REPORT OF CONTACTS DATABASE APP

As a project work for course

PYTHON PROGRAMMING (INT 213)

Name : Prakhyat Singhal

Registration Number : 12005809

Program : B.Tech CSE

Semester : Third

School : School of Computer Science and Engineering

Name of University : Lovely Professional University

Date of Submission : 19 November 2021

CONTACTS DATABASE APP

ABOUT :-

It is a desktop app developed for managing contacts using Python and its framework Tkinter. Tkinter is used for developing GUI of the app. Tkinter provided various controls such as buttons, labels and text boxes in the app. I used SQLite Database in the app. SQLite is a software library that provides a relational database system (RDBMS). SQLite is integrated with the application that accesses the database. The application interact with the database, read and write directly from the database files stored on disk.

ACKNOWLEDGEMENT:-

I would like to thank my mentor - Prof. Sagar Pande for his advice on this project and for giving me this opportunity to create a application.

TEAM MEMBERS:-

TEAM LEADER :-

Prakhyat Singhal:-

1. Coding (Complete Project)
2. Managing Database
3. Creating GUI
4. Creating the report

I decided to do it alone because by creating the entire project, I will get to know about every single detail of the project. It will increase my knowledge and efficiency for Python, its framework Tkinter and database SQLite.

TOOLS :-

Tkinter

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

To create a tkinter app:

i. Importing the module – tkinter
ii. Create the main window (container)
iii. Add any number of widgets to the main window
iv. Apply the event Trigger on the widgets.

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application.

SQLite

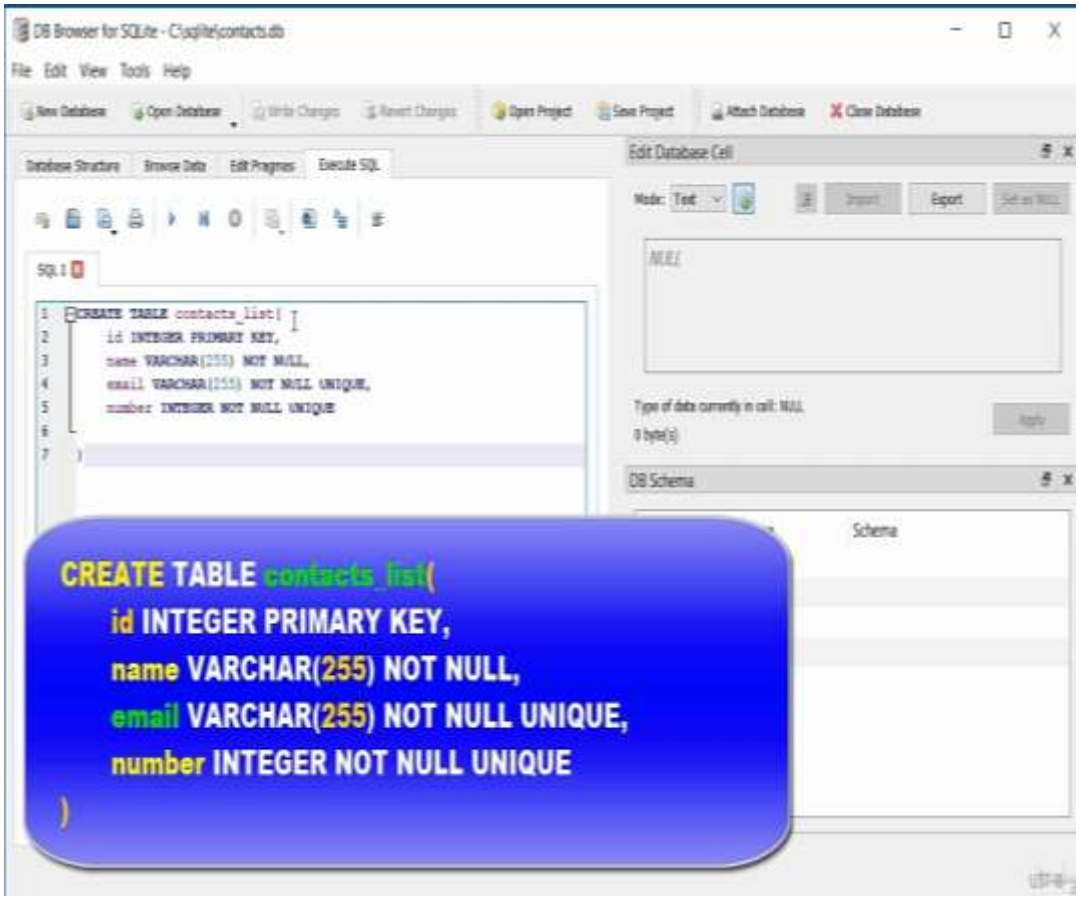
SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine in the world. It is an in-process library and its code is publicly available. It is free for use for any purpose, commercial or private. It is basically an embedded SQL database engine.

