THE ADITYA BIRLA PUBLIC SCHOOL

ALL INDIA SENIOR SCHOOL CERTIFICATE EXAMINATION OF CBSE

SESSION 2020-2021

COMPUTER SCIENCE

Made BY- Devansh Joshi & Asad Khan

All INDIA SENIOR SCHOOL CERTIFICATE EXAMINATION

CENTRAL BOARD OF SECONDARY EDUCATION

CLASS- XII,SCIENCE 2020-2021

SUBJECT : *Computer Science*

SCHOOL : *Aditya Birla Public School, Renusagar*

DATE OF SUBMISSION :

GUIDED BY : Piyush Joshi
(Computer Science)
Aditya Birla Public School,Renusagar

DECLARATION

I,hereby declare that I have completed my project to the best of my ability and knowledge. It has taken a lot of efforts on my part to successfully complete this project .

I,hereby declare that my project is completed in all respects and this project is an effort of my study and scientific work.

(Devansh Joshi & Asad Khan)

CERTIFICATE

This is to certify that Devansh Joshi & Asad Khan has completed his project work with much sincerity and obedience. He is an obedient student and has completed his project well before the completion of this tenure. I assure that he has not resorted to any unfair means and has done the project with great sincerity.

This project may be consider as fulfillment for A.I.S.S.C.E. conducted by C.B.S.E. in 2020-2021.

I hereby certify that he is well behaved and obedient student to the best of my knowledge and belief. He bears a good moral character.

Piyush Joshi
(Computer Science)
Aditya Birla Public
School,Renusagar

Bidya Chatterjee
PRINCIPAL
Aditya Birla Public
School,Renusagar

ACKNOWLEDGEMENT

The goal was fixed, moves were calculated & I moved, full of enthusiasm, vigor & keen interest. There were times when it proved an uphill last goal going beyond my reach. The work however progressed & my will power grew more strongly. The successful completion of this work further confirms my belief that firm determination always lead to success.

It has rightly been said that we are built on the shoulders of others for everything I have achieved the credit goes to my project guide Mr. Piyush Joshi . He has motivating personality who provided me time, encouragement & determination to complete the project.

Lastly my sincere thanks to all those who helped me directly or indirectly in developing & evolving

(Devansh Joshi & Asad Khan)

About The Project

THIS PROGRAM IS DESIGNED TO STORE AND MANAGE CONTACTS.

OUR PROGRAM CAN DO:-

- ADD AND DELETE CONTACTS
- SAVE AND UPDATE CONTACTS
- IMPORT AND EXPORT CONTACTS
- EXCLUSIVE THEMES(DARK,LIGHT AND BETA THEMES)
- EXCLUSIVE INTERFERENCE

