“Student Information”

In python

Ansari Arif(3117001)

Memon Shaziya(3117026)

Mishra Saloni(3117027)

Second Year Computer Engineering(Sem-IV)

Subject: Open Source Tech Lab(OSTL)

Prof. Asadullah Shaikh

Department of Computer Engineering

M. H. Saboo Siddik College of Engineering

8,Saboo Siddik Polytechnic Road, Byculla,

Mumbai, Maharashtra 400008

Index

Problem Statement:

Contact Management System in python. This GUI based Contact Management system provides the simplest management of contact details. In short, this projects mainly focus on CRUD. There's an external database connection file. This is a simple GUI based project which is very easy to understand and use. Talking about the system, it contains all the required functions which include adding, viewing, deleting and updating contact lists.

Theory:

Contact Management System project is written in Python. The project file contains a python script (index.py). In contact management system, while adding the contact of a person, he/she has to provide first name, last name, gender, address and contact details. The user can also update the contact list if he/she wants to. For this, the user has to double-click on a record that he/she wishes to edit. The system shows the contact details in a list view. And also the user easily delete any contact details.

In order to run the project, you must have installed Python, on your PC. This is a simple GUI Based system, specially written for the beginners. Contact Management System in Python project with source code is free to download. Use for education purpose only! For the project demo, have a look at the image slider below.

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

Explaination:

To create a tkinter:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x is 'tkinter'.

```
import tkinter
```

There are two main methods used you the user need to remember while creating the Python application with GUI.

1. **Tk(screenName=None, baseName=None, className ='Tk', useTk=1):** To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can

change the className to the desired one. The basic code used to create the main window of the application is:

```
m=tkinter.Tk() where m is the name of the main window
object
```

2. **mainloop():** There is a method known by the name mainloop() is used when you are ready for the application to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.

```
m.mainloop()
```

tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes class.

1. **pack() method:**It organizes the widgets in blocks before placing in the parent widget.
2. **grid() method:**It organizes the widgets in grid (table-like structure) before placing in the parent widget.
3. **place() method:**It organizes the widgets by placing them on specific positions directed by the programmer.

There are a number of widgets which you can put in your tkinter application. Some of the major widgets are explained below:

Button:To add a button in your application, this widget is used. The general syntax is:

```
w=Button(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **activebackground**: to set the background color when button is under the cursor.
- **activeforeground**: to set the foreground color when button is under the cursor.
- **bg**: to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the button.
- **width**: to set the width of the button.
- **height**: to set the height of the button.

Entry: It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used.
The general syntax is:

```
w=Entry(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas.

Label: It refers to the display box where you can put any text or image which can be updated any time as per the code.
The general syntax is:

```
w=Label(master, option=value)
```

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas.

Database Connection using sqlit3:

SQLite3 can be integrated with Python using sqlite3 module, which was written by Gerhard Haring. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249. You do not need to install this module separately because it is shipped by default along with Python version 2.5.x onwards.

To use sqlite3 module, you must first create a connection object that represents the database and then optionally you can create a cursor object, which will help you in executing all the SQL statements.

Sr.No.	API & Description
1	**sqlite3.connect(database [,timeout ,other optional arguments])** This API opens a connection to the SQLite database file. You can use ":memory:" to open a database connection to a database that resides in RAM instead of on disk. If

	database is opened successfully, it returns a connection object. When a database is accessed by multiple connections, and one of the processes modifies the database, the SQLite database is locked until that transaction is committed. The timeout parameter specifies how long the connection should wait for the lock to go away until raising an exception. The default for the timeout parameter is 5.0 (five seconds). If the given database name does not exist then this call will create the database. You can specify filename with the required path as well if you want to create a database anywhere else except in the current directory.
2	**connection.cursor([cursorClass])** This routine creates a **cursor** which will be used throughout of your database programming with Python. This method accepts a single optional parameter cursorClass. If supplied, this must be a

	custom cursor class that extends sqlite3.Cursor.
3	**cursor.execute(sql [, optional parameters])** This routine executes an SQL statement. The SQL statement may be parameterized (i. e. placeholders instead of SQL literals). The sqlite3 module supports two kinds of placeholders: question marks and named placeholders (named style). **For example** – cursor.execute("insert into people values (?, ?)", (who, age))
4	**connection.execute(sql [, optional parameters])** This routine is a shortcut of the above execute method provided by the cursor object and it creates an intermediate cursor object by calling the cursor method, then calls the cursor's execute method with the parameters given.
5	**connection.executemany(sql[, parameters])**

