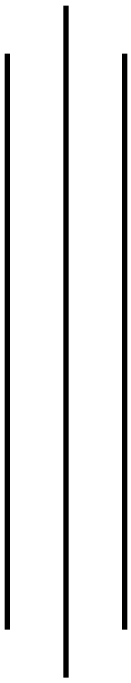
National College of Computer Studies

Paknajol, Kathmandu



Report on
Contact Management System

Submitted by:

Kapil Chhatkuli

Khusbu Shakya

Nischal Maharjan

Pujan Shakya

Shubhanjali Shakya

Student Declaration

This is to certify that I have completed the project entitled “Contact Management” under the guidance of “Mr. Subash Shrestha” in the partial fulfillment of the requirement for the degree of Bachelor of IT Management at the Faculty of Management, Tribhuvan University. This is the original work and we have not submitted earlier elsewhere.

Acknowledgement

“It is not possible to prepare a project report without the assistance and encouragement of other people. This one is certainly no exception. “

On the very outset of this report, we would like to extend our sincere and heartfelt obligations towards all the personages who have helped us in this endeavor. Without their active guidance help, co-operation and encouragement, we would not have made headway in the project.

We are ineffably indebted to Mr. Subash Shrestha for conscientious guidance and encouragement to accomplish this assignment. We are extremely thankful and pay our gratitude for his valuable guidance and support on completion of this project in its presently.

We extend our gratitude to National College of Computer Studies for giving us this opportunity.

We also acknowledge with the deep sense of reverence, our gratitude towards our parents and members of our family, who has always supported us morally as well as emotionally.

At last but not least gratitude goes to all our friends who directly or indirectly helped us to complete this project report.

Any omission in this brief acknowledgement does not mean lack of gratitude.

Thanking you

Abstract

Contact management system is a program designed especially for processing the contact. It is a contact especially designed for contact management like in call center, institution, offices where large number of contacts is to be processed. The system focuses on processing the name of the person, unique id, contact, address along with the concept of storing the customer details and displaying them when needed. This system also provides the feature of updating, viewing, deleting the customer records. New contact can be added easily and the existing contact can be updated easily. This contact management system is very effective as it is stored systematically and scientifically in the computer.

Contents

Chapter 1: Introduction

Python is an open-source (free) programming language that is used in web programming, data science, artificial intelligence, and many scientific applications. Learning Python allows the programmer to focus on solving problems, rather than focusing on syntax. Its relative size and simplified syntax give it an edge over languages like Java and C++, yet the abundance of libraries gives it the power needed to accomplish great things.

1.1 History

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wickenden & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "benevolent dictator for life", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a five-member "Steering Council" to lead the project.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector (in addition to reference counting) for memory management and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

1.2 What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

1.3 Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Chapter 2: APPENDIX

2.1 About program

This program is used to store the contact of the person. This program is fully developed in python. For Graphical User Interface we have imported the tkinter Library and sqlite3 for database.

2.2 User Manual:

2.2.1 How to Add New contact

First Press the “Add New” Button then enter the First name, Last name, Gender, Age, Address, Mobile no and Phone no. Then press the save button.

2.2.2 How to update contact

First select the contact that you want to update, click on the “update button”. Then change the data and click save.

2.2.3 How to delete contact

Select the contact you want to delete then click on delete button. Then the selected report will be deleted.

2.3 Source Code

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

root = Tk() #initialize tkinter object creation
root.title("Contact List")
width = 800
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="#a5afd9")
#============================VARIABLES=============================
=====
FNAME = StringVar()
LNAME = StringVar()
GENDER = StringVar()
AGE = StringVar()
ADDRESS = StringVar()
MOBILENO = StringVar()
PHONENO = StringVar()
#============================METHODS==============================
======
def Database():
    conn = sqlite3.connect("Contact_management.db")#data base bata data tana ko
    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, mobile TEXT,phoneno 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():
    conn = sqlite3.connect("Contact_management.db")
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM `member` WHERE `mobile` = ?
",[str(MOBILENO.get())])#data base bata mobile number hara ko
    fetch = cursor.fetchone()
    cursor.execute("SELECT * FROM `member` WHERE `phoneno` = ?
",[str(PHONENO.get())])
    fetchphone = cursor.fetchone()
    num = re.compile(r"^[0-9]+")