Requirement

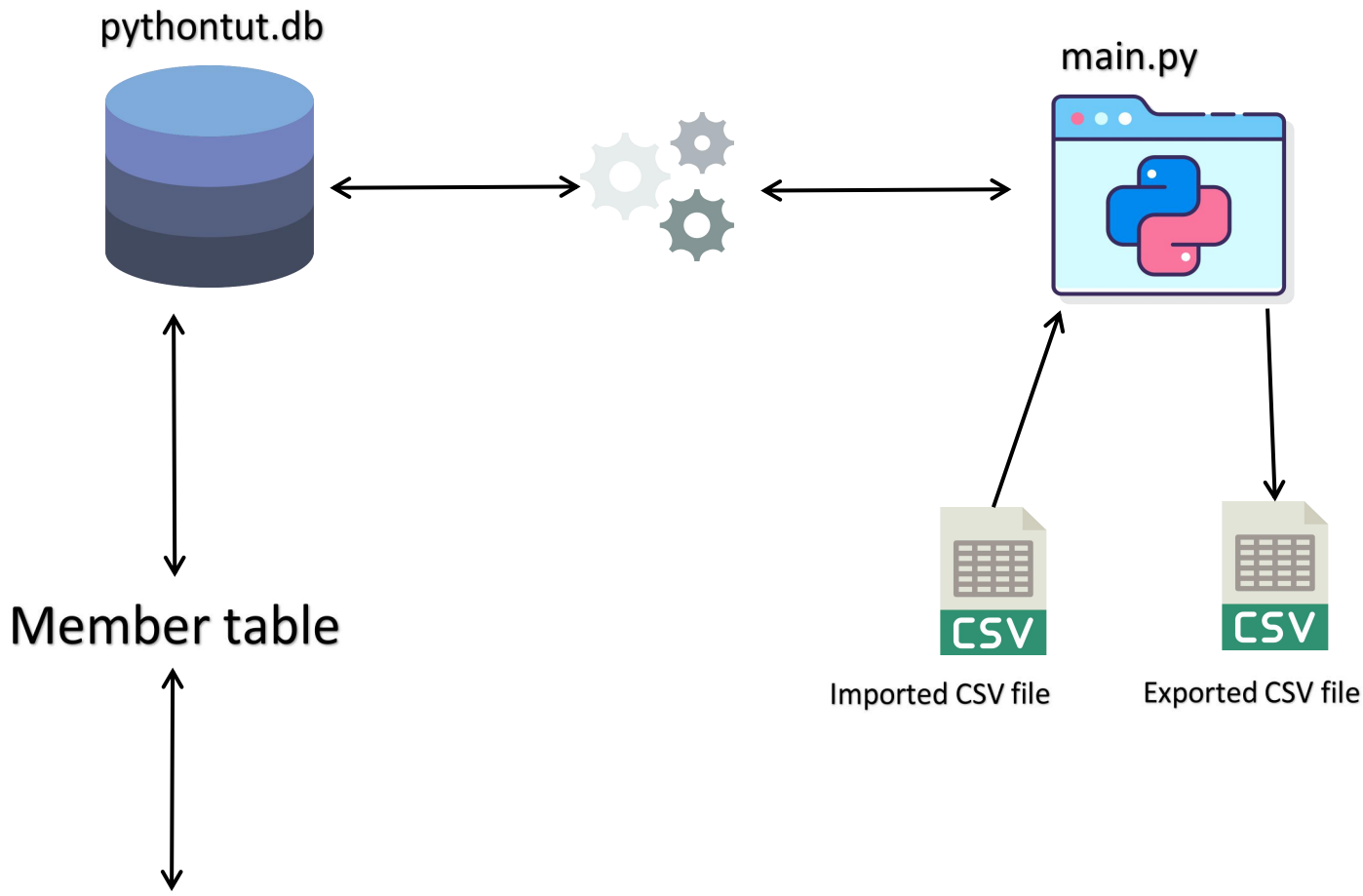
HARDWARE REQUIREMENTS

- RAM - 1 GB
- PROCESSOR - 1Ghz(PENTINUM III)
- HARDDISK - 100GB
- GRAPHIC CARD - DIRCTEX 9 OR WDDM 1.0

SOFTWARE REQUIREMENTS

- WINDOWS VERSION - 7 OR GREATER
- SYSTEM TYPE- 64-BIT
- PYTHON 3.7.x
- TTK THEMES(PYTHON MODULE)

DATA BASE STRUCTURE



FILEDS	TYPE	NULL	KEY	EXTRA
MEMBER ID	INT	NO	PRIMARY	AUTOINCREMENT
FIRSTNAME	TEXT	YES	-	-
LASTNAME	TEXT	YES	-	-
GENDER	TEXT	YES	-	-
AGE	INT	YES	-	-
ADDRESS	TEXT	YES	-	-
CONTACT	INT	YES	-	-