	This routine is a shortcut that creates an intermediate cursor object by calling the cursor method, then calls the cursor.s executemany method with the parameters given.
6	**cursor.executescript(sql_script)** This routine executes multiple SQL statements at once provided in the form of script. It issues a COMMIT statement first, then executes the SQL script it gets as a parameter. All the SQL statements should be separated by a semi colon (;).
7	**connection.executescript(sql_script)** This routine is a shortcut that creates an intermediate cursor object by calling the cursor method, then calls the cursor's executescript method with the parameters given.
8	**connection.total_changes()** This routine returns the total number of database rows that have been modified, inserted, or deleted since the database connection was opened.

9	**connection.commit()** This method commits the current transaction. If you don't call this method, anything you did since the last call to commit() is not visible from other database connections.
10	**connection.close()** This method closes the database connection. Note that this does not automatically call commit(). If you just close your database connection without calling commit() first, your changes will be lost!
11	**cursor.fetchmany([size = cursor.arraysize])** This routine fetches the next set of rows of a query result, returning a list. An empty list is returned when no more rows are available. The method tries to fetch as many rows as indicated by the size parameter.
12	**cursor.fetchall()**

	This routine fetches all (remaining) rows of a query result, returning a list. An empty list is returned when no rows are available.

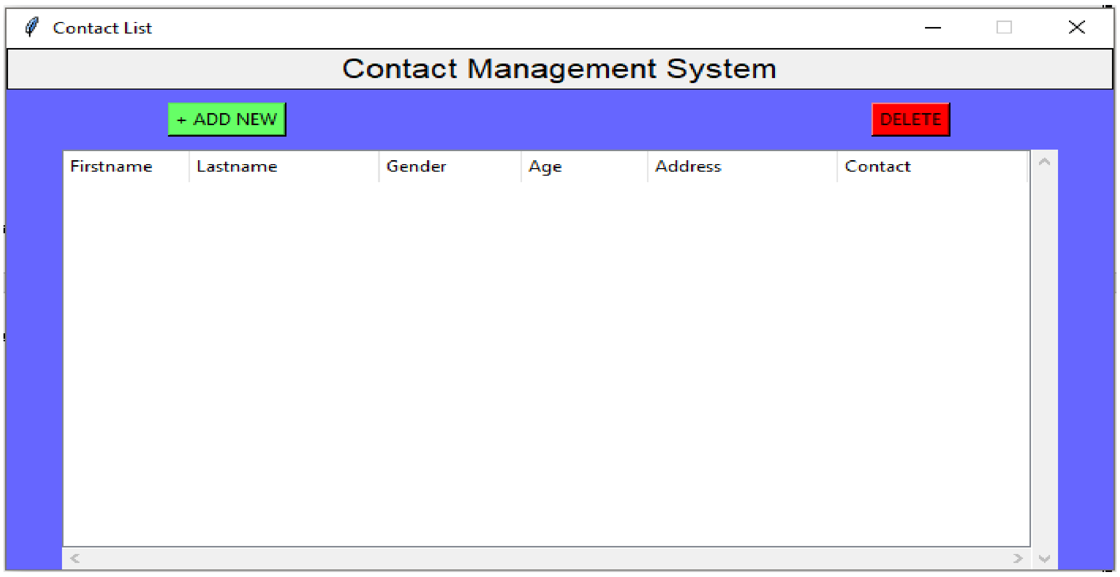
Program:

```
from tkinter import *
import sqlite3
import tkinter.ttk as ttk
import tkinter.messagebox as tkMessageBox

root = Tk()
root.title("Contact List")
width = 700
height = 400
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)
root.geometry("%dx%d+%d+%d" % (width, height, x, y))
root.resizable(0, 0)
root.config(bg="#6666ff")

#============================VARIABLES===================================
FIRSTNAME = StringVar()
LASTNAME = StringVar()
GENDER = StringVar()
AGE = StringVar()
ADDRESS = StringVar()
CONTACT = StringVar()



#============================METHODS=====================================

def Database():
    conn = sqlite3.connect("pythontut.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id INTEGER NOT
NULL  PRIMARY KEY AUTOINCREMENT, firstname TEXT, lastname TEXT, gender TEXT,
age TEXT, address TEXT, contact TEXT)")
    cursor.execute("SELECT * FROM `member` ORDER BY `lastname` ASC")
    fetch = cursor.fetchall()
    for data in fetch:
        tree.insert('', 'end', values=(data))
    cursor.close()
    conn.close()

def SubmitData():
    if  FIRSTNAME.get() == "" or LASTNAME.get() == "" or GENDER.get() == ""
or AGE.get() == "" or ADDRESS.get() == "" or CONTACT.get() == "":
```

