\*\* CONTENTS \*\*

Topic

Introduction……………………………………………………………… 3

Purpose…………………………………………………………. 4

Software and Hardware Requirement…………………………………… 5

Literature Survey………………………………………………………… 6

Technical feasibility……………………………………………. 6

Operational feasibility………………………………………….. 6

Economical feasibility………………………………………….. 7

Code Optimization………………………………………………………. 8

Output Screen……………………………………………………………. 32

Conclusion………………………………………………………………. 37

Future Enhancement…………………………………………………….. 38

APPENDICES…………………………………………………………... 39

Reference………………………………………………………………... 40

Bibliography…………………………………………………………….. 41

Project Title:

PHONEBOOK

Technology Used :PYTHON(SQLITE,TKINTER)

Language :PYTHON language

Model Type :Individual

College :LNCT

Report Format

Submitted Department: T&P

Submitted To : Rambabu Lovewanshi

Name of student :SRIJAN MISHRA

Branch : EX

Semester : 5th

Enrollment No. : 0103EX181098

Submission Date : 31-DEC-2020

INTRODUCTION

PHONEBOOKis a console application. As it is a system that has been designed to collect data in the form of tables where data is the user’s contact numbers that is to be collected. The objective of this project is to establish a system where multiple contact details can be stored with the features of manipulations of data of the given users. In this project we will make an easier task ofcollecting and managing mobiles details of phone no. of multiple users.

As we know there was a time when mobile data was stored manually in a handbook or a diary written by the users with a pen known as phone book but here we are trying to merge that old tradition with technological driven world. Its obvious that managing the hard copies is always a tough task than soft copies. Papers, books ,hardcopies of anything can be moulded , destroyed , misplaced or easily weathered, but data stored in an app is forever until and unless you have not disturbed the data.

This application has been developed to provide best phone data management services to customers. This phonebookhas been developed to provide a management platform where a customer can store multiple contact details and manage them as per their convenience. The data can easily be manipulated , can be added , deleted, edited and all other things that’s required in an phonebook is available in here.

1.1 PURPOSE

Data collection&management

We can add a contact no. by entering the add button.

User can delete the no. which is no more required.

User can edit the stored contact details by clicking edit button. And many more .

EASY INTERACTION WITH INFORMATIVE CONSOLE

Good range of options available. With this system you can:

With this we have made easy data collection of the customers and manipulation of the data easy for future use.

If something went wrong, we can edit the data instead of deleting which will result in Relief from Tedious work of entering details every-time.

Notifies and checks for permission before deleting the contact already present.

Can refresh data on spot after any changes made to existing data.

Software and Hardware requirements

SOFTWARE REQUIREMENT SPECIFICATION

Operating System :Windows 10.1

Programming Language : PYTHON

User Interface : Console Application

Database :SQLITE

HARDWARE REQUIREMENTS SPECIFICATION

Processor : Pentium IV processor or higher

RAM : Minimum of 1GB RAM

Memory :500 MB or higher

Literature survey /Review of Literature

INTRODUCTION

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it’s worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

3.1 TECHNICAL FEASIBILITY

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis. Understand the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies. Is the required technology available with the organization ?.

3.2 OPERATIONAL FEASIBILITY

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.

Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.

Have the user been involved in the planning and development of the project?

Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

3.3 ECONOMIC FEASIBILITY

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

CODE SNIPPETS FOR TRAINING

1)ROOT CODE :-

from tkinter import \*

import datetime

from mypeople import Mypeople

from addpeople import AddPeople

from aboutus import About

date = datetime.datetime.now().date()

date = str(date)

class Application(object):

def \_init\_(self,master):

self.master = master

self.top = Frame(master, height= 150, bg= 'yellow')

self.top.pack(fill=X)

self.bottom = Frame(master, height= 500, bg= '#14075c')

self.bottom.pack(fill=X)

self.heading = Label(self.top, text='My phonebook', font='arial 15 bold', bg='yellow', fg='#5c0e10')

self.heading.place(x=260, y=50)

self.date\_lbl = Label(self.top, text="Today's date:"+date, font='arial 9 bold', fg = 'black', bg='yellow')

self.date\_lbl.place(x=495, y=110)

self.viewButton = Button(self.bottom, text= ' My people ', font= 'arial 12 bold', bg='#2c4982', command=self.my\_people)

self.viewButton.place(x=250, y=70)

self.addButton = Button(self.bottom, text='Add people', font='arial 12 bold', bg='#2c4982', command=self.direct\_AD)

self.addButton.place(x=250, y=180)

self.aboutButton = Button(self.bottom, text=' About us ', font='arial 12 bold', bg='#2c4982', command=self.about\_us)

self.aboutButton.place(x=250, y=290)

def my\_people(self):

people = Mypeople()

def about\_us(self):

about\_page = About()

def direct\_AD(self):

direct = AddPeople()

def main():

root = Tk()

app = Application(root)

root.title("Phonebook app")

root.geometry("650x550+350+200")

root.resizable(False, False)

root.mainloop()

if \_name\_ == '\_main\_':

main()