Features of SQLite

1. The transactions follow ACID properties i.e. atomicity, consistency, isolation, and durability even after system crashes and power failures.
2. The configuration process is very easy, no setup or administration needed.
3. All the features of SQL are implemented in it with some additional features like partial indexes, indexes on expressions, JSON, and common table expressions.

DB Browser for SQLite

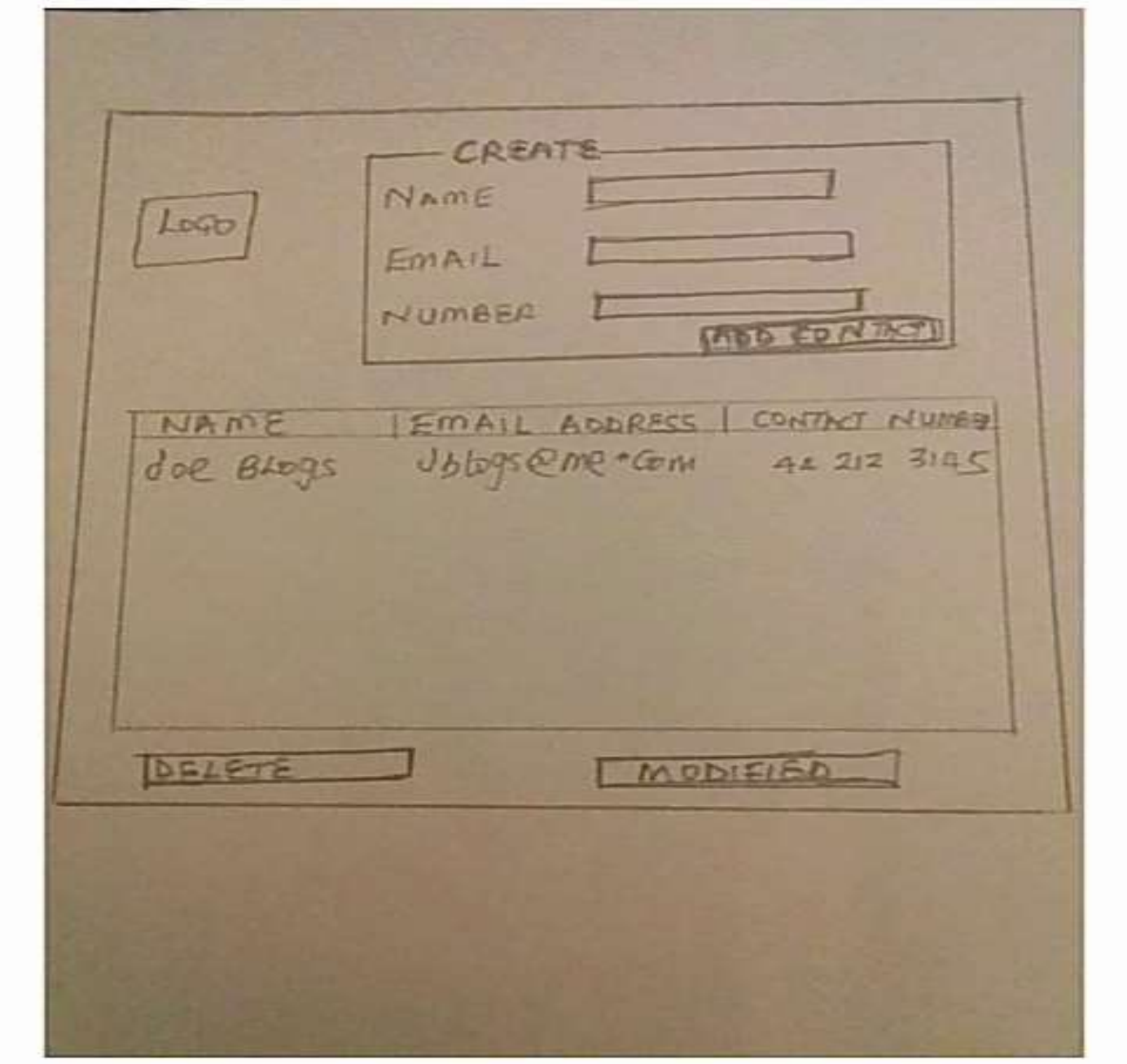
1. High quality, visual open source tool to create, design and edit database files compatible with SQLite.
2. DB Management tool with GUI.