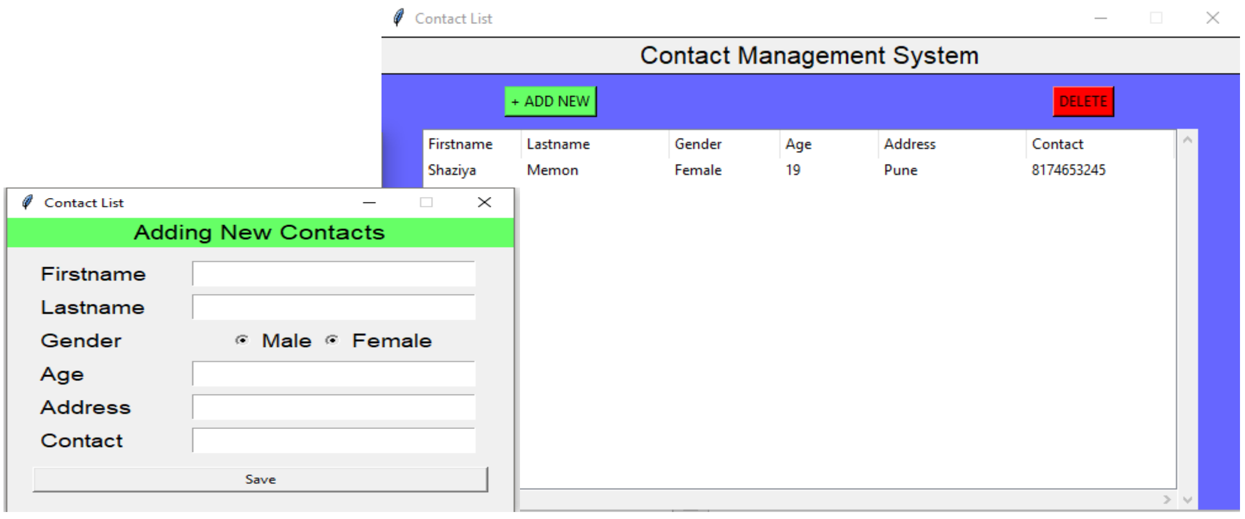
```
            result = tkMessageBox.showwarning('', 'Please Complete The Required
    Field', icon="warning")
        else:
            tree.delete(*tree.get_children())
            conn = sqlite3.connect("pythontut.db")
            cursor = conn.cursor()
            cursor.execute("INSERT INTO `member` (firstname, lastname, gender,
    age, address, contact) VALUES(?, ?, ?, ?, ?, ?)", (str(FIRSTNAME.get()),
    str(LASTNAME.get()), str(GENDER.get()), int(AGE.get()), str(ADDRESS.get()),
    str(CONTACT.get())))
            conn.commit()
            cursor.execute("SELECT * FROM `member` ORDER BY `lastname` ASC")
            fetch = cursor.fetchall()
            for data in fetch:
                tree.insert('', 'end', values=(data))
            cursor.close()
            conn.close()
            FIRSTNAME.set("")
            LASTNAME.set("")
            GENDER.set("")
            AGE.set("")
            ADDRESS.set("")
            CONTACT.set("")

    def UpdateData():
        if GENDER.get() == "":
           result = tkMessageBox.showwarning('', 'Please Complete The Required
    Field', icon="warning")
        else:
            tree.delete(*tree.get_children())
            conn = sqlite3.connect("pythontut.db")
            cursor = conn.cursor()
            cursor.execute("UPDATE `member` SET `firstname` = ?, `lastname` = ?,
    `gender` =?, `age` = ?,  `address` = ?, `contact` = ? WHERE `mem_id` = ?",
    (str(FIRSTNAME.get()), str(LASTNAME.get()), str(GENDER.get()),
    str(AGE.get()), str(ADDRESS.get()), str(CONTACT.get()), int(mem_id)))
            conn.commit()
            cursor.execute("SELECT * FROM `member` ORDER BY `lastname` ASC")
            fetch = cursor.fetchall()
            for data in fetch:
                tree.insert('', 'end', values=(data))
            cursor.close()
            conn.close()
            FIRSTNAME.set("")
            LASTNAME.set("")
            GENDER.set("")
            AGE.set("")
            ADDRESS.set("")
            CONTACT.set("")


    def OnSelected(event):
        global mem_id, UpdateWindow
        curItem = tree.focus()
        contents =(tree.item(curItem))
        selecteditem = contents['values']
        mem_id = selecteditem[0]
        FIRSTNAME.set("")
        LASTNAME.set("")
        GENDER.set("")
        AGE.set("")
        ADDRESS.set("")
        CONTACT.set("")
        FIRSTNAME.set(selecteditem[1])
        LASTNAME.set(selecteditem[2])
        AGE.set(selecteditem[4])
        ADDRESS.set(selecteditem[5])
        CONTACT.set(selecteditem[6])
        UpdateWindow = Toplevel()
        UpdateWindow.title("Contact List")
        width = 400
        height = 300
        screen_width = root.winfo_screenwidth()
        screen_height = root.winfo_screenheight()
        x = ((screen_width/2) + 450) - (width/2)
        y = ((screen_height/2) + 20) - (height/2)
        UpdateWindow.resizable(0, 0)
        UpdateWindow.geometry("%dx%d+%d+%d" % (width, height, x, y))
