## **Introduction**

Through this project, we have attempted to develop an application named PhotoEditor, which can be used to edit photos with provided functionalities. The functionalities included in this project include adjustment of brightness, contrast, temperature and tint of an image using sliders as per one's requirements. Further functions for colour inversion, greyscale inversion, and for flipping and rotating images too are provided. Further, we provide buttons for undo, redo, and save operations.

All these functionalities are developed using Python PIL module and tkinter module for user interactive GUI and the back end is developed using MySQL.

The main advantage of this application is that it provides a generalized tool for the users to edit an image  and they can use it to edit images however they require based on their needs and requirements.

Further enhancements of application functionalities are possible.

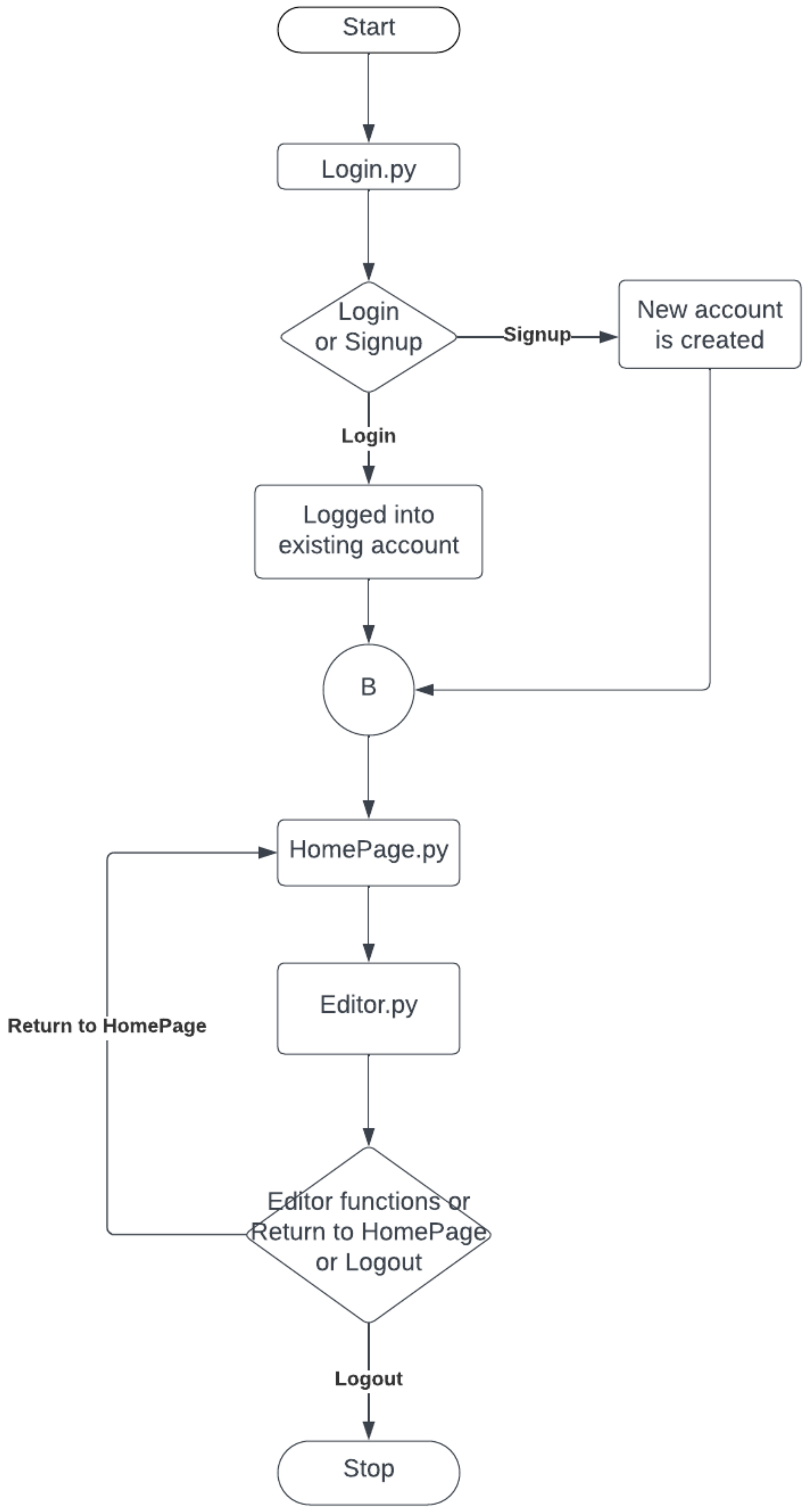
## Software used

Python 3.11.1 on Windows 11

Python Modules:

- tkinter (filedialog, ttk, messagebox)
- PIL (Image, ImageEnhnace, ImageTk, ImageOps, ImageStat)
- mysql.connector
- subprocess

All user account credentials are stored in **accounts** table in MySQL database **12ceditor**


Start
Login.py
Login or Signup
Signup
New account is created
Login
Logged into existing account
B
HomePage.py
Editor.py
Return to HomePage
Editor functions or Return to HomePage or Logout
Logout
Stop

## SOURCE CODE:

## Module 1: Login.py

```
import tkinter as tk
from tkinter import messagebox
from PIL import Image,ImageTk
import mysql.connector
import subprocess
import os



mycon= conn=mysql.connector.connect(
        host="localhost",
        user="root",
        password="admin"
    )

cr=mycon.cursor()
sql="CREATE DATABASE IF NOT EXISTS 12ceditor;"
cr.execute(sql)
sql="USE 12ceditor;"
cr.execute(sql)
sql="CREATE TABLE IF NOT EXISTS accounts(username
varchar(50),password varchar(50));"
cr.execute(sql)
```

```
def login():
    # Authenticate the user based on the input fields
    l_username=login_un_var.get()
    l_password=login_pw_var.get()

    # Connect to the MySQL database and query the accounts table to check if 
the username and password are valid
    conn=mysql.connector.connect(
        host="localhost",
        user="root",
        password="admin",
        database="12ceditor"
    )
    cursor=conn.cursor()
    cursor.execute("SELECT * FROM accounts WHERE username='{}' AND 
password='{}';".format(l_username,l_password))
    account=cursor.fetchone()

    if account!=None:
        # If the username and password are correct, then opening editor window

        main_window.destroy()

subprocess.run(['python',r"C:\Users\Ahila\OneDrive\Desktop\PROJECT12C\H
omePage.py"])
    else:
        # If either of username or password is incorrect, diaplaying an error message
        messagebox.showinfo('Invalid','Invalid Username or Password!')

    conn.close()



def signup():
    # Insert the new user account into the accounts table
    s_username = signup_un_var.get()
    s_password = signup_pw_var.get()

    # Connect to the MySQL database and insert the new user account into the accounts table
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="admin",
        database="12ceditor"
    )
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM accounts WHERE username='{}' AND password='{}';".format(s_username,s_password))
    account=cursor.fetchone()
    if not account:
        cursor.execute("INSERT INTO accounts (username, password) VALUES ('{}','{}')".format(s_username,s_password))
        conn.commit()
        messagebox.showinfo('Success','Success! new account is created!')
        main_window.destroy()
        hpath=os.path.join("PROJECT12C","HomePage.py")
subprocess.run(['python',r"C:\Users\Ahila\OneDrive\Desktop\PROJECT12C\H
omePage.py"])



    else:
        messagebox.showinfo('Oops!','An acccount with these details already
exists! Retry with new details!')



    # Close the database connection and display a success message
    conn.close()




# Create the main window
main_window = tk.Tk()
main_window.title('PhotoEditor - Login Page')

# Get the screen width and height
wth = main_window.winfo_screenwidth()
ht = main_window.winfo_screenheight()
main_window.geometry('{}x{}'.format(wth,ht))
try:
    img=Image.open(r"C:\Users\Ahila\Downloads\bgimage.png")
    resized_img=img.resize((wth,ht))
    photo_image=ImageTk.PhotoImage(resized_img)
    bglabel=tk.Label(main_window,image=photo_image)
    bglabel.place(x=0,y=0)

    f1=tk.Frame(main_window,width=wth//2,height=ht//2)
    fwth =f1.winfo_screenwidth()
    fht = f1.winfo_screenheight()
    f1.place(x=(wth//2)-(fwth//4),y=(ht//2)-(fht//3))

except:
    main_window.configure(bg='blue')
    f1=tk.Frame(main_window,width=wth//2,height=ht//2)
    fwth =f1.winfo_screenwidth()
    fht = f1.winfo_screenheight()
    f1.place(x=(wth//2)-(fwth//4),y=(ht//2)-(fht//3))
    f1.configure(bg='cyan')



signup_un_var=tk.StringVar()
signup_pw_var=tk.StringVar()

login_un_var=tk.StringVar()
login_pw_var=tk.StringVar()




l1=tk.Label(f1,text='WELCOME TO 12C
PHOTOEDITOR.',font=('Arial',15,'bold'))
```

```
l1.place(x=150,y=20)

l2=tk.Label(f1,text='New here? Signup to continue..',font=('Arial',10,'bold'))
l2.place(x=200,y=60)

l3= tk.Label(f1, text="Username",font=('Arial',10,'bold'))
l3.place(x=100,y=100)

e1 = tk.Entry(f1,textvariable=signup_un_var)
e1.place(x=200,y=100,width=300,height=25)

l4= tk.Label(f1,text="Password",font=('Arial',10,'bold'))
l4.place(x=100,y=140)

e2= tk.Entry(f1,textvariable=signup_pw_var,show="*")
e2.place(x=200,y=140,width=300,height=25)

b1=tk.Button(f1,height=2,text='Signup',command=signup)
b1.place(x=300,y=180)

l5=tk.Label(f1,text="Already a user? Login and let's
begin.",font=('Arial',10,'bold'))
l5.place(x=200,y=220)

l6= tk.Label(f1, text="Username",font=('Arial',10,'bold'))
l6.place(x=100,y=260)

e3 = tk.Entry(f1,textvariable=login_un_var)
```

```
e3.place(x=200,y=260,width=300,height=25)

l7= tk.Label(f1,text="Password",font=('Arial',10,'bold'))
l7.place(x=100,y=300)

e4= tk.Entry(f1,textvariable=login_pw_var,show="*")
e4.place(x=200,y=300,width=300,height=25)

b2=tk.Button(f1,height=2,text='Login',command=login)
b2.place(x=300,y=340)



main_window.mainloop()
```

**Module 2: HomePage.py**

```
import tkinter as tk
from PIL import Image,ImageTk
import subprocess

w1=tk.Tk()
wth=w1.winfo_screenwidth()
ht=w1.winfo_screenheight()
w1.title('PhotoEditor - HomePage')
w1.geometry('{}x{}'.format(wth,ht))
w1.configure(bg='red')
```

```
def click():
    w1.destroy()

subprocess.run(['python',r"C:\Users\Ahila\OneDrive\Desktop\PROJECT12C\ed
itor.py"])



ltext="""Welcome to 12C Editor.

Photos are a great way of preserving great moments of our life in memory.

We offer features to enhance and manipulate your photos so as to make them
even more beautiful..

Some of them include Brightness, Contrast, Temperature and Tint adjustment
that you could perform by moving

the sliders.. Moreover, there are options to rotate an image, get mirror image of
an image too.

On clicking the button below, you will be redirected to the editor page,

wherein you can edit an image ... Happy Editing !!!"""



text1=tk.Label(w1,text=ltext,font=('Arial',15,'bold'))
text1.place(x=wth//6.5,y=ht//6)
text1.configure(bg='bisque1')
```

```
b1=tk.Button(w1,text='Choose an image to start
editing',width=25,height=5,command=click)
b1.place(x=wth//2.1,y=ht//1.5)

w1.mainloop()
```