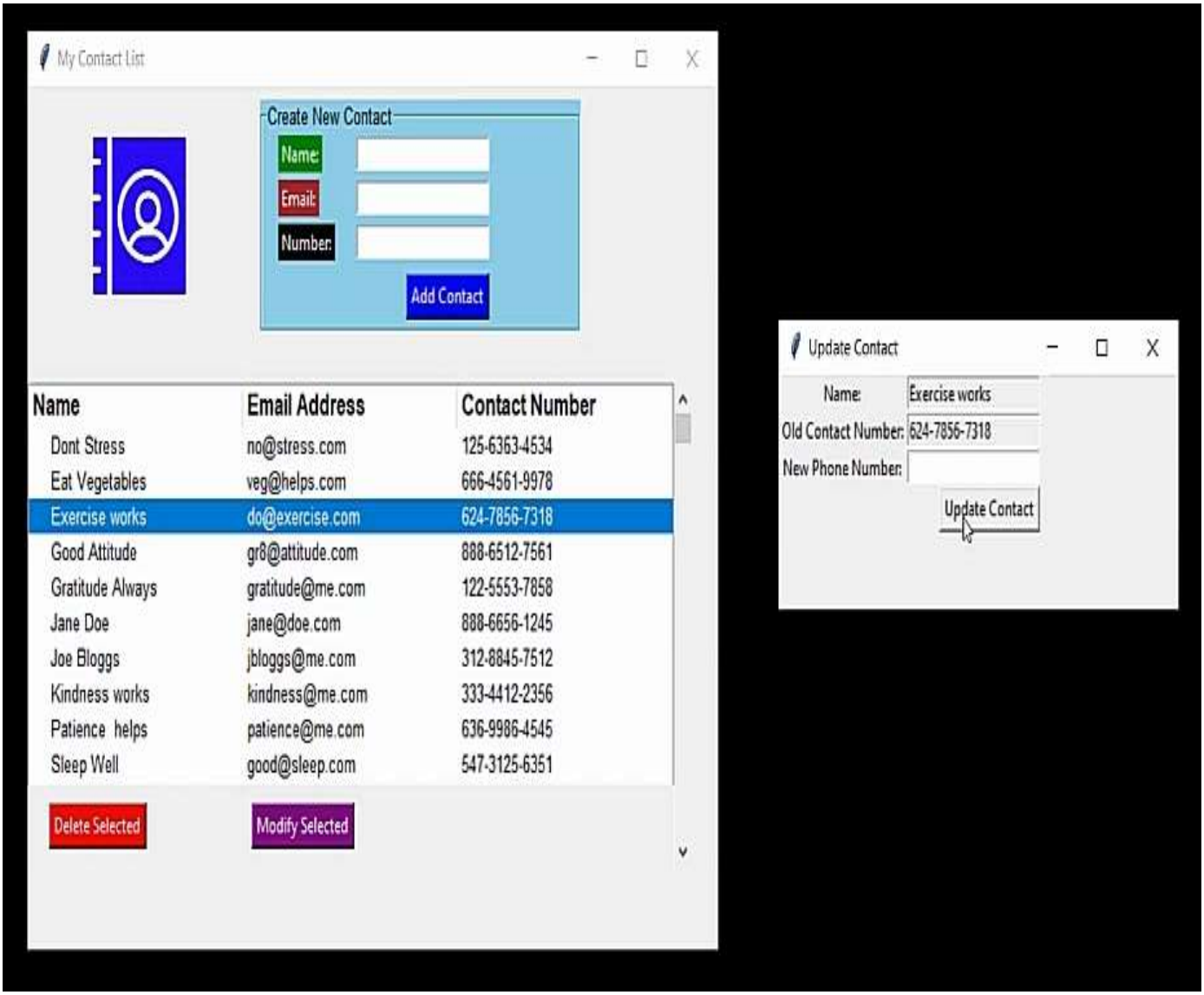
Application Sketch

It is always a good idea to have a rough design sketch of the program we are going to create.

