(),uname.get())
                                                cur.execute(sql,val)
                                                con.commit()
                                                messagebox.showinfo('','Password for '+uname.get()+'\
nchanged.',parent=passwd_win)
                                                passwd_win.destroy()
                                        else:
                                                messagebox.showinfo('','Password for '+uname.get()+'
has not been changed..\nThe databasehas not\nbeen modified.',parent=passwd_win)
                                else:
                                        messagebox.showerror('Error','Username \''+uname.get()+'\'
does not exist.',parent=passwd_win)
                        else:
                                messagebox.showerror('','Do not leave any fields
blank.',parent=passwd_win)

                img14=tk.PhotoImage(file='icons/passwd.png')
                img=tk.Label(passwd_win,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(passwd_win,text='Change password\nfor
user',font=h1fnt,justify=tk.LEFT).grid(column=1,row=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select uname from users')
                d=cur.fetchall()
                users_list=[]
                for i in d:
                        users_list.append(i[0])

                n=tk.StringVar()


tk.Label(passwd_win,text='Username',font=fnt).grid(column=0,row=5,sticky=tk.E,padx=10,pady=10)
                uname=ttk.Combobox(passwd_win,textvariable=n,font=fnt,width=19)
                uname.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=10)
                uname['values']=users_list
                uname.current(0)

                tk.Label(passwd_win,text='New
password',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
                npass=tk.Entry(passwd_win,font=fnt,show='*')
                npass.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

                subbtn=tk.Button(passwd_win,text='Make changes',font=fntit,command=ch_user_passwd)
                subbtn.grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)
                passwd_win.bind('<Return>',lambda event:ch_user_passwd())

        def register():#adds user
                uuid='U'+str(rd.randint(10000,99999))

                def reguser(): #adds user to db

                        reg_fname_inp=reg_fname.get()
                        reg_email_inp=reg_email.get()
                        reg_num_inp=reg_num.get()
                        reg_uname_inp=reg_uname.get().lower()
                        reg_passwd_inp=reg_passwd.get()

                        cur.execute('select uname from users')
                        users=cur.fetchall()

                        b=(reg_uname_inp,)
                        if (not reg_fname_inp.isspace()==True and not reg_fname_inp=='') and (not
reg_email_inp.isspace()==True and not reg_email_inp=='') and (not reg_num_inp.isspace()==True and
not reg_num_inp=='') and (not reg_uname_inp.isspace()==True and not reg_uname_inp=='') and (not
reg_passwd_inp.isspace()==True and not reg_passwd_inp==''):         #checks if inputs are not empty
or contains spaces
                                if b not in users:
                                        if '@' in reg_email_inp and '.' in reg_email_inp:
```

```
                                        if len(reg_num_inp) == 10:
                                                regsql='insert into users values(%s,%s,%s,%s,
%s,%s)'

        regval=(uuid,reg_fname_inp,reg_email_inp,reg_num_inp,reg_uname_inp,reg_passwd_inp)

                                                cur.execute(regsql,regval)
                                                con.commit()

                                                messagebox.showinfo('','The new user
'+reg_uname_inp+'\nhas been successfully registered.',parent=regwin)
                                                regwin.destroy()
                                        else:
                                                messagebox.showerror('Error','Invalid phone
number entered.',parent=regwin)
                                    else:
                                                messagebox.showerror('Error','Invalid electronic mail
ID entered.',parent=regwin)
                                else:

                                        messagebox.showerror('Error','Username '+reg_uname_inp+'\
nalready exists.',parent=regwin)
                        else:
                                messagebox.showerror('Error','Please do not leave any fields
blank.',parent=regwin)

                regwin=tk.Toplevel()
                regwin.title('Add user')
                regwin.resizable(False, False)

                img15=tk.PhotoImage(file='icons/adduser.png')
                img=tk.Label(regwin,image=img15,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10,sticky=tk.E)
                img.image=img15

                tk.Label(regwin,text='Add
user ... ',font=h1fnt).grid(column=1,row=0,padx=10,pady=10,sticky=tk.W)

                tk.Label(regwin,text='ID',font=fnt).grid(column=0,row=3,sticky=tk.E,padx=10,pady=10)
                tk.Label(regwin,text=uuid,font=fnt).grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

                tk.Label(regwin,text='1. Personal
info',font=fntit).grid(column=0,row=5,sticky=tk.W,padx=10,pady=10)


        tk.Label(regwin,text='Name',font=fnt).grid(column=0,row=6,sticky=tk.E,padx=10,pady=10)
                reg_fname=tk.Entry(regwin,font=fnt)
                reg_fname.grid(column=1,row=6,sticky=tk.EW,padx=10,pady=10)

                tk.Label(regwin,text='Electronic mail
ID',font=fnt).grid(column=0,row=7,sticky=tk.E,padx=10,pady=10)
                reg_email=tk.Entry(regwin,font=fnt)
                reg_email.grid(column=1,row=7,sticky=tk.EW,padx=10,pady=10)

                tk.Label(regwin,text='Phone
number',font=fnt).grid(column=0,row=8,sticky=tk.E,padx=10,pady=10)
                reg_num=tk.Entry(regwin,font=fnt)
                reg_num.grid(column=1,row=8,sticky=tk.EW,padx=10,pady=10)

                tk.Label(regwin,text='2. Login
info',font=fntit).grid(column=0,row=10,sticky=tk.W,padx=10,pady=10)


        tk.Label(regwin,text='Username',font=fnt).grid(column=0,row=11,sticky=tk.E,padx=10,pady=10)
                reg_uname=tk.Entry(regwin,font=fnt)
                reg_uname.grid(column=1,row=11,sticky=tk.EW,padx=10,pady=10)


        tk.Label(regwin,text='Password',font=fnt).grid(column=0,row=12,sticky=tk.E,padx=10,pady=10)
                reg_passwd=tk.Entry(regwin,show='*',font=fnt)
                reg_passwd.grid(column=1,row=12,sticky=tk.EW,padx=10,pady=10)

                regsubmit=tk.Button(regwin,text='Register',command=reguser,font=fntit)
                regsubmit.grid(column=1,row=14,padx=10,pady=10,sticky=tk.W)
                regwin.bind('<Return>',lambda event:reguser())
```