    if  FNAME.get() == "" or LNAME.get() == "" or GENDER.get() == "" or AGE.get() == "" or
ADDRESS.get() == "" or MOBILENO.get() == "":
        result = tkMessageBox.showwarning('', 'Please Complete The Required Field',
icon="warning")
    elif len(FNAME.get())>=20:
        result = tkMessageBox.showwarning('', 'First Name Can Be More Than 20 Characters!!!',
icon="warning")
    elif len(LNAME.get())>=20:
        result = tkMessageBox.showwarning('', 'Last Name Can Be More Than 20 Characters!!!',
icon="warning")
    elif(re.fullmatch(num,AGE.get())== None):
        result = tkMessageBox.showwarning('', 'Age can not be character!!!', icon="warning")
    elif (int(AGE.get())>110):
        result = tkMessageBox.showwarning('', 'Age Cant be More Than 110!!!', icon="warning")
    elif(int(AGE.get())<0):
        result = tkMessageBox.showwarning('', 'Age Cant be Less Than 0!!!', icon="warning")
    elif len(ADDRESS.get())>=20:
        result = tkMessageBox.showwarning('', 'Address can MOre Than 20 Characters!!!',
icon="warning")
    elif(re.fullmatch(num,MOBILENO.get())== None):
        result = tkMessageBox.showwarning('', 'Mobile no can not be character!!!',
icon="warning")
    elif len(MOBILENO.get())!=10:
        result = tkMessageBox.showwarning('', 'Mobile Can Not Be Less Than 10!!!',
icon="warning")
    elif(re.fullmatch(num,PHONENO.get())== None and PHONENO.get()!=""):
        result = tkMessageBox.showwarning('', 'Phone no can not be character!!!', icon="warning")
```

```
    elif (len(PHONENO.get())!=7 and PHONENO.get()!=""):
        result = tkMessageBox.showwarning('', 'Phone Cant Not Be Less Than 7!!!',
icon="warning")
    elif(fetch!=None):
        result = tkMessageBox.showwarning('', 'Mobile Number Already Exists!!!',
icon="warning")
    elif(fetchphone!=None and  PHONENO.get()!=""):
        result = tkMessageBox.showwarning('', 'Phone Number Already Exists!!!', icon="warning")
    else:
        tree.delete(*tree.get_children())
        conn = sqlite3.connect("Contact_management.db")
        cursor = conn.cursor()
        cursor.execute("INSERT INTO `member` (firstname, lastname, gender, age, address,
mobile,phoneno) VALUES(?, ?, ?, ?, ?, ?,?)", (str(FNAME.get()), str(LNAME.get()),
str(GENDER.get()), int(AGE.get()), str(ADDRESS.get()),
str(MOBILENO.get()),str(PHONENO.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()
        FNAME.set("")
        LNAME.set("")
        GENDER.set("")
        AGE.set("")
        ADDRESS.set("")
        MOBILENO.set("")
        PHONENO.set("")

def UpdateData():
    conn = sqlite3.connect("Contact_management.db")
    cursor = conn.cursor()
 #   cursor.execute("SELECT * FROM `member` WHERE `mobile` = ?
",[str(MOBILENO.get())])#data base bata mobile number hara ko
   # fetch = cursor.fetchone()
   # cursor.execute("SELECT * FROM `member` WHERE `phoneno` = ?
",[str(PHONENO.get())])
   # fetchphone = cursor.fetchone()
```

