```python
import tkinter as tk
from tkinter import messagebox, ttk

# Create the main window for the application
root = tk.Tk()
root.title("Contact Book")  # Title of the window
root.state("zoomed")  # Set window to maximize by default

# Fonts and styles
large_font = ("Arial", 11)  # Font style used throughout the UI

# Frames for different sections of the contact book (Save, Delete, Update)
add_frame = tk.Frame(root, background="gray64", width=400, height=320, borderwidth=1, relief="solid")
add_frame.place(x=30, y=310)  # Position the "Add Contact" section
save_contact_label = ttk.Label(add_frame, text="SAVE CONTACT", font=("Arial", 13, "bold"))
save_contact_label.place(x=130, y=20)  # Title for the Save Contact section

delete_frame = tk.Frame(root, background="gray74", width=400, height=320, borderwidth=1, relief="solid")
delete_frame.place(x=440, y=310)  # Position the "Delete Contact" section
delete_contact_label = ttk.Label(delete_frame, text="DELETE CONTACT", font=("Arial", 13, "bold"))
delete_contact_label.place(x=130, y=20)  # Title for the Delete Contact section
sub_heading = ttk.Label(delete_frame, text="Using either the Name or the Phone Number.", font=("Arial", 9, "italic"))
sub_heading.place(x=80, y=50)  # Sub-heading for clarification

update_frame = tk.Frame(root, background="gray84", width=400, height=320, borderwidth=1, relief="solid")
update_frame.place(x=850, y=310)  # Position the "Update Contact" section
update_contact_label = ttk.Label(update_frame, text="UPDATE CONTACT", font=("Arial", 13, "bold"))
update_contact_label.place(x=130, y=20)  # Title for the Update Contact section
upd_sub_heading = ttk.Label(update_frame, text="Search and Update contact details.", font=("Arial", 9, "italic"))
upd_sub_heading.place(x=110, y=50)  # Sub-heading for clarification

# Listbox to display the saved contacts
display_saved_contact = tk.Listbox(root, height=15, width=100, bg="white smoke", borderwidth=1, relief="solid", selectmode=tk.SINGLE)
display_saved_contact.pack()
display_saved_contact.insert(0, "   Name   |    Phone Number    |   Email ID    |   Address     ")  # Header in the Listbox

# Search Contact Section
```

