

A

Project Report

On

## **PHONE BOOK**

Submitted in partial fulfilment of

The requirements for the 4<sup>th</sup> Semester Sessional Examination of

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

Session: 2019-23

Submitted by:

Mayank Mohak, Registration No: 1901060170

A. Himadri Prasad Achary, Regd. No.- 1901070008

Md. Faizan Ansari, Regd. No- 1901060164

Under the guidance of

Mr. P Sudheer Babu



Department of Computer Science & Engineering

GIET UNIVERSITY, Gunupur, Odisha

2020-2021

## CERTIFICATE

This is to certify that the project work entitled “**PHONE BOOK**” is done by **Mayank Mohak (Regd. No.- 1901060170), A. Himadri Prasad Achary (Regd. No.- 1901070008), Md. Faizan Ansari, Regd. No- 1901060164,** in partial fulfilment of the requirements for the 4<sup>th</sup> Semester Sessional Examination of Bachelor of Technology in **Computer Science and Engineering** during the academic year 2020-21. This work is submitted to the department as a Part of evaluation of 4<sup>th</sup> Semester Project.

**Mr. P Sudheer Babu**  
Class Teacher

**Prof. (Dr) .Sanjay Kumar Kuanar**  
HoD, CSE

**EXCELLENCE - OUR ESSENCE**

## **ACKNOWLEDGEMENT**

We deem it a pleasure to acknowledge your sense of gratitude our project guide Mr. P Sudheer Babu under whom we have carried out the project work, her incisive and objective guidance timely advice encouraged constant flow of energy to continue the work. We wish to reciprocate in full measure the kindness shown by Mr. P Sudheer Babu who inspired us with her suggestions in successfully completing the project work.

We are also grateful to our HOD Prof. (Dr) Sanjay Kumar Kuanar of CSE department for providing Moral support and adequate facility in completing this project.

Finally we must say that no height is ever achieved without some sacrifice made at some end and it is here where we owe our special debt to our parents and our friends for showing their generous love and care throughout the entire period of time.

GIETU, ODISHA

MD. FAIZAN ANSARI

A. HIMADRI PRASAD ACHARY

MAYANK MOHAK

EXCELLENCE - OUR ESSENCE

## **ABSTRACTION:**

This project (**PHONE BOOK**) deals with the maintenance of the Contact details. It generates the Contact on basis of Data Entered. Phone Contact can be Entered Edited and Deleted with input validation.

We have made this project by using **PYTHON ( tkinter )**. The main purpose of this project is to easily manage contacts on desktop.

This is built on python's GUI tool tkinter and is very easy to handle and secure as we use SQLite as database.

Hence It was just the briefly description of the project. Let's discuss about the project thoroughly.



## INTRODUCTION:

### ➤ Purpose:-

This project solves problem of checking contacts on phone and tough to manage two device.

### ➤ Project Scope:-

The project is very simple to understand, And the main point is, it will take very less time to manage contacts and do irritating task in a fun way.

### ➤ Product Features:-

This project is made by using PYTHON (tkinter).

First upon when we run the code a graphical user interface will be shown. That's the home page or we can say main page of our Application. There is a panel to enter Edit and delete contact separately.

## System Analysis

### Software Requirements:-

- OS: as PYTHON is platform independent so we can run on any platform
- Language used: PYTHON 3
- Graphic user interface: tkinter
- Any Terminal