About the language

What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Program Code

```
import tkinter as tk
import csv
from random import choice
from tkinter import filedialog
from tkinter import ttk
from time import sleep
import sqlite3
from tkinter import messagebox

try:
    from ttkthemes import themed_tk
    root = themed_tk.ThemedTk()
    root.set_theme("arc")
except:
    root = tk.Tk()

root.iconphoto(False,tk.PhotoImage(file=".\\img\\icon.png"))
root.title("Contact Manager")
root.minsize(800,500)
root.geometry("800x500")

def animateframe():
    global frames
    for i in range(1,58):
        frames.append(tk.PhotoImage(file=f".\\img\\{i}.png"))
frames=[]
animateframe()
animation = tk.Label(root,image="",bg="white")
animation.pack(fill=tk.BOTH,expand=tk.TRUE)
for i in range(57):
    animation.config(image=frames[i])
    root.update()
    sleep(0.05)
sleep(1.5)
animation.destroy()
```

```
#  variables
FIRSTNAME = tk.StringVar()
LASTNAME = tk.StringVar()
GENDER = tk.StringVar()
AGE = tk.StringVar()
ADDRESS = tk.StringVar()
CONTACT = tk.StringVar()

##########################################################
# FUNCTIONS
def Database():
    conn = sqlite3.connect("pythontut.db")
    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, contact 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():
    if  FIRSTNAME.get() == "" or LASTNAME.get() == "" or GENDER.get() == "" or AGE.get() == "" or ADDRESS.get() == "" or CONTACT.get() == "":
        result = messagebox.showwarning('', 'Please Complete The Required Field', icon="warning")
    else:
        tree.delete(*tree.get_children())
        conn = sqlite3.connect("pythontut.db")
        cursor = conn.cursor()
        cursor.execute("INSERT INTO `member` (firstname, lastname, gender, age, address, contact) VALUES(?, ?, ?, ?, ?, ?)", (str(FIRSTNAME.get()).capitalize(), str(LASTNAME.get()).capitalize(), str(GENDER.get()), int(AGE.get()), str(ADDRESS.get()), str(CONTACT.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()
        FIRSTNAME.set("")
        LASTNAME.set("")
        GENDER.set("")
        AGE.set("")
        ADDRESS.set("")
        CONTACT.set("")

def DeleteData():
    if not tree.selection():
       result = messagebox.showwarning('', 'Please Select Something First!', icon="warning")
    else:
        result = messagebox.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("pythontut.db")
            cursor = conn.cursor()
            cursor.execute("DELETE FROM `member` WHERE `mem_id` = %d" % selecteditem[0])
            conn.commit()
            cursor.close()
            conn.close()


def UpdateData():
    if GENDER.get() == "":
       result = messagebox.showwarning('', 'Please Complete The Required Field', icon="warning")
    else:
        tree.delete(*tree.get_children())
```

```
        conn = sqlite3.connect("pythontut.db")
        cursor = conn.cursor()
        cursor.execute("UPDATE `member` SET `firstname` = ?, `lastname` = ?, `gender` =?, `age` = ?, `address` = ?, `contact` = ? WHERE `mem_id` = ?", (str(FIRSTNAME.get()), str(LASTNAME.get()), str(GENDER.get()), str(AGE.get()), str(ADDRESS.get()), str(CONTACT.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()
        FIRSTNAME.set("")
        LASTNAME.set("")
        GENDER.set("")
        AGE.set("")
        ADDRESS.set("")
        CONTACT.set("")


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("")
    GENDER.set(selecteditem[3])
    FIRSTNAME.set(selecteditem[1])
    LASTNAME.set(selecteditem[2])
    AGE.set(selecteditem[4])
    ADDRESS.set(selecteditem[5])
    CONTACT.set(selecteditem[6])
    UpdateWindow = tk.Toplevel()
```

```
    UpdateWindow.iconphoto(False,tk.PhotoImage(file=".\\img\\icon.png"))
    UpdateWindow.title("Contact List")
    # UpdateWindow.resizable(0, 0)
    UpdateWindow.geometry("400x300")
    if 'NewWindow' in globals():
        NewWindow.destroy()


#====================FRAMES==========================
=