```
        tk.Grid.columnconfigure(manageuserwin,0,weight=1)

        #FRAME 1
        tk.Grid.rowconfigure(manageuserwin,0,weight=1)
        f1=tk.Frame(manageuserwin)
        f1.grid(row=0,column=0,sticky=tk.NSEW)

        #frame 1 grid
        tk.Grid.columnconfigure(f1,0,weight=1)
        tk.Grid.columnconfigure(f1,1,weight=1)

        tk.Grid.rowconfigure(f1,0,weight=1)
        img6=tk.PhotoImage(file='icons/people.png')
        tk.Label(f1,image=img6).grid(column=0,row=0,sticky=tk.E,padx=10,pady=10)
        himg=tk.Label(f1,text=('Manage the users ... '),font=h1fnt)
        himg.grid(column=1,row=0,sticky=tk.W,padx=10,pady=10)
        himg.image=img6
        tk.Label(f1,text=('Connected to database: '+con.database),font=('IBM Plex
Sans',12),justify=tk.LEFT,fg='green').grid(column=1,row=1,sticky=tk.W,padx=10,pady=10)


ttk.Separator(f1,orient='horizontal').grid(column=0,row=2,sticky=tk.EW,padx=10,pady=10,columnspan=2)

        #FRAME 2
        tk.Grid.rowconfigure(manageuserwin,1,weight=1)
        f2=tk.Frame(manageuserwin)
        f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

        #frame 2 grid
        tk.Grid.columnconfigure(f2,0,weight=1)
        tk.Grid.columnconfigure(f2,1,weight=1)
        tk.Grid.columnconfigure(f2,2,weight=1)
        tk.Grid.columnconfigure(f2,3,weight=1)

        tk.Label(f2,text='You
can:',font=fntit,justify=tk.LEFT).grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

        tk.Grid.rowconfigure(f2,5,weight=1)

        img8=tk.PhotoImage(file='icons/preview.png')
        tbviewbtn=tk.Button(f2,text='view all',image=img8,font=fnt,command=viewall)
        tbviewbtn.grid(column=0,row=5,padx=10,pady=10,sticky=tk.E)
        tbviewbtn.image=img8
        tk.Label(f2,text='View all user
details.',font=fnt,fg='blue').grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

        img10=tk.PhotoImage(file='icons/searchusr.png')
        viewbtn=tk.Button(f2,text='viewone',image=img10,font=fnt,command=viewone)
        viewbtn.grid(column=2,row=5,padx=10,pady=10,sticky=tk.E)
        viewbtn.imageg=img10
        tk.Label(f2,text='View a single user\'s
details.',font=fnt).grid(column=3,row=5,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,6,weight=1)

        img7=tk.PhotoImage(file='icons/adduser.png')
        tbviewbtn=tk.Button(f2,text='add',image=img7,font=fnt,command=register)
        tbviewbtn.grid(column=0,row=6,padx=10,pady=10,sticky=tk.E)
        tbviewbtn.image=img7
        tk.Label(f2,text='Add a
user.',font=fnt,fg='green').grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)

        img11=tk.PhotoImage(file='icons/passwd.png')
        passbtn=tk.Button(f2,text='passwd',image=img11,font=fnt,command=passwd)
        passbtn.grid(column=2,row=6,padx=10,pady=10,sticky=tk.E)
        passbtn.image=img11
        tk.Label(f2,text='Change the password for a
user.',font=fnt).grid(column=3,row=6,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,7,weight=1)
        img12=tk.PhotoImage(file='icons/ban_user.png')
        delbtn=tk.Button(f2,text='del',image=img12,font=fnt,command=delone)
        delbtn.grid(column=0,row=7,padx=10,pady=10,sticky=tk.E)
```

```
        delbtn.image=img12
        tk.Label(f2,text='Delete a
user.',font=fnt,fg='red').grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,8,weight=1)
        tk.Message(f2,text='WARNING: This will delete\na user\'s profile\nfrom the system
permanently.',width=500,font=fnt,fg='white',bg='red').grid(column=1,row=8,padx=10,pady=10,sticky=tk.
NW)

        tk.Grid.rowconfigure(f2,16,weight=1)

        tk.Grid.rowconfigure(f2,17,weight=1)

def manage_db():               #Manage db
        import mysql.connector as ms
        import tkinter as tk
        import platform as pf
        import ctypes
        import pandas as pd
        from tkinter import ttk
        import os
        from tkinter import messagebox
        from tkinter import scrolledtext

        #Enables DPI scaling on supported versions of Windows
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass

        #mysql connection
        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
        cur=con.cursor()

        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)
        menufnt=('IBM Plex Mono',11)

        dbmainwin=tk.Toplevel()
        dbmainwin.title('Database Manager')

        cur.execute('show tables')                    #creating list of available tables for dropbox
        a=cur.fetchall()
        tables_list=[]
        for i in a:
                tables_list.append(i[0])

        def showtb():  #Show selected table
                if not table.get()=='' and not table.get().isspace():
                        if table.get() in tables_list:
                                dbwin=tk.Toplevel()
                                dbwin.resizable(False,False)
                                dbwin.title(table.get()+' table')
                                sql=str('show columns from '+table.get())            #getting headers
for table
                                cur.execute(sql)
                                a=cur.fetchall()
                                headers_list=[]
                                for x in a:
                                        headers_list.append(x[0])
                                        header=tuple(headers_list)

                                sql2=str('select * from '+table.get())                       #getting
data from table
                                cur.execute(sql2)
                                data=[header]+cur.fetchall()
        #appending header to data

                                rows=len(data)
                                cols=len(data[0])

