CONTACT BOOK

CLASS ASSESMENT 2 REPORT

by

PRANSHU YADAV 11913078
RAVI SINGH 11913094

Section: K19PG

RollNumbers: 44 , 46

Department of Intelligent Systems,
School of Computer Science Engineering,
Lovely Professional University, Jalandhar

November, 2020

Student Declaration

This is to declare that this report has been written by us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. we aver that if any part of the report is found to be copied, we shall take full responsibility for it.

Signature:Pranshu yadav
Ravi singh
Name: Pranshu yadav
Ravi singh
Roll Number: 44
46

Place: LPU, Jalandhar
Date: 30th October 2020

TABLE OF CONTENTS

BONAFIDE CERTIFICATE

Certified that this project report “CONTACT BOOK ” is the bonafide work of “PRANSHU YADAV and RAVI SINGH” who carried out the project work under my supervision.

DR.DHANPRATAP SINGH

DR.DHANPRATAP SINGH

ASSOCIATE PROFESSOR

25706

INTELLIGENCE DEPARTMENT

OBJECTIVE:

The main objective of this project/contact book is to store contacts. Each contact enteries usually contains few standard fields i.e first name , last name ,company name ,email ,address, phone number ,fax number , telephone number.

INTRODUCTION:

WHAT IS A CONTACT BOOK:

An Contact book or an address book is a book or a database used for storing entries called contacts. Each contact entry usually consists of a few standard fields (for example: first name, last name, company name, address, telephone number, e-mail address, fax number, mobile number). Most such systems store the details in alphabetical order of people's names, although in paper-based address books entries can easily end up out of order as the owner inserts details of more individuals or as people move.
Address books allow easy access to the user's friends, family, business associates and others by maintaining their email and other contact details on their computer. Address books can be software based, or accessed online or through a network. Users may also be able to export contacts from their address books to mobile phones, PDAs and other portable electronic devices.

Network address books can be managed through a singular interface in order to combine a user's entire network. These can include contacts for social networks, emails, mobile phones and PDAs.

Some people believe that future network address books will have capabilities that supersede social networks when they expand to reveal the real-time activities of address book contacts, including current social network posts, blog posts and other electronic activities. For example, Facebook has friend suggestion applications, where users can suggest that their contacts become friends with other friends; this can be thought of as a type of shared address book.

A contact book can be pen and paper based old fashioned contact book ro it can be a digital contact book that are currently used in mobile phones for storing of contacts as per accordance of the user.

TASKS THAT CAN BE PERFORMED :

1. Addition of contacts
2. Deletion of contacts
3. Updation of contacts
4. Searching of contacts
5. Copying of contacts
6. Moving of contacts

MODULES THAT ARE USED IN THIS PROJECT :

Tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task. Import the module then make a root window on which we display the content with the help of various widgets provide by this module.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Sql lite:

SQLite3 is a very easy to use database engine. It is self-contained, serverless, zero-configuration and transactional. It is very fast and lightweight, and the entire database is stored in a single disk file. It is used in a lot of applications as internal data storage. The Python Standard Library includes a module called "sqlite3" intended for working with this database. This module is a SQL interface compliant with the DB-API 2.0 specification.

Using Python's SQLite Module
To use the SQLite3 module we need to add an import statement to our python script:

Connecting SQLite to the Database
We use the function sqlite3.connect to connect to the database. We can use the argument ":memory:" to create a temporary DB in the RAM or pass the name of a file to open or create it.

When we are done working with the DB we need to close the connection.

MessageBox Widget:

MessageBox Widget is used to display the message boxes in the python applications. This module is used to display a message using provides a number of functions.

FUNCTIONS USED :

1: create_database

2: put_data.