    FormTitle = ttk.Frame(UpdateWindow)
    FormTitle.pack(side=tk.TOP,fill=tk.X)
    ContactForm = ttk.Frame(UpdateWindow)
    ContactForm.pack(side=tk.TOP,fill=tk.BOTH,expand=True)
    lf=ttk.Frame(ContactForm)
    lf.pack(side=tk.LEFT,fill=tk.BOTH,expand=True)
    rf=ttk.Frame(ContactForm)
    rf.pack(side=tk.RIGHT,fill=tk.BOTH,expand=True)
    RadioGroup = ttk.Frame(rf)
    Male = ttk.Radiobutton(RadioGroup, text="Male", variable=GENDER,
value="Male" ).pack(side=tk.LEFT)
    Female = ttk.Radiobutton(RadioGroup, text="Female", variable=GENDER,
value="Female" ).pack(side=tk.LEFT)


#====================LABELS==========================
    lbl_title = ttk.Label(FormTitle, text="Updating Contacts")
    lbl_title.pack(fill=tk.X)
    lbl_firstname = ttk.Label(lf, text="Firstname")
    lbl_firstname.pack(fill=tk.BOTH,expand=True)
    lbl_lastname = ttk.Label(lf ,text="Lastname")
    lbl_lastname.pack(fill=tk.BOTH,expand=True)
    lbl_gender = ttk.Label(lf, text="Gender")
    lbl_gender.pack(fill=tk.BOTH,expand=True)
    lbl_age = ttk.Label(lf, text="Age")
    lbl_age.pack(fill=tk.BOTH,expand=True)
    lbl_address = ttk.Label(lf, text="Address")
    lbl_address.pack(fill=tk.BOTH,expand=True)
    lbl_contact = ttk.Label(lf, text="Contact")
    lbl_contact.pack(fill=tk.BOTH,expand=True)
```

```
#====================ENTRY=============================
=
   firstname = ttk.Entry(rf, textvariable=FIRSTNAME  )
   firstname.pack(fill=tk.BOTH,expand=True)
   lastname = ttk.Entry(rf, textvariable=LASTNAME )
   lastname.pack(fill=tk.BOTH,expand=True)
   RadioGroup.pack(fill=tk.BOTH,expand=True)
   age = ttk.Entry(rf, textvariable=AGE   )
   age.pack(fill=tk.BOTH,expand=True)
   address = ttk.Entry(rf, textvariable=ADDRESS   )
   address.pack(fill=tk.BOTH,expand=True)
   contact = ttk.Entry(rf, textvariable=CONTACT   )
   contact.pack(fill=tk.BOTH,expand=True)


#==================BUTTONS============================
=
   btn_updatecon = ttk.Button(UpdateWindow, text="Update",
command=UpdateData)
   btn_updatecon.pack(side=tk.BOTTOM,fill=tk.X)


def AddNewWindow():
   global NewWindow
   FIRSTNAME.set("")
   LASTNAME.set("")
   GENDER.set("")
   AGE.set("")
   ADDRESS.set("")
   CONTACT.set("")
   NewWindow = tk.Toplevel()
   NewWindow.iconphoto(False,tk.PhotoImage(file=".\\img\\icon.png"))
   NewWindow.title("Contact List")
   width = 400
   height = 300
   NewWindow.geometry("400x300")
   NewWindow.resizable(0, 0)
   if 'UpdateWindow' in globals():
```

```
    UpdateWindow.destroy()


#==================FRAMES=============================
=
   FormTitle = tk.Frame(NewWindow)
   FormTitle.pack(side=tk.TOP,fill=tk.X)
   ContactForm = tk.Frame(NewWindow)
   ContactForm.pack(side=tk.TOP,fill=tk.BOTH,expand=True)
   lf=ttk.Frame(ContactForm)
   lf.pack(side=tk.LEFT,fill=tk.BOTH,expand=True)
   rf=ttk.Frame(ContactForm)
   rf.pack(side=tk.RIGHT,fill=tk.BOTH,expand=True)
   RadioGroup = ttk.Frame(rf)
   Male = ttk.Radiobutton(RadioGroup, text="Male", variable=GENDER,
value="Male" ).pack(side=tk.LEFT)
   Female = ttk.Radiobutton(RadioGroup, text="Female", variable=GENDER,
value="Female" ).pack(side=tk.LEFT)


#==================LABELS=============================
   lbl_title = ttk.Label(FormTitle, text="Adding New Contacts")
   lbl_title.pack(fill=tk.X)
   lbl_firstname = ttk.Label(lf, text="Firstname")
   lbl_firstname.pack(fill=tk.BOTH,expand=True)
   lbl_lastname = ttk.Label(lf ,text="Lastname")
   lbl_lastname.pack(fill=tk.BOTH,expand=True)
   lbl_gender = ttk.Label(lf, text="Gender")
   lbl_gender.pack(fill=tk.BOTH,expand=True)
   lbl_age = ttk.Label(lf, text="Age")
   lbl_age.pack(fill=tk.BOTH,expand=True)
   lbl_address = ttk.Label(lf, text="Address")
   lbl_address.pack(fill=tk.BOTH,expand=True)
   lbl_contact = ttk.Label(lf, text="Contact")
   lbl_contact.pack(fill=tk.BOTH,expand=True)


#==================ENTRY=============================
=
   firstname = ttk.Entry(rf, textvariable=FIRSTNAME )
```

