**19CSE212**

**DSA**

**LAB – 3**

**TITLE – PHONE DIRECTORY**

**Problem Statement:** Phone Directory Management System with Sorting and Customized Search

**Problem Description:**

Our goal is to develop a phone directory management system that allows users to store and manage their contacts efficiently. The system should provide functionality for inserting, updating, deleting, and deleting all contacts. Additionally, the system should support sorting contacts alphabetically and enable users to fetch contact details by specifying the attribute and value of their choice.

**Approach to the problem:**

Approach to the Problem Using Linked List Data Structure:

To address the phone directory management problem, we have implemented a linear data structure called a linked list. The linked list consists of nodes, where each node represents a single contact in the phone directory.

**1. Insertion:**

- We maintain a pointer called `head` that points to the first node in the linked list.

- When a user wants to add a new contact, we create a new instance of a contact class that holds the contact's details.

- We then insert this contact if it the first one then self.head, means first node itself. And if adding the contacts to an existing phone directory then to the end of the linked list by updating the `head` pointer.

**2. Updating:**

- To update a contact's details, we traverse the linked list starting from the `head` node.

- For each node, we compare the name (in lowercase) with the user's input.

- If a match is found, we update the value of the desired attribute (e.g., phone number, email address) with the user's provided input.

**3. Deleting:**

- To delete a specific contact, we traverse the linked list starting from the `head` node.

- We compare the name of each contact with the user's input.

- If a match is found, we remove that particular node from the linked list by adjusting the links between adjacent nodes.

**4. Deleting All Contacts:**

- To delete all contacts, we simply remove all the links from the linked list by setting the `head` pointer to null.

- This action effectively removes all the nodes, leaving an empty phone directory.

**5. Sorting Alphabetically:**

- We implement a static method within the phone directory class to sort the contacts alphabetically.

- This method uses a temporary variable to traverse the linked list and swap adjacent nodes if they are out of order.

- The sorting is typically done based on the names of the contacts.

**6. Searching Attribute:**

- To search for a contact based on a specific attribute (e.g., name), we traverse the linked list.

- When a contact's name matches the user's desired name, we update the value of the specified attribute with the user's input.

By utilizing a linked list data structure, we can efficiently perform operations such as insertion, updating, deletion, deleting all contacts, sorting alphabetically, and searching for attributes within the phone directory management system.

**SOURCE CODE**

importtkinterastk

fromtkinterimportmessagebox

**class**Contact:

**def**\_\_init\_\_(self, name, number, email, category):

        self.name= name

        self.number= number

        self.email= email

        self.category= category

        self.next=None

**class**PhoneDirectory:

**def**\_\_init\_\_(self):

        self.head=None

**def**insert\_contact(self, name, number, email, category):

        new\_contact=Contact(name, number, email, category)

        ifself.headisNone:

            self.head=new\_contact

        else:

            current=self.head

            whilecurrent.nextisnotNone:

                current=current.next

            current.next=new\_contact

**def**delete\_contact(self, name):

        ifself.headisNone:

            messagebox.showerror("Error", "Phone Directory is empty.")

            return

        ifself.head.name.lower() ==name.lower():

            self.head=self.head.next

            messagebox.showinfo("Success", "Contact deleted successfully.")

            return

        current=self.head

        prev=None

        whilecurrentisnotNone:

            ifcurrent.name.lower() ==name.lower():

                prev.next=current.next

                messagebox.showinfo("Success", "Contact deleted successfully.")

                return

            prev=current

            current=current.next

        messagebox.showerror("Error", "Contact not found.")

**def**delete\_all\_contacts(self):

        self.head=None

        messagebox.showinfo("Success", "All contacts deleted.")

**def**update\_contact(self, name, attribute, value):

        ifself.headisNone:

            messagebox.showerror("Error", "Phone Directory is empty.")

            return

        current=self.head

        whilecurrentisnotNone:

            ifcurrent.name.lower() ==name.lower():

                ifattribute.lower() =='name':

                    current.name= value

                elifattribute.lower() =='number':

                    current.number= value

                elifattribute.lower() =='email':

                    current.email= value

                elifattribute.lower() =='category':

                    current.category= value

                messagebox.showinfo("Success", "Contact updated successfully.")

                return

            current=current.next

        messagebox.showerror("Error", "Contact not found.")