        if 'NewWindow' in globals():
            NewWindow.destroy()

        #===================FRAMES==============================
        FormTitle = Frame(UpdateWindow)
        FormTitle.pack(side=TOP)
        ContactForm = Frame(UpdateWindow)
        ContactForm.pack(side=TOP, pady=10)
        RadioGroup = Frame(ContactForm)
        Male = Radiobutton(RadioGroup, text="Male", variable=GENDER,
    value="Male",  font=('arial', 14)).pack(side=LEFT)
        Female = Radiobutton(RadioGroup, text="Female", variable=GENDER,
    value="Female",  font=('arial', 14)).pack(side=LEFT)

        #===================LABELS==============================
        lbl_title = Label(FormTitle, text="Updating Contacts", font=('arial',
    16), bg="orange",  width = 300)
        lbl_title.pack(fill=X)
        lbl_firstname = Label(ContactForm, text="Firstname", font=('arial', 14),
    bd=5)
        lbl_firstname.grid(row=0, sticky=W)
        lbl_lastname = Label(ContactForm, text="Lastname", font=('arial', 14),
    bd=5)
        lbl_lastname.grid(row=1, sticky=W)
        lbl_gender = Label(ContactForm, text="Gender", font=('arial', 14), bd=5)
        lbl_gender.grid(row=2, sticky=W)
        lbl_age = Label(ContactForm, text="Age", font=('arial', 14), bd=5)
        lbl_age.grid(row=3, sticky=W)
        lbl_address = Label(ContactForm, text="Address", font=('arial', 14),
    bd=5)
        lbl_address.grid(row=4, sticky=W)
        lbl_contact = Label(ContactForm, text="Contact", font=('arial', 14),
    bd=5)
        lbl_contact.grid(row=5, sticky=W)

        #===================ENTRY===============================
        firstname = Entry(ContactForm, textvariable=FIRSTNAME, font=('arial',
    14))
        firstname.grid(row=0, column=1)
        lastname = Entry(ContactForm, textvariable=LASTNAME, font=('arial', 14))
        lastname.grid(row=1, column=1)
        RadioGroup.grid(row=2, column=1)
        age = Entry(ContactForm, textvariable=AGE,  font=('arial', 14))
        age.grid(row=3, column=1)
        address = Entry(ContactForm, textvariable=ADDRESS,  font=('arial', 14))
        address.grid(row=4, column=1)
        contact = Entry(ContactForm, textvariable=CONTACT,  font=('arial', 14))
        contact.grid(row=5, column=1)


        #==================BUTTONS==============================
        btn_updatecon = Button(ContactForm, text="Update", width=50,
    command=UpdateData)
        btn_updatecon.grid(row=6, columnspan=2, pady=10)


    #fn1353p
    def DeleteData():
        if not tree.selection():
           result = tkMessageBox.showwarning('', 'Please Select Something
    First!', icon="warning")
        else:
            result = tkMessageBox.askquestion('', 'Are you sure you want to
    delete this record?', icon="warning")
            if result == 'yes':
                curItem = tree.focus()
                contents =(tree.item(curItem))
                selecteditem = contents['values']