                                for i in range(rows):
        #drawing the table in GUI
```

```
                                        for j in range(cols):
                                                entry =
tk.Label(dbwin,borderwidth=1,relief='solid',height=2,font=fnt,padx=10)
                                                entry.grid(row=i, column=j,padx=2,pady=2,sticky=tk.EW)
                                                entry.configure(text=data[i][j])
                                                if i==0:
                                                        entry.configure(fg='red',font=fntit)  #colors
and italicises header
                        else:
                                messagebox.showerror('Error','Table '+table.get()+' does not
exist.',parent=dbmainwin)
                else:
                        messagebox.showerror('Error','Please choose a table.',parent=dbmainwin)

        def droptb():  #Drop selected tablr
                if not table.get()=='' and not table.get().isspace():
                        if table.get() in tables_list:
                                messagebox.showwarning('WARNING','The table chosen will be dropped\
nfrom the database permanently.\nContinue?',parent=dbmainwin)
                                confirm=messagebox.askyesno('','Do you wish to drop the
table \''+table.get()+'\'\nalong with its contents ?',parent=dbmainwin)
                                if confirm == True:
                                        sql=str('drop table '+table.get())
                                        cur.execute(sql)
                                        con.commit()
                                        messagebox.showinfo('','The table \''+table.get()+'\'\nhas
been dropped\nfrom the database.',parent=dbmainwin)
                                else:
                                        messagebox.showinfo('','DROP TABLE operation
on \''+table.get()+'\' cancelled.\nThe database has not been modified.',parent=dbmainwin)
                                        pass
                        else:
                                messagebox.showerror('Error','Table '+table.get()+' does not
exist.',parent=dbmainwin)
                else:
                        messagebox.showerror('Error','Please choose a table.',parent=dbmainwin)

        def deltb():   #Delete contents of selected tables
                if not table.get()=='' and not table.get().isspace():
                        if table.get() in tables_list:
                                messagebox.showwarning('WARNING','All the contents of the table chosen
will be deleted permanently.\nContinue?',parent=dbmainwin)
                                confirm=messagebox.askyesno('','Do you wish to delete\nall records
from the table \''+table.get()+'\'?',parent=dbmainwin)
                                if confirm == True:
                                        sql=str('delete from '+table.get())
                                        cur.execute(sql)
                                        con.commit()
                                        messagebox.showinfo('','All records in table \''+table.get()
+'\'\nhave been permenantly deleted\nfrom the database.',parent=dbmainwin)
                                else:
                                        messagebox.showinfo('','DELETE FROM TABLE operation
on \''+table.get()+'\' cancelled.\nThe database has not been modified.',parent=dbmainwin)
                                        pass
                        else:
                                messagebox.showerror('Error','Table '+table.get()+' does not
exist.',parent=dbmainwin)
                else:
                        messagebox.showerror('Error','Please choose a table.',parent=dbmainwin)

        def exporttb():        #Export selected table to CSV
                if not table.get()=='' and not table.get().isspace():
                        if table.get() in tables_list:
                                df=pd.read_sql('select * from '+table.get(),con)
                                df.set_index(df.columns[0],inplace=True)

                                def export_to_csv():
                                        path_input=path.get()
                                        if not path_input=='' and not path_input.isspace():
                                                df.reset_index(inplace=True)
                                                os.chdir('export')
                                                df.to_csv(path_input,index=False)
                                                os.chdir('./..')
                                                messagebox.showinfo('','Table '+table.get()+' exported
to '+path_input+'.',parent=export_win)
```

```
                                        export_win.destroy()
                                    else:
                                        messagebox.showerror('Error','Please enter a
filename.',parent=export_win)

                                export_win=tk.Toplevel()
                                export_win.resizable(False,False)
                                export_win.title('Export to CSV')

                                tk.Label(export_win,font=h1fnt,text='Export to CSV
file ... ').grid(row=0,column=0,padx=10,pady=10,sticky=tk.NW)

                                tk.Label(export_win,font=('IBM Plex Mono',12,'bold
italic'),text='Data',justify=tk.LEFT).grid(row=1,column=0,padx=10,pady=10,sticky=tk.W)

                                tk.Label(export_win,font=fnt,text='Enter the name of the\nCSV file.\
nThe file will be saved to the\n\'export\'
folder.',justify=tk.LEFT).grid(row=3,column=0,padx=10,pady=10,sticky=tk.W)

                                path=tk.Entry(export_win,font=fnt)
                                path.grid(row=5,column=0,padx=10,pady=10,sticky=tk.EW)


        submit=tk.Button(export_win,font=fnt,text='Export',command=export_to_csv)
                                submit.grid(row=6,column=0,padx=10,pady=10)

                                #Binds Enter key to export function
                                export_win.bind('<Return>',lambda event:export_to_csv())
                        else:
                                messagebox.showerror('Error','Table '+table.get()+' does not
exist.',parent=dbmainwin)
                else:
                        messagebox.showerror('Error','Please choose a table.',parent=dbmainwin)

        def help():             #View help page.
                helpwin=tk.Toplevel()
                helpwin.resizable(False,False)
                helpwin.title('Help')

                img14=tk.PhotoImage(file='icons/help.png')
                img=tk.Label(helpwin,image=img14)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(helpwin,text='What is the difference between\n\'deleting from\'
and \'dropping\' a
table?',font=h1fnt,justify=tk.LEFT).grid(row=0,column=1,padx=10,pady=10,sticky=tk.W)
                txt='''Deleting' from a table performs the SQL DELETE FROM
operation, which, by default, deletes all records
from the table, whilst keeping the table structure
intact.

On the other hand, 'dropping' a table performs the
SQL DROP TABLE deletes the table structure from the
database along with its contents.'''

                a=scrolledtext.ScrolledText(helpwin,wrap=tk.WORD,width=30,height=10,font=fnt)
                a.grid(row=3,column=1,padx=10,pady=10,sticky=tk.EW)
                a.insert(tk.INSERT,txt)
                a.configure(state='disabled')

        menubar=tk.Menu(dbmainwin)

        user=tk.Menu(menubar,tearoff=0)
        menubar.add_cascade(label='Help',menu=user,font=menufnt)

        user.add_command(label='DELETE FROM vs DROP table',command=help,font=menufnt,underline=0)

        dbmainwin.config(menu=menubar)

        tk.Grid.columnconfigure(dbmainwin,0,weight=1)

        #FRAME 1
        tk.Grid.rowconfigure(dbmainwin,0,weight=1)
        f1=tk.Frame(dbmainwin)
```

```
        f1.grid(row=0,column=0,sticky=tk.NSEW)