2.) my people:-

from tkinter import \*

import sqlite3

from addpeople import AddPeople

from updatepeople import Update

from display import Display

from tkinter import messagebox

con = sqlite3.connect('database.db')

cur = con.cursor()

class Mypeople(Toplevel):

def \_init\_(self):

Toplevel.\_init\_(self)

self.geometry("650x650+600+200")

self.title("My People")

self.resizable(FALSE, FALSE)

self.top = Frame(self, height=150, bg='red')

self.top.pack(fill=X)

self.bottom = Frame(self, height=500, bg='orange')

self.bottom.pack(fill=X)

self.heading = Label(self.top, text='My people', font='arial 15 bold', bg='red', fg='orange')

self.heading.place(x=260, y=50)

self.scroll = Scrollbar(self.bottom, orient=VERTICAL)

self.listbox = Listbox(self.bottom, width=50, height=35)

self.listbox.grid(row=0, column=0, padx=(40,0))

self.scroll.config(command=self.listbox.yview)

self.listbox.config(yscrollcommand=self.scroll.set)

count = 0

persons = cur.execute("select \* from 'address book'").fetchall()

print(persons)

for person in persons:

self.listbox.insert(count, str(person[0])+ "."+person[1]+ " " +person[2])

count += 1

self.scroll.grid(row=1, column=1, sticky=N+S)

btnadd = Button(self.bottom, text="Add", width="12", font="Sans 12 bold", command=self.add\_people)

btnadd.grid(row=0, column=2, padx=20, pady=10, sticky=N)

btnupdate = Button(self.bottom, text="Update", width="12", font="Sans 12 bold", command=self.update\_function)

btnupdate.grid(row=0, column=2, padx=20, pady=50, sticky=N)

btndisplay = Button(self.bottom, text="Display", width="12", font="Sans 12 bold", command=self.display\_people)

btndisplay.grid(row=0, column=2, padx=20, pady=90, sticky=N)

btndelete = Button(self.bottom, text="Delete", width="12", font="Sans 12 bold", command= self.delete\_person)

btndelete.grid(row=0, column=2, padx=20, pady=130, sticky=N)

def delete\_person(self):

selected\_item = self.listbox.curselection()

person = self.listbox.get(selected\_item)

person\_id = person.split(".")[0]

query = "delete from 'address book' where person\_id = {}".format(person\_id)

answer = messagebox.askquestion("warning", "are you sure you wanna delete")

if answer == 'yes':

try:

cur.execute(query)

con.commit()

messagebox.showinfo("success", "Deleted")

self.destroy()

except Exception as e:

messagebox.showinfo("Info", str(e))

def add\_people(self):

add = AddPeople()

self.destroy()

def update\_function(self):

selected\_item = self.listbox.curselection()

person = self.listbox.get(selected\_item)

person\_id = person.split(".")[0]

updatepage = Update(person\_id)

def display\_people(self):

selected\_item = self.listbox.curselection()

person = self.listbox.get(selected\_item)

person\_id = person.split(".")[0]

displaypage = Display(person\_id)

3.) UPDATE:-

from tkinter import \*

import sqlite3

from tkinter import messagebox

con = sqlite3.connect('database.db')

cur = con.cursor()

class Update(Toplevel):

def \_init\_(self, person\_id):