                tree.delete(curItem)
                conn = sqlite3.connect("pythontut.db")
                cursor = conn.cursor()
                cursor.execute("DELETE FROM `member` WHERE `mem_id` = %d" %
    selecteditem[0])
                conn.commit()
                cursor.close()
                conn.close()

    def AddNewWindow():
        global NewWindow
        FIRSTNAME.set("")
        LASTNAME.set("")
        GENDER.set("")
        AGE.set("")
        ADDRESS.set("")
        CONTACT.set("")
        NewWindow = Toplevel()
        NewWindow.title("Contact List")
        width = 400
        height = 300
        screen_width = root.winfo_screenwidth()
        screen_height = root.winfo_screenheight()
        x = ((screen_width/2) - 455) - (width/2)
        y = ((screen_height/2) + 20) - (height/2)
        NewWindow.resizable(0, 0)
        NewWindow.geometry("%dx%d+%d+%d" % (width, height, x, y))
        if 'UpdateWindow' in globals():
            UpdateWindow.destroy()

        #===================FRAMES==============================
        FormTitle = Frame(NewWindow)
        FormTitle.pack(side=TOP)
        ContactForm = Frame(NewWindow)
        ContactForm.pack(side=TOP, pady=10)
        RadioGroup = Frame(ContactForm)
        Male = Radiobutton(RadioGroup, text="Male", variable=GENDER,
    value="Male",  font=('arial', 14)).pack(side=LEFT)
        Female = Radiobutton(RadioGroup, text="Female", variable=GENDER,
    value="Female",  font=('arial', 14)).pack(side=LEFT)

        #===================LABELS==============================
        lbl_title = Label(FormTitle, text="Adding New Contacts", font=('arial',
    16), bg="#66ff66",  width = 300)
        lbl_title.pack(fill=X)
        lbl_firstname = Label(ContactForm, text="Firstname", font=('arial', 14),
    bd=5)
        lbl_firstname.grid(row=0, sticky=W)
        lbl_lastname = Label(ContactForm, text="Lastname", font=('arial', 14),
    bd=5)
        lbl_lastname.grid(row=1, sticky=W)
        lbl_gender = Label(ContactForm, text="Gender", font=('arial', 14), bd=5)
        lbl_gender.grid(row=2, sticky=W)
        lbl_age = Label(ContactForm, text="Age", font=('arial', 14), bd=5)
        lbl_age.grid(row=3, sticky=W)
        lbl_address = Label(ContactForm, text="Address", font=('arial', 14),
    bd=5)
        lbl_address.grid(row=4, sticky=W)
        lbl_contact = Label(ContactForm, text="Contact", font=('arial', 14),
    bd=5)
        lbl_contact.grid(row=5, sticky=W)

        #===================ENTRY===============================
        firstname = Entry(ContactForm, textvariable=FIRSTNAME, font=('arial',
    14))
        firstname.grid(row=0, column=1)
        lastname = Entry(ContactForm, textvariable=LASTNAME, font=('arial', 14))
        lastname.grid(row=1, column=1)
        RadioGroup.grid(row=2, column=1)
        age = Entry(ContactForm, textvariable=AGE,  font=('arial', 14))
        age.grid(row=3, column=1)
        address = Entry(ContactForm, textvariable=ADDRESS,  font=('arial', 14))