        #frame 1 grid
        tk.Grid.columnconfigure(f1,0,weight=1)
        tk.Grid.columnconfigure(f1,1,weight=1)

        tk.Grid.rowconfigure(f1,0,weight=1)

        img6=tk.PhotoImage(file='icons/dataset.png')
        himg=tk.Label(f1,image=img6)
        himg.grid(column=0,row=0,padx=10,pady=10,sticky=tk.E)
        himg.image=img6

        tk.Label(f1,text=('Manage the
databases ... '),font=h1fnt).grid(column=1,row=0,sticky=tk.W,padx=10,pady=10)
        tk.Grid.rowconfigure(f1,1,weight=1)
        tk.Label(f1,text=('Connected to database: '+con.database),font=('IBM Plex
Sans',12),justify=tk.LEFT,fg='green').grid(column=1,row=1,sticky=tk.W,padx=10,pady=1)

ttk.Separator(f1,orient='horizontal').grid(column=0,row=2,sticky=tk.EW,padx=10,pady=10,columnspan=2)
        #FRAME 2
        tk.Grid.rowconfigure(dbmainwin,1,weight=1)
        f2=tk.Frame(dbmainwin)
        f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

        #frame 2 grid
        tk.Grid.columnconfigure(f2,0,weight=1)
        tk.Grid.columnconfigure(f2,1,weight=1)
        tk.Grid.columnconfigure(f2,2,weight=1)
        tk.Grid.columnconfigure(f2,3,weight=1)

        tk.Label(f2,text='Choose a
table.',font=fntit,justify=tk.LEFT).grid(column=1,row=4,sticky=tk.W,padx=10,pady=10)
        img7=tk.PhotoImage(file='icons/table.png')
        h2img=tk.Label(f2,image=img7)
        h2img.grid(column=0,row=4,sticky=tk.E,padx=10,pady=10)
        h2img.image=img7

        tk.Grid.rowconfigure(f2,5,weight=1)
        n=tk.StringVar()
        table=ttk.Combobox(f2,textvariable=n,font=fnt,state='readonly')
        table.grid(row=5,column=1,padx=10,pady=10,sticky=tk.EW)
        table['values']=tables_list

        tk.Label(f2,text='You
can:',font=fntit,justify=tk.LEFT).grid(column=1,row=6,sticky=tk.W,padx=10,pady=10)

        tk.Grid.rowconfigure(f2,7,weight=1)
        img8=tk.PhotoImage(file='icons/preview.png')
        tbviewbtn=tk.Button(f2,text='viewtable',image=img8,font=fnt,command=showtb)
        tbviewbtn.grid(column=0,row=7,padx=10,pady=10,sticky=tk.E)
        tbviewbtn.image=img8
        tk.Label(f2,text='View the
table.',font=fnt,fg='blue').grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,8,weight=1)
        img9=tk.PhotoImage(file='icons/export.png')
        tbexportbtn=tk.Button(f2,text='export table',image=img9,font=fnt,command=exporttb)
        tbexportbtn.grid(column=0,row=8,padx=10,pady=10,sticky=tk.E)
        tbexportbtn.image=img9
        tk.Label(f2,text='Export the table\nto CSV
file.',font=fnt,fg='green',justify=tk.LEFT).grid(column=1,row=8,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,9,weight=1)
        img10=tk.PhotoImage(file='icons/delete.png')
        deltbbtn=tk.Button(f2,text='deltable',image=img10,font=fnt,command=deltb)
        deltbbtn.grid(column=0,row=9,padx=10,pady=10,sticky=tk.E)
        deltbbtn.image=img10
        tk.Label(f2,text='Delete all the contents\nof the
table.',font=fnt,justify=tk.LEFT).grid(column=1,row=9,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,10,weight=1)

        tk.Message(f2,text='WARNING:\nThis will delete all the contents of the table chosen
permanently.',font=fnt,fg='white',bg='orange').grid(column=1,row=10,padx=10,sticky=tk.NW)
```

```
        img11=tk.PhotoImage(file='icons/remove.png')
        drptbbtn=tk.Button(f2,text='droptable',image=img11,font=fnt,command=droptb)
        drptbbtn.grid(column=2,row=9,padx=10,pady=10,sticky=tk.E)
        drptbbtn.image=img11
        tk.Label(f2,text='Drop the
table.',font=fnt,fg='red').grid(column=3,row=9,padx=10,pady=10,sticky=tk.W)

        tk.Message(f2,text='WARNING:\nThis will drop the table chosen\nand its contents
permanently.',font=fnt,fg='white',bg='red').grid(column=3,row=10,padx=10,sticky=tk.NW)

        #Bind Enter to show table function
        dbmainwin.bind('<Return>',lambda event:showtb())
```

7. managebkgs.py

```
def bus():     #manage bus bookings
        import mysql.connector as ms
        import tkinter as tk
        import platform as pf
        import ctypes
        from tkinter import ttk
        from tkinter import messagebox

        #Enables DPI scaling on supported versions of Windows
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass

        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
        cur=con.cursor()

        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)

        managebusbkgs=tk.Toplevel()
        managebusbkgs.title('Bus Bookings Manager')


        def viewall(): #View all bookings
                viewall_win=tk.Toplevel()
                viewall_win.title('All bus bookings')
                viewall_win.resizable(False,False)

                header=('Booking ID','Timestamp','Number of
Passengers','Origin','Destination','Date','Time','Bus Type')

                sql2=str('select * from bus_bkgs')                    #getting data from table

                cur.execute(sql2)
                data=[header]+cur.fetchall()                                                #appending header
to data

                rows=len(data)
                cols=len(data[0])

                for i in range(rows):                                                       #drawing the
table in GUI
                        for j in range(cols):
                                entry =
tk.Label(viewall_win,borderwidth=1,relief='solid',padx=10,height=2,font=fnt)
                                entry.grid(row=i, column=j,padx=2,pady=2,sticky=tk.EW)
                                entry.configure(text=data[i][j])
                                if i==0:
                                        entry.configure(fg='red',font=fntit)  #colors and italicises
header

        def viewone(): #View one booking
                def get_busbkginfo():

