# TASK5-CONTACT BOOK

**PROJECT TITLE:**

**Contact Book Management System**

**DESCRIPTION:**

**A GUI-based contact management system using CustomTkinter that allows users to store, update, delete, and view contact information.**

**FEATURES:**

**1.Input fields for name, phone, email, and address.**
**2.Validation for phone number format.**
**3.Add, update, delete operations with data persistence in JSON.**
**4.Search and filter contacts by name.**
**5.All data stored in a JSON file.**

**TECHNOLOGIES USED:**

**1.Python**
**2.CustomTkinter**
**3.JSON, OS modules**

**TARGET USERS:**

**1.Students and users looking for digital contact management.**

**2.Demonstration for beginner-level GUI and file I/O in Python.**

**APPENDIX:**

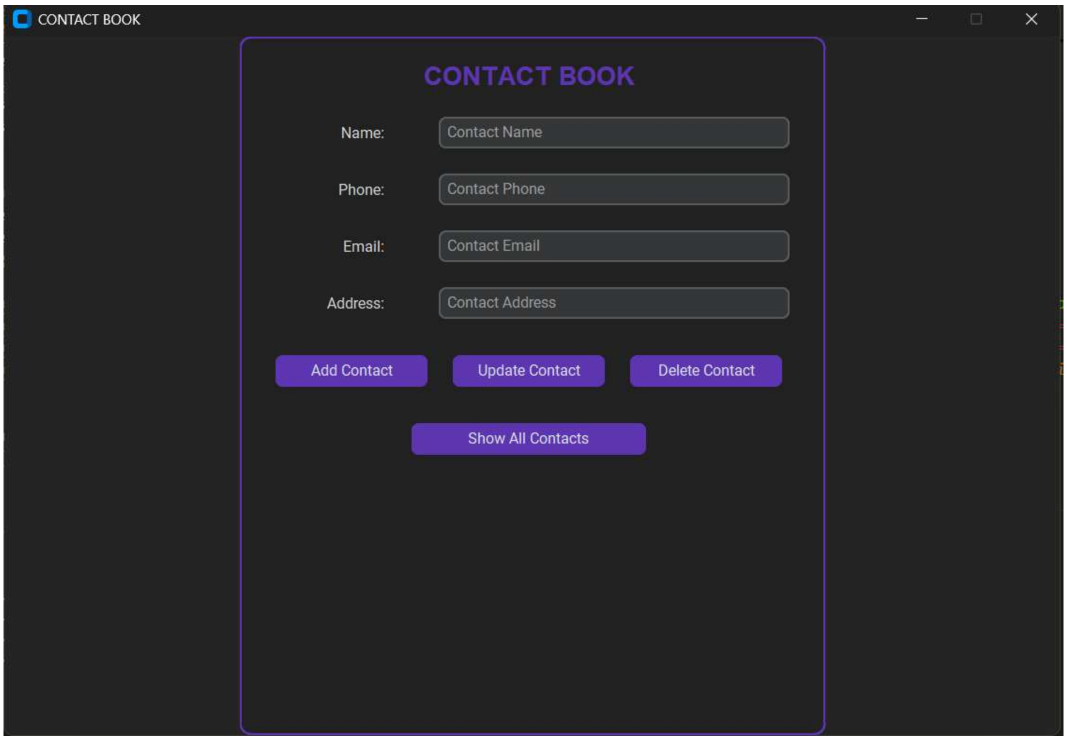
**Contact_Book.py**

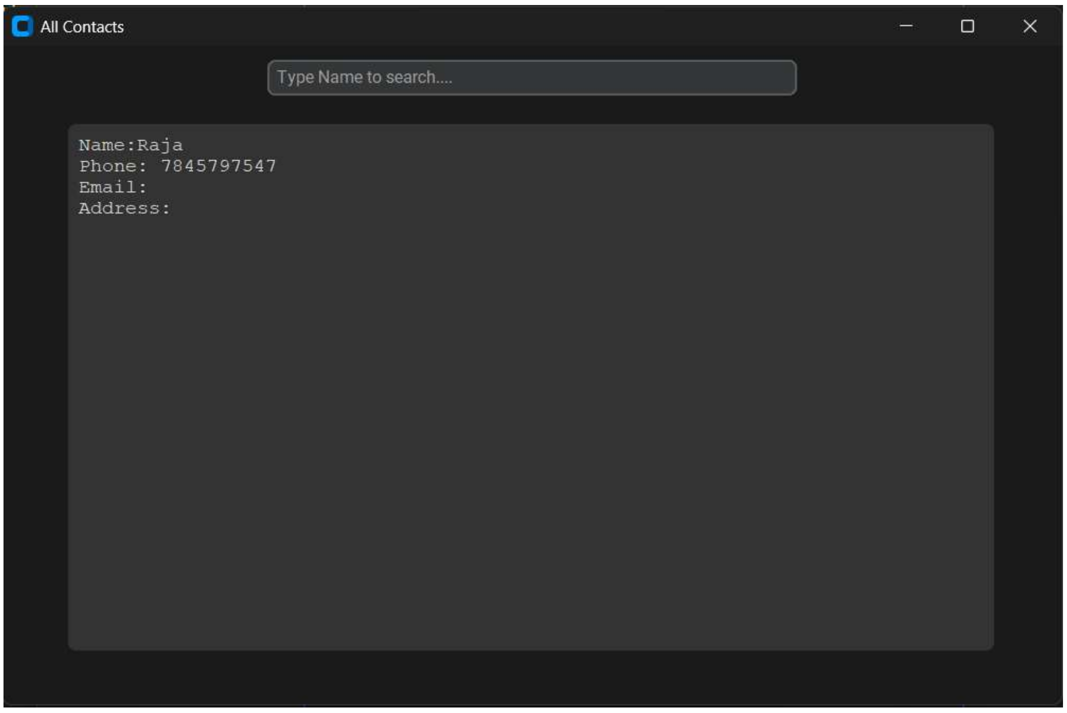
```
import customtkinter as ctk
import tkinter.messagebox as msg
import json
import os
Contact_file="contacts.json"
#for loading the contact
def load_contacts():
   if not os.path.exists(Contact_file):
      return []
   with open(Contact_file,'r') as f:
      return json.load(f)
# for Saving the contact
def save_contacts(data):
   with open(Contact_file,'w') as f:
      json.dump(data,f,indent=4)

#Creating a class for Contact Book App
class contactbook(ctk.CTk):
   def __init__(self):
      super().__init__()
      self.title("CONTACT BOOK")
      self.geometry("900x600")
      self.resizable(False,False)
      ctk.set_appearance_mode("dark")
      ctk.set_default_color_theme("dark-blue")

      self.contacts = load_contacts()
      self.create_ui()

   #Creating Functions
   def clear_fields(self):
      self.name_entry.delete(0,'end')
      self.phone_entry.delete(0,'end')
      self.email_entry.delete(0,'end')
      self.address_entry.delete(0,'end')
```

```
    def add_contact(self):
        name=self.name_entry.get().strip()
        phone=self.phone_entry.get().strip()
        email=self.email_entry.get().strip()
        address=self.address_entry.get().strip()
        if not name or not phone:
            msg.showwarning("Missing Info","Name and Phone are
required.")
            return
        if not phone.isdigit() or len(phone)!=10:
            msg.showerror("Invalid Phone","Phone Number must be 10
digits.")
            return
        for contact in self.contacts:
            if contact["name"]==name or contact['phone']==phone:
                msg.showerror("Duplicate","Contact with same name or
phone exits.")
                return
        self.contacts.append({
            'name':name,
            'phone':phone,
            'email':email,
            "address":address
        })

        save_contacts(self.contacts)
        msg.showinfo("Success","Contacts Added.")
        self.clear_fields()

    def update_contact(self):
        name=self.name_entry.get().strip()
        phone=self.phone_entry.get().strip()

        for contact in self.contacts:
            if contact['name']==name:
                contact['phone']=phone
                contact['email']=self.email_entry.get().strip()
                contact['address']=self.address_entry.get().strip()
                save_contacts(self.contacts)
                msg.showinfo("Updated",f"Contact {name} updated.")
                return
        msg.showerror("Not Found","Contact not found.")

    def delete_contact(self):
        name=self.name_entry.get().strip()
        for contact in self.contacts:
            if contact['name']==name:
                self.contacts.remove(contact)
                save_contacts(self.contacts)
                msg.showinfo("Deleted",f"Contact {name} deleted.")
                self.clear_fields()
                return
        msg.showerror("Not Found","Contact not found.")


    def show_all_contacts(self):
        popup=ctk.CTkToplevel(self)
        popup.title("All Contacts")
        popup.geometry("800x500")
        popup.grab_set()

        def filter_Contacts(event=None):
            search_text=search_entry.get().strip().lower()
            display.delete('0.0',"end")
            filtered = self.contacts if not search_text else [c for c in
self.contacts if search_text in c['name'].lower()]
            if filtered:
                for c in filtered:
                  display.insert("end", f"Name:{c['name']}\nPhone:
{c['phone']}\nEmail: {c['email']}\nAddress: {c['address']}\n\n")
            else:
                display.insert("end","No Contact found.")

        search_entry=ctk.CTkEntry(popup,placeholder_text="Type
Name to search....",width=400)
        search_entry.pack(pady=10)
        search_entry.bind("<KeyRelease>",filter_Contacts)

        display=
ctk.CTkTextbox(popup,width=700,height=400,font=("Courier",14)
)
        display.pack(padx=20,pady=10)