**Module 3: editor.py**

```
import tkinter as tk
from tkinter import filedialog
from tkinter import ttk
from tkinter import messagebox
from PIL import Image,ImageEnhance,ImageTk,ImageOps,ImageStat
import subprocess


w=tk.Tk()

wid=w.winfo_screenwidth()
hgt=w.winfo_screenheight()
w.configure(width=wid,height=hgt,bg='black')
w.title('PhotoEditor - Editor Window')



fwid=0.25*wid
```

```
fhgt=0.50*hgt
f1=tk.Frame(w,width=fwid,height=fhgt)
f1.place(x=wid//25,y=hgt//15)

f2=tk.Frame(w,width=fwid,height=fhgt)
f2.place(x=wid//3,y=hgt//15)

f3=tk.Frame(w,width=1.5*fwid,height=fhgt,bg='black')
f3.place(x=wid//1.6,y=hgt//15)

f4=tk.Frame(w,width=wid-100,height=fhgt//2,bg="black")
f4.place(x=wid//25,y=hgt//1.7)

canvas_original=tk.Canvas(f1,
width=fwid,height=fhgt,highlightbackground='red')
canvas_original.place(x=0,y=0)

canvas_edited=tk.Canvas(f2, width=fwid,height=fhgt)
canvas_edited.place(x=0,y=0)

# Function to update the canvas with a new image
def update_canvas(image):
    global image_tk
    image_tk=ImageTk.PhotoImage(image)
    canvas_edited.create_image(0, 0, anchor=tk.NW, image=image_tk)
```

```
def open_new():
    global file_path,resized_img,orig_stat,original_image,current_image
    file_path=filedialog.askopenfilename()
    img=Image.open(file_path)

    # Resize the image to fit the canvas widgets
    resized_img=img.resize((int(fwid),int(fhgt)))
    photo_original=ImageTk.PhotoImage(resized_img)
    canvas_original.create_image(0, 0, anchor=tk.NW, image=photo_original)
    canvas_original.image=photo_original
    update_canvas(resized_img)

    original_image = resized_img

    # Create an ImageStat object for the original image
    orig_stat = ImageStat.Stat(original_image)

    # Define global variables to store the current state of the image and the
undo/redo stacks
    global
undo,redo,levelt1,levelt2,brightness_value,contrast_value,temperature_value,tin
t_value
    current_image=resized_img
    undo=[current_image]
    redo=[]
    levelt1=levelt2=0

    # Defineing global variables to store the current values of the sliders
    brightness_value=tk.DoubleVar(value=1.0)
    contrast_value=tk.DoubleVar(value=1.0)
    temperature_value=tk.IntVar(value=0)
    tint_value=tk.IntVar(value=0)

open_new()







def saveslider():
    # Save the current state of the image to the undo list
    undo.append(current_image.copy())

    brightness_value.set(1.0)
    l1.config(text="Brightness Value: 1")
    contrast_value.set(0)
    l2.config(text="Contrast Value: 1")
    temperature_value.set(0)
    l3.config(text="Temperature Value: 0")
    tint_value.set(0)
    l4.config(text="Tint Value: 0")
```

```
# Function to adjust the brightness of an image
def adjust_brightness(level):
    global current_image,undo
    level=float(s1.get())
    l1.config(text="Brightness Value: {}".format(int(level)))
    level=((level+100)/200)*1.5+0.5

    current_stat=ImageStat.Stat(current_image)
    # Get the brightness value of the original image
    orig_brightness=orig_stat.mean[0]
    # Adjust the brightness of the current image to match the original brightness
    enhancer=ImageEnhance.Brightness(current_image)
    current_image= enhancer.enhance(orig_brightness/current_stat.mean[0])
    enhancer=ImageEnhance.Brightness(current_image)
    modified_image=enhancer.enhance(level)
    current_image=modified_image
    update_canvas(modified_image)

# Function to adjust the contrast of an image
def adjust_contrast(level):
    global current_image,undo
    level=float(s2.get())
    l2.config(text="Contrast Value: {}".format(int(level)))
    level=((level+100)/200)*1.5+0.5
    current_stat=ImageStat.Stat(current_image)
    current_contrast=current_stat.stddev[0]

    # Get the contrast value of the original image and modify ot new image
    orig_contrast= orig_stat.stddev[0]
    enhancer=ImageEnhance.Contrast(current_image)
    current_image=enhancer.enhance(orig_contrast/current_contrast)
    enhancer=ImageEnhance.Contrast(current_image)
    modified_image=enhancer.enhance(level)
    current_image=modified_image
    update_canvas(modified_image)

# Function to adjust the temperature of an image
def adjust_temperature(level):
    global current_image,undo,levelt1
    level=float(level)
    l3.config(text="Temperature Value: {}".format(int(level)))

    r,g,b=current_image.split()[0:3]
    r1=r.point(lambda x:x+level-levelt1)
    b1=b.point(lambda x:x-level+levelt1)
    levelt1=level
    modified_image=Image.merge("RGB", (r1, g, b1))
    current_image=modified_image
    update_canvas(modified_image)
```

```
# Function to adjust the tint of an image
def adjust_tint(level):
    global current_image,undo,levelt2
    level=float(level)
    l4.config(text="Tint Value: {}".format(int(level)))

    r,g,b=current_image.split()[0:3]
    r1=r.point(lambda x:x+level-levelt2)
    g1=g.point(lambda x:x+level-levelt2)
    levelt2=level
    modified_image=Image.merge("RGB", (r1, g1, b))
    current_image=modified_image
    update_canvas(modified_image)



def rotate():
    global current_image,undo
    undo.append(current_image.copy())
    flipped_image=ImageOps.flip(current_image)
    current_image=flipped_image
    update_canvas(flipped_image)

def mirror():
    global current_image,undo
    undo.append(current_image.copy())
    mirror_image=ImageOps.mirror(current_image)
    current_image=mirror_image
    update_canvas(mirror_image)

def greyscale():
    global current_image,undo
    if current_image.mode!='L':
        undo.append(current_image.copy())
        greyscale_image=ImageOps.grayscale(current_image)
        current_image=greyscale_image
        update_canvas(greyscale_image)



# Function to invert the colors of an image
def invert_colors():
    global current_image,undo
    undo.append(current_image.copy())
    inverted_image=ImageOps.invert(current_image.convert("RGB"))
    current_image=inverted_image
    update_canvas(inverted_image)

# Define the save function
def save():


    filename=filedialog.asksaveasfilename(defaultextension='.png')
    if not filename:
        return

    # Save the edited image to the specified filename
    current_image.save(filename)

# Function to undo the last change made to the image
def undo_change():
    global current_image,undo,redo
    # Check if there are any changes to undo
    if len(undo)>1:



        brightness_value.set(1.0)
        l1.config(text="Brighntess Value: 1")
        contrast_value.set(0)
        l2.config(text="Contrast Value: 1")
        temperature_value.set(0)
        l3.config(text="Temperature Value: 0")
        tint_value.set(0)
        l4.config(text="Tint Value: 0")

        # Pop the current image state from the undo
        current_image=undo.pop()
        # Get the previous image state from the undo
        previous_image=undo[-1]
        current_image=previous_image
        # Save the current image state to the redo
        redo.append(current_image.copy())
        # Display the previous image state on the canvas
        update_canvas(previous_image)
    else:
        # There are no changes to undo
        messagebox.showinfo("Can't Undo!","There are no previous image states to perform undo operation.")

#Function to redo the last change made to the image
def redo_change():
    global current_image,undo,redo
    # Check if there are any changes to redo
    if len(redo)>0:
        #Pop the next image state from the redo
        next_image=redo.pop()

        #Save the current image state to the undo
        undo.append(current_image.copy())

        #Update the current image with the next image state
        current_image=next_image

        #Display the next image state on the canvas
        update_canvas(next_image)

    else:
        pass
        #There are no changes to redo
```

```
def revert_to_original():
    global
current_image,undo,brightness_value,contrast_value,temperature_value,tint_val
ue


    brightness_value.set(1.0)
    l1.config(text="Brightness Value: 1")
    contrast_value.set(0)
    l2.config(text="Contrast Value: 1")
    temperature_value.set(0)
    l3.config(text="Temperature Value: 0")
    tint_value.set(0)
    l4.config(text="Tint Value: 0")

    current_image=original_image.copy()
    update_canvas(original_image)

# Creating a custom style for the Scale widget
style = ttk.Style()
style.configure("Custom.Horizontal.TScale",
background="yellow",foreground="lightblue",troughcolor="red")

def logout_window():
    w.destroy()

subprocess.run(['python',r"C:\Users\Ahila\OneDrive\Desktop\PROJECT12C\Lo
gin.py"])

def survey():
```

```
subprocess.run(['python',r"C:\Users\Ahila\OneDrive\Desktop\PROJECT12C\su
rvey.py"])



# Create 4 sliders and their names
s1=ttk.Scale(f3,from_=-
100,to=100,orient=tk.HORIZONTAL,length=200,variable=brightness_value,sty
le="Custom.Horizontal.TScale",command=adjust_brightness)
s1.grid(row=0,column=0,padx=10,pady=50)
l1=tk.Label(f3,text="Brightness Value: 1")
l1.grid(row=1,column=0,pady=10)

s2=ttk.Scale(f3,from_=-
100,to=100,orient=tk.HORIZONTAL,length=200,variable=contrast_value,style
="Custom.Horizontal.TScale",command=adjust_contrast)
s2.grid(row=0,column=1,padx=50,pady=50)
l2=tk.Label(f3,text="Contrast Value: 1")
l2.grid(row=1,column=1,pady=10)

s3=ttk.Scale(f3,from_=-
75,to=75,orient=tk.HORIZONTAL,length=200,variable=temperature_value,styl
e="Custom.Horizontal.TScale",command=adjust_temperature)
s3.grid(row=4,column=0,padx=10,pady=50)
l3=tk.Label(f3,text="Temperature Value: 0")
l3.grid(row=5,column=0,pady=10)

s4=ttk.Scale(f3,from_=-
100,to=100,orient=tk.HORIZONTAL,length=200,variable=tint_value,style="C
ustom.Horizontal.TScale",command=adjust_tint)
s4.grid(row=4,column=1,padx=50,pady=50)
```

```
l4=ttk.Label(f3,text="Tint Value: 0")

l4.grid(row=5,column=1,pady=10)



# creating buttons for rotate,mirror,color inversion and greyscale, saving
changes of sliders and placing them in 1st row


rotate_b=tk.Button(f4,text="Rotate",width=20,height=2,bg="orange",command
=rotate)

rotate_b.grid(row=0,column=0,padx=50,pady=10)


mirror_b=tk.Button(f4,text="Mirror",width=20,height=2,bg="lightgreen",comm
and=mirror)

mirror_b.grid(row=0,column=1,padx=50,pady=10)


color_inversion_b=tk.Button(f4,text="Color
Inversion",width=20,height=2,bg="yellow",command=invert_colors)

color_inversion_b.grid(row=0,column=2,padx=50,pady=10)


greyscale_b=tk.Button(f4,text="Greyscale",width=20,height=2,bg="lightblue",c
ommand=greyscale)

greyscale_b.grid(row=0,column=3,padx=50,pady=10)



slider=tk.Button(f4,text='Save changes of
sliders',width=20,height=2,bg='yellow',command=saveslider)

slider.grid(row=0,column=4,padx=50,pady=10)
```

```
# creating buttons for undo,redo,revert,logout,save and placing them in second
row

undo_b=tk.Button(f4,text="Undo",width=20,height=2,bg="lightblue",command
=undo_change)
undo_b.grid(row=1,column=0,padx=50,pady=10)

redo_b=tk.Button(f4,text="Redo",width=20,height=2,bg="yellow",command=r
edo_change)
redo_b.grid(row=1,column=1,padx=50,pady=10)

revert_b=tk.Button(f4,text='Revert To
Original',width=20,height=2,bg='white',command=revert_to_original)
revert_b.grid(row=1,column=2,padx=50,pady=10)

log_b=tk.Button(f4,text='Logout',width=20,height=2,bg='lightgreen',command=
logout_window)
log_b.grid(row=1,column=3,padx=50,pady=10)

save_b=tk.Button(f4,text="Save",width=20,height=2,bg="red",command=save)
save_b.grid(row=1,column=4,padx=50,pady=10)


# Button for Returning to HomePage

choose_b=tk.Button(f4,text='Choose Another
Image',width=20,height=2,bg='lightgreen',command=open_new)
choose_b.grid(row=2,column=0,padx=50,pady=10)
```

```
survey_b=tk.Button(f4,text='Attend
Survey',width=20,height=2,bg='lightblue',command=survey)
survey_b.grid(row=2,column=1,padx=50,pady=10)

w.mainloop()
```