**def**search\_contacts\_by\_attribute(self, attribute, value):

        contacts= []

        current=self.head

        whilecurrentisnotNone:

            ifattribute.lower() =='name':

                ifcurrent.name.lower() ==value.lower():

                    contacts.append(current)

            elifattribute.lower() =='number':

                ifcurrent.number.lower() ==value.lower():

                    contacts.append(current)

            elifattribute.lower() =='email':

                ifcurrent.email.lower() ==value.lower():

                    contacts.append(current)

            elifattribute.lower() =='category':

                ifcurrent.category.lower() ==value.lower():

                    contacts.append(current)

            current=current.next

        returncontacts

**def**sort\_alphabetically(self):

        ifself.headisNone:

            return

        current=self.head

        whilecurrent.nextisnotNone:

            smallest=current

            temp=current.next

            whiletempisnotNone:

                iftemp.name.lower() <smallest.name.lower():

                    smallest=temp

                temp=temp.next

            ifsmallest!=current:

                self.swap(current, smallest)

            current=current.next

    @staticmethod

**def**swap(contact1, contact2):

        temp\_name= contact1.name

        temp\_number= contact1.number

        temp\_email= contact1.email

        temp\_category= contact1.category

        contact1.name = contact2.name

        contact1.number = contact2.number

        contact1.email = contact2.email

        contact1.category = contact2.category

        contact2.name =temp\_name

        contact2.number =temp\_number

        contact2.email =temp\_email

        contact2.category =temp\_category

**def**insert\_contact():

    name=entry\_name.get()

    number=entry\_number.get()

    email=entry\_email.get()

    category=entry\_category.get()

    ifname=='':

        messagebox.showerror("Error", "Name field cannot be empty.")

        return

    phone\_directory.insert\_contact(name, number, email, category)

    messagebox.showinfo("Success", "Contact inserted successfully.")

    entry\_name.delete(0, tk.END)

    entry\_number.delete(0, tk.END)

    entry\_email.delete(0, tk.END)

    entry\_category.delete(0, tk.END)

**def**display\_all\_contacts():

    contacts\_text.delete('1.0', tk.END)

    current=phone\_directory.head

    whilecurrentisnotNone:

        contacts\_text.insert(tk.END, **f**"Name: {current.name}\nNumber: {current.number}\nEmail: {current.email}\nCategory: {current.category}\n\n")

        current=current.next

**def**search\_contact\_by\_name():

    name=entry\_search.get()

    ifname=='':

        messagebox.showerror("Error", "Name field cannot be empty.")

        return

    contacts=phone\_directory.search\_contacts\_by\_attribute('name', name)

    display\_search\_results(contacts)

**def**search\_contact\_by\_attribute():

    attribute=entry\_search\_attribute.get()

    value=entry\_search\_value.get()

    ifattribute==''orvalue=='':

        messagebox.showerror("Error", "All fields are required.")

        return

    contacts=phone\_directory.search\_contacts\_by\_attribute(attribute, value)

    display\_search\_results(contacts)

**def**display\_search\_results(contacts):

    contacts\_text.delete('1.0', tk.END)

    iflen(contacts) >0:

        forcontactin contacts:

            contacts\_text.insert(tk.END, **f**"Name: {contact.name}\nNumber: {contact.number}\nEmail: {contact.email}\nCategory: {contact.category}\n\n")

    else:

        messagebox.showinfo("Information", "Contact not found.")

**def**delete\_contact():

    name=entry\_delete.get()

    ifname=='':

        messagebox.showerror("Error", "Name field cannot be empty.")

        return

    phone\_directory.delete\_contact(name)

    entry\_delete.delete(0, tk.END)

**def**delete\_all\_contacts():

    response=messagebox.askyesno("Confirmation", "Are you sure you want to delete all contacts?")

    ifresponse==tk.YES:

        phone\_directory.delete\_all\_contacts()

        contacts\_text.delete('1.0', tk.END)

**def**update\_contact():

    name=entry\_update\_name.get()

    attribute=entry\_update\_attribute.get()

    value=entry\_update\_value.get()

    ifname==''orattribute==''orvalue=='':

        messagebox.showerror("Error", "All fields are required.")

        return

    phone\_directory.update\_contact(name, attribute, value)

    entry\_update\_name.delete(0, tk.END)

    entry\_update\_attribute.delete(0, tk.END)

    entry\_update\_value.delete(0, tk.END)