3: new_contact

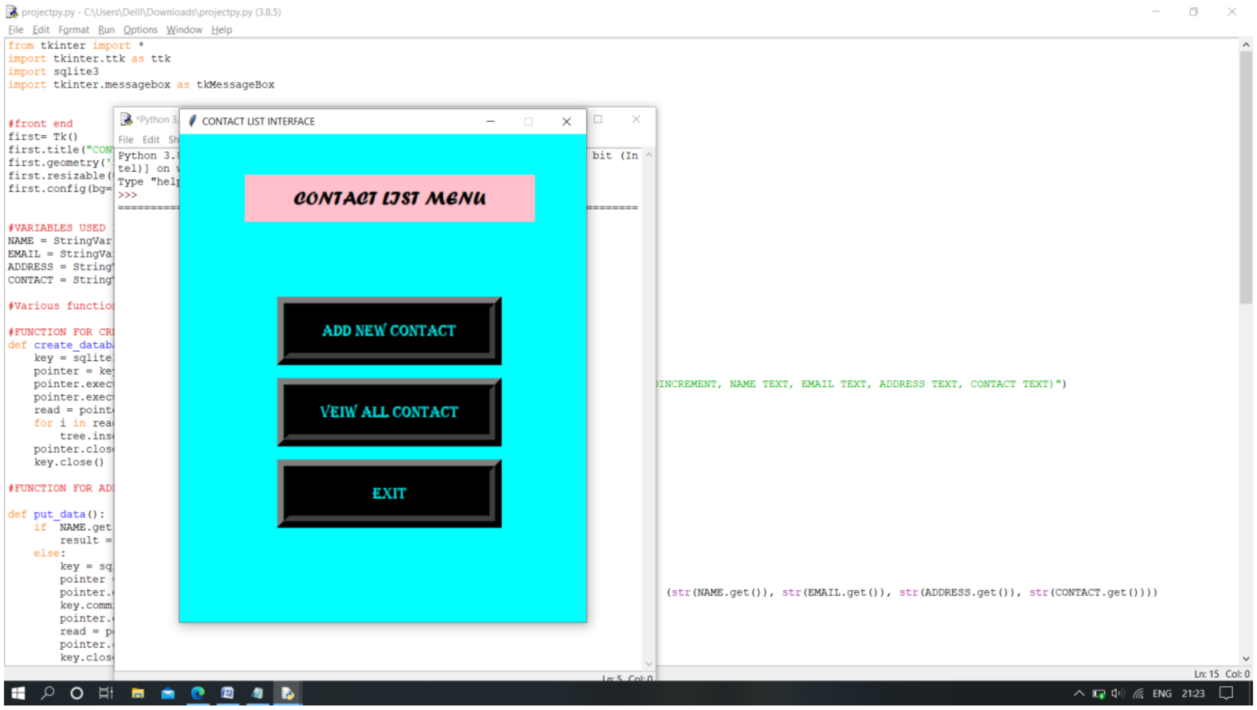
SOURCE CODE :

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



#front end
first= Tk()
```

```
first.title("CONTACT LIST INTERFACE")
first.geometry('500x600')
first.resizable(0,0)
first.config(bg="#00ffff")



#VARIABLES USED IN THE CODE
NAME = StringVar()
EMAIL = StringVar()
ADDRESS = StringVar()
CONTACT = StringVar()

#Various functions used

#FUNCTION FOR CREATING THE DATABASE
def create_database():
    key = sqlite3.connect("Ravi.db")
    pointer = key.cursor()
    pointer.execute("CREATE TABLE IF NOT EXISTS 'contact' (mem_id INTEGER NOT NULL
    PRIMARY KEY AUTOINCREMENT, NAME TEXT, EMAIL TEXT, ADDRESS TEXT,
    CONTACT TEXT)")
    pointer.execute("SELECT * FROM 'contact' ORDER BY 'NAME' ASC")
    read = pointer.fetchall()
    for i in read:
        tree.insert('', 'end', values=(i))
    pointer.close()
    key.close()

#FUNCTION FOR ADDING NEW DATA TO THE DATABASE


def put_data():
    if  NAME.get() == "" or EMAIL.get() == "" or ADDRESS.get() == "" or CONTACT.get() == "":
        result = tkMessageBox.showwarning('', 'Fill Up The Required Field', icon="warning")
    else:
        key = sqlite3.connect("Ravi.db")
        pointer = key.cursor()
        pointer.execute("INSERT INTO 'contact' (NAME, EMAIL, ADDRESS, CONTACT )
    VALUES(?, ?, ?, ?)", (str(NAME.get()), str(EMAIL.get()), str(ADDRESS.get()),
    str(CONTACT.get())))
        key.commit()
        pointer.execute("SELECT * FROM 'contact' ORDER BY 'NAME' ASC")
        read = pointer.fetchall()
        pointer.close()
        key.close()
        NAME.set("")
        EMAIL.set("")
        ADDRESS.set("")
        CONTACT.set("")