                        if not bkgid.get()=='' and not bkgid.get().isspace():
                                if bkgid.get() in bus_bkgid_list:
                                        sql='select * from bus_bkgs where bkgid=%s'
                                        val=(bkgid.get(),)
                                        cur.execute(sql,val)
                                        c=cur.fetchall()
                                        bkg_id=c[0][0]
                                        bkg_ts=c[0][1]
                                        bkg_passno=c[0][2]
                                        bkg_org=c[0][3]
                                        bkg_dest=c[0][4]
                                        bkg_date=c[0][5]
                                        bkg_time=c[0][6]
```

```
                                        bkg_type=c[0][7]

                                        e=[('Booking ID',bkg_id),('Timestamp',bkg_ts),('Number of
passengers',bkg_passno),('Origin',bkg_org),('Destination',bkg_dest),('Date',bkg_date),
('Time',bkg_time),('Bus Type',bkg_type)]


                                        rows=len(e)
                                        cols=len(e[0])

        tk.Label(frame3,font=fntit,text='Data').grid(row=0,column=0,sticky=tk.W)
                                        for i in range(rows):
        #drawing the table in GUI
                                                for j in range(cols):
                                                        entry =
tk.Label(frame2,borderwidth=1,relief='solid',padx=10,width=30,height=2,font=fnt)

        entry.grid(row=i,column=j,padx=2,pady=2,sticky=tk.EW)
                                                        entry.configure(text=e[i][j])
                                                        if j==0:
                                                                entry.configure(fg='red',font=fntit)
#colors and italicises header
                                else:
                                        messagebox.showerror('Error','Booking \''+bkgid.get()+'\' does
not exist.',parent=viewone_win)
                        else:
                                messagebox.showerror('Error','Please enter the
booking.',parent=viewone_win)
                viewone_win=tk.Toplevel()
                viewone_win.title('View bus booking')
                viewone_win.resizable(False,False)

                frame1=tk.Frame(viewone_win)
                frame1.grid(row=0,column=0,padx=10,pady=10,sticky=tk.EW)

                frame2=tk.Frame(viewone_win)
                frame2.grid(row=2,column=0,padx=10,pady=10,sticky=tk.EW)

                frame3=tk.Frame(viewone_win)
                frame3.grid(row=1,column=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select bkgid from bus_bkgs')
                a=cur.fetchall()
                bus_bkgid_list=[]
                for i in a:
                        bus_bkgid_list.append(i[0])

                img14=tk.PhotoImage(file='icons/searchusr.png')
                img=tk.Label(frame1,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(frame1,font=h1fnt,text='View bus booking
details').grid(row=0,column=1,padx=10,pady=10,sticky=tk.W)

                tk.Label(frame1,font=fnt,text='Enter booking
ID.').grid(row=4,column=1,padx=10,pady=10,sticky=tk.W)
                n=tk.StringVar()
                bkgid=ttk.Combobox(frame1,textvariable=n,font=fnt)
                bkgid.grid(row=5,column=1,padx=10,pady=10,sticky=tk.EW)
                bkgid['values']=bus_bkgid_list

                submit=tk.Button(frame1,font=fntit,text='Submit',command=get_busbkginfo)
                submit.grid(row=5,column=2,padx=10,pady=10)
                viewone_win.bind('<Return>',lambda event:get_busbkginfo())

        def delone():  #Delete booking
                delone_win=tk.Toplevel()
                delone_win.resizable(False,False)
                delone_win.title('Delete bus booking')

                def delete_busbkg():
                        if not bkgid.get()=='' and not bkgid.get().isspace():
                                if bkgid.get() in bus_bkgid_list:
```

```
                                        messagebox.showwarning('','This operation will delete\nthe
booking selected permanently.\nContinue?',parent=delone_win)
                                        confirm=messagebox.askyesno('','Do you wish to delete the
booking '+bkgid.get()+'?',parent=delone_win)
                                        if confirm == True:
                                                sql='delete from bus_bkgs where bkgid =%s'
                                                val=(bkgid.get(),)
                                                cur.execute(sql,val)
                                                con.commit()
                                                messagebox.showinfo('','Booking '+bkgid.get()+'
deleted.',parent=delone_win)
                                                delone_win.destroy()
                                        else:
                                                messagebox.showinfo('','Booking '+bkgid.get()+' not
deleted.\nThe database has not been modified.',parent=delone_win)
                                else:
                                        messagebox.showerror('Error','Bookinge \''+bkgid.get()+'\'
does not exist.',parent=delone_win)
                        else:
                                messagebox.showerror('','Please enter the booking
ID.',parent=delone_win)

                img14=tk.PhotoImage(file='icons/delete_bkgs.png')
                img=tk.Label(delone_win,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(delone_win,text='Delete a bus
booking',font=h1fnt).grid(column=1,row=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select bkgid from bus_bkgs')
                d=cur.fetchall()
                bus_bkgid_list=[]
                for i in d:
                        bus_bkgid_list.append(str(i[0]))

                tk.Label(delone_win,text='Select a
booking.',font=fntit).grid(column=1,row=4,padx=10,pady=10,sticky=tk.W)

                n=tk.StringVar()
                bkgid=ttk.Combobox(delone_win,textvariable=n,font=fnt,width=19)
                bkgid.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=10)
                bkgid['values']=bus_bkgid_list

                delbtn=tk.Button(delone_win,text='Delete',font=fntit,command=delete_busbkg,fg='red')
                delbtn.grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)
                delone_win.bind('<Return>',lambda event:delete_busbkg())

        tk.Grid.columnconfigure(managebusbkgs,0,weight=1)

        #FRAME 1
        tk.Grid.rowconfigure(managebusbkgs,0,weight=1)
        f1=tk.Frame(managebusbkgs)
        f1.grid(row=0,column=0,sticky=tk.NSEW)

        #frame 1 grid
        tk.Grid.columnconfigure(f1,0,weight=1)
        tk.Grid.columnconfigure(f1,1,weight=1)

        tk.Grid.rowconfigure(f1,0,weight=1)
        img6=tk.PhotoImage(file='icons/bus.png')
        himg=tk.Label(f1,image=img6)
        himg.grid(column=0,row=0,sticky=tk.E,padx=10,pady=10)
        himg.image=img6
        tk.Label(f1,text=('Manage the bus
booking ... '),font=h1fnt).grid(column=1,row=0,sticky=tk.W,padx=10,pady=10)
        tk.Label(f1,text=('Connected to database: '+con.database),font=('IBM Plex
Sans',12),justify=tk.LEFT,fg='green').grid(column=1,row=1,sticky=tk.W,padx=10,pady=10)

ttk.Separator(f1,orient='horizontal').grid(column=0,row=2,sticky=tk.EW,padx=10,pady=10,columnspan=2)
        #FRAME 2
        tk.Grid.rowconfigure(managebusbkgs,1,weight=1)
        f2=tk.Frame(managebusbkgs)
        f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)
```