```
    num = re.compile(r"^[0-9]+")
    if GENDER.get() == "":
       result = tkMessageBox.showwarning('', 'Please Complete The Required Field In Gender',
icon="warning")
    if  FNAME.get() == "" or LNAME.get() == "" or GENDER.get() == "" or AGE.get() == "" or
ADDRESS.get() == "" or MOBILENO.get() == "":
        result = tkMessageBox.showwarning('', 'Please Complete The Required Field',
icon="warning")
    elif len(FNAME.get())>=20:
        result = tkMessageBox.showwarning('', 'First Name Can Be More Than 20 Characters!!!',
icon="warning")
    elif len(LNAME.get())>=20:
        result = tkMessageBox.showwarning('', 'Last Name Can Be More Than 20 Characters!!!',
icon="warning")
    elif(re.fullmatch(num,AGE.get())== None):
        result = tkMessageBox.showwarning('', 'Age can not be character!!!', icon="warning")
    elif (int(AGE.get())>110):
        result = tkMessageBox.showwarning('', 'Age Cant be More Than 110!!!', icon="warning")
    elif(int(AGE.get())<0):
        result = tkMessageBox.showwarning('', 'Age Cant be Less Than 0!!!', icon="warning")
    elif len(ADDRESS.get())>=20:
        result = tkMessageBox.showwarning('', 'Address can MOre Than 20 Characters!!!',
icon="warning")
    elif(re.fullmatch(num,MOBILENO.get())== None):
        result = tkMessageBox.showwarning('', 'Mobile no can not be character!!!',
icon="warning")
    elif len(MOBILENO.get())!=10:
        result = tkMessageBox.showwarning('', 'Mobile Can Not Be Less Than 10!!!',
icon="warning")
    elif(re.fullmatch(num,PHONENO.get())== None and PHONENO.get()!=""):
        result = tkMessageBox.showwarning('', 'Phone no can not be character!!!', icon="warning")
    elif (len(PHONENO.get())!=7 and PHONENO.get()!=""):
        result = tkMessageBox.showwarning('', 'Phone Cant Not Be Less Than 7!!!',
icon="warning")
    #elif(fetch!=None):
    #    result = tkMessageBox.showwarning('', 'Mobile Number Already Exists!!!',
icon="warning")
    #elif(fetchphone!=None and  PHONENO.get()!=""):
    #   result = tkMessageBox.showwarning('', 'Phone Number Already Exists!!!',
icon="warning")
```

```
    else:
        tree.delete(*tree.get_children())
        conn = sqlite3.connect("Contact_management.db")
        cursor = conn.cursor()
        cursor.execute("UPDATE `member` SET `firstname` = ?, `lastname` = ?, `gender` =?, `age`
= ?,  `address` = ?, `mobile` = ?,`phoneno` = ? WHERE `mem_id` = ?", (str(FNAME.get()),
str(LNAME.get()), str(GENDER.get()), str(AGE.get()), str(ADDRESS.get()),
str(MOBILENO.get()), str(PHONENO.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()
        FNAME.set("")
        LNAME.set("")
        GENDER.set("")
        AGE.set("")
        ADDRESS.set("")
        MOBILENO.set("")
        PHONENO.set("")

def Update():
    if not tree.selection():
       result = tkMessageBox.showwarning('', 'Please Select Item to Update!', icon="warning")
    global mem_id, UpdateWindow
    curItem = tree.focus()
    contents =(tree.item(curItem))
    selecteditem = contents['values']
    mem_id = selecteditem[0]
    FNAME.set("")
    LNAME.set("")
    GENDER.set("")
    AGE.set("")
    ADDRESS.set("")
    MOBILENO.set("")
    PHONENO.set("")
    FNAME.set(selecteditem[1])
    LNAME.set(selecteditem[2])
```

```
    AGE.set(selecteditem[4])
    ADDRESS.set(selecteditem[5])
    MOBILENO.set(selecteditem[6])
    PHONENO.set(selecteditem[7])
    UpdateWindow = Toplevel()
    UpdateWindow.title("Contact List")
    width = 400
    height = 350
    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="Mobile NO *", font=('arial', 14), bd=5)
    lbl_contact.grid(row=5, sticky=W)
    lbl_phone = Label(ContactForm, text="Phone NO", font=('arial', 14), bd=5)
    lbl_phone.grid(row=6, sticky=W)

    #===================ENTRY=============================
    firstname = Entry(ContactForm, textvariable=FNAME, font=('arial', 14))
    firstname.grid(row=0, column=1)
    lastname = Entry(ContactForm, textvariable=LNAME, 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=MOBILENO,  font=('arial', 14))
    contact.grid(row=5, column=1)
    phone = Entry(ContactForm, textvariable=PHONENO,  font=('arial', 14))
    phone.grid(row=6, column=1)


    #===================BUTTONS=============================
    btn_updatecon = Button(ContactForm, text="Update", width=50, command=UpdateData)
    btn_updatecon.grid(row=7, columnspan=2, pady=10)