**def**sort\_contacts():

    phone\_directory.sort\_alphabetically()

    messagebox.showinfo("Success", "Contacts sorted alphabetically.")

    display\_all\_contacts()

phone\_directory=PhoneDirectory()

window=tk.Tk()

window.title("Phone Directory")

window.configure(bg='black')  *# Set the background color of the window to gray*

label\_name=tk.Label(window, text="Name:", bg="black", fg="white")

label\_name.grid(row=0, column=0, sticky=tk.W)

entry\_name=tk.Entry(window, bg="gray",)

entry\_name.grid(row=0, column=1)

label\_number=tk.Label(window, text="Number:", bg="black", fg="white")

label\_number.grid(row=1, column=0, sticky=tk.W)

entry\_number=tk.Entry(window, bg="gray")

entry\_number.grid(row=1, column=1)

label\_email=tk.Label(window, text="Email:", bg="black", fg="white")

label\_email.grid(row=2, column=0, sticky=tk.W)

entry\_email=tk.Entry(window, bg="gray")

entry\_email.grid(row=2, column=1)

label\_category=tk.Label(window, text="Category:", bg="black", fg="white")

label\_category.grid(row=3, column=0, sticky=tk.W)

entry\_category=tk.Entry(window, bg="gray")

entry\_category.grid(row=3, column=1)

button\_insert=tk.Button(window, text="Insert", command=insert\_contact, bg='gray', fg='black')  *# Set the button color to gray and text color to black*

button\_insert.grid(row=4, column=1, pady=(10, 0))

button\_display=tk.Button(window, text="Display All", command=display\_all\_contacts, bg='gray', fg='black')  *# Set the button color to gray and text color to black*

button\_display.grid(row=5, column=1, pady=(10, 0))

label\_search=tk.Label(window, text="Search:", bg="black", fg="white")

label\_search.grid(row=6, column=0, sticky=tk.W)

entry\_search=tk.Entry(window, bg="gray")

entry\_search.grid(row=6, column=1)

button\_search=tk.Button(window, text="Search by Name", command=search\_contact\_by\_name, bg='gray', fg='black')  *# Set the button color to gray and text color to black*

button\_search.grid(row=7, column=1, pady=(10, 0))

label\_search\_attribute=tk.Label(window, text="Attribute:", bg="black", fg="white")

label\_search\_attribute.grid(row=8, column=0, sticky=tk.W)

entry\_search\_attribute=tk.Entry(window, bg="gray")

entry\_search\_attribute.grid(row=8, column=1)

label\_search\_value=tk.Label(window, text="Value:", bg="black", fg="white")

label\_search\_value.grid(row=9, column=0, sticky=tk.W)

entry\_search\_value=tk.Entry(window, bg="gray")

entry\_search\_value.grid(row=9, column=1)

button\_search\_attribute=tk.Button(window, text="Search by Attribute", command=search\_contact\_by\_attribute, bg='gray', fg='black')  *# Set the button color to gray and text color to black*

button\_search\_attribute.grid(row=10, column=1, pady=(10, 0))

label\_delete=tk.Label(window, text="Delete:", bg="black", fg="white")

label\_delete.grid(row=11, column=0, sticky=tk.W)

entry\_delete=tk.Entry(window, bg="gray")

entry\_delete.grid(row=11, column=1)

button\_delete=tk.Button(window, text="Delete", command=delete\_contact, bg='gray', fg='black')  *# Set the button color to gray and text color to black*

button\_delete.grid(row=12, column=1, pady=(10, 0))

button\_delete\_all=tk.Button(window, text="Delete All", command=delete\_all\_contacts, bg='gray', fg='black')  *# Set the button color to gray and text color to black*

button\_delete\_all.grid(row=13, column=1, pady=(10, 0))

label\_update\_name=tk.Label(window, text="Name:", bg="black", fg="white")

label\_update\_name.grid(row=14, column=0, sticky=tk.W)

entry\_update\_name=tk.Entry(window, bg="gray")

entry\_update\_name.grid(row=14, column=1)

label\_update\_attribute=tk.Label(window, text="Attribute:", bg="black", fg="white")

label\_update\_attribute.grid(row=15, column=0, sticky=tk.W)

entry\_update\_attribute=tk.Entry(window, bg="gray")