#FUNCTION FOR CREATING NEW CONTACT WINDOW
def new_contact():
    global add_mem
    NAME.set("")
    EMAIL.set("")
    ADDRESS.set("")
    CONTACT.set("")
    add_mem = Toplevel()
    add_mem.title("New Contact")
    add_mem.resizable(0, 0)
    add_mem.geometry('400x340')



    #FRAMES USED FOR NEW CONTACT WINDOW
    FormTitle = Frame(add_mem)
    FormTitle.pack(side=TOP)
    ContactForm = Frame(add_mem)
    ContactForm.pack(side=TOP, pady=20)

    #LABELS USED FOR NEW CONTACT WINDOW
    heading = Label(FormTitle, text="ENTER THE DETAILS", font=('arial', 16), bg="cyan", bd=15,
    width = 280)
    heading.pack(fill=X)
    new_name = Label(ContactForm, text="NAME", font=('arial', 14), bd=10)
    new_name.grid(row=0, sticky=W)
    new_email = Label(ContactForm, text="E-MAIL", font=('arial', 14), bd=10)
    new_email.grid(row=1, sticky=W)
    new_address = Label(ContactForm, text="ADDRESS", font=('arial', 14), bd=10)
    new_address.grid(row=2, sticky=W)
    new_contact = Label(ContactForm, text="CONTACT", font=('arial', 14), bd=10)
    new_contact.grid(row=3, sticky=W)

    #ENTRY USED FOR NEW CONTACT FORM
    in_name = Entry(ContactForm, textvariable=NAME, font=('arial', 14))
    in_name.grid(row=0, column=1)
    in_email = Entry(ContactForm, textvariable=EMAIL, font=('arial', 14))
    in_email.grid(row=1, column=1)
    in_address = Entry(ContactForm, textvariable=ADDRESS,  font=('arial', 14))
    in_address.grid(row=2, column=1)
    in_contact = Entry(ContactForm, textvariable=CONTACT,  font=('arial', 14))
    in_contact.grid(row=3, column=1)



    #BUTTON USED IN THE NEW CONTACT WINDOW
    finalize = Button(ContactForm, text="DONE",bd=10,bg='brown', width=50, command=put_data)
    finalize.grid(row=4, columnspan=2, pady=10)