### Brief overview of the Technology:

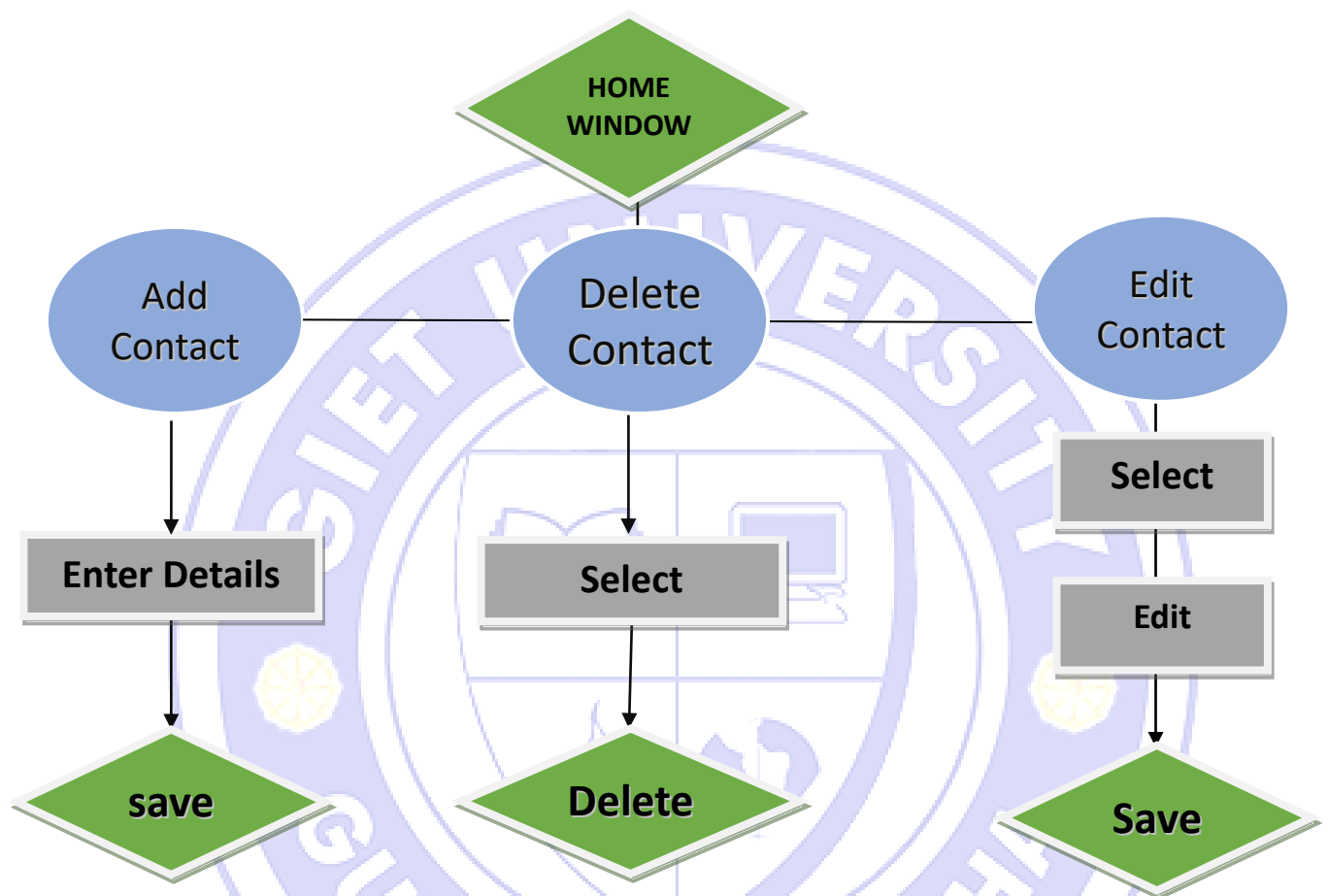
**PYTHON:-** Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.

**Tkinter:-** Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name Tkinter comes from Tk interface.

**SQLite:-** SQLite is a relational database management system contained in a C library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program. SQLite generally follows PostgreSQL syntax.