        filter_Contacts()


    def create_ui(self):

self.frame=ctk.CTkFrame(self,border_width=2,border_color="#5e
35b1",corner_radius=10,height=600,width=500)
        self.frame.place(relx=0.5,rely=0.5,anchor="center")
        ctk.CTkLabel(self.frame,text="CONTACT
BOOK",text_color="#5e35b1",font=("Helvetica",22,"bold")).grid(ro
w=0,column=0,columnspan=2,pady=(20,10))

ctk.CTkLabel(self.frame,text="Name:",width=100,anchor='e').grid(
row=1,column=0,padx=10,pady=10)

self.name_entry=ctk.CTkEntry(self.frame,placeholder_text="Conta
ct Name",width=300)
        self.name_entry.grid(row=1,column=1,padx=10,pady=10)

ctk.CTkLabel(self.frame,text="Phone:",width=100,anchor="e").gri
d(row=2,column=0,padx=10,pady=10)

self.phone_entry=ctk.CTkEntry(self.frame,placeholder_text="Conta
ct Phone",width=300)
        self.phone_entry.grid(row=2,column=1,padx=10,pady=10)

ctk.CTkLabel(self.frame,text="Email:",width=100,anchor='e').grid(
row=3,column=0,padx=10,pady=10)
        self.email_entry
=ctk.CTkEntry(self.frame,placeholder_text="Contact
Email",width=300)
        self.email_entry.grid(row=3,column=1,padx=10,pady=10)
```

```
ctk.CTkLabel(self.frame,text="Address:",width=100,anchor="e").gr
id(row=4,column=0,padx=10,pady=10)

self.address_entry=ctk.CTkEntry(self.frame,placeholder_text="Con
tact Address",width=300)
      self.address_entry.grid(row=4,column=1,padx=10,pady=10)


   #Creating Buttons

button_frame=ctk.CTkFrame(self.frame,fg_color="transparent")

button_frame.grid(row=5,column=0,columnspan=2,pady=20,padx
=20)
      self.frame.grid_propagate(False)

      ctk.CTkButton(button_frame,text="Add
Contact",fg_color="#5e35b1",hover_color='#009933',command=
self.add_contact,width=130).pack(side="left",padx=10)
      ctk.CTkButton(button_frame,text="Update
Contact",fg_color="#5e35b1",hover_color='#0066cc',command=
self.update_contact,width=130).pack(side="left",padx=10)
      ctk.CTkButton(button_frame,text="Delete
Contact",fg_color="#5e35b1",hover_color="#cc0000",command=
self.delete_contact,width=130).pack(side="left",padx=10)
      ctk.CTkButton(self.frame,text="Show All
Contacts",fg_color="#5e35b1",hover_color='#7e57c2',command=
self.show_all_contacts,width=200).grid(row=6,column=0,columnsp
an=2,pady=10)


if __name__=="__main__":
   app=contactbook()
   app.mainloop()
```

## OUTPUTS:





```
Name:Raja
Phone: 7845797547
Email:
Address:
```