Toplevel.\_init\_(self)

self.geometry("650x650+600+200")

self.title("Update person")

self.resizable(False, False)

print("person id =", person\_id)

query = "select \* from 'address book' where person\_id = '{}'".format(person\_id)

result = cur.execute(query).fetchone()

print(result)

self.person\_id = person\_id

person\_name = result[1]

person\_surname = result[2]

person\_email = result[5]

person\_phine = result[3]

person\_address = result[4]

print("person name", person\_name)

print("person surname", person\_surname)

print("person email", person\_email)

print("person phone", person\_phine)

print("person address", person\_address)

self.top = Frame(self, height=150, bg='white')

self.top.pack(fill=X)

self.bottom = Frame(self, height=500, bg='grey')

self.bottom.pack(fill=X)

self.heading = Label(self.top, text='Update people', font='arial 15 bold', bg='white', fg='grey')

self.heading.place(x=260, y=50)

self.label\_name = Label(self.bottom, text="Name", font='arial 15 bold', fg="white", bg="grey")

self.label\_name.place(x=49, y=40)

self.entry\_name = Entry(self.bottom, width=30, bd=4)

self.entry\_name.insert(0, person\_name)

self.entry\_name.place(x=150, y=40)

self.label\_surname = Label(self.bottom, text="Surname", font='arial 15 bold', fg="white", bg="grey")

self.label\_surname.place(x=49, y=100)

self.entry\_surname = Entry(self.bottom, width=30, bd=4)

self.entry\_surname.insert(0, person\_surname)

self.entry\_surname.place(x=150, y=100)

self.label\_email = Label(self.bottom, text="Email", font='arial 15 bold', fg="white", bg="grey")

self.label\_email.place(x=49, y=160)

self.entry\_email = Entry(self.bottom, width=30, bd=4)

self.entry\_email.insert(0, person\_email)

self.entry\_email.place(x=150, y=160)

self.label\_Phone\_number = Label(self.bottom, text="Phone no.", font='arial 15 bold', fg="white", bg="grey")

self.label\_Phone\_number.place(x=49, y=220)

self.entry\_Phone\_number = Entry(self.bottom, width=30, bd=4)

self.entry\_Phone\_number.insert(0, person\_phine)

self.entry\_Phone\_number.place(x=150, y=220)

self.label\_Address = Label(self.bottom, text="Address", font='arial 15 bold', fg="white", bg="grey")

self.label\_Address.place(x=49, y=280)

self.entry\_Address = Text(self.bottom, width=50, height=6)

self.entry\_Address.insert(1.0, person\_address)

self.entry\_Address.place(x=150, y=280)

button = Button(self.bottom, text="Update Person", font="arial 15 bold", fg="white", bg="black",

command=self.update\_people)

button.place(x=270, y=420)

def update\_people(self):

id = self.person\_id

name = self.entry\_name.get()

surname = self.entry\_surname.get()

email = self.entry\_email.get()

phone = self.entry\_Phone\_number.get()

address = self.entry\_Address.get(1.0, 'end-1c')

query = "update 'address book' set person\_name = '{}', person\_surname = '{}', person\_email = '{}', person\_phine = {}, person\_address = '{}' where person\_id = {}".format(name, surname, email, phone, address, id)

try:

cur.execute(query)

con.commit()

messagebox.showinfo("Success", "contact updated")

except Exception as e:

print(e)

4.) to DISPLAY:-

from tkinter import \*

import sqlite3

from tkinter import messagebox

con = sqlite3.connect('database.db')

cur = con.cursor()

class Display(Toplevel):

def \_init\_(self, person\_id):

Toplevel.\_init\_(self)

self.geometry("650x650+600+200")

self.title("Display person")

self.resizable(False, False)

print("person id =", person\_id)

query = "select \* from 'address book' where person\_id = '{}'".format(person\_id)

result = cur.execute(query).fetchone()

print(result)

self.person\_id = person\_id

person\_name = result[1]

person\_surname = result[2]

person\_email = result[5]

person\_phine = result[3]

person\_address = result[4]

print("person name", person\_name)

print("person surname", person\_surname)

print("person email", person\_email)

print("person phone", person\_phine)

print("person address", person\_address)

self.top = Frame(self, height=150, bg='white')

self.top.pack(fill=X)

self.bottom = Frame(self, height=500, bg='grey')

self.bottom.pack(fill=X)

self.heading = Label(self.top, text='People details', font='arial 15 bold', bg='white', fg='grey')

self.heading.place(x=260, y=50)

self.label\_name = Label(self.bottom, text="Name", font='arial 15 bold', fg="white", bg="grey")

self.label\_name.place(x=49, y=40)

self.entry\_name = Entry(self.bottom, width=30, bd=4)

self.entry\_name.insert(0, person\_name)

self.entry\_name.config(state="disable")

self.entry\_name.place(x=150, y=40)

self.label\_surname = Label(self.bottom, text="Surname", font='arial 15 bold', fg="white", bg="grey")

self.label\_surname.place(x=49, y=100)

self.entry\_surname = Entry(self.bottom, width=30, bd=4)

self.entry\_surname.insert(0, person\_surname)