**Module 4: survey.py**

```
import tkinter as tk
from tkinter import ttk
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import mysql.connector
import subprocess

# Connect to MySQL database
mc2=mysql.connector.connect(
    host="localhost",
    user="root",
    password="admin",
    database="12ceditor"
)

# Create a cursor
cr2=mc2.cursor()

# create table
```

```
cr2.execute("CREATE TABLE IF NOT EXISTS survey (q1 varchar(30),q2
varchar(30),q3 varchar(30),q4 varchar(30),q5 varchar(30));")

# Function to submit survey responses to the database
def submit_survey():
    # Get responses
    q1_response=q1_var.get()
    q2_response=q2_var.get()
    q3_response=q3_var.get()
    q4_response=q4_var.get()
    q5_response=q5_var.get()

    # Save responses to MySQL database
    cr2.execute("INSERT INTO survey (q1,q2,q3,q4,q5) VALUES
('{}','{}','{}','{}','{}')".format(q1_response,q2_response,q3_response,q4_respons
e,q5_response))
    mc2.commit()



# Function to update details from MySQL database
def update_details():
    cr2.execute("SELECT * FROM survey;")
    data=cr2.fetchall()

    #Count responses
    q1_count={}
    q2_count={}
    q3_count={}
```

```
q4_count={}
q5_count={}
for row in data:
    if row[0] not in q1_count:
        q1_count[row[0]]=1
    elif row[0] in q1_count:
        q1_count[row[0]]+=1

    if row[1] not in q2_count:
        q2_count[row[1]]=1
    elif row[1] in q2_count:
        q2_count[row[1]]+=1

    if row[2] not in q3_count:
        q3_count[row[2]]=1
    elif row[2] in q3_count:
        q3_count[row[2]]+=1

    if row[3] not in q4_count:
        q4_count[row[3]]=1
    elif row[2] in q3_count:
        q4_count[row[3]]+=1

    if row[4] not in q5_count:
        q5_count[row[4]]=1
    elif row[4] in q5_count:
        q5_count[row[4]]+=1
count=[q1_count,q2_count,q3_count,q4_count,q5_count]


def generate_piechart():
    nonlocal qno

    ax.clear()
    ax.pie(count[qno-1].values(),labels=count[qno-
1].keys(),autopct='%1.1f%%', startangle=90)
    ax.set_title("Q"+questions[qno-1])
    canvas.draw()
    if qno<5:
        qno+=1
    else:
        qno=1


window.destroy()
new_window=tk.Tk()
new_window.title("PhotoEditor - Survey Results")
wid=new_window.winfo_screenwidth()
hgt=new_window.winfo_screenheight()
new_window.configure(width=wid,height=hgt,bg='black')

fig, ax = plt.subplots()
ax.axis('equal')
# Update pie chart on the Tkinter window
canvas=FigureCanvasTkAgg(fig, master=new_window)
canvas.get_tk_widget().config(width=wid/1.5,height=hgt/1.5)
canvas.get_tk_widget().place(x=100,y=100)

qno=1
generate_piechart()
next_button=tk.Button(new_window,text="Next
Question",width=20,height=2,command=generate_piechart,bg='yellow')
next_button.place(x=wid/1.2,y=hgt/2.5)



new_window.mainloop()

# Create main window
window = tk.Tk()
window.title("Photo Editor - Survey Page")



wid=window.winfo_screenwidth()
hgt=window.winfo_screenheight()
window.geometry('{}x{}'.format(wid,hgt))
window.configure(bg='black')

f1=tk.Frame(window,width=wid//1.2,height=hgt//1.5)
fwth =f1.winfo_screenwidth()
fht = f1.winfo_screenheight()
```

```
f1.place(x=(wid//3.5)-(fwth//4),y=(hgt//2.5)-(fht//3))
f1.configure(bg='cyan')

style = ttk.Style()
style.configure("TRadiobutton", font=('Helvetica', 14))

questions=["1. How frequently do you use a photo editor to edit photos?","2.
How was your experience of using this app?","3. Which set of functionalities
did you find more useful?","4. Would you recommend this app to a friend or a
relative?","5. Which one of the following functionalities would you suggest to
add to the editor?"]

# Question 1
q1_label = ttk.Label(f1, text=questions[0],font=('Arial',15,'bold'))
q1_label.place(x=50,y=20)

q1_var = tk.StringVar()
q1_options = ["Always", "Sometimes", "Rarely", "Not at all"]
for i, option in enumerate(q1_options):
    rb=ttk.Radiobutton(f1,text=str(i + 1) + ". " + option, variable=q1_var,
value=option)
    rb['style']='TRadiobutton'
    rb.place(x=((i==0 and 50) or 200*i) ,y=50)

# Question 2
q2_label = ttk.Label(f1, text=questions[1],font=('Arial',15,'bold'))
q2_label.place(x=50,y=100)

q2_var = tk.StringVar()
```

```
q2_options = ["Excellent", "Good", "Average", "Worst Experience"]
for i,option in enumerate(q2_options):
    ttk.Radiobutton(f1, text=str(i + 1) + ". " + option, variable=q2_var,
value=option).place(x=((i==0 and 50) or 200*i),y=150)

# Question 3
q3_label = ttk.Label(f1, text=questions[2],font=('Arial',15,'bold'))
q3_label.place(x=50,y=200)

q3_var = tk.StringVar()
q3_options = ["Rotate and Mirror", "Brightness and Contrast", "Colour
Inversion", "Greyscale"]
for i, option in enumerate(q3_options):
    ttk.Radiobutton(f1, text=str(i + 1) + ". " + option, variable=q3_var,
value=option).place(x=((i==0 and 50) or 300*i),y=250)

# Question 4
q4_label = ttk.Label(f1, text=questions[3],font=('Arial',15,'bold'))
q4_label.place(x=50,y=300)

q4_var = tk.StringVar()
q4_options = ["Yes", "No", "Maybe", "Not sure"]
for i, option in enumerate(q4_options):
    ttk.Radiobutton(f1, text=str(i + 1) + ". " + option, variable=q4_var,
value=option).place(x=((i==0 and 50) or 200*i),y=350)

# Question 5
q5_label = ttk.Label(f1, text=questions[4],font=('Arial',15,'bold'))
q5_label.place(x=50,y=400)
```

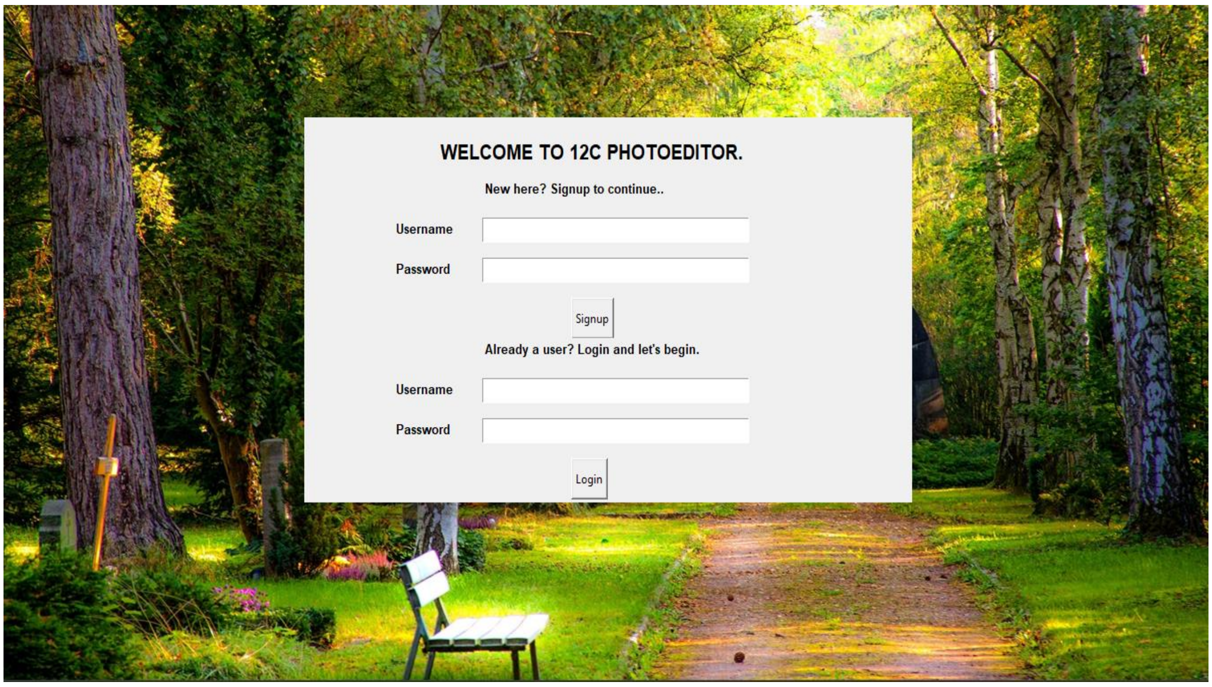
```
q5_var = tk.StringVar()
q5_options = ["Crop", "Resize", "Draw","A better interface"]
for i, option in enumerate(q5_options):
    ttk.Radiobutton(f1, text=str(i + 1) + ". " + option, variable=q5_var,
value=option).place(x=((i==0 and 50) or 200*i),y=450)



# Submit Button
submit_button = tk.Button(window,
text="Submit",width=20,height=2,command=submit_survey)
submit_button.place(x=wid//4.5,y=hgt//1.3)

view_results_button = tk.Button(window,text="View
Results",width=20,height=2,command=update_details)
view_results_button.place(x=wid//2.5,y=hgt//1.3)



# Run the Tkinter event loop
window.mainloop()
```

## APPLICATION FLOW:

### Login.py

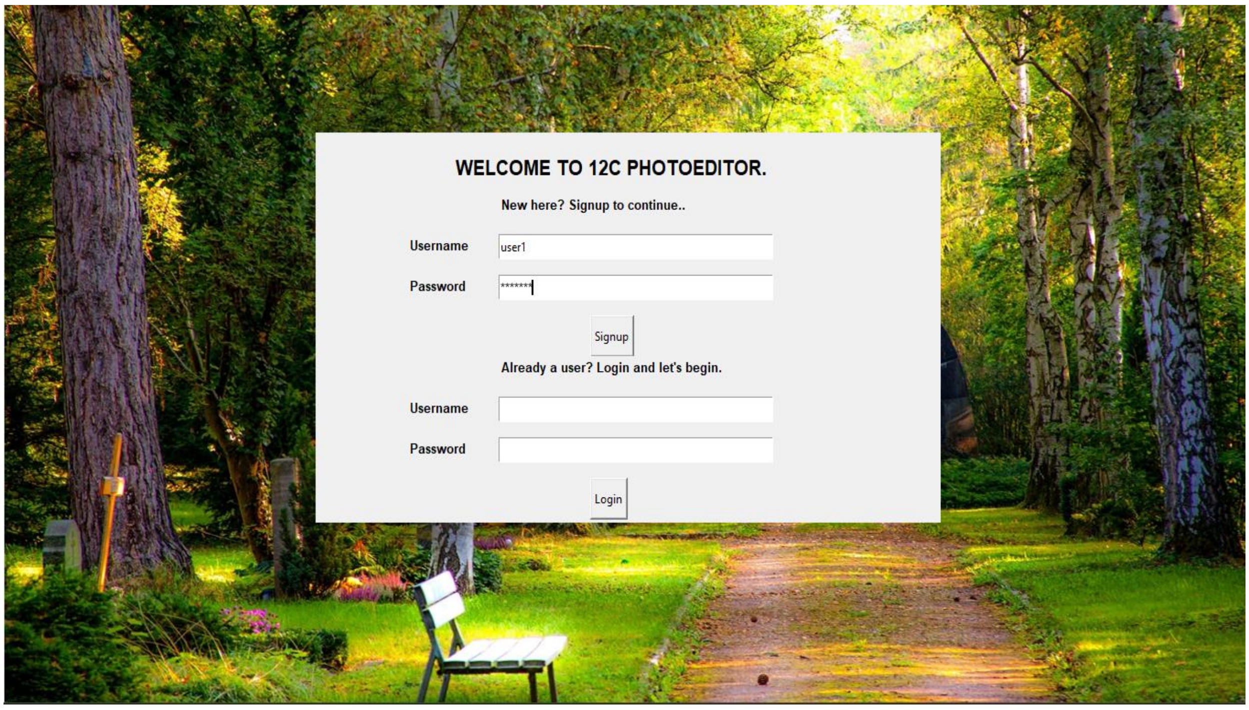
- Connection to database:

The username and password for connection to database asked from user.


WELCOME TO 12C PHOTOEDITOR.
New here? Signup to continue..
Username
Password
Signup
Already a user? Login and let's begin.
Username
Password
Login

Registering:

Users can signup through entryboxes provided below Signup label.


WELCOME TO 12C PHOTOEDITOR.

New here? Signup to continue..

Username user1

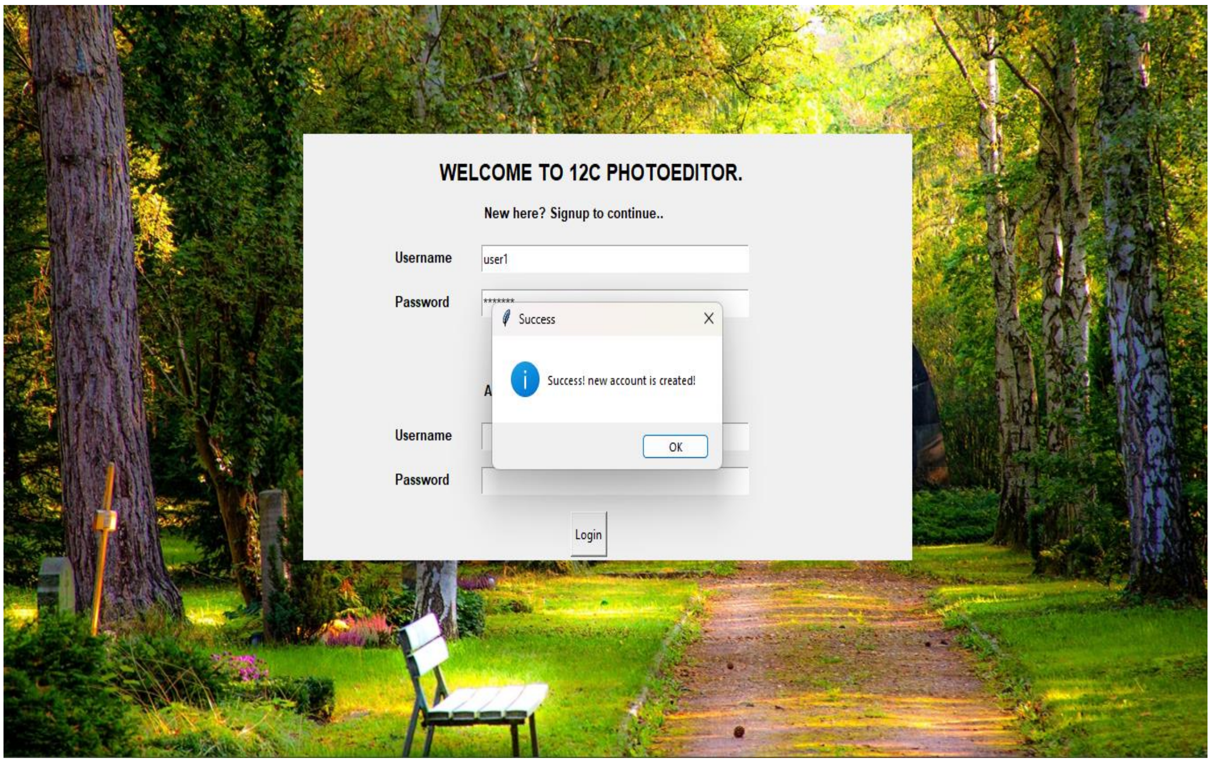
Password *******

Already a user? Login and let's begin.

Username

Password

Success messagebox is displayed after creation of new account and addition of user account details into the database 12ceditor.


WELCOME TO 12C PHOTOEDITOR.