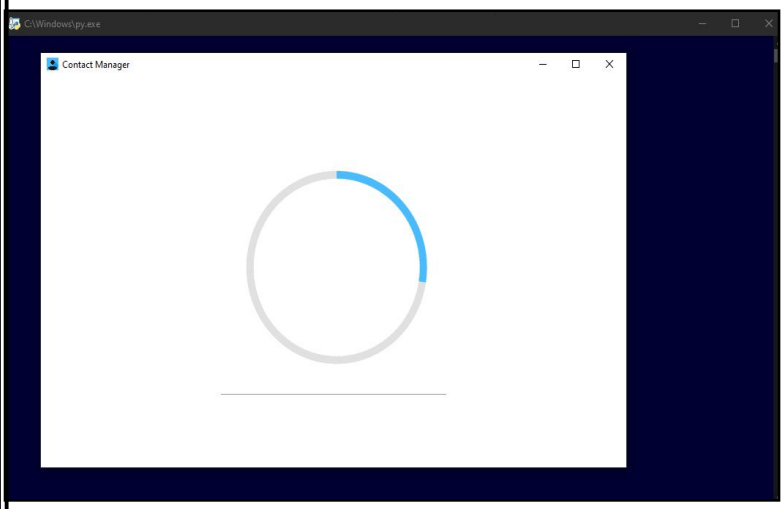
```
    firstname.pack(fill=tk.BOTH,expand=True)
    lastname = ttk.Entry(rf, textvariable=LASTNAME )
    lastname.pack(fill=tk.BOTH,expand=True)
    RadioGroup.pack(fill=tk.BOTH,expand=True)
    age = ttk.Entry(rf, textvariable=AGE   )
    age.pack(fill=tk.BOTH,expand=True)
    address = ttk.Entry(rf, textvariable=ADDRESS   )
    address.pack(fill=tk.BOTH,expand=True)
    contact = ttk.Entry(rf, textvariable=CONTACT   )
    contact.pack(fill=tk.BOTH,expand=True)


#==================BUTTONS=============================
=
    btn_addcon = ttk.Button(NewWindow, text="Save",
width=50,command=SubmitData)
    btn_addcon.pack(side=tk.BOTTOM,fill=tk.X)

def exportcontact():
    askdir = filedialog.asksaveasfilename(title="Save Contacts as
CSV",defaultextension = ".csv")
    if askdir.strip() == "" :
        messagebox.showerror(title="No File Name Give",message="NO file Name Was
Given \n stoping process of exporting...")
    else:
        lis=[["ID",'firstname', 'lastname', 'gender', 'age', 'address', 'contact']]
        conn = sqlite3.connect("pythontut.db")
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM `member` ORDER BY `lastname` ASC")
        fetch = cursor.fetchall()
        for data in fetch:
            lis.append(data)
        cursor.close()
        conn.close()
        with open (askdir,'a') as g:
            obj=csv.writer(g)
            obj.writerows(lis)

def importcontact():
```

```
    askdir = filedialog.askopenfilename(title="Save Contacts as CSV",filetypes = [("CSV FILE","*.csv")])
    if askdir.strip() == "" :
        messagebox.showerror(title="No File Name Give",message="NO file Name Was Given \n stoping process of importing...")
    else:
        conn = sqlite3.connect("pythontut.db")
        cursor = conn.cursor()
        with open(askdir,'r') as f:
            obj=csv.reader(f)
            for i in obj:
                if i != []:
                    cursor.execute("INSERT INTO `member` (firstname, lastname, gender, age, address, contact) VALUES(?, ?, ?, ?, ?, ?)", (i[0].capitalize(),i[1].capitalize(),i[2].capitalize(),i[3] ,i[4],i[5]))
        conn.commit()
        cursor.execute("SELECT * FROM `member` ORDER BY `lastname` ASC")
        fetch = cursor.fetchall()
        tree.delete(*tree.get_children())
        for data in fetch:
            tree.insert('', 'end', values=(data))
        cursor.close()
        conn.close()

def themechanger(theme):
    try:
        if theme == "arc":
            root.set_theme("arc")
            lbl_title.config(bg="white",fg="black")
        elif theme == "equilux":
            root.set_theme("equilux")
            lbl_title.config(bg="gray",fg="white")
        else:
            lbl_title.config(bg="white",fg="black")
            allthemes=['aquativo',"winxpblue",'default', 'black', 'kroc', 'vista','scidblue','ubuntu']
            root.set_theme(choice(allthemes))
    except:
        messagebox.showerror(title="Theme ERROR",message="ttkthemes lib/mod not found !! \n please install it from 'pipy.org'")
```