self.entry\_surname.config(state="disable")

self.entry\_surname.place(x=150, y=100)

self.label\_email = Label(self.bottom, text="Email", font='arial 15 bold', fg="white", bg="grey")

self.label\_email.place(x=49, y=160)

self.entry\_email = Entry(self.bottom, width=30, bd=4)

self.entry\_email.insert(0, person\_email)

self.entry\_email.config(state="disable")

self.entry\_email.place(x=150, y=160)

self.label\_Phone\_number = Label(self.bottom, text="Phone no.", font='arial 15 bold', fg="white", bg="grey")

self.label\_Phone\_number.place(x=49, y=220)

self.entry\_Phone\_number = Entry(self.bottom, width=30, bd=4)

self.entry\_Phone\_number.insert(0, person\_phine)

self.entry\_Phone\_number.config(state="disable")

self.entry\_Phone\_number.place(x=150, y=220)

self.label\_Address = Label(self.bottom, text="Address", font='arial 15 bold', fg="white", bg="grey")

self.label\_Address.place(x=49, y=280)

self.entry\_Address = Text(self.bottom, width=50, height=6)

self.entry\_Address.insert(1.0, person\_address)

self.entry\_Address.config(state="disable")

self.entry\_Address.place(x=150, y=280)

5.) ADD:-

from tkinter import \*

import sqlite3

from tkinter import messagebox

con = sqlite3.connect('database.db')

cur = con.cursor()

class AddPeople(Toplevel):

def \_init\_(self):

Toplevel.\_init\_(self)

self.geometry("650x650+600+200")

self.title("Add People")

self.resizable(FALSE, FALSE)

self.top = Frame(self, height=150, bg='white')

self.top.pack(fill=X)

self.bottom = Frame(self, height=500, bg='grey')

self.bottom.pack(fill=X)

self.heading = Label(self.top, text='Add new people', font='arial 15 bold', bg='white', fg='grey')

self.heading.place(x=260, y=50)

self.label\_name = Label(self.bottom, text="Name", font='arial 15 bold', fg="white", bg="grey")

self.label\_name.place(x=49, y=40)

self.entry\_name = Entry(self.bottom, width=30, bd=4)

self.entry\_name.insert(0, "Enter Name")

self.entry\_name.place(x=150, y=40)

self.label\_surname = Label(self.bottom, text="Surname", font='arial 15 bold', fg="white", bg="grey")

self.label\_surname.place(x=49, y=100)

self.entry\_surname = Entry(self.bottom, width=30, bd=4)

self.entry\_surname.insert(0, "Enter Surname")

self.entry\_surname.place(x=150, y=100)

self.label\_email = Label(self.bottom, text="Email", font='arial 15 bold', fg="white", bg="grey")

self.label\_email.place(x=49, y=160)

self.entry\_email = Entry(self.bottom, width=30, bd=4)

self.entry\_email.insert(0, "Enter Email")

self.entry\_email.place(x=150, y=160)

self.label\_Phone\_number = Label(self.bottom, text="Phone no.", font='arial 15 bold', fg="white", bg="grey")

self.label\_Phone\_number.place(x=49, y=220)

self.entry\_Phone\_number = Entry(self.bottom, width=30, bd=4)

self.entry\_Phone\_number.insert(0, "Enter Phone Number")

self.entry\_Phone\_number.place(x=150, y=220)

self.label\_Address = Label(self.bottom, text="Address", font='arial 15 bold', fg="white", bg="grey")

self.label\_Address.place(x=49, y=280)

self.entry\_Address = Text(self.bottom, width=50, height=6)

self.entry\_Address.place(x=150, y=280)

button = Button(self.bottom, text="Add Person", font="arial 15 bold", fg="white", bg="black", command=self.add\_people)

button.place(x=270, y=420)

def add\_people(self):

name = self.entry\_name.get()

surname = self.entry\_surname.get()

email = self.entry\_email.get()

phone = self.entry\_Phone\_number.get()

address = self.entry\_Address.get(1.0, 'end-1c')

if name and surname and email and phone and address != "":

try:

query ="insert into 'address book' (person\_name, person\_surname, person\_email, person\_phine, person\_address)values(?,?,?,?,?)"

cur.execute(query, (name, surname, email, phone, address))

con.commit()

messagebox.showinfo("success", "contact added")

except Exception as e:

messagebox.showerror("Error", str(e))

else:

messagebox.showerror("Error", "fill all the fields", icon='warning')

output screen

The following pictures will show the series of output screens and the actual console screen ad working of phonebook takes place :-