```
        #frame 2 grid
        tk.Grid.columnconfigure(f2,0,weight=1)
        tk.Grid.columnconfigure(f2,1,weight=1)
        tk.Grid.columnconfigure(f2,2,weight=1)
        tk.Grid.columnconfigure(f2,3,weight=1)

        tk.Label(f2,text='You
can:',font=fntit,justify=tk.LEFT).grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

        tk.Grid.rowconfigure(f2,5,weight=1)
        img8=tk.PhotoImage(file='icons/preview.png')
        tbviewbtn=tk.Button(f2,text='view all',image=img8,font=fnt,command=viewall)
        tbviewbtn.grid(column=0,row=5,padx=10,pady=10,sticky=tk.E)
        tbviewbtn.image=img8
        tk.Label(f2,text='View all booking
details.',font=fnt,fg='blue').grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

        img10=tk.PhotoImage(file='icons/search_bkgs.png')
        viewbtn=tk.Button(f2,text='viewone',image=img10,font=fnt,command=viewone)
        viewbtn.grid(column=2,row=5,padx=10,pady=10,sticky=tk.E)
        viewbtn.image=img10
        tk.Label(f2,text='View a single booking
details.',font=fnt).grid(column=3,row=5,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,7,weight=1)
        img12=tk.PhotoImage(file='icons/delete_bkgs.png')
        delbtn=tk.Button(f2,text='del',image=img12,font=fnt,command=delone)
        delbtn.grid(column=0,row=7,padx=10,pady=10,sticky=tk.E)
        delbtn.image=img12
        tk.Label(f2,text='Delete a
booking.',font=fnt,fg='red').grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)
        tk.Grid.rowconfigure(f2,8,weight=1)
        tk.Message(f2,text='WARNING: This will delete\nthe booking selected\nfrom the system
permanently.',width=500,font=fnt,fg='white',bg='red').grid(column=1,row=8,padx=10,pady=10,sticky=tk.
NW)


        tk.Grid.rowconfigure(f2,16,weight=1)

def taxi():    #Manage taxi bookings
        import mysql.connector as ms
        import tkinter as tk
        import platform as pf
        import ctypes
        from tkinter import ttk
        from tkinter import messagebox

        #Enables DPI scaling on supported versions of Windows
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass

        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')
        cur=con.cursor()

        fnt=('IBM Plex Mono',12)
        fntit=('IBM Plex Mono',12,'italic')
        h1fnt=('IBM Plex Sans',24)

        managetaxibkgs=tk.Toplevel()
        managetaxibkgs.title('Taxi Bookings Manager')

        def viewall():  #View all bookings
                viewall_win=tk.Toplevel()
                viewall_win.title('All taxi bookings')
                viewall_win.resizable(False,False)

                header=('Booking ID','Timestamp','Origin','Destination','Date','Time','Taxi Type')

                sql2=str('select * from taxi_bkgs')                  #getting data from table
                cur.execute(sql2)
                data=[header]+cur.fetchall()                                        #appending header
to data
```

```
                rows=len(data)
                cols=len(data[0])

                for i in range(rows):                                                   #drawing the
table in GUI
                        for j in range(cols):
                                entry =
tk.Label(viewall_win,borderwidth=1,relief='solid',padx=10,height=2,font=fnt)
                                entry.grid(row=i, column=j,padx=2,pady=2,sticky=tk.EW)
                                entry.configure(text=data[i][j])
                                if i==0:
                                        entry.configure(fg='red',font=fntit)  #colors and italicises
header

        def viewone(): #View one bookings
                def get_taxibkginfo():

                        if not bkgid.get()=='' and not bkgid.get().isspace():
                                if bkgid.get() in taxi_bkgid_list:
                                        sql='select * from taxi_bkgs where bkgid=%s'
                                        val=(bkgid.get(),)
                                        cur.execute(sql,val)
                                        c=cur.fetchall()
                                        bkg_id=c[0][0]
                                        bkg_ts=c[0][1]
                                        bkg_org=c[0][2]
                                        bkg_dest=c[0][3]
                                        bkg_date=c[0][4]
                                        bkg_time=c[0][5]
                                        bkg_type=c[0][6]

                                        e=[('Booking ID',bkg_id),('Timestamp',bkg_ts),
('Origin',bkg_org),('Destination',bkg_dest),('Date',bkg_date),('Time',bkg_time),('Taxi
Type',bkg_type)]

                                        rows=len(e)
                                        cols=len(e[0])

        tk.Label(frame3,font=fntit,text='Data').grid(row=0,column=0,sticky=tk.W)
                                        for i in range(rows):
        #drawing the table in GUI
                                                for j in range(cols):
                                                        entry =
tk.Label(frame2,borderwidth=1,relief='solid',padx=10,width=30,height=2,font=fnt)

        entry.grid(row=i,column=j,padx=2,pady=2,sticky=tk.EW)
                                                        entry.configure(text=e[i][j])
                                                        if j==0:
                                                                entry.configure(fg='red',font=fntit)
#colors and italicises header
                                else:
                                        messagebox.showerror('Error','Booking \''+bkgid.get()+'\' does
not exist.',parent=viewone_win)
                        else:
                                messagebox.showerror('Error','Please enter the
booking.',parent=viewone_win)
                viewone_win=tk.Toplevel()
                viewone_win.title('View taxi booking')
                viewone_win.resizable(False,False)

                frame1=tk.Frame(viewone_win)
                frame1.grid(row=0,column=0,padx=10,pady=10,sticky=tk.EW)

                frame2=tk.Frame(viewone_win)
                frame2.grid(row=2,column=0,padx=10,pady=10,sticky=tk.EW)

                frame3=tk.Frame(viewone_win)
                frame3.grid(row=1,column=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select bkgid from taxi_bkgs')
                a=cur.fetchall()
                taxi_bkgid_list=[]
                for i in a:
                        taxi_bkgid_list.append(i[0])
```