New here? Signup to continue..

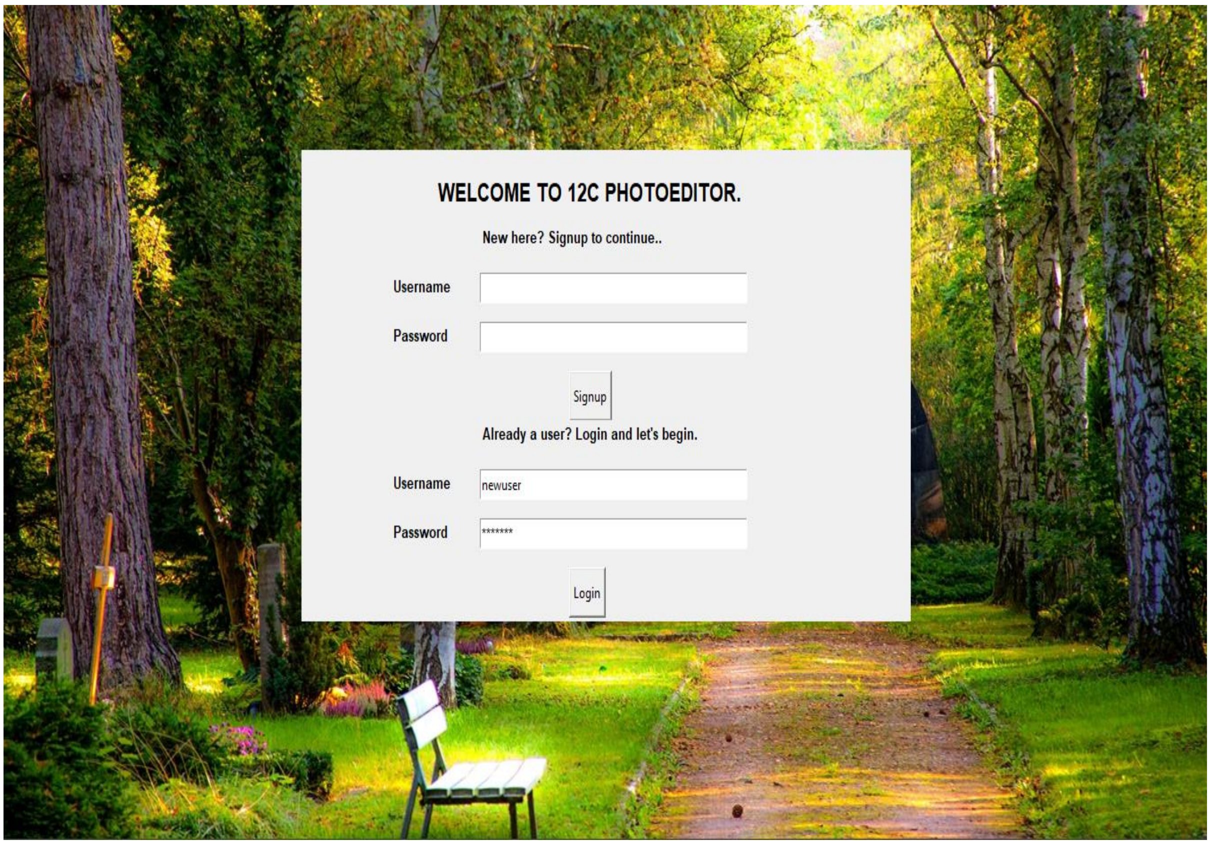
Username user1

Password *******

Success

Success! new account is created!

Login:



WELCOME TO 12C PHOTOEDITOR.

New here? Signup to continue..

Username

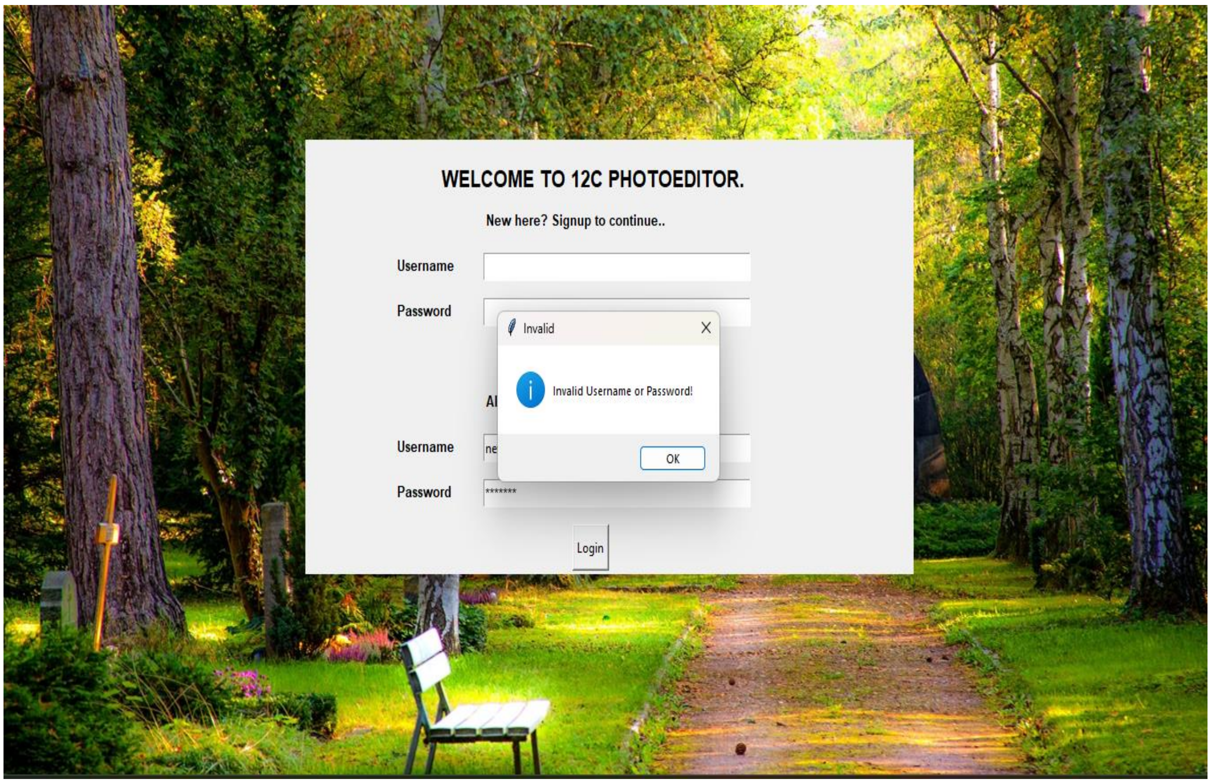
Password

Already a user? Login and let's begin.

Username newuser

Password *******

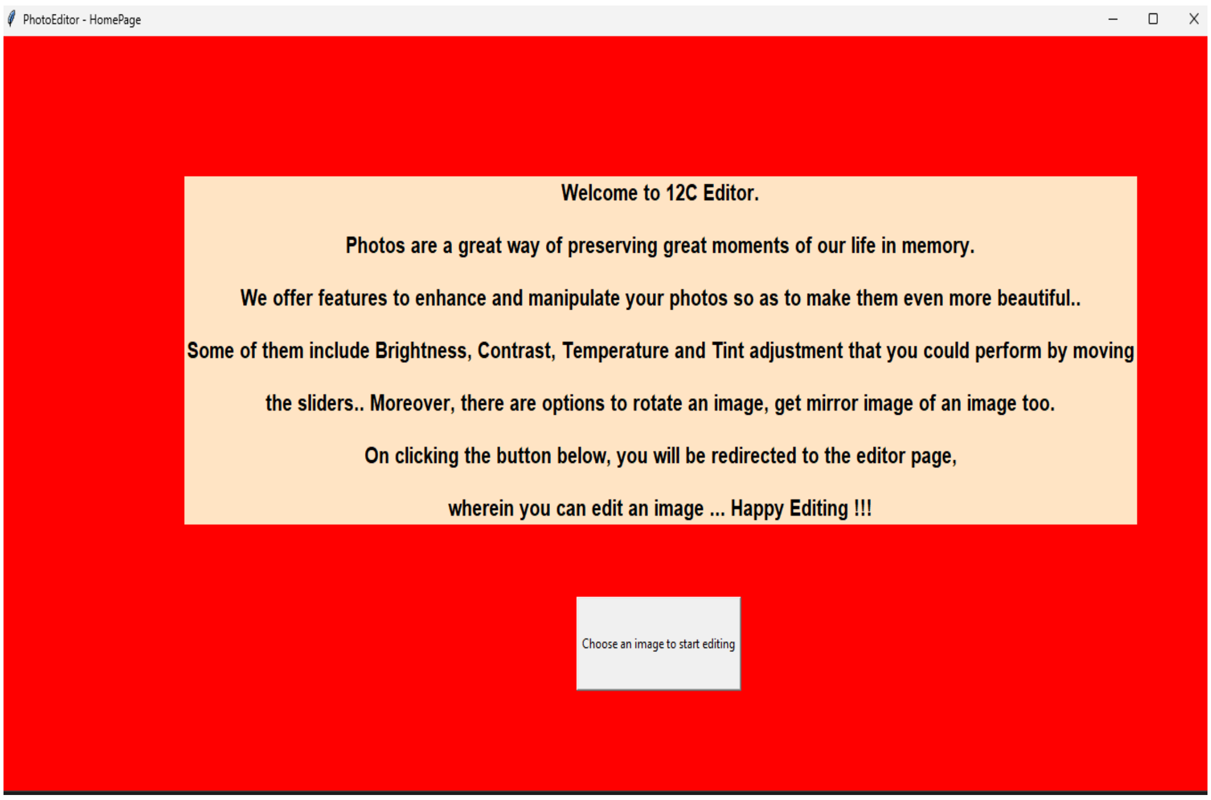
In case the user enters invalid credentials, a mesagebox indicating the same is displayed on the screen.



Invalid Username or Password!

## HomePage window:

HomePage window is designed to welcome users.



PhotoEditor - HomePage

Welcome to 12C Editor.

Photos are a great way of preserving great moments of our life in memory.

We offer features to enhance and manipulate your photos so as to make them even more beautiful..

Some of them include Brightness, Contrast, Temperature and Tint adjustment that you could perform by moving

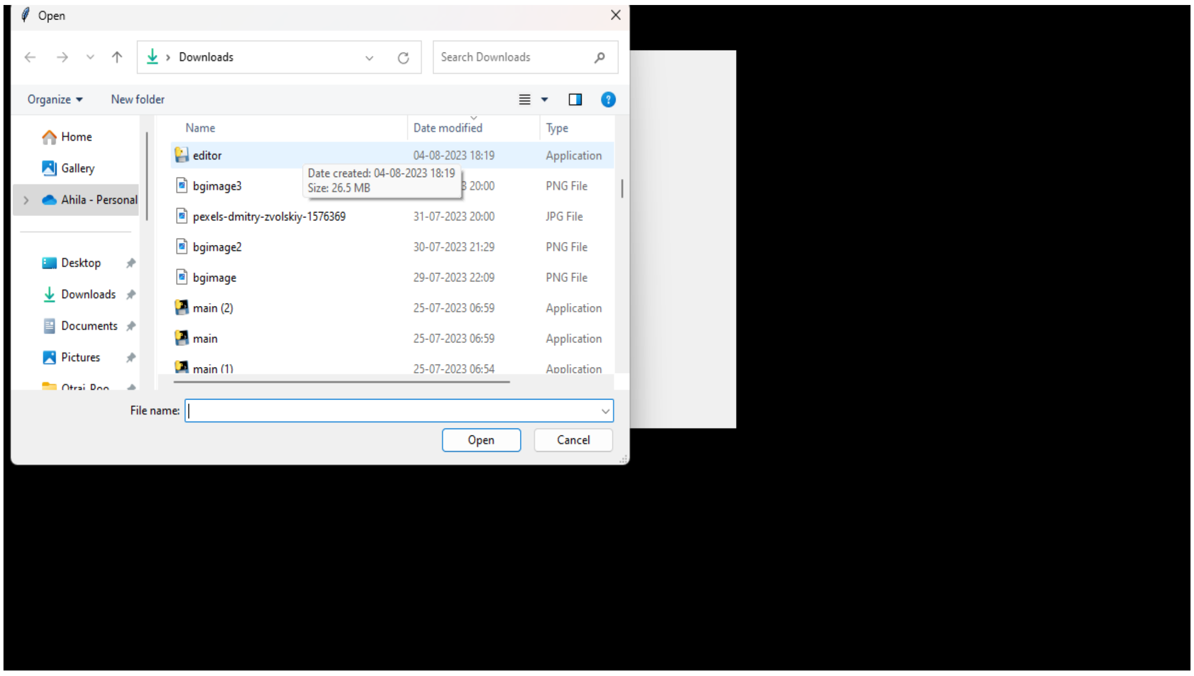
the sliders.. Moreover, there are options to rotate an image, get mirror image of an image too.

On clicking the button below, you will be redirected to the editor page,

wherein you can edit an image ... Happy Editing !!!