>>The first page of phonebooklooks as shown in figure 1. The cover page provides an interface to the user such that the user can access any feature present in console application. The page is shown below :-

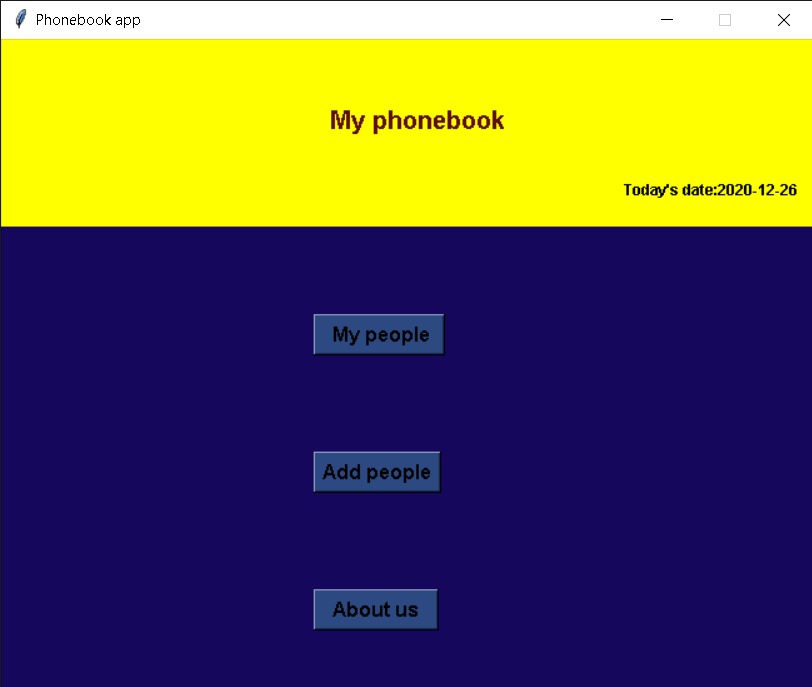


Figure 8.1

>>The second console screen shows the add user option where the user can add the numerous contact details. The login page figure 8.2 is shown below :-