```
                img14=tk.PhotoImage(file='icons/searchusr.png')
                img=tk.Label(frame1,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(frame1,font=h1fnt,text='View taxi booking
details ... ').grid(row=0,column=1,padx=10,pady=10,sticky=tk.W)

                tk.Label(frame1,font=fnt,text='Enter booking
ID.').grid(row=4,column=1,padx=10,pady=10,sticky=tk.W)
                n=tk.StringVar()
                bkgid=ttk.Combobox(frame1,textvariable=n,font=fnt)
                bkgid.grid(row=5,column=1,padx=10,pady=10,sticky=tk.EW)
                bkgid['values']=taxi_bkgid_list

                submit=tk.Button(frame1,font=fntit,text='Submit',command=get_taxibkginfo)
                submit.grid(row=5,column=2,padx=10,pady=10)
                viewone_win.bind('<Return>',lambda event:get_taxibkginfo())

        def delone(): #Delete one booking.
                delone_win=tk.Toplevel()
                delone_win.resizable(False,False)
                delone_win.title('Delete taxi booking')
                def delete_taxi_bkg():
                        if not bkgid.get()=='' and not bkgid.get().isspace():
                                if bkgid.get() in taxi_bkgid_list:
                                        messagebox.showwarning('','This operation will delete\nthe
booking selected permanently.\nContinue?',parent=delone_win)
                                        confirm=messagebox.askyesno('','Do you wish to delete the
booking '+bkgid.get()+'?',parent=delone_win)
                                        if confirm == True:
                                                sql='delete from taxi_bkgs where bkgid =%s'
                                                val=(bkgid.get(),)
                                                cur.execute(sql,val)
                                                con.commit()
                                                messagebox.showinfo('','Booking '+bkgid.get()+'
deleted.',parent=delone_win)
                                                delone_win.destroy()
                                        else:
                                                messagebox.showinfo('','Booking '+bkgid.get()+' not
deleted.\nThe database has not been modified.',parent=delone_win)
                                else:
                                        messagebox.showerror('Error','Bookinge \''+bkgid.get()+'\'
does not exist.',parent=delone_win)
                        else:
                                messagebox.showerror('','Please enter the booking
ID.',parent=delone_win)

                img14=tk.PhotoImage(file='icons/delete_bkgs.png')
                img=tk.Label(delone_win,image=img14,font=h1fnt)
                img.grid(column=0,row=0,padx=10,pady=10)
                img.image=img14

                tk.Label(delone_win,text='Delete a taxi
booking',font=h1fnt).grid(column=1,row=0,padx=10,pady=10,sticky=tk.W)

                cur.execute('select bkgid from taxi_bkgs')
                d=cur.fetchall()
                taxi_bkgid_list=[]
                for i in d:
                        taxi_bkgid_list.append(str(i[0]))

                tk.Label(delone_win,text='Select a
booking.',font=fntit).grid(column=1,row=4,padx=10,pady=10,sticky=tk.W)

                n=tk.StringVar()
                bkgid=ttk.Combobox(delone_win,textvariable=n,font=fnt,width=19)
                bkgid.grid(column=1,row=5,sticky=tk.EW,padx=10,pady=10)
                bkgid['values']=taxi_bkgid_list


        delbtn=tk.Button(delone_win,text='Delete',font=fntit,command=delete_taxi_bkg,fg='red')
                delbtn.grid(column=1,row=6,padx=10,pady=10,sticky=tk.W)
                delone_win.bind('<Return>',lambda event:delete_taxi_bkg())
```

```
        tk.Grid.columnconfigure(managetaxibkgs,0,weight=1)

        #FRAME 1
        tk.Grid.rowconfigure(managetaxibkgs,0,weight=1)
        f1=tk.Frame(managetaxibkgs)
        f1.grid(row=0,column=0,sticky=tk.NSEW)

        #frame 1 grid
        tk.Grid.columnconfigure(f1,0,weight=1)
        tk.Grid.columnconfigure(f1,1,weight=1)

        tk.Grid.rowconfigure(f1,0,weight=1)
        img6=tk.PhotoImage(file='icons/taxi.png')
        himg=tk.Label(f1,image=img6)
        himg.grid(column=0,row=0,sticky=tk.E,padx=10,pady=10)
        himg.image=img6
        tk.Label(f1,text=('Manage the taxi
booking ... '),font=h1fnt).grid(column=1,row=0,sticky=tk.W,padx=10,pady=10)

        tk.Label(f1,text=('Connected to database: '+con.database),font=('IBM Plex
Sans',12),justify=tk.LEFT,fg='green').grid(column=1,row=1,sticky=tk.W,padx=10,pady=10)

ttk.Separator(f1,orient='horizontal').grid(column=0,row=2,sticky=tk.EW,padx=10,pady=10,columnspan=2)
        #FRAME 2
        tk.Grid.rowconfigure(managetaxibkgs,1,weight=1)
        f2=tk.Frame(managetaxibkgs)
        f2.grid(row=1,column=0,padx=10,pady=10,sticky=tk.NSEW)

        #frame 2 grid
        tk.Grid.columnconfigure(f2,0,weight=1)
        tk.Grid.columnconfigure(f2,1,weight=1)
        tk.Grid.columnconfigure(f2,2,weight=1)
        tk.Grid.columnconfigure(f2,3,weight=1)

        tk.Label(f2,text='You
can:',font=fntit,justify=tk.LEFT).grid(column=1,row=3,sticky=tk.W,padx=10,pady=10)

        tk.Grid.rowconfigure(f2,5,weight=1)
        img8=tk.PhotoImage(file='icons/preview.png')
        tbviewbtn=tk.Button(f2,text='view all',image=img8,font=fnt,command=viewall)
        tbviewbtn.grid(column=0,row=5,padx=10,pady=10,sticky=tk.E)
        tbviewbtn.image=img8
        tk.Label(f2,text='View all booking
details.',font=fnt,fg='blue').grid(column=1,row=5,padx=10,pady=10,sticky=tk.W)