## Editor.py:

A new open dialog box appears using which the user can navigate his files to select an image for editing:

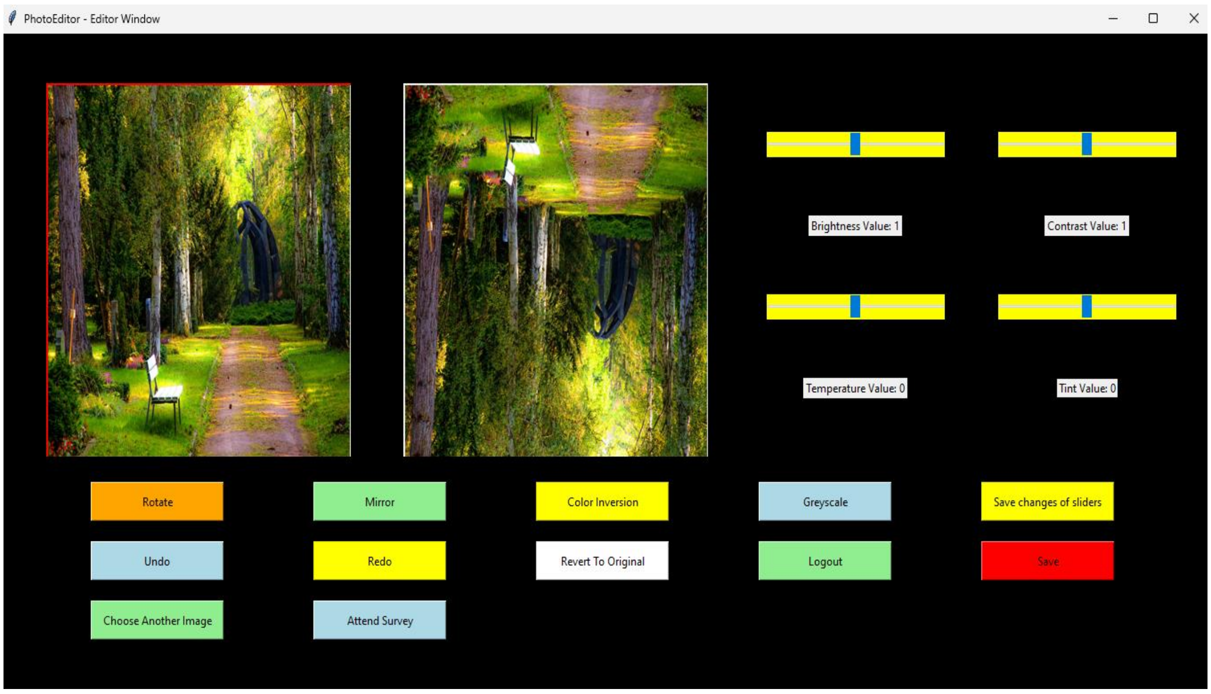


| Name | Date modified | Type |
|---|---|---|
| editor | 04-08-2023 18:19 | Application |
| bgimage3 | 3 20:00 | PNG File |
| pexels-dmitry-zvolskiy-1576369 | 31-07-2023 20:00 | JPG File |
| bgimage2 | 30-07-2023 21:29 | PNG File |
| bgimage | 29-07-2023 22:09 | PNG File |
| main (2) | 25-07-2023 06:59 | Application |
| main | 25-07-2023 06:59 | Application |
| main (1) | 25-07-2023 06:54 | Application |

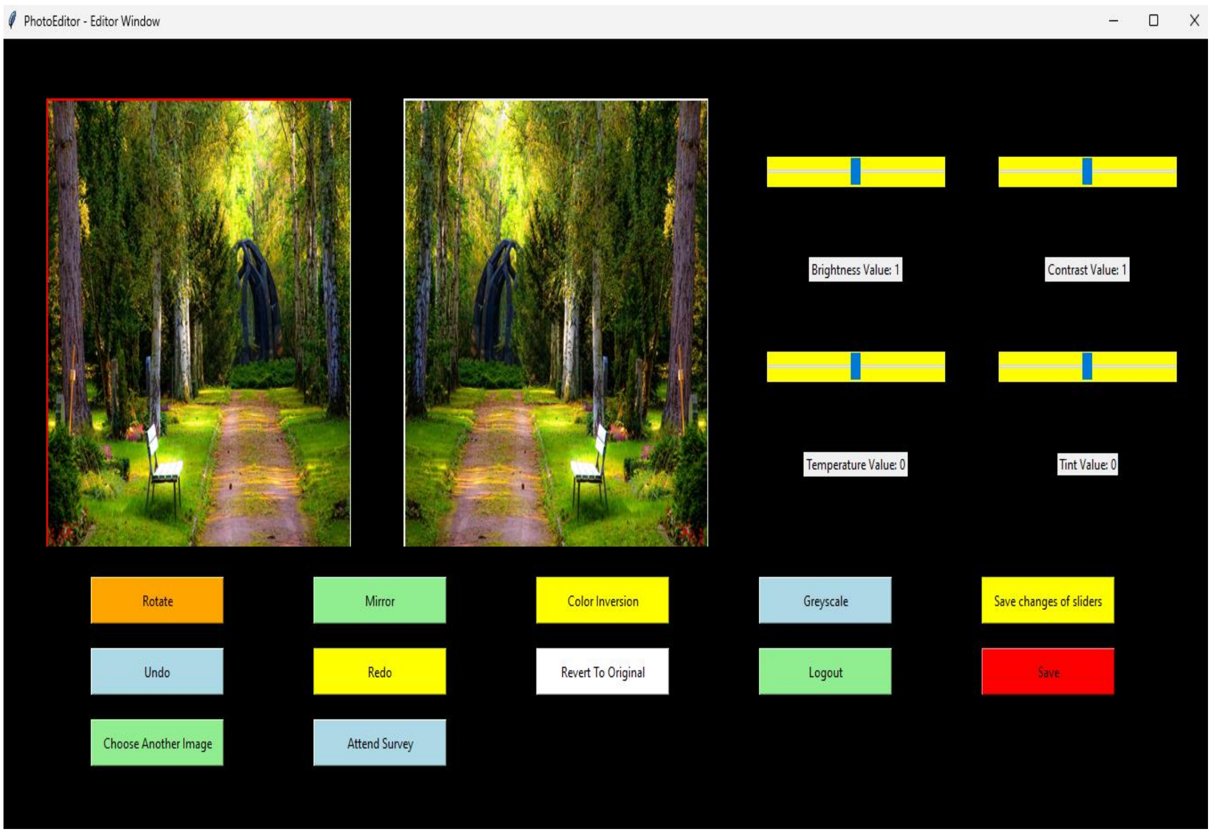
Date created: 04-08-2023 18:19
Size: 26.5 MB

Demonstration of functions:

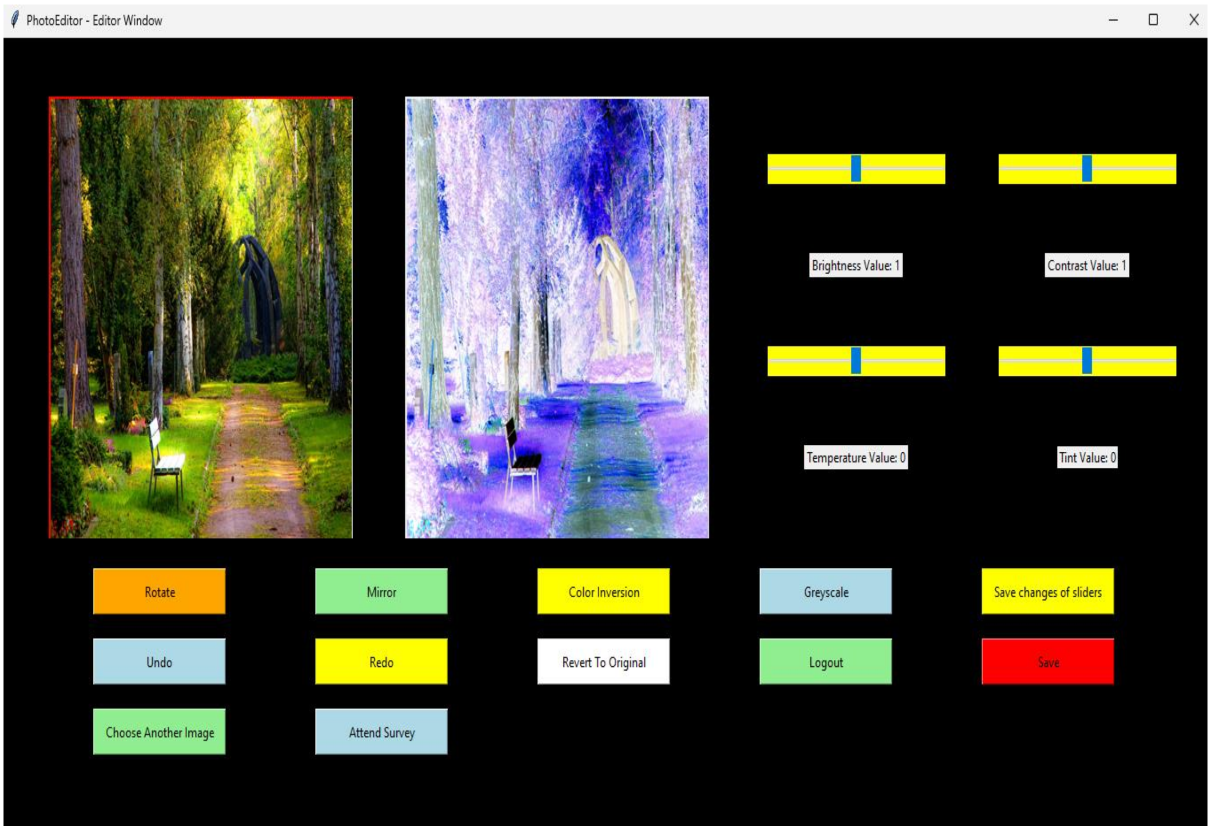
1. Rotating an image using Rotate button:



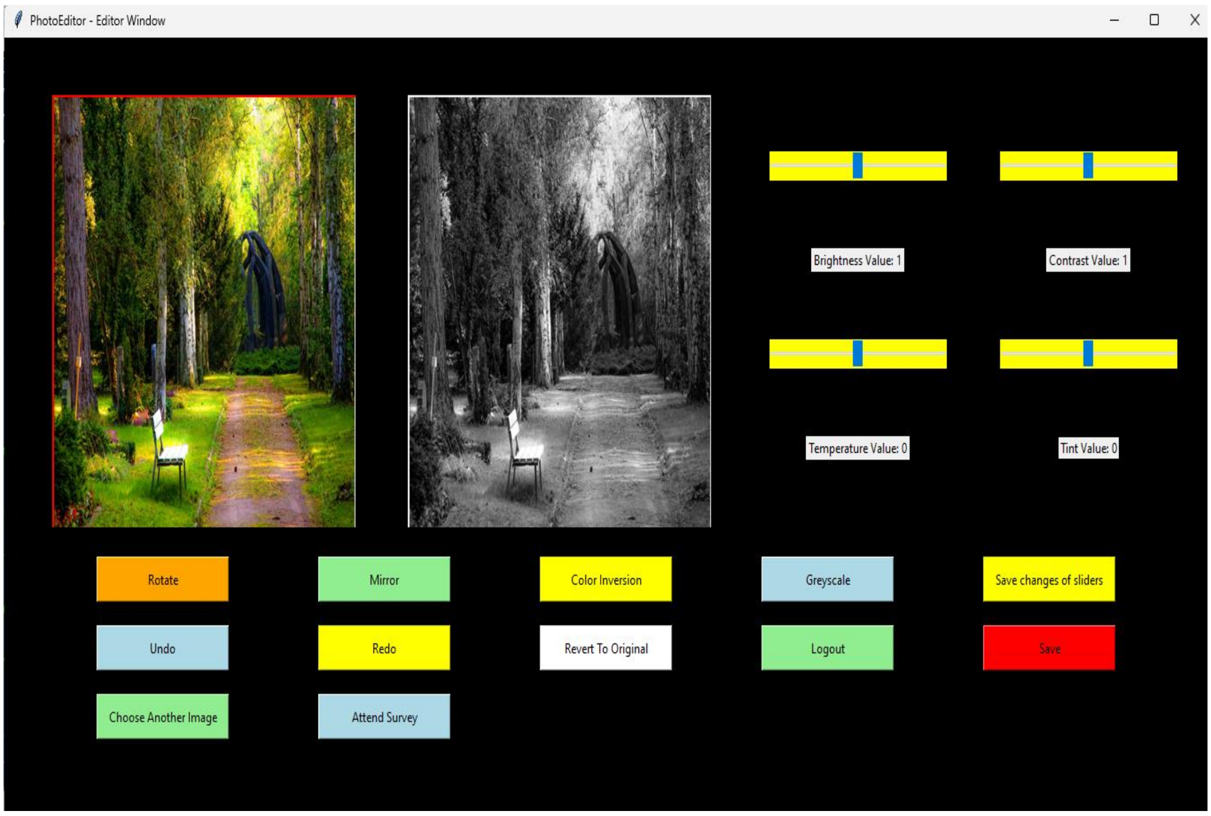
2. Getting mirror image of an image using Mirror button:



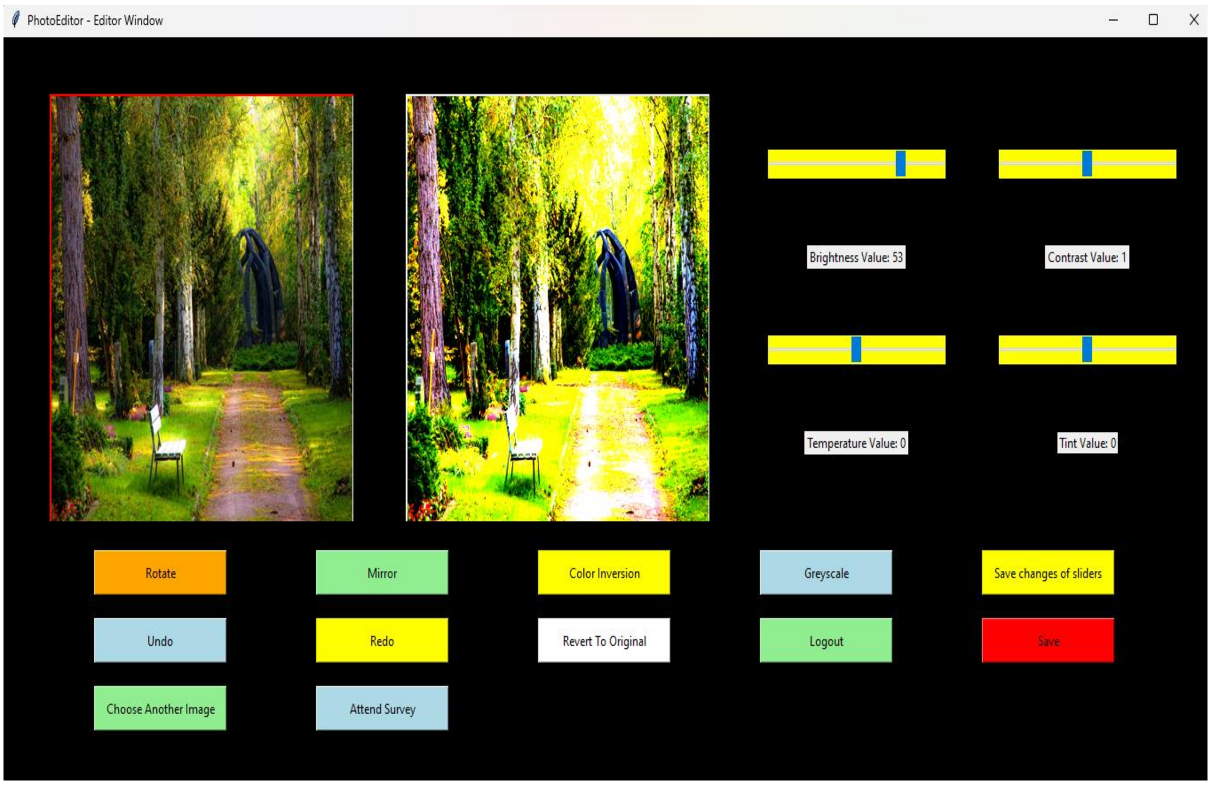
3. Colour Inversion button to invert colours of an image:



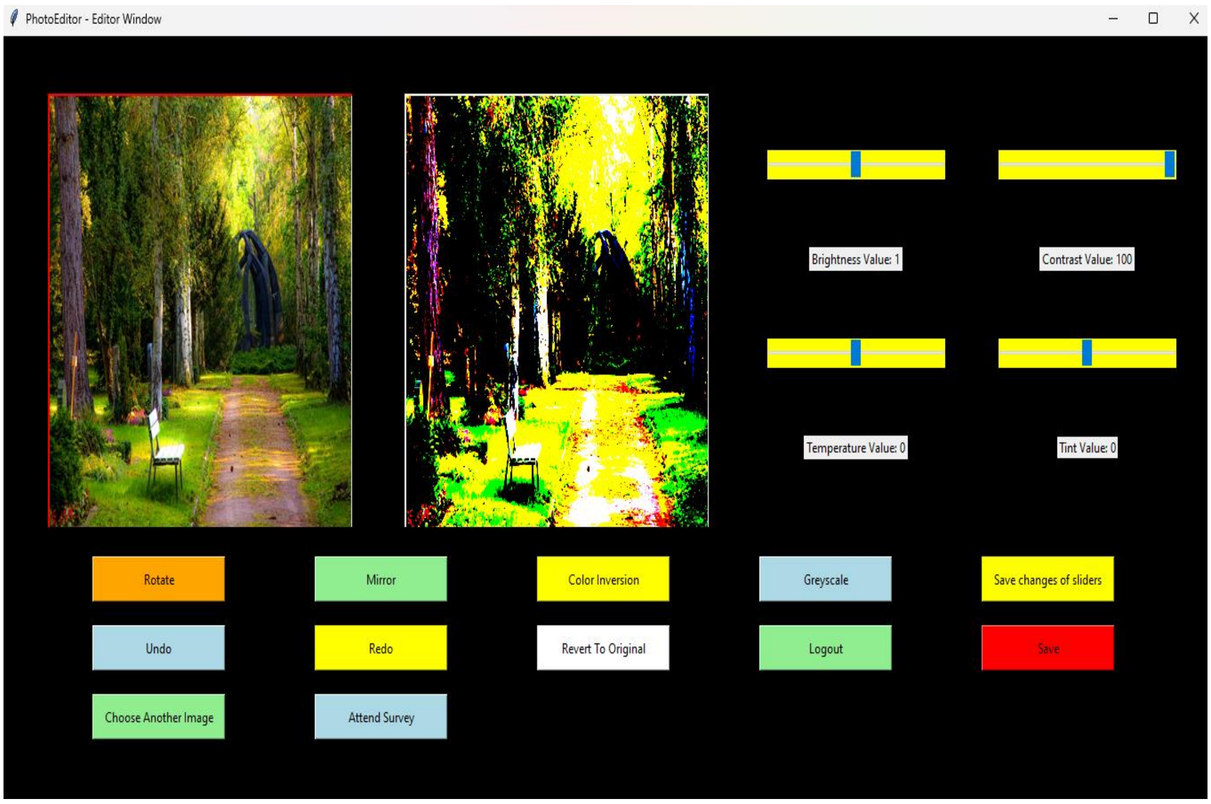
4. Greyscale button to convert an image to it s greyscale form: similar to black and white



