```
from tkinter import *
from PIL import Image, ImageTk
import pandas as pd
from tkinter import filedialog
import inception as ict
from tkinter.ttk import Treeview
from tkinter import messagebox as ms
import sqlite3

with sqlite3.connect("login.db") as db:
    cursor = db.cursor()
cursor.execute("CREATE TABLE IF NOT EXISTS user(username TEXT NOT NULL, password TEXT NOT NULL);")
cursor.execute("SELECT * FROM user")
db.commit()
db.close()

class main:
    def __init__(self,roop):
        self.roop = roop
        self.roop.geometry("500x685+750+100")
        self.ID = StringVar()
        self.password = StringVar()
        self.widgets()

    def login(self):
        with sqlite3.connect("login.db") as db:
            cursor = db.cursor()
        find_user = ("SELECT * FROM user WHERE username = ?AND password = ?")
        cursor.execute(find_user, [(self.ID.get()), (self.password.get())])
        results = cursor.fetchall()
        if results:
            toplevel = Toplevel(self.roop)
            menu(toplevel)
            self.roop.withdraw()
        else:
            ms.showerror("Oops!!", "Username not matched! ")
            self.ID.set("")
            self.password.set("")
        db.close()

    def Generator(self):
        toplevel = Toplevel(self.roop)
        Generator_Frame(toplevel)

    def quit(self):
        sys.exit()

    def widgets(self):

        self.loginf = Frame(self.roop)
        self.logimg = Image.open('./wine_test/wine_tatle.jpg').resize((465, 550), Image.ANTIALIAS)
        self.logimg = ImageTk.PhotoImage(self.logimg)
        Label(self.loginf, image=self.logimg).pack(pady=5)
        Label(self.loginf, text="ID").pack()
        Entry(self.loginf, textvariable=self.ID).pack()
        Label(self.loginf, text="password").pack()
        Entry(self.loginf, textvariable=self.password).pack()
        self.loginf.pack()
        self.login_btn = Frame(self.loginf)
        Button(self.login_btn, text="LOGIN", command=self.login).pack(side=LEFT, padx=10)
        Button(self.login_btn, text='NEW_ID', command=self.Generator).pack(side=LEFT, padx=10)
        Button(self.login_btn, text="EXIT", command=self.quit).pack(side=LEFT, padx=10)
        self.login_btn.pack(pady=5)
```

```
class Generator_Frame:
    def __init__(self,roop):
        self.roop = roop
        self.roop.geometry("300x300+850+250")
        self.new_ID = StringVar()
        self.new_password = StringVar()
        self.widgets()

    def Generator(self):
        with sqlite3.connect("login.db") as db:
            cursor = db.cursor()
        find_user = ("SELECT * FROM user WHERE username = ? ")
        cursor.execute(find_user, [(self.new_ID.get())])
        if cursor.fetchall():
            ms.showerror('Error!', 'Username Taken Try a Diffrent One.')
            self.reset()
        else:
            insert = "INSERT INTO user(username,password) VALUES(?,?)"
            cursor.execute(insert, [self.new_ID.get(), (self.new_password.get())])
            ms.showinfo("success!!", "Account Created")
            self.reset()
            self.roop.withdraw()
        db.commit()
        db.close()

    def reset(self):
        self.new_ID.set("")
        self.new_password.set("")

    def widgets(self):
        self.new_user = Frame(self.roop)
        Label(self.new_user, text="Generator ID").pack()
        Label(self.new_user, text="ID").pack()
        Entry(self.new_user, textvariable=self.new_ID).pack()
        Label(self.new_user, text="password").pack()
        Entry(self.new_user, textvariable=self.new_password).pack()
        Button(self.new_user, text="Generator", command=self.Generator).pack()
        Button(self.new_user, text="EXIT").pack()
        self.new_user.pack()

class menu:
    def __init__(self,roop):
        self.roop = roop
        self.roop.geometry("320x470+850+190")
        self.imglist = []
        self.imglistbtn = ["./wine_test/wine_images.jpg",
                           "./wine_test/wine_recommandation.jpg",
                           "./wine_test/wine_back.jpg"]
        for i in self.imglistbtn:
            imgbtn = Image.open(i).resize((200, 100), Image.ANTIALIAS)
            imgbtn = ImageTk.PhotoImage(imgbtn)
            self.imglist.append(imgbtn)
        self.widgets()

    def photo(self):
        toplevel = Toplevel(self.roop)
        Photographic_recognition(toplevel)
        self.roop.withdraw()

    def Recommended(self):
        toplevel = Toplevel(self.roop)
        Recommended(toplevel)
        self.roop.withdraw()

    def gellery(self):
```

```
        toplevel = Toplevel(self.roop)
        Gellery(toplevel)
        self.roop.withdraw()

    def exit(self):
        toplevel = Toplevel(self.roop)
        main(toplevel)
        self.roop.withdraw()

    def widgets(self):

        self.menu = Frame(self.roop)
        Label(self.menu, text="MENU").pack()
        btn1 = Button(self.menu, command=self.photo)
        btn1.pack()
        btn1.config(image=self.imglist[0],compound=CENTER)
        btn2 = Button(self.menu, command=self.Recommended)
        btn2.pack()
        btn2.config(image=self.imglist[1], compound=CENTER)
        btn3 = Button(self.menu, command=self.gellery)
        btn3.pack()
        btn3.config(image=self.imglist[1], compound=CENTER)
        btn4 = Button(self.menu, command=self.exit)
        btn4.pack()
        btn4.config(image=self.imglist[2], compound=CENTER)
        self.menu.pack()

class Photographic_recognition:
    def __init__(self,roop):
        self.roop = roop
        self.roop.geometry("600x430+700+220")
        self.widgets()

    def load_image(self):
        global img
        wine_list = pd.read_csv('data_set.csv', header=0, index_col=0, encoding="cp949")
        try:
            fname = filedialog.askopenfilename(initialdir="E:/Images", title="choose your file",
                                               filetypes=(("jpeg files", "*.jpg", "jpeg"), ("all files", "*.*")))
            img = Image.open(fname).resize((335, 325), Image.ANTIALIAS)
            img = ImageTk.PhotoImage(img)
            self.imgL.config(image=img, width=335, height=325)
            data = ict.inseption(fname)
            self.text.config(state='normal')
            self.text.delete(1.0, END)
            self.text.insert(END, '* 예 측 결 과 *')
            self.text.insert(END, ('\nname : {0}'.format(data[0])))
            self.text.insert(END, ('\n정확도 : %.2f%%' % (data[1] * 100)))
            wine_data1 = wine_list.loc[data[0]]
            for i, n in zip(wine_list.keys(), wine_data1.values):
                self.text.insert(END, '\n{0}'.format("======== {0} ========\n{1}".format(i, n)))
            self.text.config(state='disabled')
        except:
            self.text.delete(1.0, END)
            self.text.insert(END, '\n관련 정보가 없습니다.')