## System Design and Specifications

### FLOW CHART



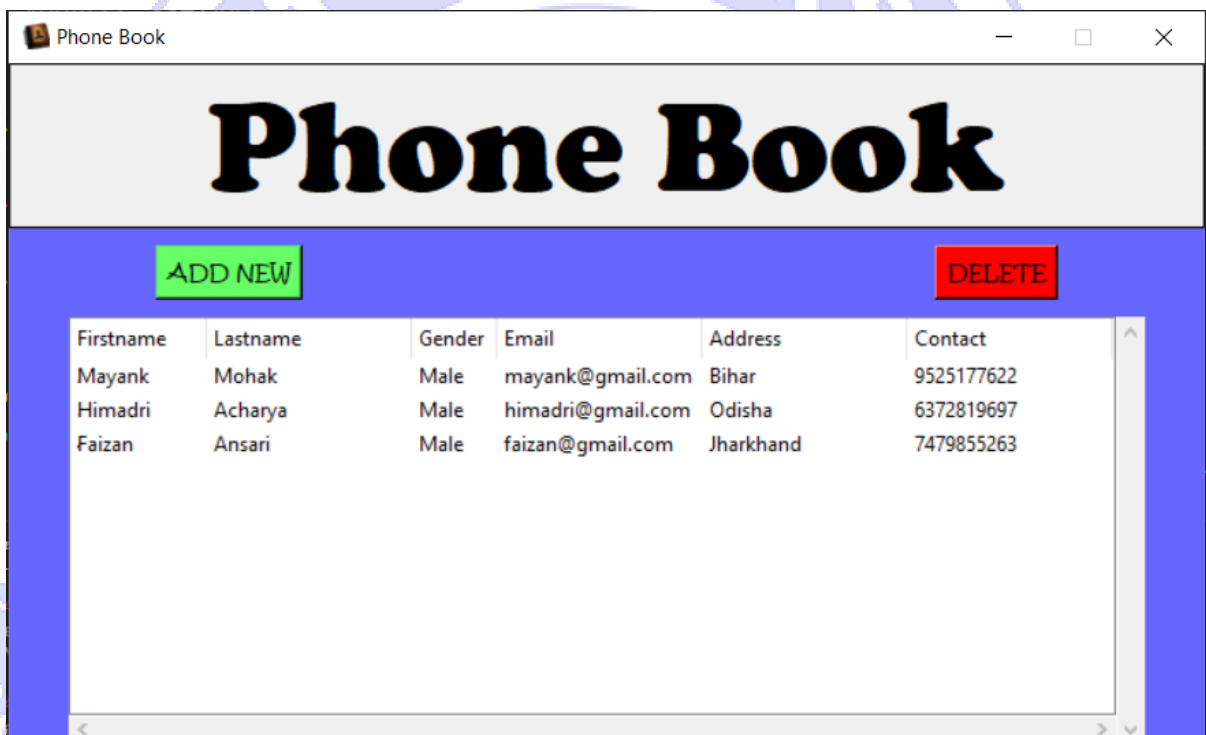
## Process Specification:

### Screenshot Diagrams:-

- ❖ Command line Terminal from where we execute this program.

```
Mohit@mayankmohak MINGW64 /d/workspace/project/4th/Contact
$ python contact.py
```

- ❖ Home Screen.



The Home Screen of the Phone Book application is displayed in a window titled "Phone Book". It features a large title "Phone Book" at the top. Below the title, there are two buttons: "ADD NEW" (green) and "DELETE" (red). A table lists the contacts with columns for Firstname, Lastname, Gender, Email, Address, and Contact.

Firstname	Lastname	Gender	Email	Address	Contact
Mayank	Mohak	Male	mayank@gmail.com	Bihar	9525177622
Himadri	Acharya	Male	himadri@gmail.com	Odisha	6372819697
Faizan	Ansari	Male	faizan@gmail.com	Jharkhand	7479855263

- ❖ New Entry Screen.



The New Entry Screen is titled "Add New" and "Adding New Contacts". It contains input fields for Firstname, Lastname, Email, Address, and Contact. The Gender field has radio buttons for Male and Female. A Save button is at the bottom.

Firstname

Lastname

Gender ☐ Male ☐ Female

Email

Address

Contact

❖ Edit Entry Screen.

Contact List

### Updating Contacts

Firstname	<input type="text" value="Mayank"/>
Lastname	<input type="text" value="Mohak"/>
Gender	<input checked="" type="radio"/> Male <input type="radio"/> Female
Email	<input type="text" value="mayank@gmail.com"/>
Address	<input type="text" value="Bihar"/>
Contact	<input type="text" value="9525177622"/>


❖ Wrong Entry

Add New

### Adding New Contacts

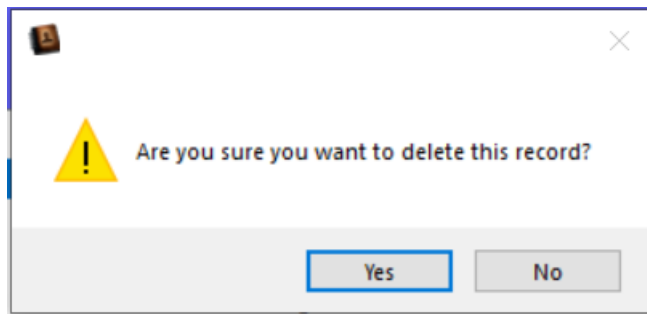
Firstname	<input type="text" value="Test"/>
Lastname	<input type="text" value="Email"/>
Gender	<input type="radio"/> Male <input checked="" type="radio"/> Female
Email	<input type="text" value="xyzexample.com"/>
Address	<input type="text" value="unknown"/>
Contact	<input type="text" value="0000000000"/>

✖

 EMAIL can be a valid email only



❖ Delete Entry Screen.