entry\_update\_attribute.grid(row=15, column=1)

label\_update\_value=tk.Label(window, text="Value:", bg="black", fg="white")

label\_update\_value.grid(row=16, column=0, sticky=tk.W)

entry\_update\_value=tk.Entry(window, bg="gray")

entry\_update\_value.grid(row=16, column=1)

button\_update=tk.Button(window, text="Update", command=update\_contact, bg='gray', fg='black')  *# Set the button color to gray and text color to black*

button\_update.grid(row=17, column=1, pady=(10, 0))

button\_sort=tk.Button(window, text="Sort Alphabetically", command=sort\_contacts, bg='gray', fg='black')  *# Set the button color to gray and text color to black*

button\_sort.grid(row=18, column=1, pady=(10, 0))

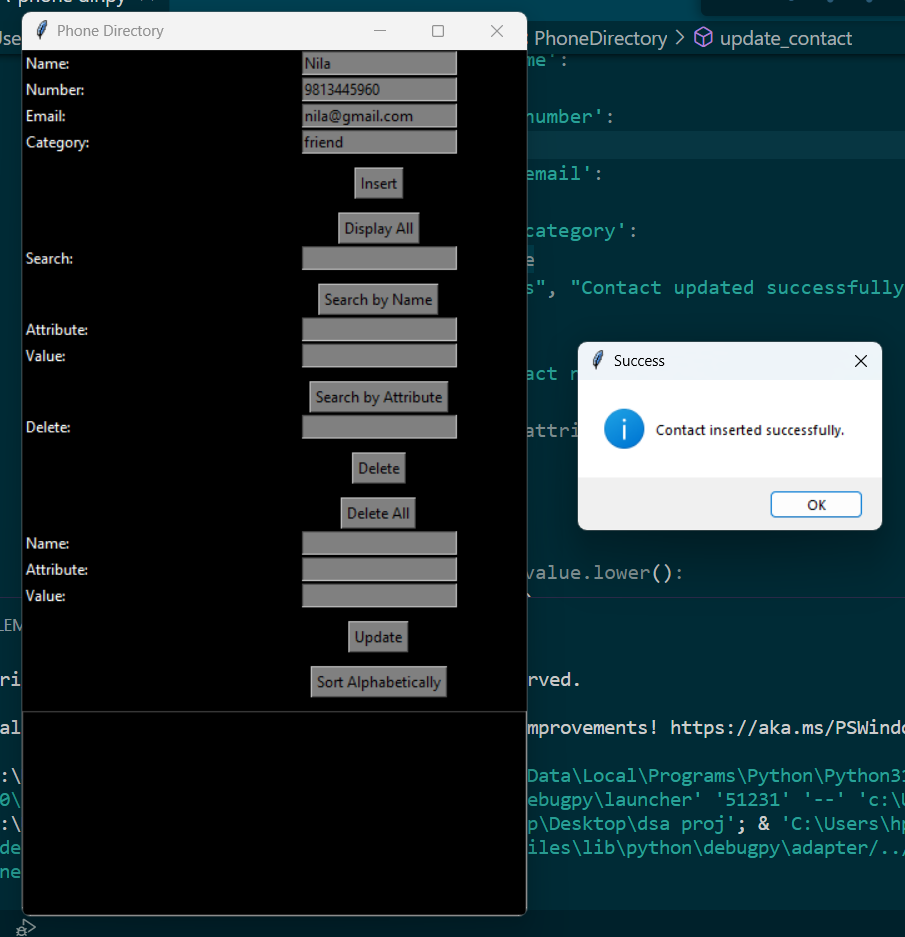
contacts\_text=tk.Text(window, height=10, width=50, bg="black", fg="white")

contacts\_text.grid(row=19, column=0, columnspan=2, pady=(10, 0))