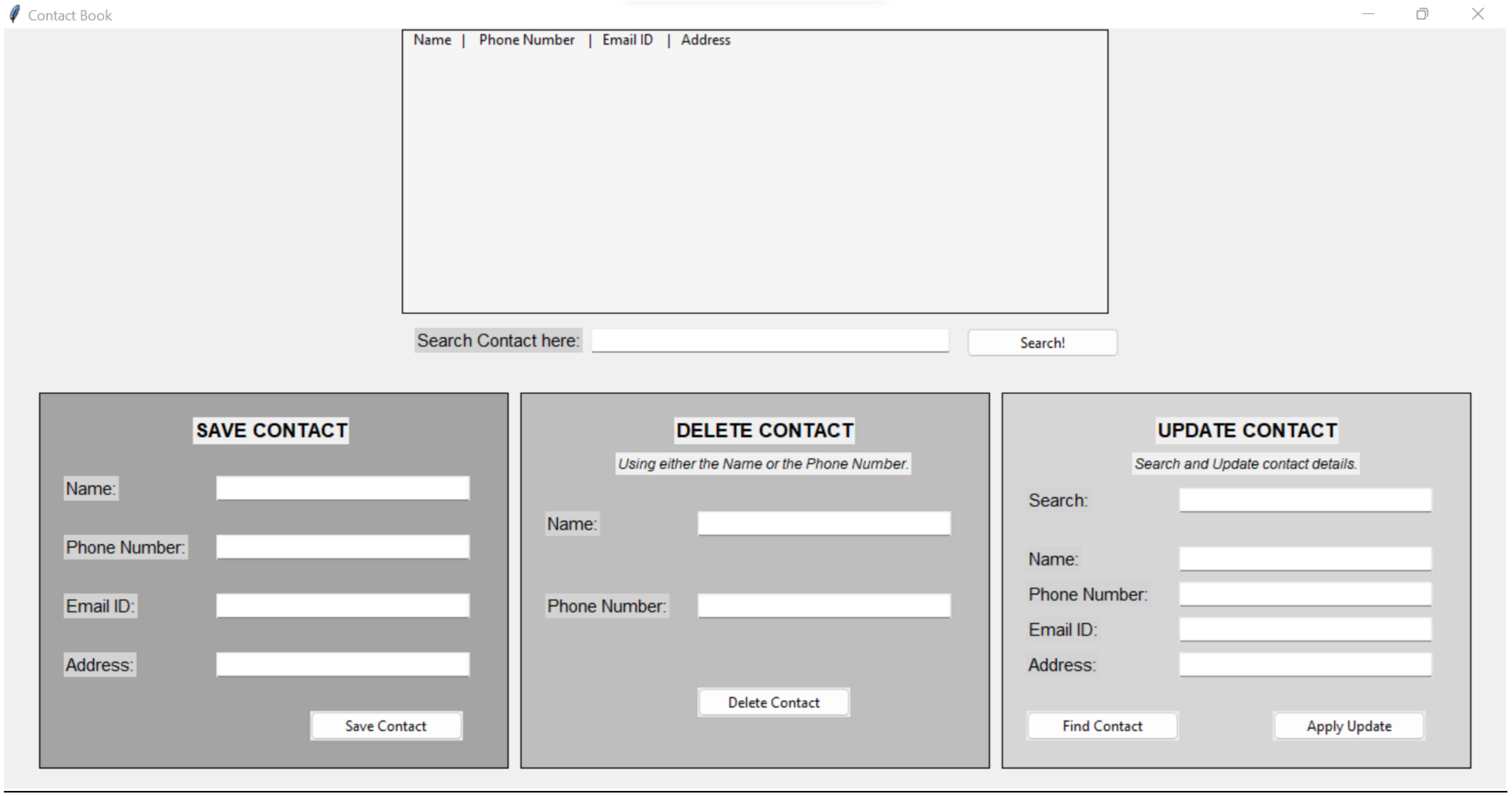
```python
search_label = ttk.Label(root, text="Search Contact here:", font=("Arial, 11"), background="light gray")
search_label.place(x=350, y=255)  # Label for search box
search_contact_entry = ttk.Entry(root, background="dim gray", width=50)
search_contact_entry.place(x=500, y=255)  # Input field for search criteria

# Function to handle searching for contacts
def search_contact():
    query = search_contact_entry.get().strip().lower()  # Get input query and convert to lowercase
    if not query:  # If query is empty
        messagebox.showinfo("Invalid Input", "Please enter a name or phone number to search.")
        return

    # Search through the saved contacts
    found = False
    for idx, entry in enumerate(display_saved_contact.get(1, tk.END), start=1):  # Skip header (idx starts from 1)
        if query in entry.lower():  # Search in lowercase for case-insensitive match
            display_saved_contact.selection_clear(0, tk.END)  # Clear previous selection
            display_saved_contact.selection_set(idx)  # Select the found contact
            display_saved_contact.see(idx)  # Scroll to the selected contact
            found = True
            break

    if not found:  # If no match was found
        messagebox.showinfo("Not Found", "No contact found matching the search criteria.")

# Button to trigger search action
search_button = ttk.Button(root, text="Search!", width=20, command=search_contact)
search_button.place(x=820, y=255)

# Initialize an empty list to store contacts
contacts = []

# Save Contact Section
def new_contact_insertion():
    new_name = contact_name_entry.get().strip()
    new_phone = phone_number_entry.get().strip()
    new_email = email_id_entry.get().strip()
    new_address = address_entry.get().strip()

    # Validation checks
    if not new_name or not new_phone:  # Check if name and phone number are provided
        messagebox.showerror("Missing Information", "Name and Phone Number are required.")
        return

    if not new_phone.isdigit():  # Ensure phone number contains digits only
        messagebox.showerror("Invalid Input", "Phone Number should contain digits only.")