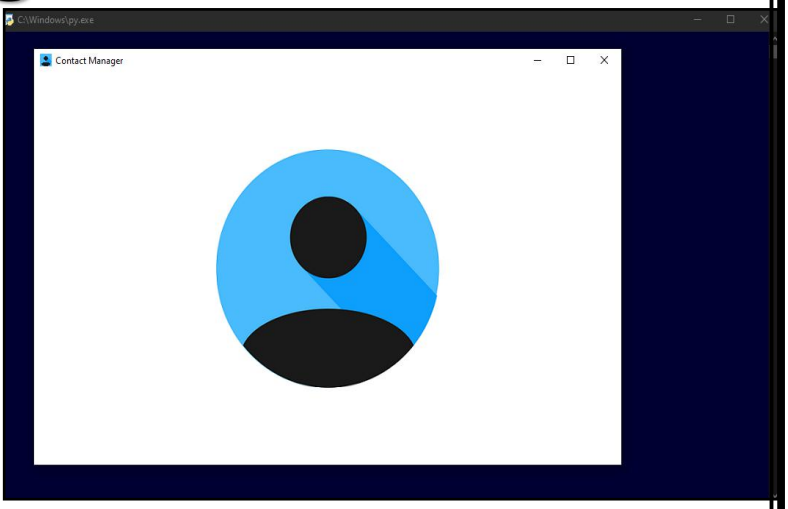
```
#####################################################

#images
addimg=tk.PhotoImage(file=".\\img\\add.png")
delimg=tk.PhotoImage(file=".\\img\\del.png")
searchimg=tk.PhotoImage(file=".\\img\\SEARCH.png")

#FRAMES
lbl_title = tk.Label(root, text="Contacts Management System", font=('arial', 20), width=500)
lbl_title.pack(fill=tk.X,side=tk.TOP)
Topframe = ttk.Frame(root)
Topframe.pack(side=tk.TOP,fill=tk.X)
bottomframe =ttk.Frame(root)
bottomframe.pack(side=tk.BOTTOM,fill=tk.BOTH,expand=tk.TRUE)
#BUTTONs
addbtn = ttk.Button(Topframe,image=addimg,command=AddNewWindow)
addbtn.pack(fill=tk.X,expand=True,side=tk.LEFT,padx=100,pady=5)
delbtn = ttk.Button(Topframe,image=delimg,command=DeleteData)
delbtn.pack(fill=tk.X,expand=True,side=tk.RIGHT,padx=100,pady=5)

#TREE VIEW

scrollbarx = ttk.Scrollbar(bottomframe, orient=tk.HORIZONTAL)
scrollbary = ttk.Scrollbar(bottomframe, orient=tk.VERTICAL)
tree = ttk.Treeview(bottomframe, 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=tk.RIGHT, fill=tk.Y)
scrollbarx.config(command=tree.xview)
scrollbarx.pack(side=tk.BOTTOM, fill=tk.X)
tree.heading('MemberID', text="MemberID", anchor=tk.W)
tree.heading('Firstname', text="Firstname", anchor=tk.W)
tree.heading('Lastname', text="Lastname", anchor=tk.W)
tree.heading('Gender', text="Gender", anchor=tk.W)
tree.heading('Age', text="Age", anchor=tk.W)
tree.heading('Address', text="Address", anchor=tk.W)
tree.heading('Contact', text="Contact", anchor=tk.W)
```

```
tree.column('#0', stretch=tk.YES, minwidth=0, width=0)
tree.column('#1', stretch=tk.YES, minwidth=0, width=80)
tree.column('#2', stretch=tk.YES, minwidth=0, width=120)
tree.column('#3', stretch=tk.YES, minwidth=0, width=90)
tree.column('#4', stretch=tk.YES, minwidth=0, width=80)
tree.column('#5', stretch=tk.YES, minwidth=0, width=120)
tree.column('#6', stretch=tk.YES, minwidth=0, width=120)
tree.pack(fill=tk.BOTH,expand=tk.TRUE)
tree.bind('<Double-Button-1>', OnSelected)

menu = tk.Menu(root)
subMenu = tk.Menu(menu,tearoff = 0)
subMenu1 = tk.Menu(menu,tearoff = 0)
menu.add_cascade(label="<< Themes >>",menu=subMenu)
menu.add_cascade(label="<< More >>",menu=subMenu1)
subMenu.add_command(label="Dark Mode",command=lambda :
themechanger("equilux"))
subMenu.add_command(label="Light Mode",command=lambda :
themechanger("arc"))
subMenu.add_command(label="Try Random (Beta) Theme",command=lambda :
themechanger("random"))
subMenu1.add_command(label="Import From CSV File",command=importcontact)
subMenu1.add_command(label="Export Into CSV File",command=exportcontact)
root.config(menu = menu)


if __name__=="__main__":
    Database()
    root.mainloop()
```