5. Brightness slider to adjust brightness of an image with current value of brightness of image below the slider.

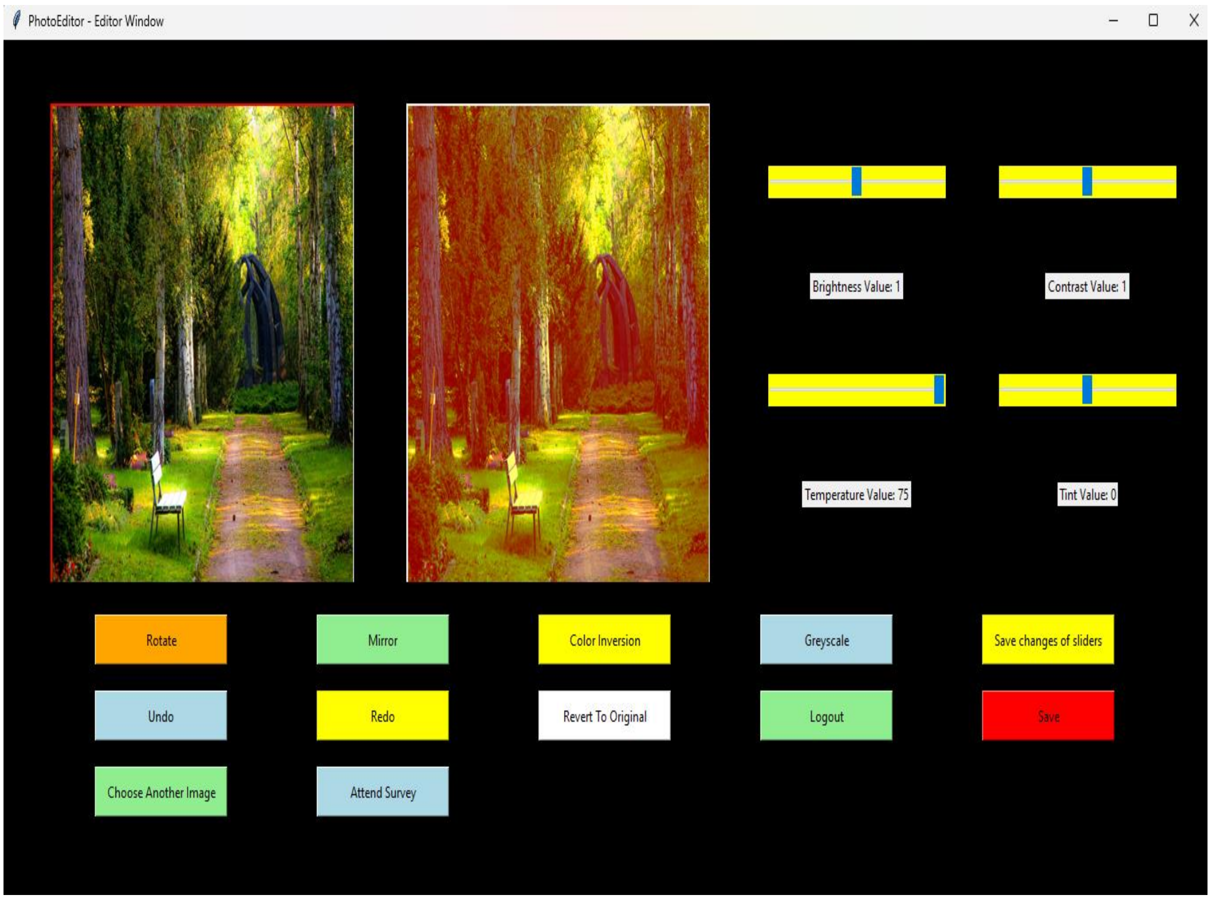


6. Contrast slider to adjust contrast of an image with current value of contrast of image below the slider.

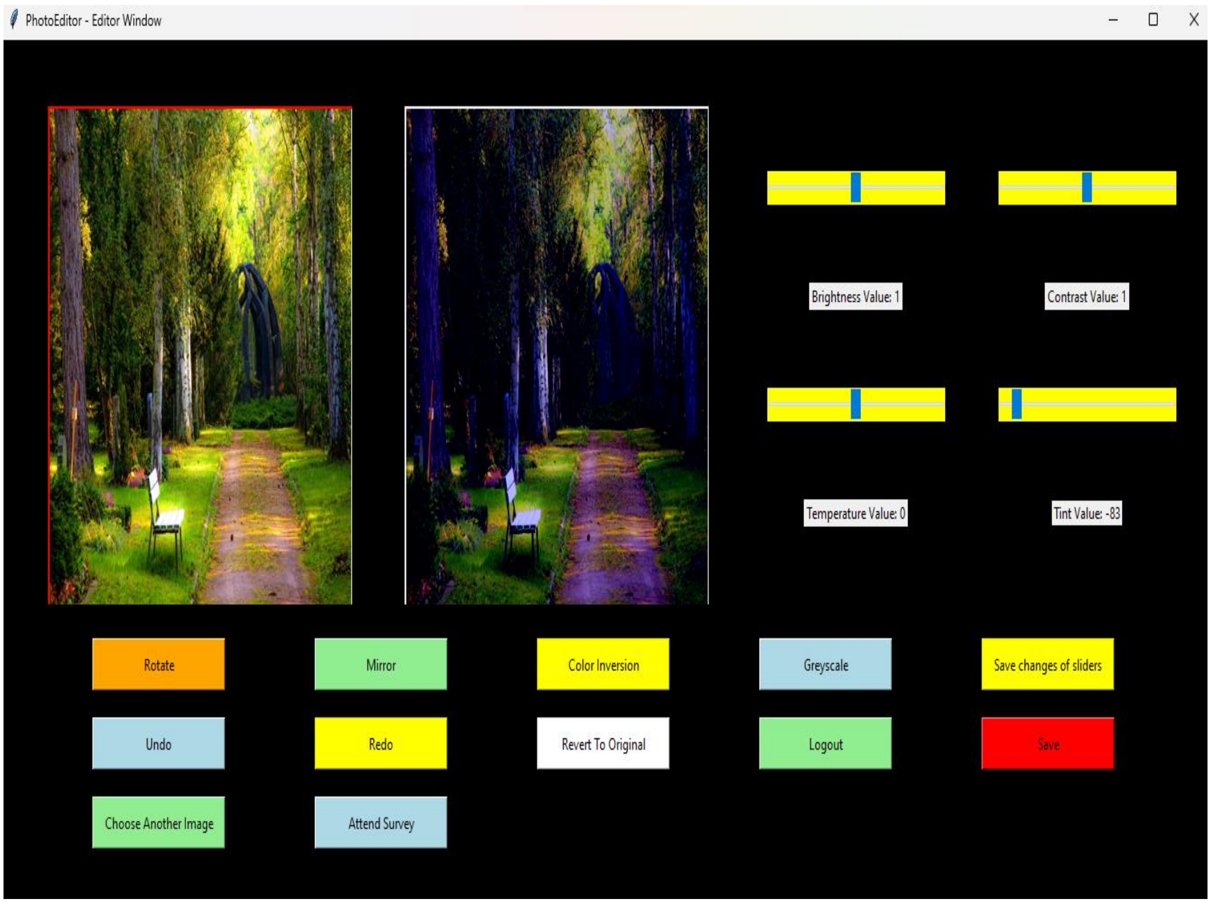


Brightness Value: 1
Contrast Value: 100
Temperature Value: 0
Tint Value: 0

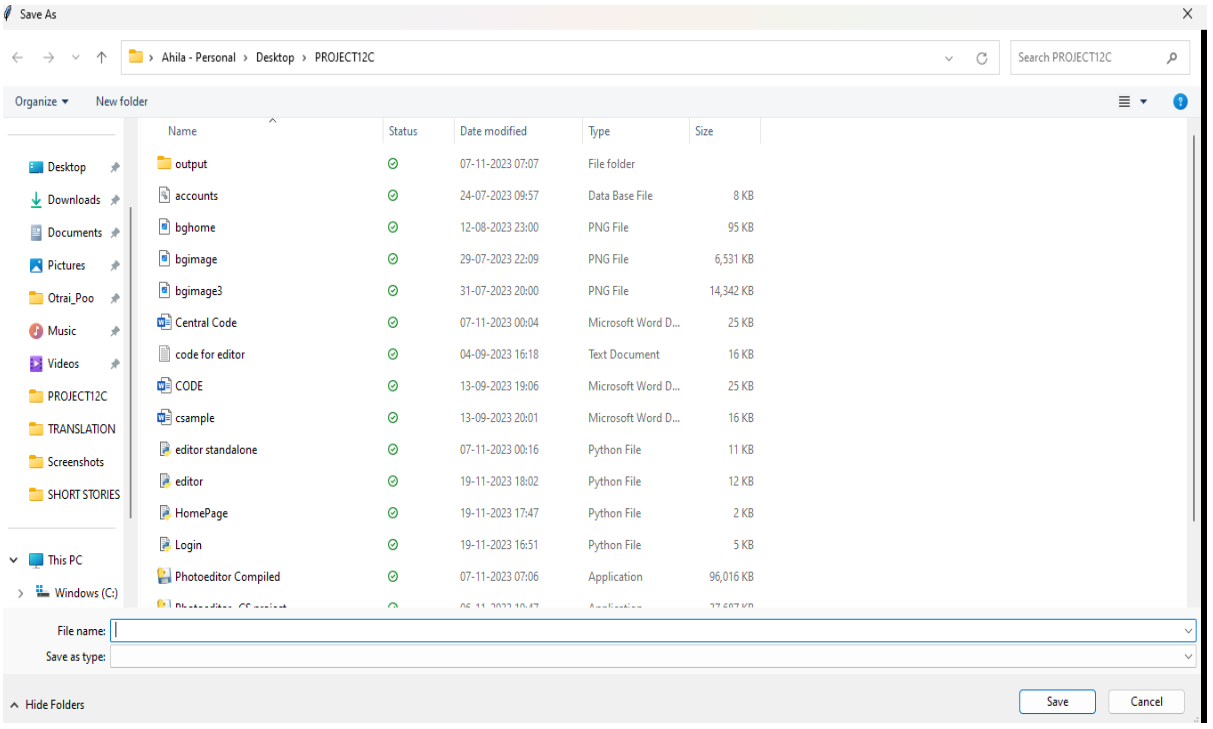
7. Temperature slider to adjust temperature of an image with current values of temperature of image indicated below the slider.



8. Tint slider to adjust tint of an image with current values of tint of image indicated below the slider.



9. Save button to save edited image in user's device: A new save dialog box is opened.


Save As

Ahila - Personal › Desktop › PROJECT12C

Search PROJECT12C

Organize ▾ New folder

Desktop
Downloads
Documents
Pictures
Otrai_Poo
Music
Videos
PROJECT12C
TRANSLATION
Screenshots
SHORT STORIES
This PC
Windows (C:)

| Name | Status | Date modified | Type | Size |
|---|---|---|---|---|
| output | ✓ | 07-11-2023 07:07 | File folder | |
| accounts | ✓ | 24-07-2023 09:57 | Data Base File | 8 KB |
| bghome | ✓ | 12-08-2023 23:00 | PNG File | 95 KB |
| bgimage | ✓ | 29-07-2023 22:09 | PNG File | 6,531 KB |
| bgimage3 | ✓ | 31-07-2023 20:00 | PNG File | 14,342 KB |
| Central Code | ✓ | 07-11-2023 00:04 | Microsoft Word D... | 25 KB |
| code for editor | ✓ | 04-09-2023 16:18 | Text Document | 16 KB |
| CODE | ✓ | 13-09-2023 19:06 | Microsoft Word D... | 25 KB |
| csample | ✓ | 13-09-2023 20:01 | Microsoft Word D... | 16 KB |
| editor standalone | ✓ | 07-11-2023 00:16 | Python File | 11 KB |
| editor | ✓ | 19-11-2023 18:02 | Python File | 12 KB |
| HomePage | ✓ | 19-11-2023 17:47 | Python File | 2 KB |
| Login | ✓ | 19-11-2023 16:51 | Python File | 5 KB |
| Photoeditor Compiled | ✓ | 07-11-2023 07:06 | Application | 96,016 KB |

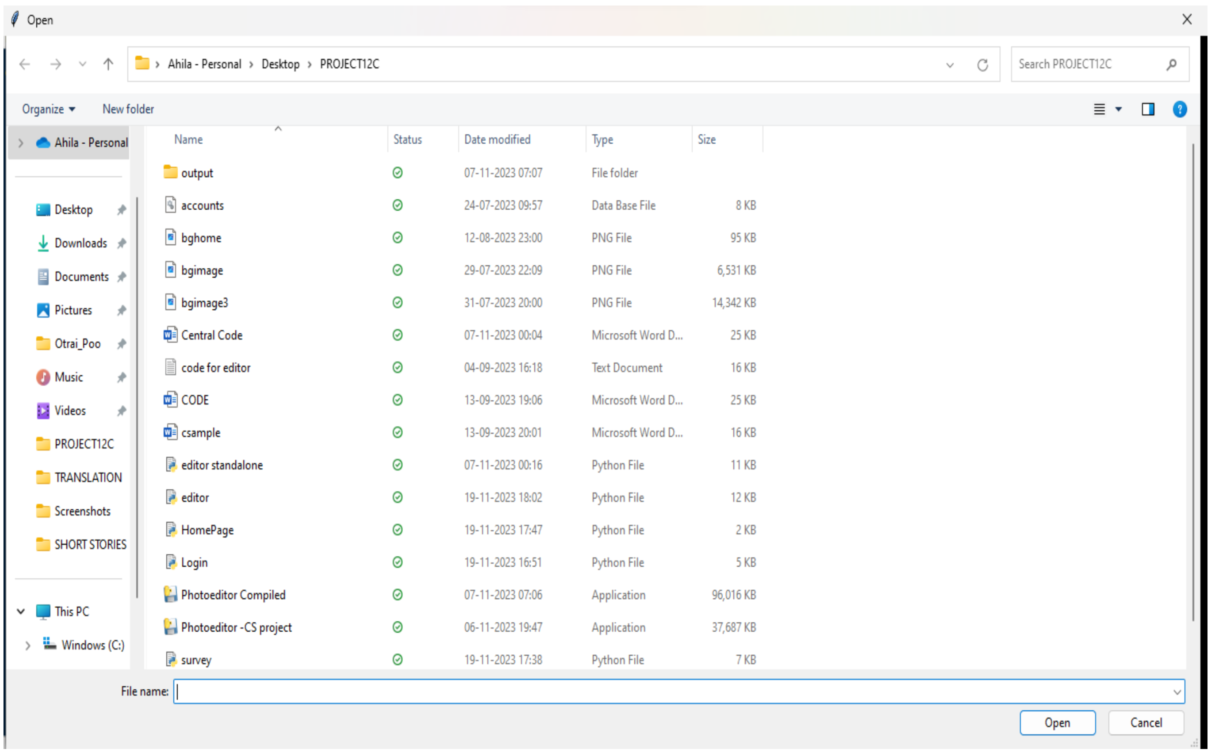
File name:

Save as type:

Hide Folders

Save Cancel

10. Choose an image button to choose new image to edit: It opens a new open dialog box.

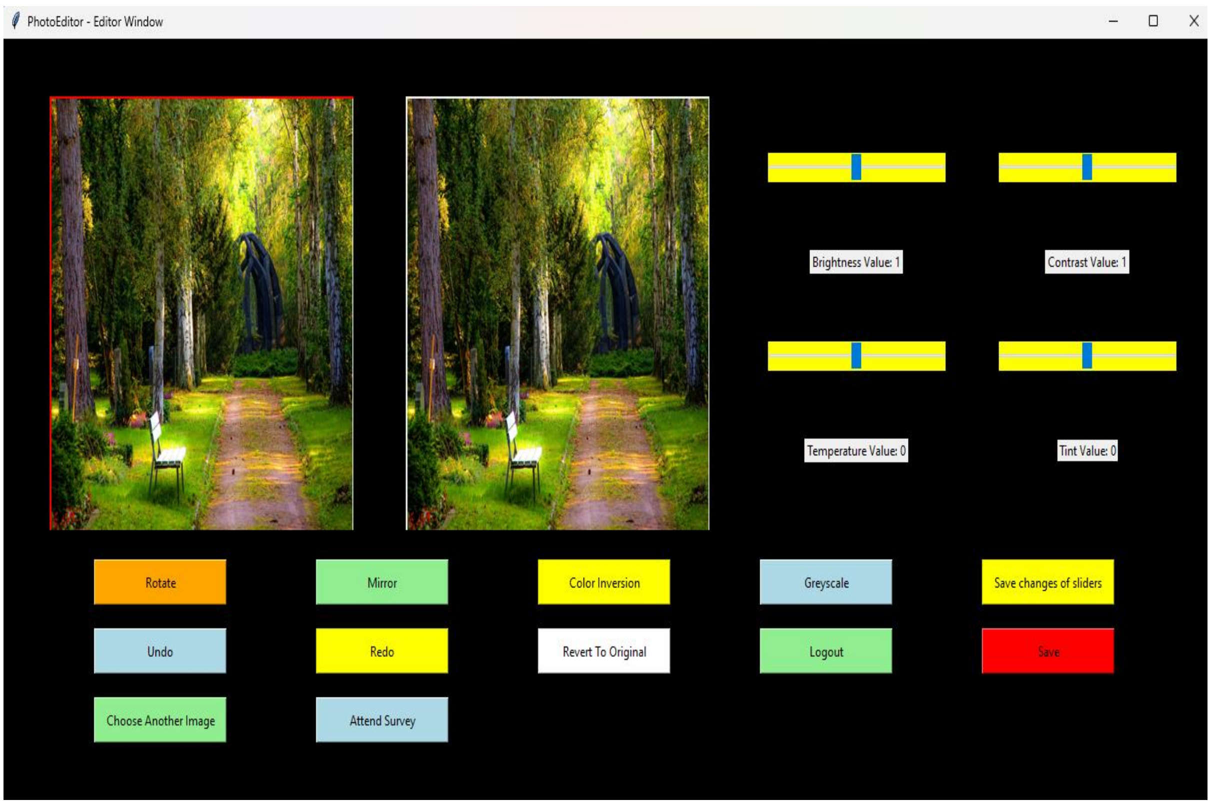


| Name | Status | Date modified | Type | Size |
|---|---|---|---|---|
| output | ✓ | 07-11-2023 07:07 | File folder | |
| accounts | ✓ | 24-07-2023 09:57 | Data Base File | 8 KB |
| bghome | ✓ | 12-08-2023 23:00 | PNG File | 95 KB |
| bgimage | ✓ | 29-07-2023 22:09 | PNG File | 6,531 KB |
| bgimage3 | ✓ | 31-07-2023 20:00 | PNG File | 14,342 KB |
| Central Code | ✓ | 07-11-2023 00:04 | Microsoft Word D... | 25 KB |
| code for editor | ✓ | 04-09-2023 16:18 | Text Document | 16 KB |
| CODE | ✓ | 13-09-2023 19:06 | Microsoft Word D... | 25 KB |
| csample | ✓ | 13-09-2023 20:01 | Microsoft Word D... | 16 KB |
| editor standalone | ✓ | 07-11-2023 00:16 | Python File | 11 KB |
| editor | ✓ | 19-11-2023 18:02 | Python File | 12 KB |
| HomePage | ✓ | 19-11-2023 17:47 | Python File | 2 KB |
| Login | ✓ | 19-11-2023 16:51 | Python File | 5 KB |
| Photoeditor Compiled | ✓ | 07-11-2023 07:06 | Application | 96,016 KB |
| Photoeditor -CS project | ✓ | 06-11-2023 19:47 | Application | 37,687 KB |
| survey | ✓ | 19-11-2023 17:38 | Python File | 7 KB |

**REPORT:**

A module named **survey.py** was created and incorporated with application.

A survey button is provided in the editor window to collect views and opinions of users about the application.