        return

    # Check if the contact already exists based on phone number
    for contact in contacts:
        if new_phone == contact[1]:  # Duplicate phone number
            messagebox.showinfo("Duplicate Contact", "Contact already exists. Use Update to modify it.")
            return

    # Add new contact to the contacts list
    contacts.append([new_name, new_phone, new_email, new_address])
    refresh_display_contacts()

    # Clear input fields after saving contact
    contact_name_entry.delete(0, tk.END)
    phone_number_entry.delete(0, tk.END)
    email_id_entry.delete(0, tk.END)
    address_entry.delete(0, tk.END)

    messagebox.showinfo("Success", "Contact saved successfully!")  # Show success message

# Function to refresh the Listbox with the latest contacts
def refresh_display_contacts():
    display_saved_contact.delete(1, tk.END)  # Clear all items in the Listbox except the header
    for contact in contacts:  # Loop through saved contacts
        formatted_entry = f"    {contact[0]}    |   {contact[1]}     |   {contact[2]}    |   {contact[3]}"
        display_saved_contact.insert(tk.END, formatted_entry)  # Insert formatted contact details into Listbox

# Delete Contact Section
```

```python
def del_contact():
    contact_to_delete = delete_contact_name_entry.get().strip()  # Get input for name to delete
    phone_to_delete = delete_phone_number_entry.get().strip()  # Get input for phone number to delete

    if not contact_to_delete and not phone_to_delete:  # If neither is provided
        messagebox.showerror("Missing Input", "Provide either Name or Phone Number to delete.")
        return

    for contact in contacts:  # Loop through contacts to find the match
        if contact_to_delete == contact[0] or phone_to_delete == contact[1]:
            contacts.remove(contact)  # Remove contact from the list
            refresh_display_contacts()  # Refresh Listbox to reflect the changes
            messagebox.showinfo("Success", "Contact deleted successfully.")
            delete_contact_name_entry.delete(0, tk.END)  # Clear input fields
            delete_phone_number_entry.delete(0, tk.END)
            return

    messagebox.showerror("Not Found", "No matching contact found.")  # If no match was found

# Update Contact Section
def update_contact():
    search_value = update_search_entry.get().strip()  # Get input search value

    if not search_value:  # If search field is empty
        messagebox.showerror("Missing Input", "Provide a Name, Email, or Address to update.")
        return

    # Loop through contacts to find a match and populate fields for updating
    for contact in contacts:
        if search_value in contact[0].lower() or search_value in contact[1].lower() or search_value in contact[2].lower() or search_value
in contact[3].lower():
            # Populate update fields with existing contact details
            update_name_entry.delete(0, tk.END)
            update_phone_number_entry.delete(0, tk.END)
            update_email_entry.delete(0, tk.END)
            update_address_entry.delete(0, tk.END)

            update_name_entry.insert(0, contact[0])
            update_phone_number_entry.insert(0, contact[1])
            update_email_entry.insert(0, contact[2])
            update_address_entry.insert(0, contact[3])
            return

    messagebox.showerror("Not Found", "No contact found with the provided details.")  # If no contact found for update

def apply_update():
    search_value = update_search_entry.get().strip()  # Get search value
    updated_name = update_name_entry.get().strip()  # Get updated contact details
    updated_phone = update_phone_number_entry.get().strip()
    updated_email = update_email_entry.get().strip()
    updated_address = update_address_entry.get().strip()

    for contact in contacts:
        if search_value in contact[0].lower() or search_value in contact[1].lower() or search_value in contact[2].lower() or search_value in contact[3].lower():
            # Update contact details
            contact[0] = updated_name if updated_name else contact[0]
            contact[1] = updated_phone if updated_phone else contact[1]
            contact[2] = updated_email if updated_email else contact[2]
            contact[3] = updated_address if updated_address else contact[3]

            refresh_display_contacts()  # Refresh Listbox with updated details
            messagebox.showinfo("Success", "Contact updated successfully.")