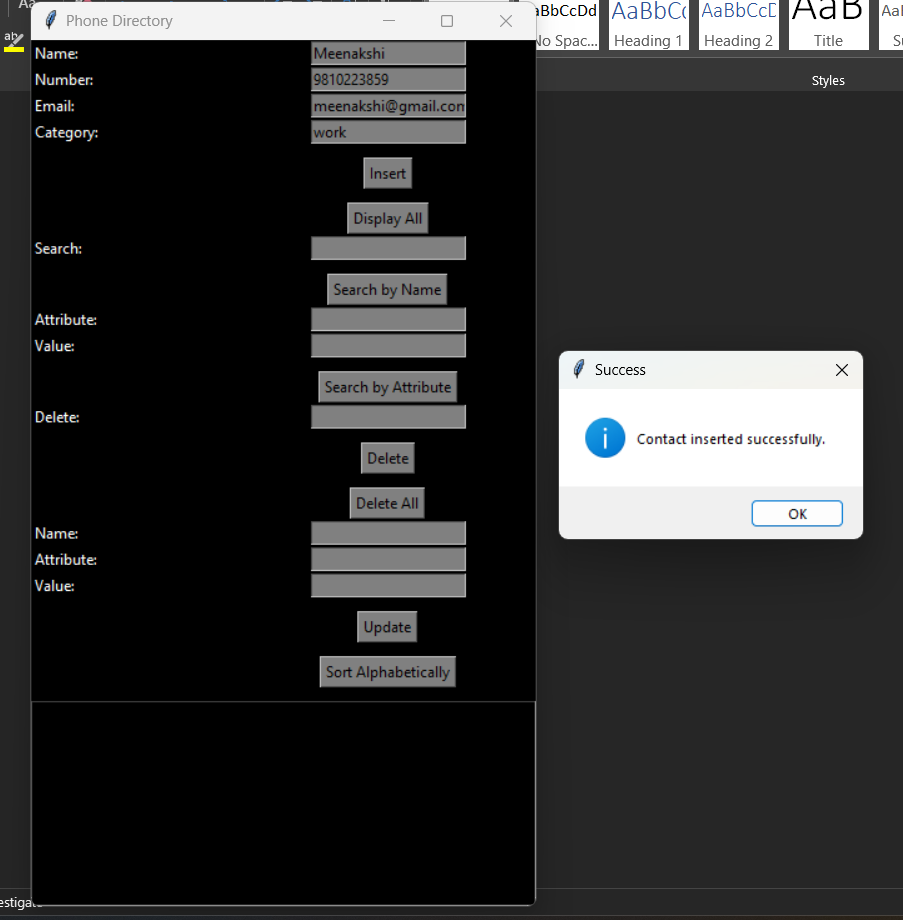
window.mainloop()