#heading title
heading= Label(first, text="CONTACT LIST MENU", bd= '15', font=('forte',18),
               bg='pink',height=1,
               width=25).place(x=80,y=50)
#buttons
b1 = Button(first,bg='black',fg='cyan', font=('algerian',14),
            text = 'ADD NEW CONTACT',bd = '15', command = new_contact,
            height=2,width=20).place(x=120,y=200)
```

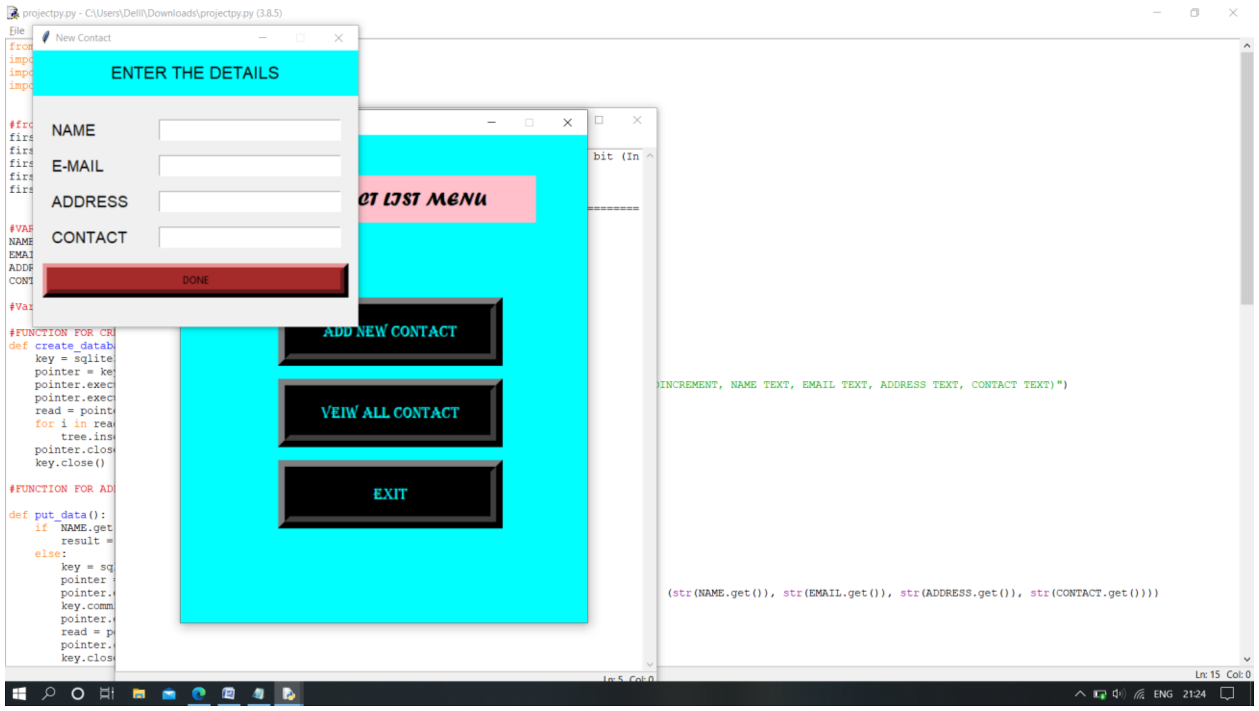
```
b2 = Button(first,bg='black',fg='cyan', font=('algerian',14),
        text = 'VEIW ALL CONTACT',bd = '15', command= NONE,
        height=2,width=20).place(x=120,y=300)
b3 = Button(first,bg='black',fg='cyan', font=('algerian',14),
        text = 'EXIT',bd = '15', command= first.destroy,
        height=2,width=20).place(x=120,y=400)
  first.mainloop()
```

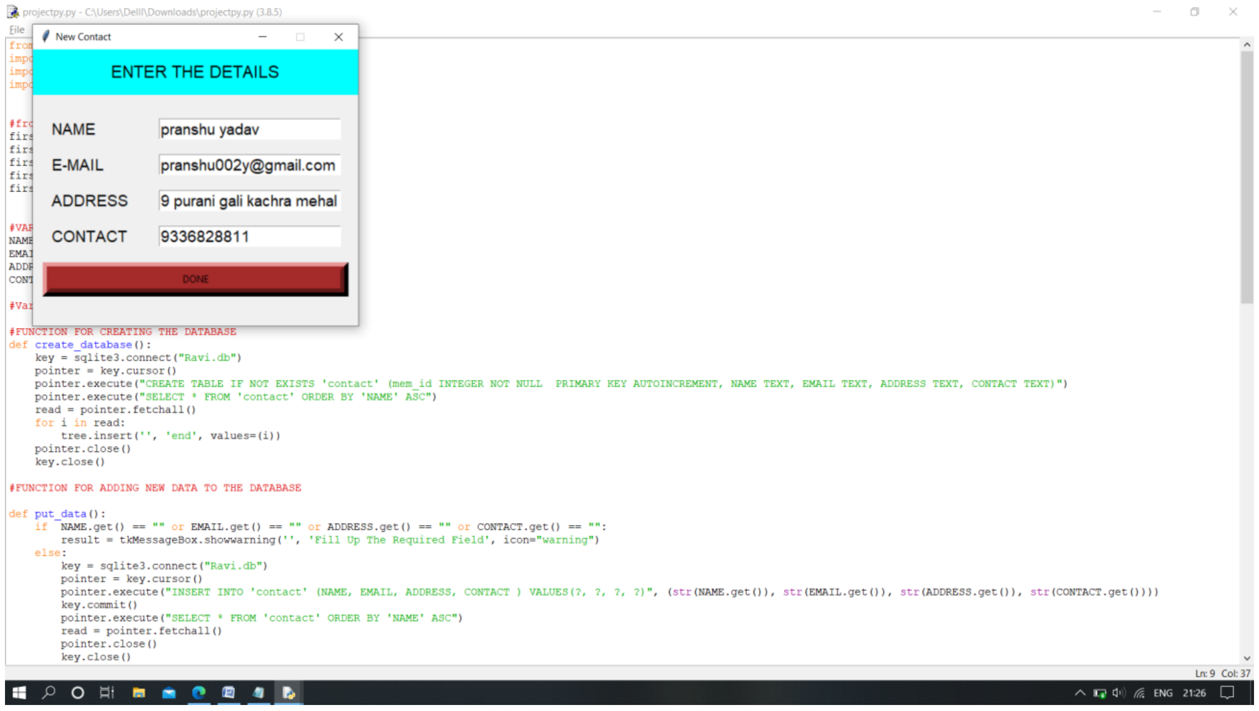
SCREENSHOTS:



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


#front end
first= Tk()
first.title("CON
first.geometry('
first.resizable(
first.config(bg=


#VARIABLES USED
NAME = StringVar
EMAIL = StringVa
ADDRESS = String
CONTACT = String

#Various functio

#FUNCTION FOR CR
def create_datab
    key = sqlite
    pointer = ke
    pointer.exec                                      )INCREMENT, NAME TEXT, EMAIL TEXT, ADDRESS TEXT, CONTACT TEXT)")
    pointer.exec
    read = point
    for i in rea
        tree.ins
    pointer.clos
    key.close()

#FUNCTION FOR AD

def put_data():
    if  NAME.get
        result =
    else:
        key = sq
        pointer
        pointer.                                       (str(NAME.get()), str(EMAIL.get()), str(ADDRESS.get()), str(CONTACT.get())))
        key.comm
        pointer.
        read = p
        pointer.
        key.clos
```

```
Python 3.                bit (In
tel)] on
Type "hel
>>>
=========                =========
```