            return

    messagebox.showerror("Not Found", "No contact found with the provided details.")  # If no contact found for update

# Input Fields for Save Contact (Name, Phone, Email, Address)
contact_name_label = ttk.Label(add_frame, text="Name:", font=large_font, background="light gray")
contact_name_label.place(x=20, y=70)
contact_name_entry = ttk.Entry(add_frame, width=35)
contact_name_entry.place(x=150, y=70)

phone_number_label = ttk.Label(add_frame, text="Phone Number:", font=large_font, background="light gray")
```

```python
phone_number_label.place(x=20, y=120)
phone_number_entry = ttk.Entry(add_frame, width=35)
phone_number_entry.place(x=150, y=120)

email_id_label = ttk.Label(add_frame, text="Email ID:", font=large_font, background="light gray")
email_id_label.place(x=20, y=170)
email_id_entry = ttk.Entry(add_frame, width=35)
email_id_entry.place(x=150, y=170)

address_label = ttk.Label(add_frame, text="Address:", font=large_font, background="light gray")
address_label.place(x=20, y=220)
address_entry = ttk.Entry(add_frame, width=35)
address_entry.place(x=150, y=220)

save_contact_button = ttk.Button(add_frame, text="Save Contact", width=20, command=new_contact_insertion)
save_contact_button.place(x=230, y=270)

# Input Fields for Delete Contact (Name and Phone Number)
delete_contact_name_label = ttk.Label(delete_frame, text="Name:", font=large_font, background="light gray")
delete_contact_name_label.place(x=20, y=100)
delete_contact_name_entry = ttk.Entry(delete_frame, width=35)
delete_contact_name_entry.place(x=150, y=100)

delete_phone_number_label = ttk.Label(delete_frame, text="Phone Number:", font=large_font, background="light gray")
delete_phone_number_label.place(x=20, y=170)
delete_phone_number_entry = ttk.Entry(delete_frame, width=35)
delete_phone_number_entry.place(x=150, y=170)

delete_contact_button = ttk.Button(delete_frame, text="Delete Contact", width=20, command=del_contact)
delete_contact_button.place(x=150, y=250)

# Input Fields for Update Contact (Search, Name, Phone, Email, Address)
update_search_label = ttk.Label(update_frame, text="Search:", font=large_font, background="light gray")
update_search_label.place(x=20, y=80)
update_search_entry = ttk.Entry(update_frame, width=35)
update_search_entry.place(x=150, y=80)

```

```python
update_name_label = ttk.Label(update_frame, text="Name:", font=large_font, background="light gray")
update_name_label.place(x=20, y=130)
update_name_entry = ttk.Entry(update_frame, width=35)
update_name_entry.place(x=150, y=130)

update_phone_number_label = ttk.Label(update_frame, text="Phone Number:", font=large_font, background="light gray")
update_phone_number_label.place(x=20, y=160)
update_phone_number_entry = ttk.Entry(update_frame, width=35)
update_phone_number_entry.place(x=150, y=160)

update_email_label = ttk.Label(update_frame, text="Email ID:", font=large_font, background="light gray")
update_email_label.place(x=20, y=190)
update_email_entry = ttk.Entry(update_frame, width=35)
update_email_entry.place(x=150, y=190)

update_address_label = ttk.Label(update_frame, text="Address:", font=large_font, background="light gray")
update_address_label.place(x=20, y=220)
update_address_entry = ttk.Entry(update_frame, width=35)
update_address_entry.place(x=150, y=220)

update_contact_button = ttk.Button(update_frame, text="Find Contact", width=20, command=update_contact)
update_contact_button.place(x=20, y=270)

apply_update_button = ttk.Button(update_frame, text="Apply Update", width=20, command=apply_update)
apply_update_button.place(x=230, y=270)

root.mainloop()  # Start the Tkinter main loop to run the app
```

OUTPUT:


Contact Book
Name | Phone Number | Email ID | Address
Search Contact here:
Search!
SAVE CONTACT
Name:
Phone Number:
Email ID:
Address:
Save Contact
DELETE CONTACT
Using either the Name or the Phone Number.
Name:
Phone Number:
Delete Contact
UPDATE CONTACT
Search and Update contact details.
Search:
Name:
Phone Number:
Email ID:
Address:
Find Contact
Apply Update