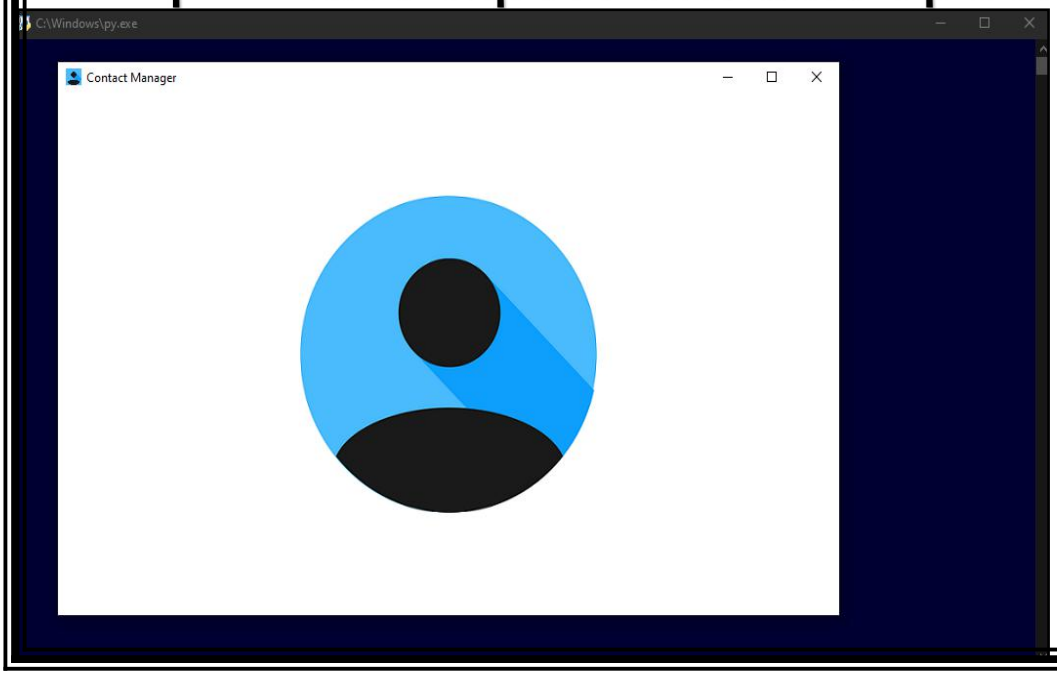
Output & Screenshot



Loading Screens

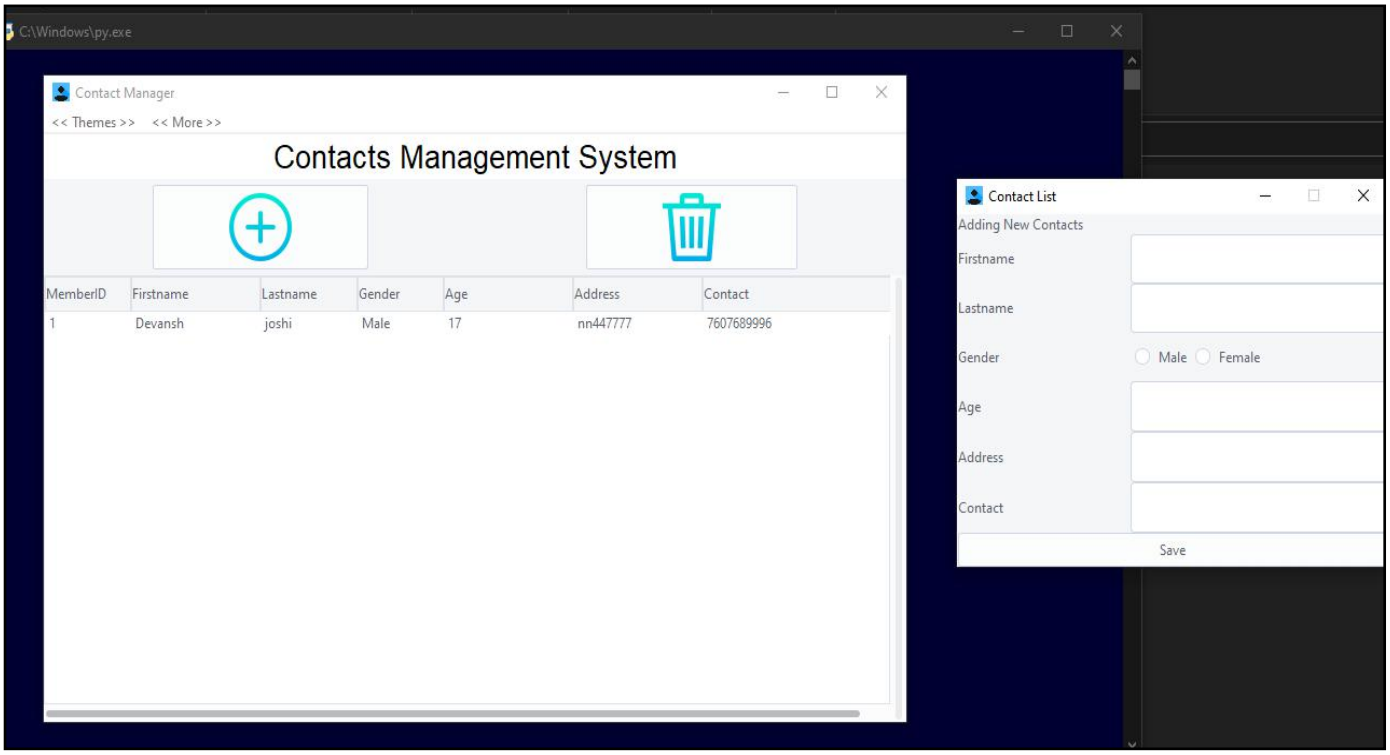


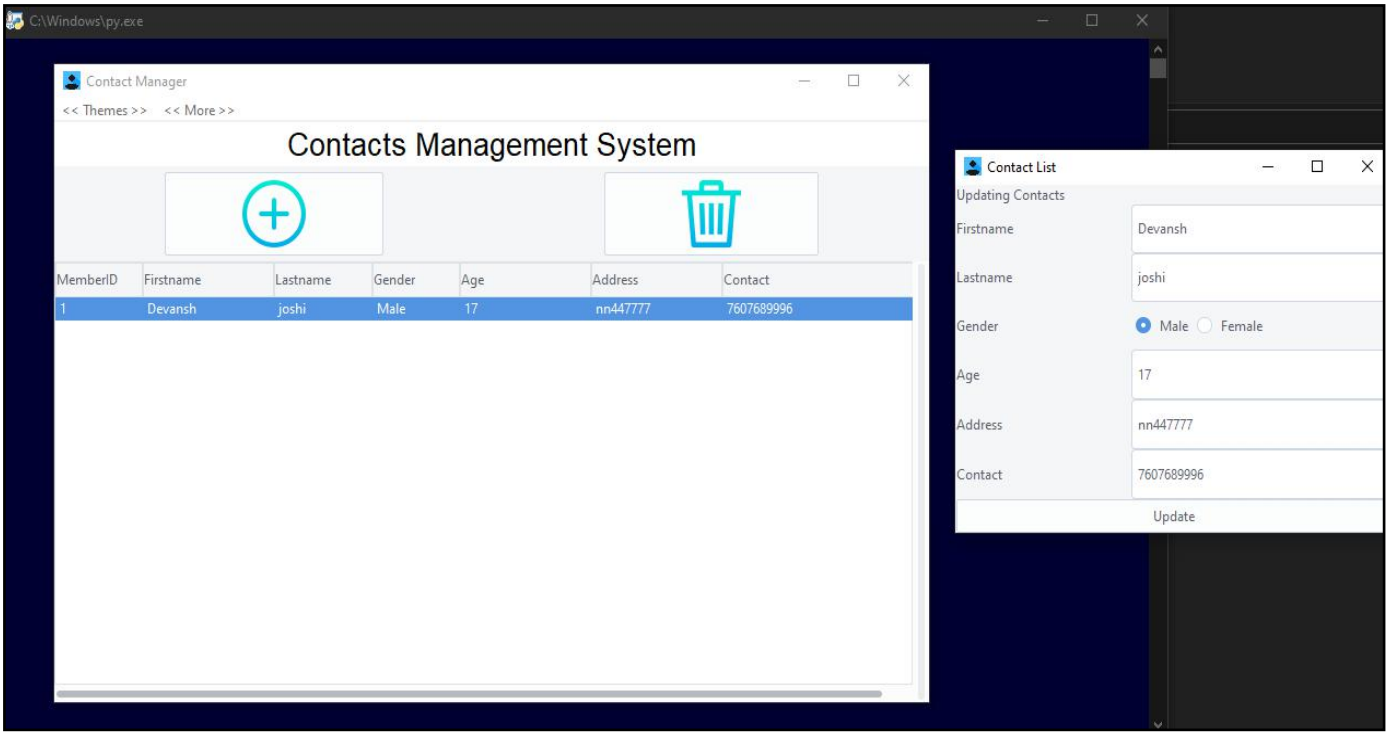
Import and export contacts option at menu bar