**OUTPUT:**

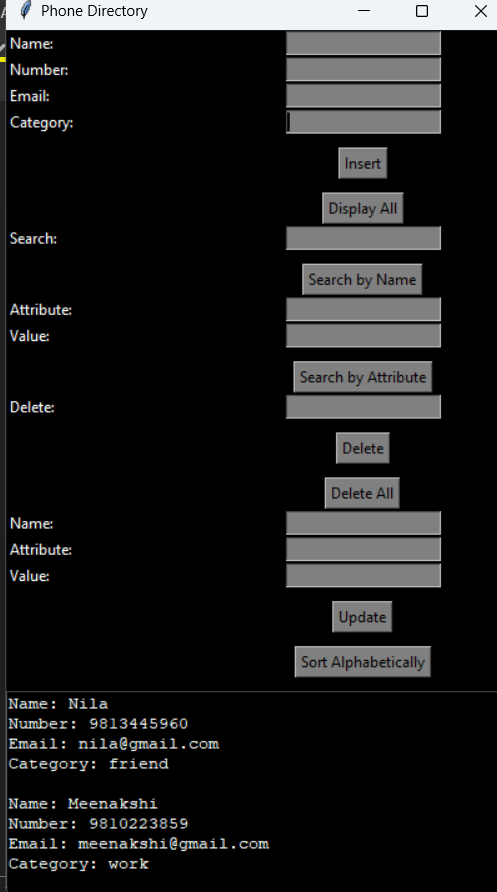
1. **Insertion of contact 1**

****

1. **Insertion of contact 2**

****

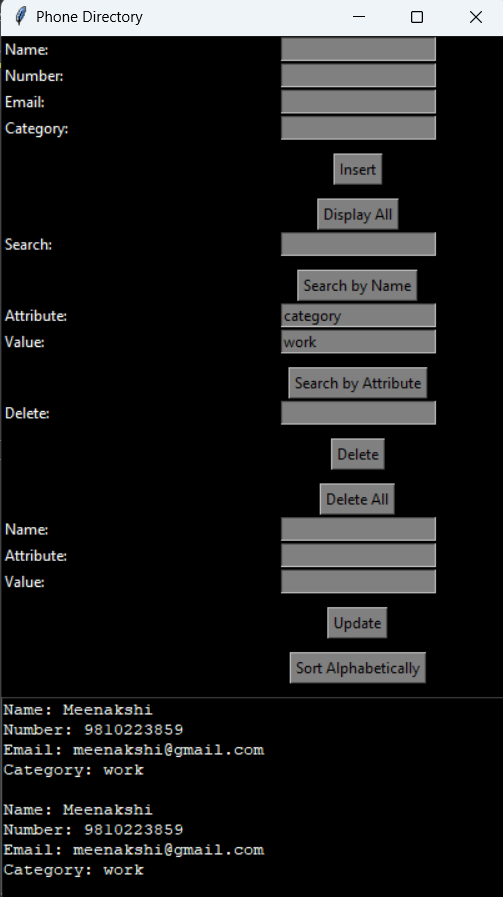
1. **Displaying the contacts**

****

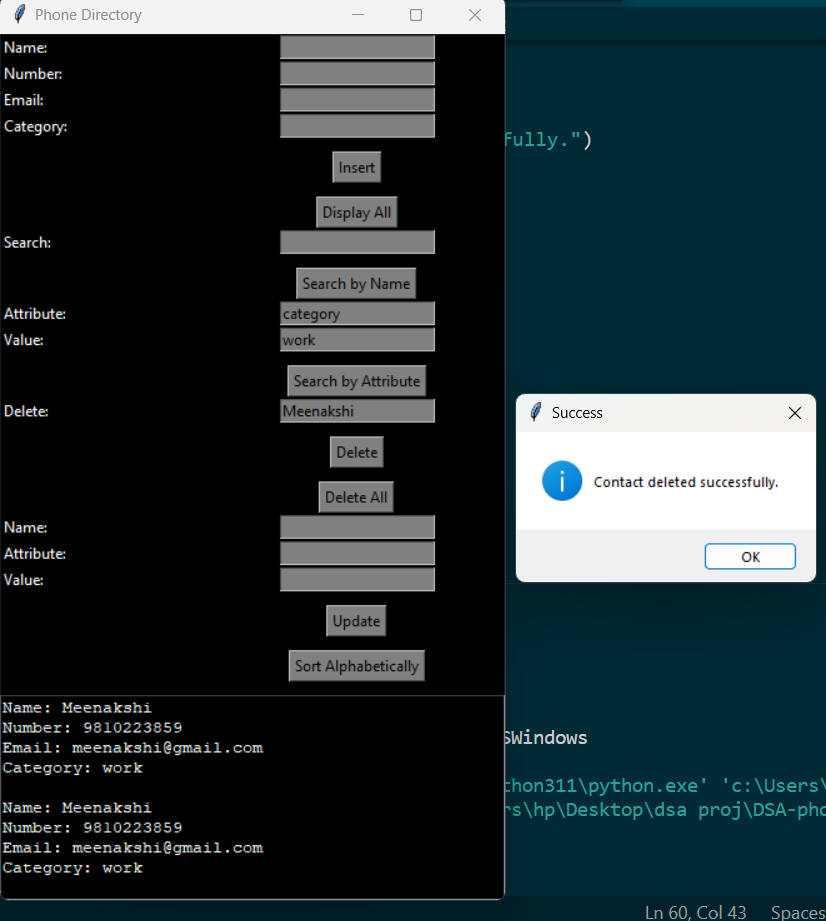