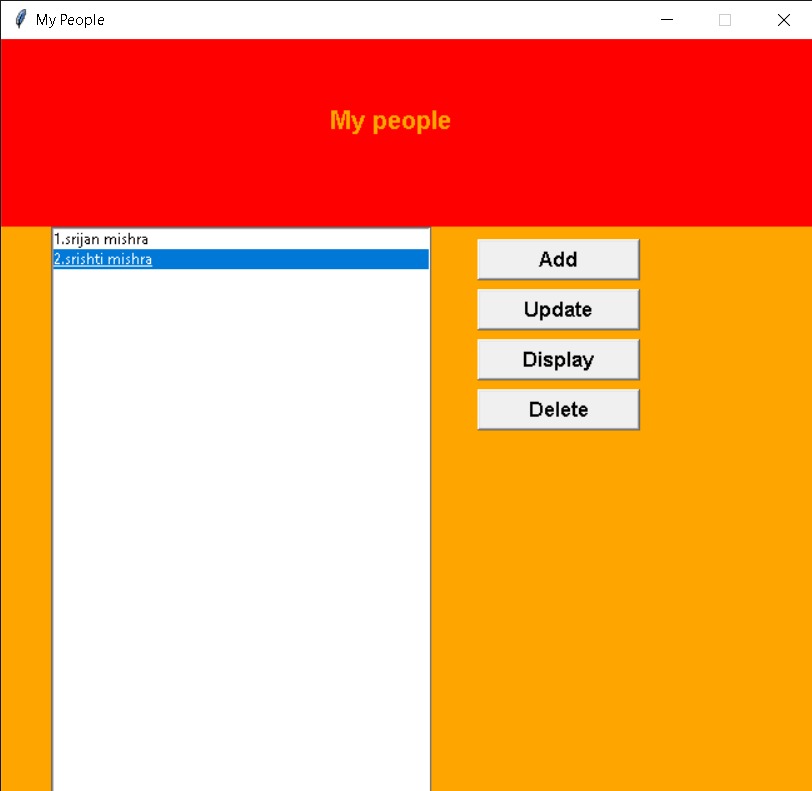


Figure 8.2

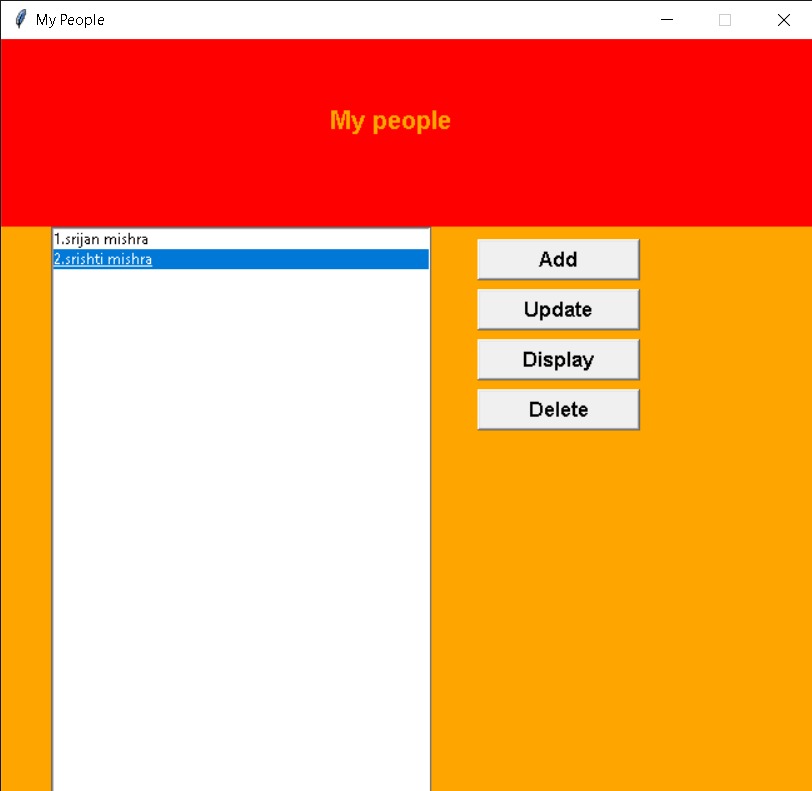
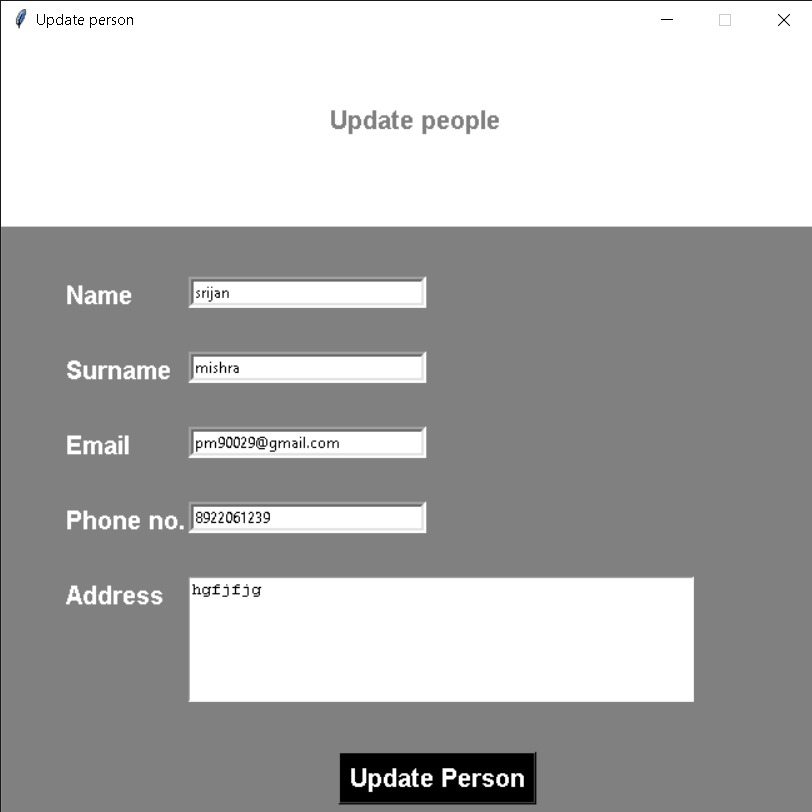
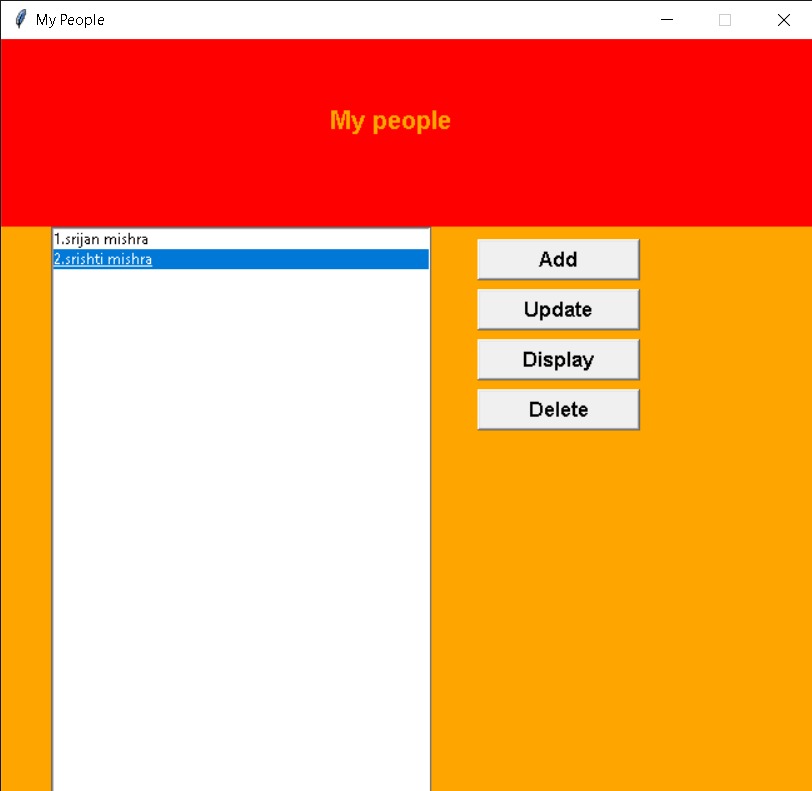
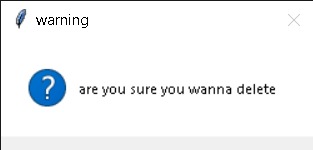
>>The third screen takes the user to a list of options available to the user. The page is shown in figure 8.3 below :-