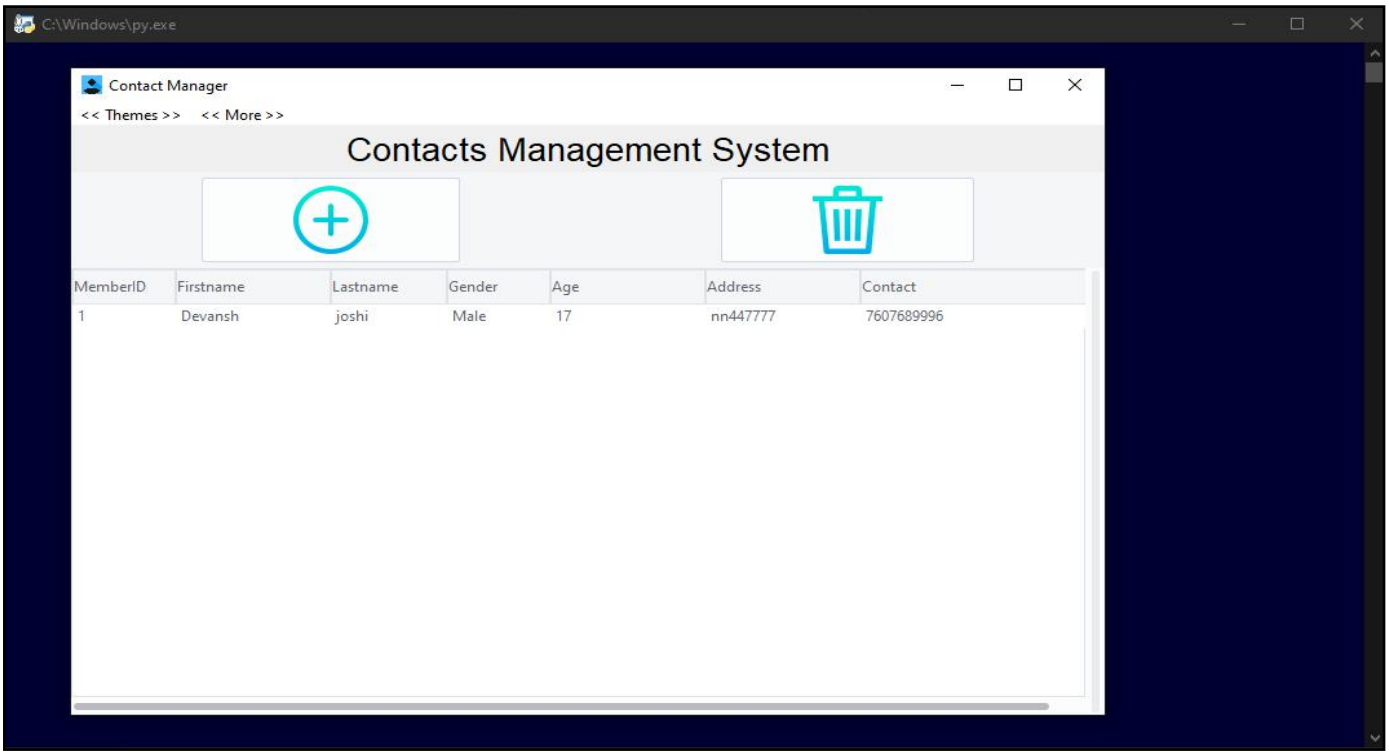
(Importing contacts can be do by CSV file and Exporting contacts outputs a CSV file with all contacts)

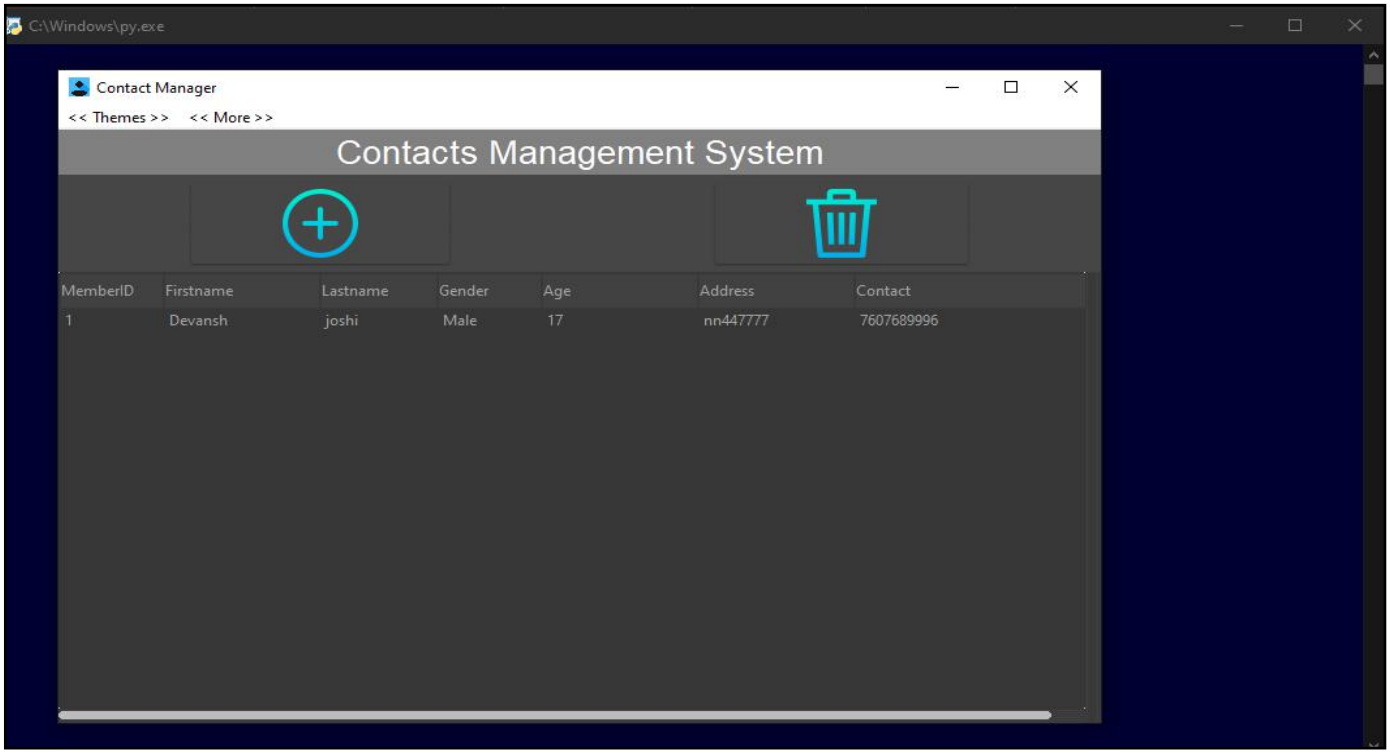
Adding and updating contacts in the program





Dark and Light Themes





Bibliography

BOOK :

- Computer Science with Python By Sumita Arora
- Python For Dummies

INTERNET :

- www.python.org
- www.stackoverflow.com