1. **Searching contact by name**

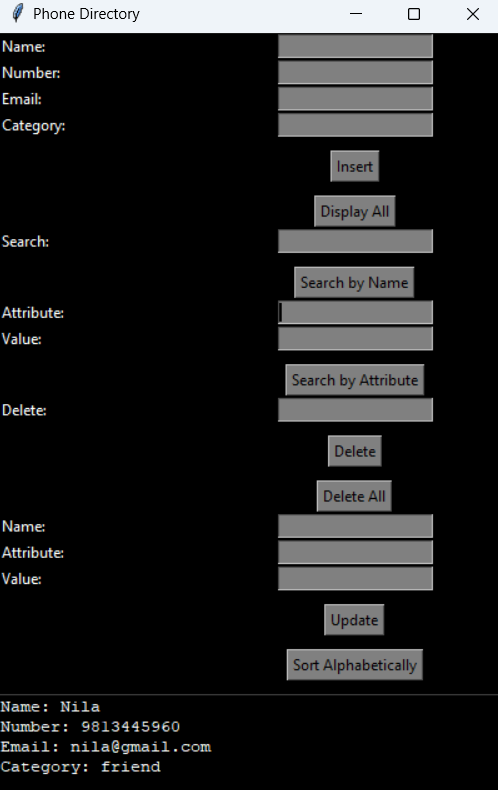
****

1. **Searching by attribute**

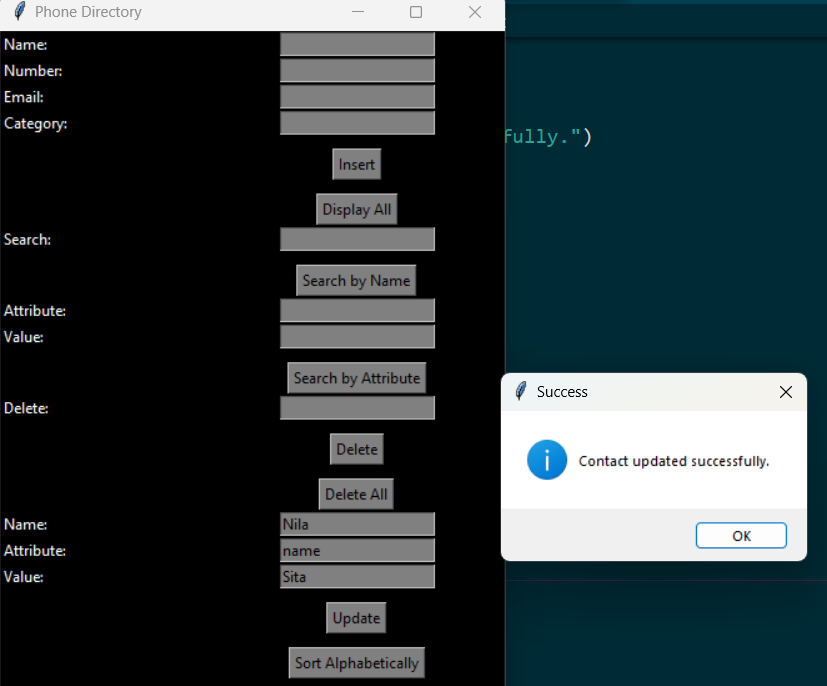
****

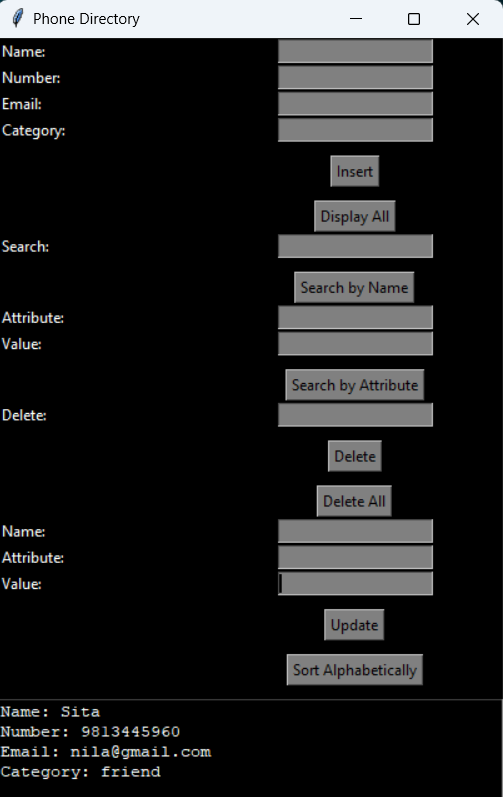
1. **Delete a contact**



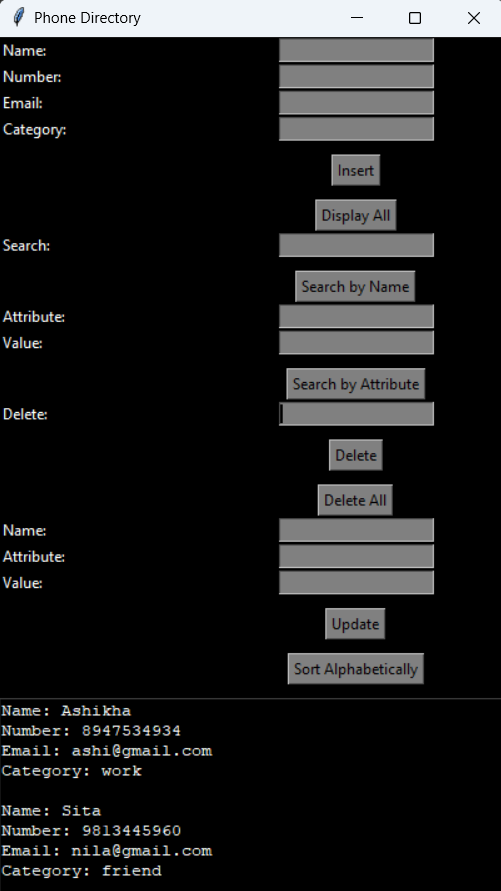