        address.grid(row=4, column=1)
        contact = Entry(ContactForm, textvariable=CONTACT,  font=('arial', 14))
        contact.grid(row=5, column=1)


        #==================BUTTONS==============================
        btn_addcon = Button(ContactForm, text="Save", width=50,
    command=SubmitData)
        btn_addcon.grid(row=6, columnspan=2, pady=10)




    #============================FRAMES======================================
    Top = Frame(root, width=500, bd=1, relief=SOLID)
    Top.pack(side=TOP)
    Mid = Frame(root, width=500,  bg="#6666ff")
    Mid.pack(side=TOP)
    MidLeft = Frame(Mid, width=100)
    MidLeft.pack(side=LEFT, pady=10)
    MidLeftPadding = Frame(Mid, width=370, bg="#6666ff")
    MidLeftPadding.pack(side=LEFT)
    MidRight = Frame(Mid, width=100)
    MidRight.pack(side=RIGHT, pady=10)
    TableMargin = Frame(root, width=500)
```

```
TableMargin.pack(side=TOP)
#============================LABELS======================================
lbl_title = Label(Top, text="Contact Management System", font=('arial', 16),
width=500)
lbl_title.pack(fill=X)

#============================ENTRY=======================================

#============================BUTTONS=====================================
btn_add = Button(MidLeft, text="+ ADD NEW", bg="#66ff66",
command=AddNewWindow)
btn_add.pack()
btn_delete = Button(MidRight, text="DELETE", bg="red", command=DeleteData)
btn_delete.pack(side=RIGHT)

#============================TABLES======================================
scrollbarx = Scrollbar(TableMargin, orient=HORIZONTAL)
scrollbary = Scrollbar(TableMargin, orient=VERTICAL)
tree = ttk.Treeview(TableMargin, columns=("MemberID", "Firstname",
"Lastname", "Gender", "Age", "Address", "Contact"), height=400,
selectmode="extended", yscrollcommand=scrollbary.set,
xscrollcommand=scrollbarx.set)
scrollbary.config(command=tree.yview)
scrollbary.pack(side=RIGHT, fill=Y)
scrollbarx.config(command=tree.xview)
scrollbarx.pack(side=BOTTOM, fill=X)
tree.heading('MemberID', text="MemberID", anchor=W)
tree.heading('Firstname', text="Firstname", anchor=W)
tree.heading('Lastname', text="Lastname", anchor=W)
tree.heading('Gender', text="Gender", anchor=W)
tree.heading('Age', text="Age", anchor=W)
tree.heading('Address', text="Address", anchor=W)
tree.heading('Contact', text="Contact", anchor=W)
tree.column('#0', stretch=NO, minwidth=0, width=0)
tree.column('#1', stretch=NO, minwidth=0, width=0)
tree.column('#2', stretch=NO, minwidth=0, width=80)
tree.column('#3', stretch=NO, minwidth=0, width=120)
tree.column('#4', stretch=NO, minwidth=0, width=90)
tree.column('#5', stretch=NO, minwidth=0, width=80)
tree.column('#6', stretch=NO, minwidth=0, width=120)
tree.column('#7', stretch=NO, minwidth=0, width=120)
tree.pack()
tree.bind('<Double-Button-1>', OnSelected)

#============================INITIALIZATION==============================
if __name__ == '__main__':
    Database()
    root.mainloop()
```