Figure 8.3

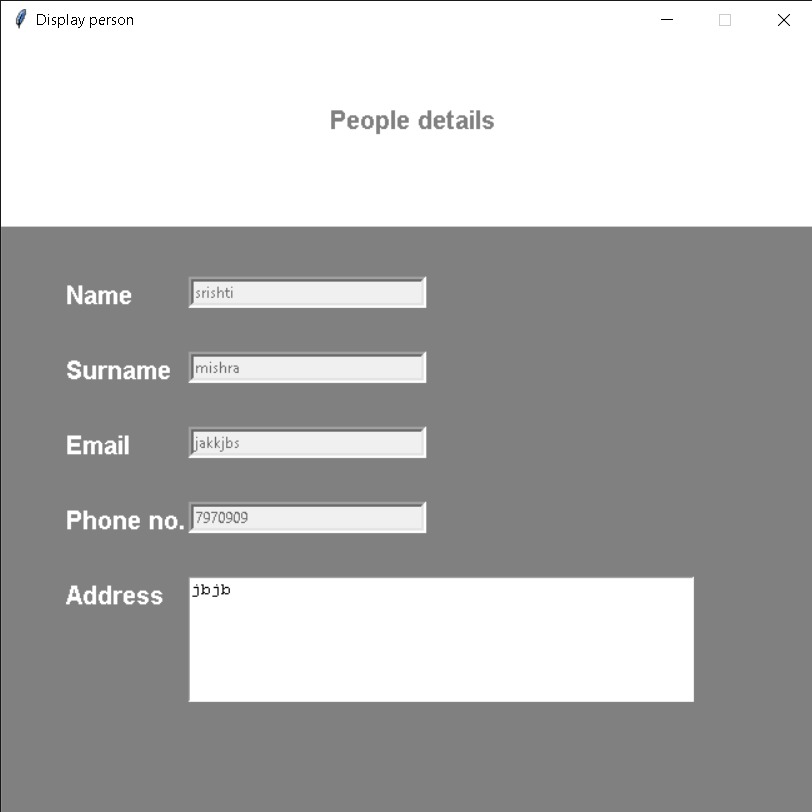
>> After the list the user can update and make changes if any in the existing lists. The phonebook console page is shown below :