1. **Updating a contact**

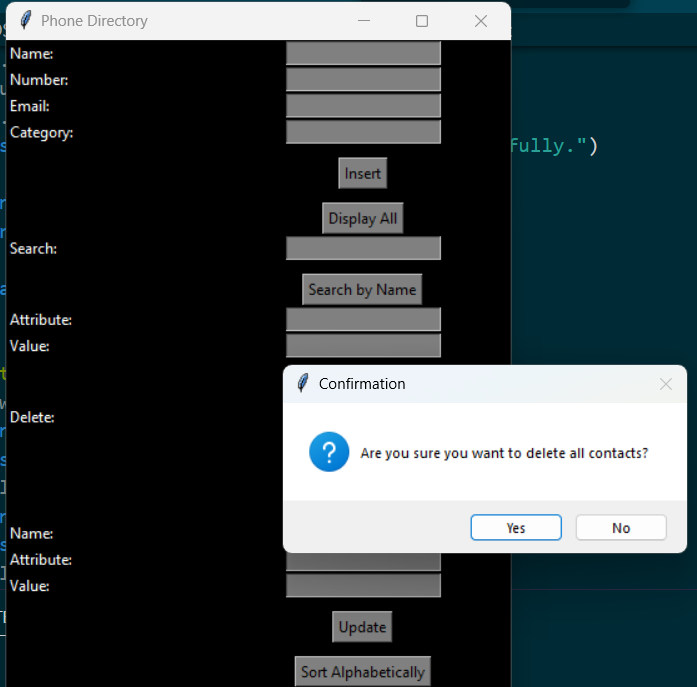
****

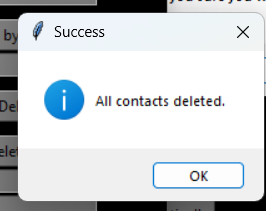
****

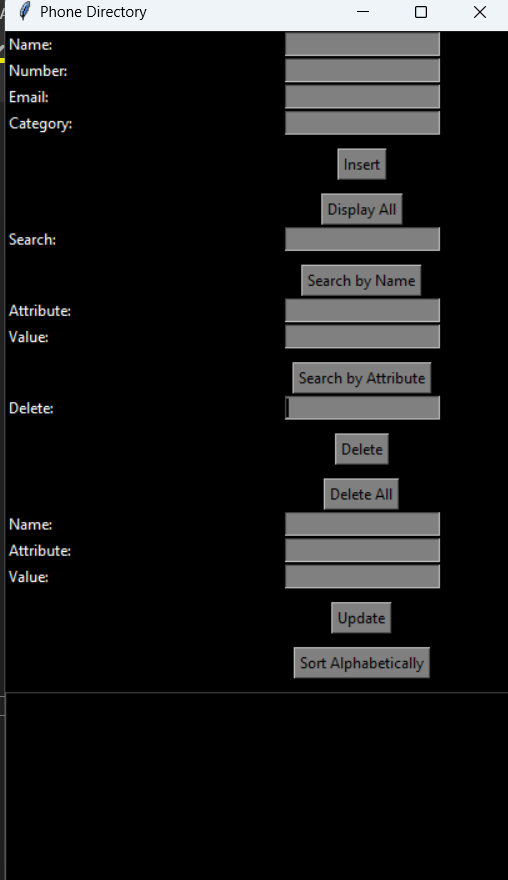
1. **Sort alphabetically**

****

1. **Delete all the contacts**

****

****

****