projectpy.py - C:\Users\Delll\Downloads\projectpy.py (3.8.5)


projectpy.py - C:\Users\Delll\Downloads\projectpy.py (3.8.5)

NAME	pranshu yadav
E-MAIL	pranshu002y@gmail.com
ADDRESS	9 purani gali kachra mehal
CONTACT	9336828811

```
#FUNCTION FOR CREATING THE DATABASE
def create_database():
    key = sqlite3.connect("Ravi.db")
    pointer = key.cursor()
    pointer.execute("CREATE TABLE IF NOT EXISTS 'contact' (mem_id INTEGER NOT NULL  PRIMARY KEY AUTOINCREMENT, NAME TEXT, EMAIL TEXT, ADDRESS TEXT, CONTACT TEXT)")
    pointer.execute("SELECT * FROM 'contact' ORDER BY 'NAME' ASC")
    read = pointer.fetchall()
    for i in read:
        tree.insert('', 'end', values=(i))
    pointer.close()
    key.close()

#FUNCTION FOR ADDING NEW DATA TO THE DATABASE

def put_data():
    if  NAME.get() == "" or EMAIL.get() == "" or ADDRESS.get() == "" or CONTACT.get() == "":
        result = tkMessageBox.showwarning('', 'Fill Up The Required Field', icon="warning")
    else:
        key = sqlite3.connect("Ravi.db")
        pointer = key.cursor()
        pointer.execute("INSERT INTO 'contact' (NAME, EMAIL, ADDRESS, CONTACT ) VALUES(?, ?, ?, ?)", (str(NAME.get()), str(EMAIL.get()), str(ADDRESS.get()), str(CONTACT.get())))
        key.commit()
        pointer.execute("SELECT * FROM 'contact' ORDER BY 'NAME' ASC")
        read = pointer.fetchall()
        pointer.close()
        key.close()
```

PRANSHU YADAV 11913078, (44)

9336828811

RAVI SINGH 11913094 ,(46)

6202480403