        img10=tk.PhotoImage(file='icons/search_bkgs.png')
        viewbtn=tk.Button(f2,text='viewone',image=img10,font=fnt,command=viewone)
        viewbtn.grid(column=2,row=5,padx=10,pady=10,sticky=tk.E)
        viewbtn.image=img10
        tk.Label(f2,text='View a single booking
details.',font=fnt).grid(column=3,row=5,padx=10,pady=10,sticky=tk.W)

        tk.Grid.rowconfigure(f2,7,weight=1)
        img12=tk.PhotoImage(file='icons/delete_bkgs.png')
        delbtn=tk.Button(f2,text='del',image=img12,font=fnt,command=delone)
        delbtn.grid(column=0,row=7,padx=10,pady=10,sticky=tk.E)
        delbtn.image=img12
        tk.Label(f2,text='Delete a
booking.',font=fnt,fg='red').grid(column=1,row=7,padx=10,pady=10,sticky=tk.W)
        tk.Grid.rowconfigure(f2,8,weight=1)
        tk.Message(f2,text='WARNING: This will delete\nthe booking selected\nfrom the system
permanently.',width=500,font=fnt,fg='white',bg='red').grid(column=1,row=8,padx=10,pady=10,sticky=tk.
NW)

        tk.Grid.rowconfigure(f2,16,weight=1)

        managetaxibkgs.mainloop()
```

8. sysinfo.py

```
def about():   #System information

        #import statements
        import platform as pf
        import tkinter as tk
        from tkinter.ttk import Separator
        import mysql.connector as ms
        from tkinter import scrolledtext
        import ctypes

        #Build number
        build='255'
        build_date='2022-09-24'

        credits_txt='''
Developed by
LIYO K. JOHN - MEGHNATH M.D. - MOHAMMED SAAD
'''

        #Enables DPI scaling on supported Windows versions
        if pf.system()=='Windows':
                try:
                        ctypes.windll.shcore.SetProcessDpiAwareness(True)
                except:
                        pass


        #mysql connection
        con=ms.connect(host='localhost',user='root',password='123456',database='taxi')

        #Fonts
        fnt=('IBM Plex Mono',12)
        h1fnt=('IBM Plex Sans',24)

        about=tk.Toplevel()
        abttitle='About this program'
        about.resizable(False, False)
        about.title(abttitle)


        #Labels
        tk.Label(about,text='About',font=h1fnt).grid(column=0,row=0,columnspan=3)
        tk.Label(about,text=('Build '+build+'
('+build_date+')'),font=fnt).grid(column=0,row=1,columnspan=3)

        logo_img=tk.PhotoImage(file='img/logo150px.png')
        logo=tk.Label(about,image=logo_img)
        logo.grid(column=0,row=2,padx=10,pady=10)
        logo.image=logo_img


        credits=tk.Label(about,font=('IBM Plex Mono',12,'bold
italic'),text=credits_txt,justify=tk.CENTER)
        credits.grid(row=2,column=2,sticky=tk.EW,padx=10,pady=10)


Separator(about,orient='horizontal').grid(column=0,row=5,sticky=tk.EW,padx=10,pady=10,columnspan=3)

        pyimgsrc=tk.PhotoImage(file='img/python.png')
        pyimg=tk.Label(about,image=pyimgsrc)
        pyimg.image=pyimgsrc
        pyimg.grid(column=0,row=6)

        tk.Label(about,text=('Python',pf.python_version()),font=fnt).grid(column=0,row=7,padx=10)
        tk.Label(about,text=('Tkinter',tk.TkVersion),font=fnt).grid(column=0,row=8,padx=10)
        tk.Label(about,text=('MySQL',con.get_server_info()),font=fnt).grid(column=0,row=9,padx=10)


        if pf.system()=='Windows':
                src=tk.PhotoImage(file='img/win.png')
```

```
        elif pf.system()=='Darwin':           #Darwin - macOS
                src=tk.PhotoImage(file='img/macos.png')
        elif pf.system()=='Linux':
                src=tk.PhotoImage(file='img/linux.png')

        osimg=tk.Label(about,image=src)
        osimg.image=src
        osimg.grid(column=2,row=6,padx=10,pady=10)

        #System info
        if pf.system()=='Windows':            #Additional info - Windows systems ONLY
                tk.Label(about,text=(pf.system(),pf.release(),pf.version()),font=('IBM Plex
Mono',12,'bold italic')).grid(column=2,row=7,padx=10)
        else:
                tk.Label(about,text=(pf.system(),pf.release()),font=('IBM Plex Mono',12,'bold
italic')).grid(column=2,row=7,padx=10)

        #Additional distribution info - Linux ONLY

        if pf.system()=='Linux':
                try:
                        linux=pf.freedesktop_os_release()
                        tk.Label(about,text=(linux['NAME']+'
'+linux['VERSION']),font=fnt).grid(column=2,row=8,padx=10)
                except:
                        pass
        else:
                pass


Separator(about,orient='horizontal').grid(column=0,row=10,sticky=tk.EW,padx=10,pady=10,columnspan=3)

        #Hostname and CPU type (e.g.i386 (32-bit); AMD64/x86_64 (64-bit) etc.)
        tk.Label(about,text=pf.node(),font=('IBM Plex Mono',12,'bold
italic')).grid(column=0,row=11,columnspan=3,padx=10)
        tk.Label(about,text=(pf.machine()+'
system'),font=fnt).grid(column=0,row=12,columnspan=3,padx=10)

Separator(about,orient='horizontal').grid(column=0,row=16,sticky=tk.EW,padx=10,pady=10,columnspan=3)


        dbinfo=tk.Label(about,text='Connected to database \''+con.database+'\'',font=fnt)
        dbinfo.grid(column=0,row=18,columnspan=3,padx=10)



Separator(about,orient='horizontal').grid(column=0,row=24,sticky=tk.EW,padx=10,pady=10,columnspan=3)

        #Closes the window
        def close():
                about.destroy()

        img1=tk.PhotoImage(file='icons/close.png')
        cls=tk.Button(about,font=fnt,text='Close',image=img1,command=close)
        cls.grid(column=0,row=25,padx=10,pady=10,columnspan=3)
        cls.image=img1
```