Output:

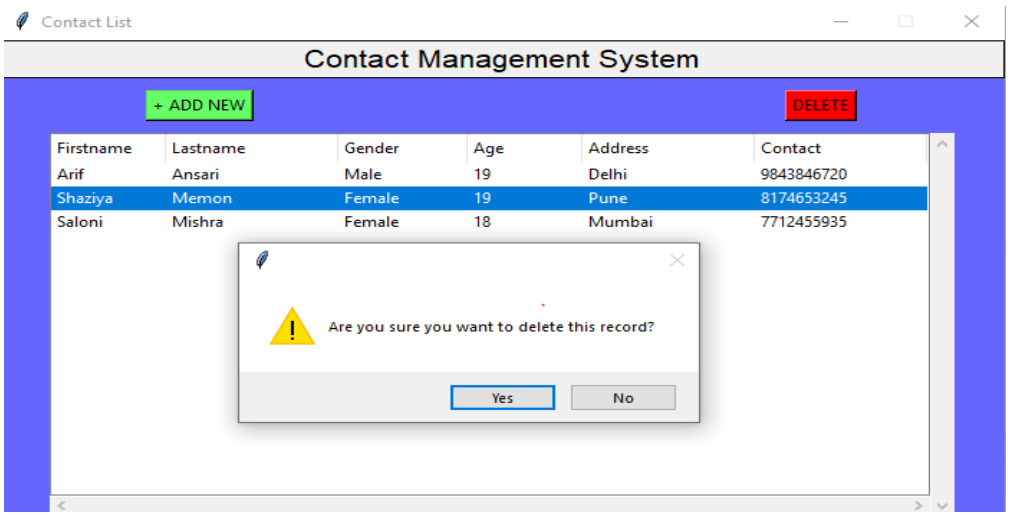


Firstname	Lastname	Gender	Age	Address	Contact



Firstname	Lastname	Gender	Age	Address	Contact
Shaziya	Memon	Female	19	Pune	8174653245

Firstname	
Lastname	
Gender	Male Female
Age	
Address	
Contact	

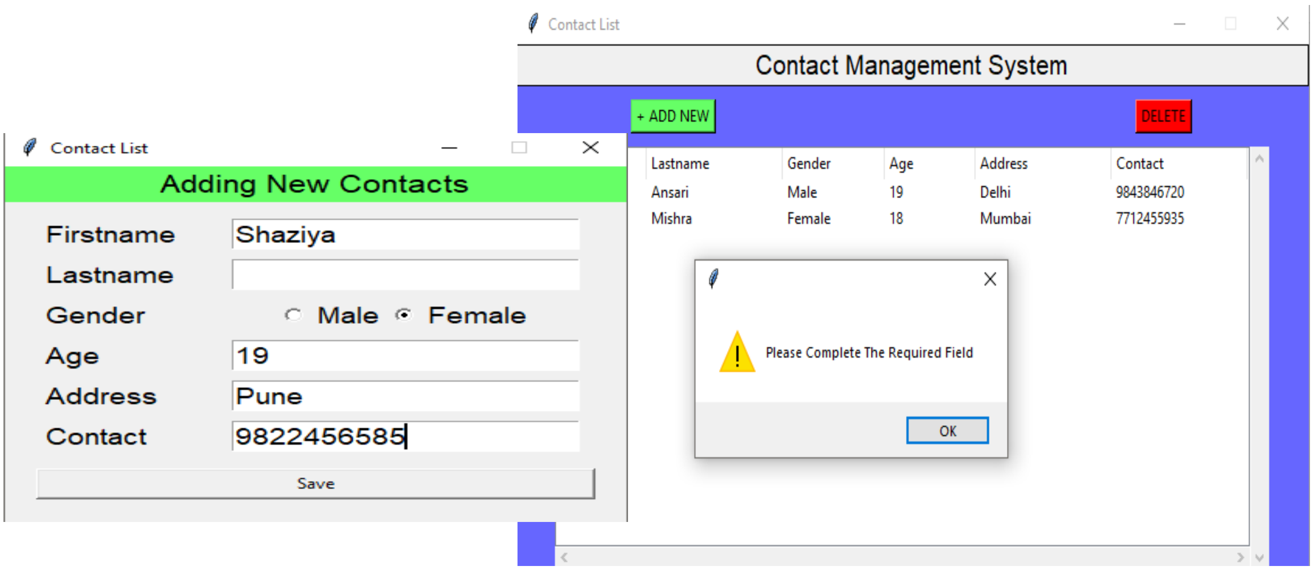


Contact List

Contact Management System

Firstname	Lastname	Gender	Age	Address	Contact
Arif	Ansari	Male	19	Delhi	9843846720
Shaziya	Memon	Female	19	Pune	8174653245
Saloni	Mishra	Female	18	Mumbai	7712455935

Are you sure you want to delete this record?



Contact List

Adding New Contacts

Firstname	Shaziya
Lastname	
Gender	Male [✓ Female]
Age	19
Address	Pune
Contact	9822456585

Contact List

Contact Management System

Lastname	Gender	Age	Address	Contact
Ansari	Male	19	Delhi	9843846720
Mishra	Female	18	Mumbai	7712455935

Please Complete The Required Field