➤ CODES :

As previously mentioned we have used JAVA's Abstract Windowing ToolKit along with the basic of JAVA(swing) to make this project. Following we have mentioned the codes here.

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

root = Tk()
root.title("Phone Book")
root.iconbitmap('icon/contact.ico')
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("db/contact.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id
    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, firstname TEXT, l
    astname TEXT, gender TEXT, age TEXT, address TEXT, contact TEXT
    )")
    cursor.execute("SELECT * FROM `member` ORDER BY `firstname`
    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 CONTA
    CT.get() == "":
        result = tkMessageBox.showwarning('', 'Please Complete
    The Required Field', icon="warning")
    else:
        tree.delete(*tree.get_children())
        conn = sqlite3.connect("db/contact.db")
        cursor = conn.cursor()
        #input validation
        fname = str(FIRSTNAME.get())
        lname = str(LASTNAME.get())
        gen = str(GENDER.get())
        age = int(AGE.get())
        addr = str(ADDRESS.get())
        cont = str(CONTACT.get())

```

```

        if not checkname(fname):
            tkMessageBox.showwarning('', 'Only A-Z in name', icon="warning")
        elif not checkname(lname):
            tkMessageBox.showwarning('', 'Only A-Z in name', icon="warning")
        elif not checkgender(gen):
            tkMessageBox.showwarning('', 'gender can only be Male or Female', icon="warning")
        elif not checkage(age):
            tkMessageBox.showwarning('', 'age can be number only', icon="warning")
        elif not checkcontact(cont):
            tkMessageBox.showwarning('', 'Ph Number 10 digits only', icon="warning")
        else:
            cursor.execute("INSERT INTO `member` (firstname, lastname, gender, age, address, contact) VALUES(?, ?, ?, ?, ?, ?)", (fname, lname, gen, age, addr, cont))
            conn.commit()
            FIRSTNAME.set("")
            LASTNAME.set("")
            GENDER.set("-")
            AGE.set("")
            ADDRESS.set("")
            CONTACT.set("")
            NewWindow.destroy()
            cursor.execute("SELECT * FROM `member` ORDER BY `firstname` ASC")
            fetch = cursor.fetchall()
            for data in fetch:
                tree.insert('', 'end', values=(data))
            cursor.close()
            conn.close()

def UpdateData():
    if GENDER.get() == "":

```

```

        result = tkMessageBox.showwarning('', 'Please Complete
The Required Field', icon="warning")
    else:
        tree.delete(*tree.get_children())
        conn = sqlite3.connect("db/contact.db")
        cursor = conn.cursor()
        #input validation
        fname = str(FIRSTNAME.get())
        lname = str(LASTNAME.get())
        gen = str(GENDER.get())
        age = int(AGE.get())
        addr = str(ADDRESS.get())
        cont = str(CONTACT.get())
        if not checkname(fname):
            tkMessageBox.showwarning('', 'Only A-
Z in name', icon="warning")
        elif not checkname(lname):
            tkMessageBox.showwarning('', 'Only A-
Z in name', icon="warning")
        elif not checkgender(gen):
            tkMessageBox.showwarning('', 'gender can only be Ma
le or Female', icon="warning")
        elif not checkage(age):
            tkMessageBox.showwarning('', 'age can be number onl
y', icon="warning")
        elif not checkcontact(cont):
            tkMessageBox.showwarning('', 'Ph Number 10 digits o
nly', icon="warning")
        else:
            cursor.execute("UPDATE `member` SET `firstname` = ?
, `lastname` = ?, `gender` = ?, `age` = ?, `address` = ?, `cont
act` = ? WHERE `mem_id` = ?", (fname, lname, gen, age, addr, co
nt, int(mem_id)))
            conn.commit()
            FIRSTNAME.set("")
            LASTNAME.set("")
            GENDER.set("-")
            AGE.set("")

```

```
        ADDRESS.set("")
        CONTACT.set("")
        UpdateWindow.destroy()
        cursor.execute("SELECT * FROM `member` ORDER BY `firstn
ame` ASC")
        fetch = cursor.fetchall()
        for data in fetch:
            tree.insert('', 'end', values=(data))
        cursor.close()
        conn.close()
```