Here is the design sketch of the project.



CONTACTS DATABASE APP



Screenshots of Project :-

```python
# Importing what we need from tkinter module
from tkinter import Tk, Button, PhotoImage, Label, LabelFrame, W, E,N,S, Entry, END,StringVar ,Scrollbar,Toplevel
from tkinter import ttk   # Provides access to the Tk themed widgets.
import sqlite3



class Contacts:
    db_filename = 'contacts.db'

    def __init__(self,root):
        self.root = root
        self.create_gui()
        ttk.style = ttk.Style()
        ttk.style.configure("Treeview", font=('helvetica',10))
        ttk.style.configure("Treeview.Heading", font=('helvetica',12, 'bold'))


    def execute_db_query(self, query, parameters=()):
        with sqlite3.connect(self.db_filename) as conn:
            print(conn)
            print('You have successfully connected to the Database')
            cursor = conn.cursor()
            query_result = cursor.execute(query, parameters)
            conn.commit()
        return query_result
```

```
    def create_gui(self):
        self.create_left_icon()
        self.create_label_frame()
        self.create_message_area()
        self.create_tree_view()
        self.create_scrollbar()
        self.create_bottom_buttons()
        self.view_contacts()


    def create_left_icon(self):
        photo = PhotoImage(file='icons/logo.gif')
        label = Label(image=photo)
        label.image = photo
        label.grid(row=0, column=0)

    def create_label_frame(self):
        labelframe = LabelFrame(self.root, text='Create New Contact',bg="sky blue",font="helvetica 10")
        labelframe.grid(row=0, column=1, padx=8, pady=8, sticky='ew')
        Label(labelframe, text='Name:',bg="green",fg="white").grid(row=1, column=1, sticky=W, pady=2,padx=15)
        self.namefield = Entry(labelframe)
        self.namefield.grid(row=1, column=2, sticky=W, padx=5, pady=2)
        Label(labelframe, text='Email:',bg="brown",fg="white").grid(row=2, column=1, sticky=W,  pady=2,padx=15)
        self.emailfield = Entry(labelframe)
        self.emailfield.grid(row=2, column=2, sticky=W, padx=5, pady=2)
        Label(labelframe, text='Number:',bg="black",fg="white").grid(row=3, column=1, sticky=W,  pady=2,padx=15)
```

```
    def create_label_frame(self):
        labelframe = LabelFrame(self.root, text='Create New Contact',bg="sky blue",font="helvetica 10")
        labelframe.grid(row=0, column=1, padx=8, pady=8, sticky='ew')
        Label(labelframe, text='Name:',bg="green",fg="white").grid(row=1, column=1, sticky=W, pady=2,padx=15)
        self.namefield = Entry(labelframe)
        self.namefield.grid(row=1, column=2, sticky=W, padx=5, pady=2)
        Label(labelframe, text='Email:',bg="brown",fg="white").grid(row=2, column=1, sticky=W,  pady=2,padx=15)
        self.emailfield = Entry(labelframe)
        self.emailfield.grid(row=2, column=2, sticky=W, padx=5, pady=2)
        Label(labelframe, text='Number:',bg="black",fg="white").grid(row=3, column=1, sticky=W,  pady=2,padx=15)
        self.numfield = Entry(labelframe)
        self.numfield.grid(row=3, column=2, sticky=W, padx=5, pady=2)
        Button(labelframe, text='Add Contact', command=self.on_add_contact_button_clicked,bg="blue",fg="white").grid(row=4,

    def create_message_area(self):
        self.message = Label(text='', fg='red')
        self.message.grid(row=3, column=1, sticky=W)

    def create_tree_view(self):
        self.tree = ttk.Treeview(height=10, columns=("email","number"),style='Treeview')
        self.tree.grid(row=6, column=0, columnspan=3)
        self.tree.heading('#0', text='Name', anchor=W)
        self.tree.heading("email", text='Email Address', anchor=W)
        self.tree.heading("number", text='Contact Number', anchor=W)
```

```
    def create_bottom_buttons(self):
        Button(text='Delete Selected', command=self.on_delete_selected_button_clicked,bg="red",fg="white").grid(row=8, column
        Button(text='Modify Selected', command=self.on_modify_selected_button_clicked,bg="purple",fg="white").grid(row=8, col

    def on_add_contact_button_clicked(self):