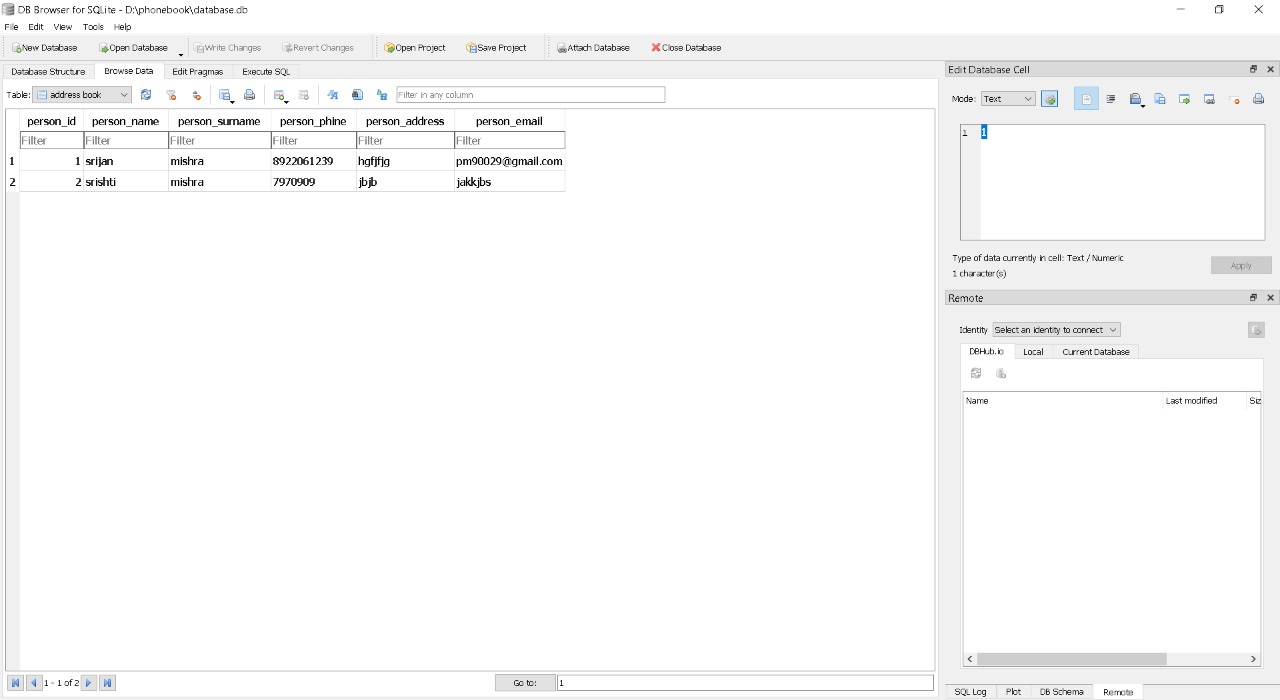
>>After the list is updated if the user want to delete any of the contact then we can delete it by clicking delete button and an confirmation box appears to confirm the decision of deleting the contact . The console for cancel is shown below :-



>>The user can view the details of the existing contacts by display buttonas shown below :-



Database picture:-



CONCLUSION

In this rapidly growing technical and digital era, maintaining a hand written notebook or manual with paper pen is gone and computerisation is done at all level. So this will perform all the work in computerized way by in this console application. By this application we can consume both time and tedious paper work of user as well as the data stored is managed easily and is safe for years. It has become more compatible because it tackles misplacement of user’s data and it has an informative console which makes manipulationof data easy for user. At last this application has every feature which makes your phone no. collection easy in every way.

FUTURE ENHANCEMENTS

Development is the most essential part of any project because it includes latest feature in the system or application. It reduces bugs and problems and creates a strong and everlasting relation with user according to their feedbacks. I will integrate some features in my phonebookand these are explained below:-

Displaying contacts profile picture for better looks and recognition.

Adding features like call outgoing and incoming.

User can add contact details to favourites and many more .

APPENDICES

Appendix A: Glossary

TERMS All the terms and abbreviations in the project are specified clearly. For further development of project evolved definitions will be specified.

ACRONYMS NO.:number of the contact list.

Appendix B: Analysis Models

This includes all the pertinent models, such as data flow diagrams and console screen outputs.

9. REFERENCES

Under this reference section, I have mentioned reference from which the data has been gathered that helped me to design this application for phonebook . These reference includes :-

We have taken only reference from a Youtubechannel :- https://www.youtube.com/c/TechGramAcademy

10. BIBLIOGRAPHY

Some other sites and papers from where the data has been gathered is listed below:-

The features to be implemented in project is taken from the youtubechannel:- https://www.youtube.com/c/TechGramAcademy

The introduction and purposes have been inspired from - <https://sites.google.com/>