```
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])
    GENDER.set(selecteditem[3])
    AGE.set(selecteditem[4])
    ADDRESS.set(selecteditem[5])
    CONTACT.set(selecteditem[6])
    UpdateWindow = Toplevel()
    UpdateWindow.title("Contact List")
    width = 400
    height = 340
    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=('Kristen ITC', 14)).pack(side=LEFT)
Female = Radiobutton(RadioGroup, text="Female", variable=GENDER, value="Female", font=('Kristen ITC', 14)).pack(side=LEFT)

#=====LABELS=====
lbl_title = Label(FormTitle, text="Updating Contacts", font=('cooper black', 20), bg="orange", width = 300)
lbl_title.pack(fill=X)
lbl_firstname = Label(ContactForm, text="Firstname", font=('Kristen ITC', 14), bd=5)
lbl_firstname.grid(row=0, sticky=W)
lbl_lastname = Label(ContactForm, text="Lastname", font=('Kristen ITC', 14), bd=5)
lbl_lastname.grid(row=1, sticky=W)
lbl_gender = Label(ContactForm, text="Gender", font=('Kristen ITC', 14), bd=5)
lbl_gender.grid(row=2, sticky=W)
lbl_age = Label(ContactForm, text="Age", font=('Kristen ITC', 14), bd=5)
lbl_age.grid(row=3, sticky=W)
lbl_address = Label(ContactForm, text="Address", font=('Kristen ITC', 14), bd=5)
lbl_address.grid(row=4, sticky=W)
lbl_contact = Label(ContactForm, text="Contact", font=('Kristen ITC', 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 Som
ething 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("db/contact.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("Add New")
    width = 400
    height = 340
    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=('Kristen ITC', 14)).pack(side=LEFT)

```



```

        Female = Radiobutton(RadioGroup, text="Female", variable=GENDER, value="Female", font=('Kristen ITC', 14)).pack(side=LEFT)
    )

#=====LABELS=====
    lbl_title = Label(FormTitle, text="Adding New Contacts", font=('cooper black', 20), bg="#66ff66", width = 300)
    lbl_title.pack(fill=X)
    lbl_firstname = Label(ContactForm, text="Firstname", font=('Kristen ITC', 14), bd=5)
    lbl_firstname.grid(row=0, sticky=W)
    lbl_lastname = Label(ContactForm, text="Lastname", font=('Kristen ITC', 14), bd=5)
    lbl_lastname.grid(row=1, sticky=W)
    lbl_gender = Label(ContactForm, text="Gender", font=('Kristen ITC', 14), bd=5)
    lbl_gender.grid(row=2, sticky=W)
    lbl_age = Label(ContactForm, text="Age", font=('Kristen ITC', 14), bd=5)
    lbl_age.grid(row=3, sticky=W)
    lbl_address = Label(ContactForm, text="Address", font=('Kristen ITC', 14), bd=5)
    lbl_address.grid(row=4, sticky=W)
    lbl_contact = Label(ContactForm, text="Contact", font=('Kristen ITC', 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, com
mand=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="Phone Book", font=('Cooper Black',
58), width=500)
lbl_title.pack(fill=X)

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

#=====BUTTONS=====
btn_add = Button(MidLeft, text="ADD NEW", font=('Kristen ITC',1
0), bg="#66ff66", command=AddNewWindow)

```

```

btn_add.pack()
btn_delete = Button(MidRight, text="DELETE", font=('Kristen ITC', 10), 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()

```

## CONCLUSION

At last we want to say this project is very simple to use. It can be also modified later by using database work. We hope it will be helpful and it's also very simple to understand.

## LIMITATIONS

We have implemented as per our knowledge so we may not say that our project is 100 % perfect but we have tried to put it 100% accuracy from our end. Some of limitation according to us would be that

- i.) We are unable to add sync feature
- ii.) Our application is window based so we cannot implement it on paper to maintain records
- iii.) This project can only be used by someone who is able to handle computer.

## REFERENCE

For making this project we had taken help from these web sources:

- ❖ **javaTpoint** (for learning *tkinter*)
- ❖ **Our Teachers** for concept of GUI and PYTHON
- ❖ **Wikipedia**([wikipedia.org](https://www.wikipedia.org))
- ❖ **GeekforGeeks**
- ❖ **The PYTHON Tutorials** (from <https://docs.python.org/3/>)
- ❖ **Stack Overflow** for getting solution of doubts([stackoverflow.com](https://stackoverflow.com))
- ❖ **Our special thanks** to getting our relevant document and doubts.