    def exit(self):
        toplevel = Toplevel(self.roop)
        menu(toplevel)
        self.roop.withdraw()

    def widgets(self):
        self.photo = Frame(self.roop)
        Label(self.photo, text="Photographic recognition").pack()
        self.imgF = Frame(self.photo)
```

```
        self.imgL = Label(self.imgF, relief='solid', width=45, height=21, image=None)
        self.imgL.grid(row=0, column=0, padx=10, pady=10)
        self.text = Text(self.imgF, relief='solid', width=30, height=25, wrap='word')
        self.text.grid(row=0, column=2, padx=10)
        self.imgF.pack()
        Button(self.photo, text='File import', fg='#5D5D5D', command=self.load_image).pack()
        Button(self.photo, text="EXIT", command=self.exit).pack()
        self.photo.pack()

class Recommended:
    def __init__(self,roop):
        self.roop = roop
        self.roop.geometry("750x520+600+220")
        self.widgets()

    def treeview_sort_column(self,tv, col, reverse):
        l = [(tv.set(k, col), k) for k in tv.get_children('')]
        l.sort(reverse=reverse)
        for index, (val, k) in enumerate(l):
            tv.move(k, '', index)
        tv.heading(col, command=lambda: self.treeview_sort_column(tv, col, not reverse))

    def exit(self):
        toplevel = Toplevel(self.roop)
        menu(toplevel)
        self.roop.withdraw()

    def widgets(self):
        wine_list = pd.read_csv('data_set.csv', header=0, encoding="cp949")

        self.recommend = Frame(self.roop)
        Label(self.recommend, text="Recommended wine").pack()

        self.radioF = Frame(self.recommend)
        self.radioF .pack()
        self.radiolist  = ["Red", "White", "Sparkling"]
        var1 = StringVar()
        for num, mane in zip(range(len(self.radiolist)), self.radiolist):
            self.rbtn = Radiobutton(self.radioF, text=mane, variable=var1, value=mane)
            self.rbtn.grid(row=0, column=num, padx=5)
        Button(self.radioF, text="surchei").grid(row=0, column=3, padx=5)

        self.treeF = LabelFrame(self.recommend)
        self.treeF.pack()
        self.treeview = Treeview(self.treeF, height=19)
        self.teeX = Scrollbar(self.treeF, orient="horizontal", command=self.treeview.xview)
        self.teeY = Scrollbar(self.treeF, orient="vertical", command=self.treeview.yview)
        self.treeview.configure(xscroll=self.teeX.set, yscroll=self.teeY.set)
        self.treeview["columns"] = (wine_list.keys())
        self.treeview.heading('#0', text="", anchor=W)
        self.treeview.column("#0", width=0, stretch=0, minwidth=0)
        for o in wine_list.get_values():
            self.treeview.insert("", "end", values=([w for w in o]))
        for i, n in zip(range(1, len(wine_list.keys()) + 1), wine_list.keys()):
            self.treeview.heading('#{}'.format(i), text=n, anchor=W,
                                  command=lambda _col="#{}".format(i): self.treeview_sort_column(self.treeview, _col, False))

        self.teeX.pack(side=BOTTOM, fill=X)
        self.teeY.pack(side=RIGHT, fill=Y)
        self.treeview.pack(fill=BOTH, expand=1, pady=1)
        self.recommend.pack()
        Button(self.recommend, text="EXIT", command=self.exit).pack(pady=10)


class Gellery:
```

```
    def __init__(self,roop):
        self.roop = roop
        self.roop.geometry("600x430+700+220")
        self.widgets()

    def exit(self):
        toplevel = Toplevel(self.roop)
        menu(toplevel)
        self.roop.withdraw()

    # def

    def widgets(self):
        self.gellery = Frame(self.roop)
        Label(self.gellery, text="GELLERY").pack()
        Text(self.gellery, relief='solid').pack(padx=13,pady=5)
        self.text = Frame(self.gellery)
        Entry(self.text, width=35).pack(side=LEFT)
        Button(self.text, text='search').pack(side=LEFT)
        Button(self.text, text="EXIT", command=self.exit).pack(side=LEFT)
        self.text.pack()
        self.gellery.pack()


roop = Tk()
main(roop)
roop.mainloop()
```