        self.add_new_contact()

    def on_delete_selected_button_clicked(self):
        self.message['text'] = ''
        try:
            self.tree.item(self.tree.selection())['values'][0]
        except IndexError as e:
            self.message['text'] = 'No item selected to delete'
            return
        self.delete_contacts()

    def on_modify_selected_button_clicked(self):
        self.message['text'] = ''
        try:
            self.tree.item(self.tree.selection())['values'][0]

        except IndexError as e:
            self.message['text'] = 'No item selected to modify'
            return
        self.open_modify_window()
```

```
    def add_new_contact(self):
        if self.new_contacts_validated():
            query = 'INSERT INTO contacts_list VALUES(NULL,?, ?,?)'
            parameters = (self.namefield.get(),self.emailfield.get(), self.numfield.get())
            self.execute_db_query(query, parameters)
            self.message['text'] = 'New Contact {} added'.format(self.namefield.get())
            self.namefield.delete(0, END)
            self.emailfield.delete(0, END)
            self.numfield.delete(0, END)
            self.view_contacts()

        else:
            self.message['text'] = 'name,email and number cannot be blank'
            self.view_contacts()

    def new_contacts_validated(self):
        return len(self.namefield.get()) != 0 and len(self.emailfield.get()) != 0 and len(self.numfield.get()) != 0

    def view_contacts(self):
        items = self.tree.get_children()
        for item in items:
            self.tree.delete(item)
        query = 'SELECT * FROM contacts_list ORDER BY name desc'
        contact_entries = self.execute_db_query(query)
        for row in contact_entries:
```

```
    def delete_contacts(self):
        self.message['text'] = ''
        name = self.tree.item(self.tree.selection())['text']
        query = 'DELETE FROM contacts_list WHERE name = ?'
        self.execute_db_query(query, (name,))
        self.message['text'] = 'Contacts for {} deleted'.format(name)
        self.view_contacts()

    def open_modify_window(self):
        name = self.tree.item(self.tree.selection())['text']
        old_number = self.tree.item(self.tree.selection())['values'][1]
        self.transient = Toplevel()
        self.transient.title('Update Contact')
        Label(self.transient, text='Name:').grid(row=0, column=1)
        Entry(self.transient, textvariable=StringVar(
            self.transient, value=name), state='readonly').grid(row=0, column=2)
        Label(self.transient, text='Old Contact Number:').grid(row=1, column=1)
        Entry(self.transient, textvariable=StringVar(
            self.transient, value=old_number), state='readonly').grid(row=1, column=2)

        Label(self.transient, text='New Phone Number:').grid(
            row=2, column=1)
        new_phone_number_entry_widget = Entry(self.transient)
        new_phone_number_entry_widget.grid(row=2, column=2)

```

```
        Button(self.transient, text='Update Contact', command=lambda: self.update_contacts(
            new_phone_number_entry_widget.get(), old_number, name)).grid(row=3, column=2, sticky=E)


        self.transient.mainloop()

    def update_contacts(self, newphone, old_phone,name):
        query = 'UPDATE contacts_list SET number=? WHERE number =? AND name =?'
        parameters = (newphone, old_phone, name)
        self.execute_db_query(query, parameters)
        self.transient.destroy()
        self.message['text'] = 'Phone number of {} modified'.format(name)
        self.view_contacts()




if __name__ == '__main__':
    root =Tk()
    root.title('My Contact List')
    root.geometry("650x450")
    root.resizable(width=False, height=False)
    application = Contacts(root)
    root.mainloop()

```

Conclusions :-

It is my hope that this report will be of huge help with understanding of my little project. It's my first project and I learnt a lot while creating it.

References:-

GeeksForGeeks

https://www.geeksforgeeks.org/python-tkinter-tutorial/

Youtube

https://www.youtube.com/