#fn1353p
def DeleteData():
    if not tree.selection():
       result = tkMessageBox.showwarning('', 'Please Select Item To Delete!', 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("Contact_management.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
   FNAME.set("")
   LNAME.set("")
   GENDER.set("")
   AGE.set("")
   ADDRESS.set("")
   MOBILENO.set("")
   PHONENO.set("")
   NewWindow = Toplevel()
   NewWindow.title("Contact List")
   width = 400
   height = 350
   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="Mobile NO *", font=('arial', 14), bd=5)
    lbl_contact.grid(row=5, sticky=W)
    lbl_phone = Label(ContactForm, text="Phone NO", font=('arial', 14), bd=5)
    lbl_phone.grid(row=6, sticky=W)

    #==================ENTRY=============================
    firstname = Entry(ContactForm, textvariable=FNAME, font=('arial', 14))
    firstname.grid(row=0, column=1)
   # firstname.insert(0,'enter first name')
   # firstname.bind("<FocusIn>", lambda args: firstname.delete('0', 'end'))
    lastname = Entry(ContactForm, textvariable=LNAME, font=('arial', 14))
    lastname.grid(row=1, column=1)
    #lastname.insert(0,'enter last name')
    #lastname.bind("<FocusIn>", lambda args: lastname.delete('0', 'end'))
    RadioGroup.grid(row=2, column=1)
    age = Entry(ContactForm, textvariable=AGE,  font=('arial', 14))
    age.grid(row=3, column=1)
    #age.insert(0,'enter age')
    #age.bind("<FocusIn>", lambda args: age.delete('0', 'end'))
    address = Entry(ContactForm, textvariable=ADDRESS,  font=('arial', 14))
    address.grid(row=4, column=1)
    #address.insert(0,'enter address')
    #address.bind("<FocusIn>", lambda args: address.delete('0', 'end'))
    contact = Entry(ContactForm, textvariable=MOBILENO,  font=('arial', 14))
    contact.grid(row=5, column=1)
    #contact.insert(0,'enter mobile number')
```

```
    #contact.bind("<FocusIn>", lambda args: contact.delete('0', 'end'))
    phone = Entry(ContactForm, textvariable=PHONENO,  font=('arial', 14))
    phone.grid(row=6, column=1)
    #phone.insert(0,'enter phone number')
    #phone.bind("<FocusIn>", lambda args: phone.delete('0', 'end'))



    #==================BUTTONS==============================
    btn_addcon = Button(ContactForm, text="Save", width=50, command=SubmitData)
    btn_addcon.grid(row=7, columnspan=2, pady=10)




#============================FRAMES==================================
=====
Top = Frame(root, width=500, bd=1, relief=SOLID)
Top.pack(side=TOP)
Mid = Frame(root, width=500,  bg="#a5afd9")
Mid.pack(side=TOP)
MidLeft = Frame(Mid, width=100,  bg="#a5afd9")
MidLeft.pack(side=LEFT, pady=10)
Cent =Frame(Mid, width=100,  bg="#a5afd9")
Cent.pack(side=LEFT, pady=10)
MidLeftPadding = Frame(Cent, width=180, bg="#b5afd9")
MidLeftPadding.pack(side=LEFT)
MidRightPadding = Frame(Cent, width=180, bg="#b5afd9")
MidRightPadding.pack(side=RIGHT)
MidRight = Frame(Mid, width=100,  bg="#a5afd9")
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),  bg="#22a181",
width=500)
lbl_title.pack(fill=X)
```

```
#============================ENTRY=================================
=====

#============================BUTTONS===============================
=====
btn_add = Button(MidLeft, text="+ ADD NEW", bg="#66ff66", command=AddNewWindow)
btn_add.pack()
btn_update = Button(Cent, text="UPDATE", bg="yellow", command= Update)
btn_update.pack()
btn_delete = Button(MidRight, text="DELETE", bg="red", command=DeleteData)
btn_delete.pack()

#============================TABLES================================
=====
scrollbarx = Scrollbar(TableMargin, orient=HORIZONTAL)
scrollbary = Scrollbar(TableMargin, orient=VERTICAL)
tree = ttk.Treeview(TableMargin, columns=("MemberID", "Firstname", "Lastname", "Gender",
"Age", "Address", "Contact","Phone"), 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="Mobile NO", anchor=W)
tree.heading('Phone', text="Phone NO", 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.column('#8', stretch=NO, minwidth=0, width=120)
```

```
tree.pack()

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

CONCLUSION

Contact Management system is effective software system which will benefit all the people for recording the contact number. It helps the process of add contact update contact as well as delete the contact if not needed. We can easily contact the people in the contact list with the help of this software and we don't need to remember the contact of people.