This is the survey window where users could enter their views through radiobuttons:

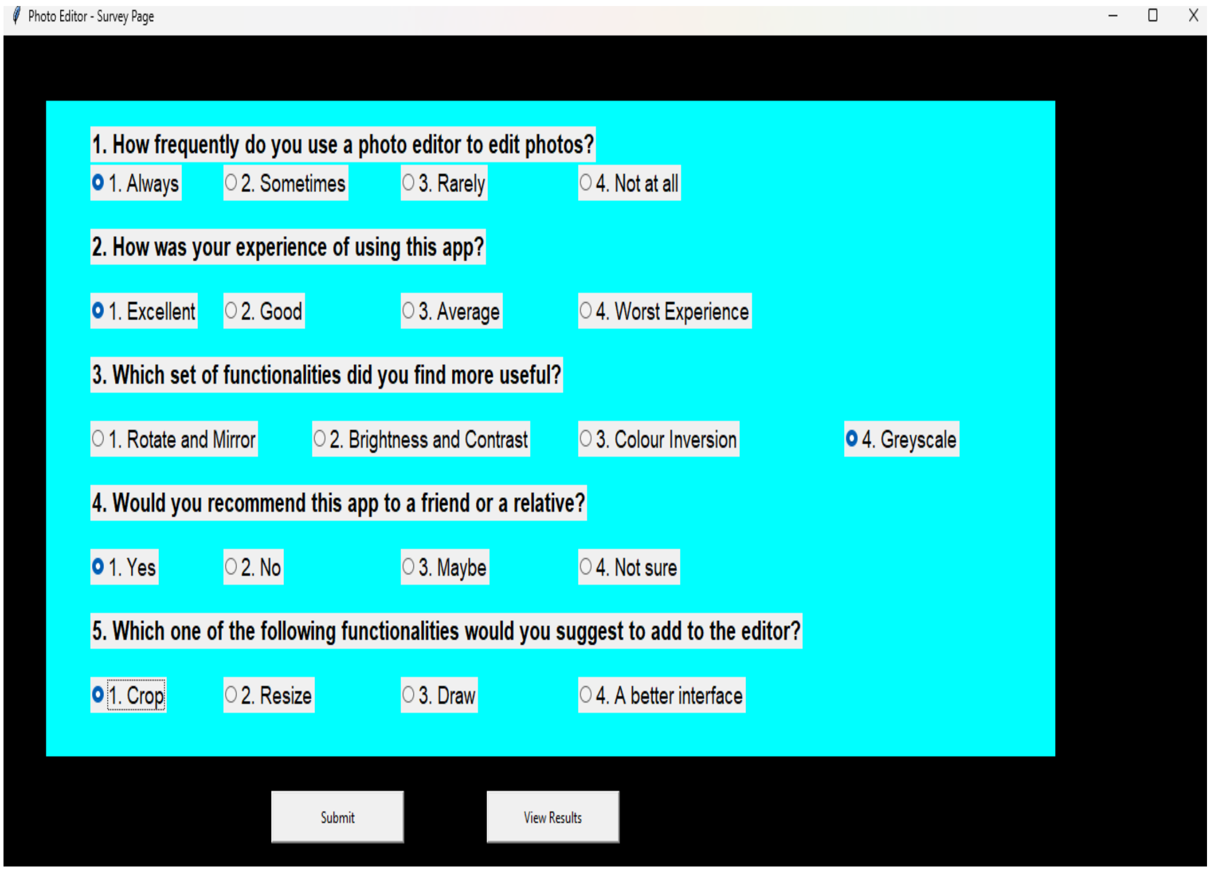


Photo Editor - Survey Page

1. How frequently do you use a photo editor to edit photos?

[✓ 1. Always] 2. Sometimes 3. Rarely 4. Not at all

2. How was your experience of using this app?

[✓ 1. Excellent] 2. Good 3. Average 4. Worst Experience

3. Which set of functionalities did you find more useful?

1. Rotate and Mirror 2. Brightness and Contrast 3. Colour Inversion [✓ 4. Greyscale]

4. Would you recommend this app to a friend or a relative?

[✓ 1. Yes] 2. No 3. Maybe 4. Not sure

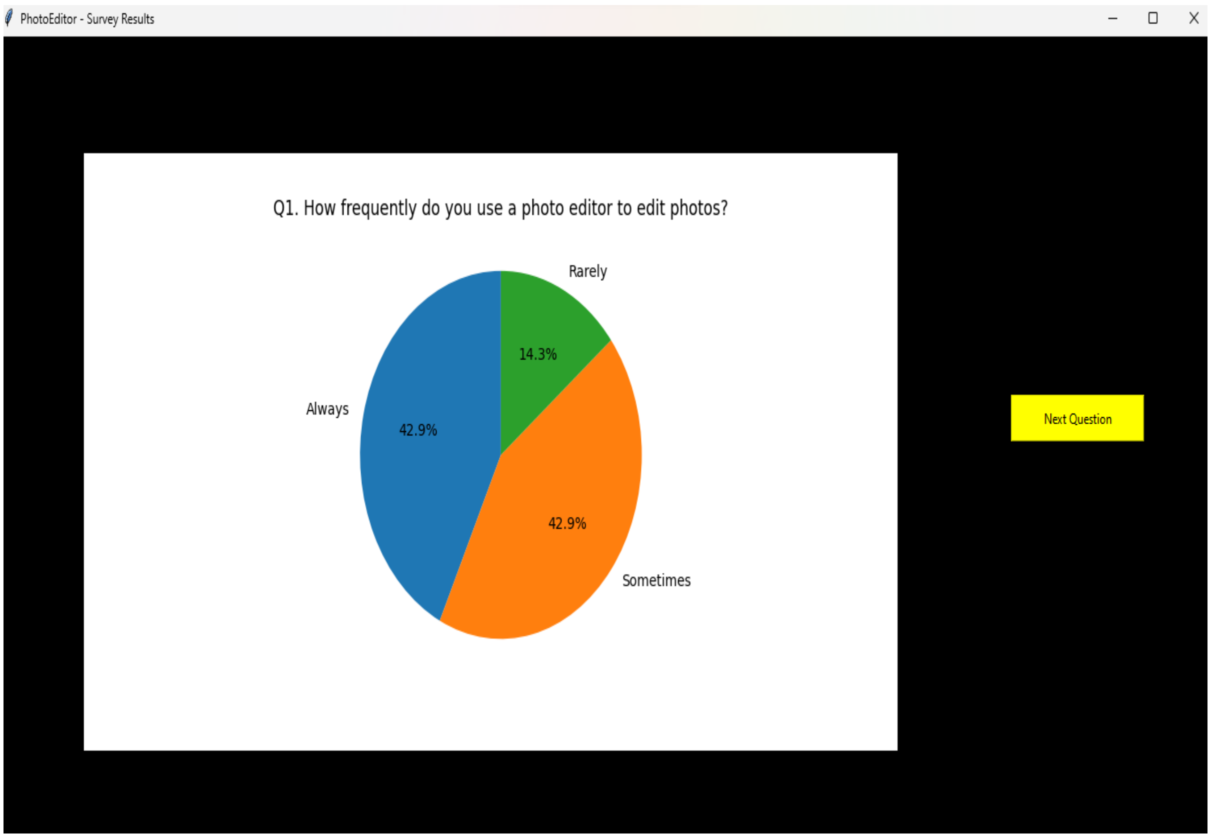
5. Which one of the following functionalities would you suggest to add to the editor?

[✓ 1. Crop] 2. Resize 3. Draw 4. A better interface

The collected responses are represented pictorially using pie charts:

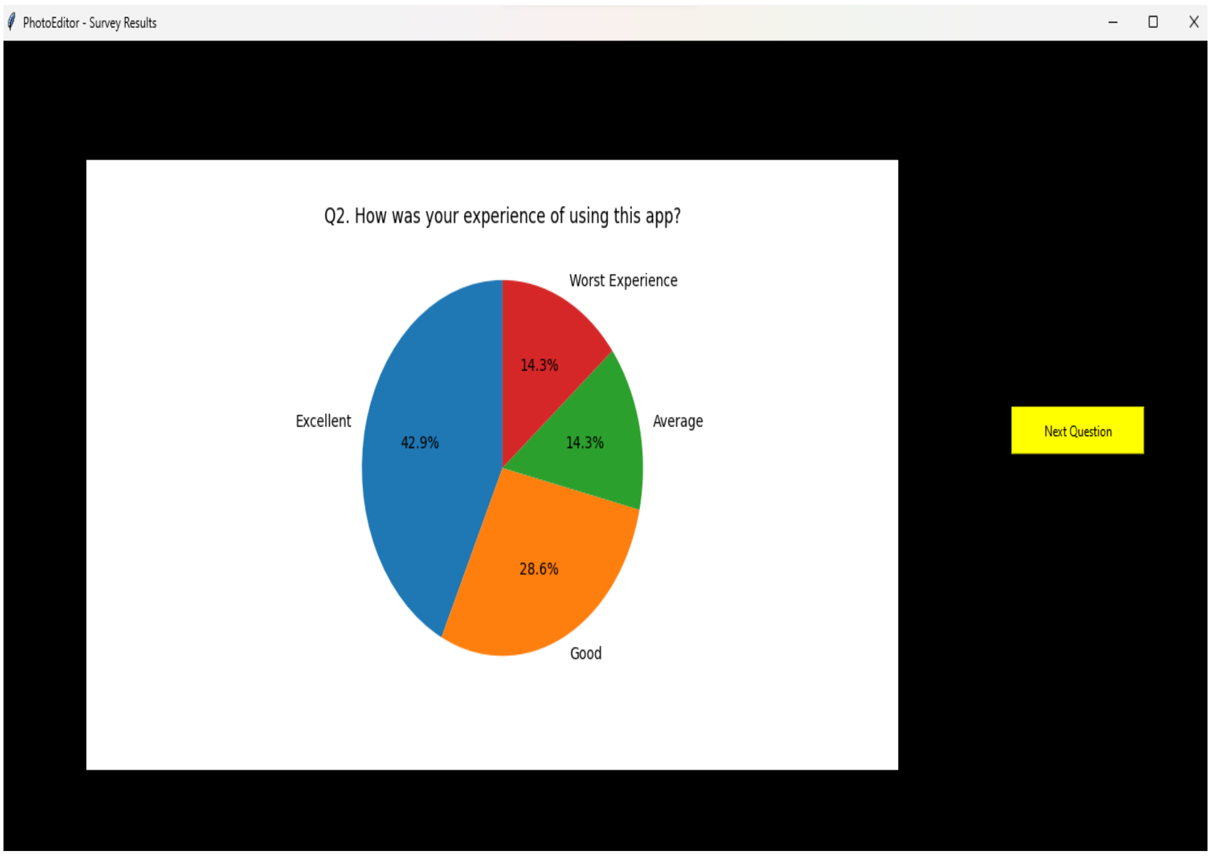
Question 1: How frequently do you use a photo editor to edit photos?

Options: Always, Sometimes, Rarely, Not at all


PhotoEditor - Survey Results
Q1. How frequently do you use a photo editor to edit photos?
Rarely
14.3%
Always
42.9%
42.9%
Sometimes
Next Question

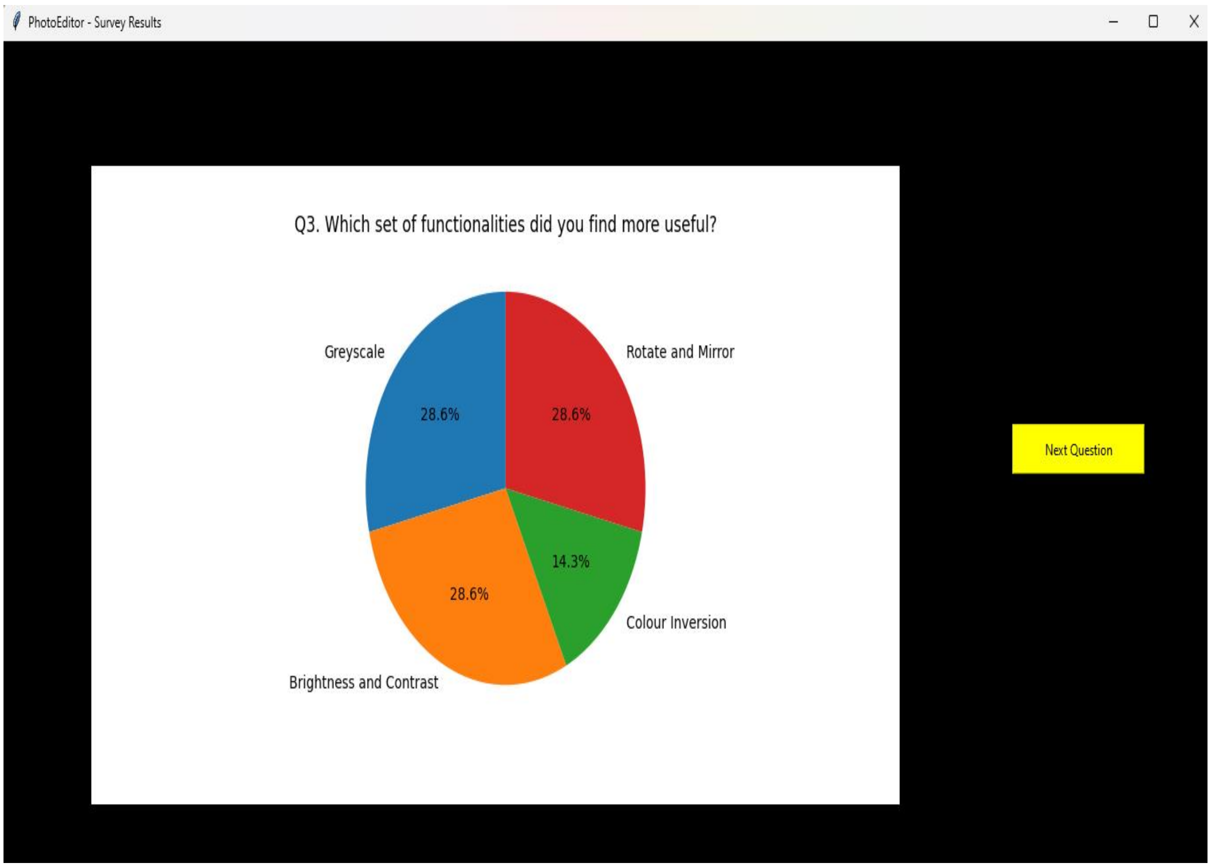
Question 2: How was your experience of using this app?

Options: Excellent, Good, Average, Worst Experience


PhotoEditor - Survey Results
Q2. How was your experience of using this app?
Worst Experience
14.3%
Excellent
42.9%
14.3%
Average
28.6%
Good
Next Question

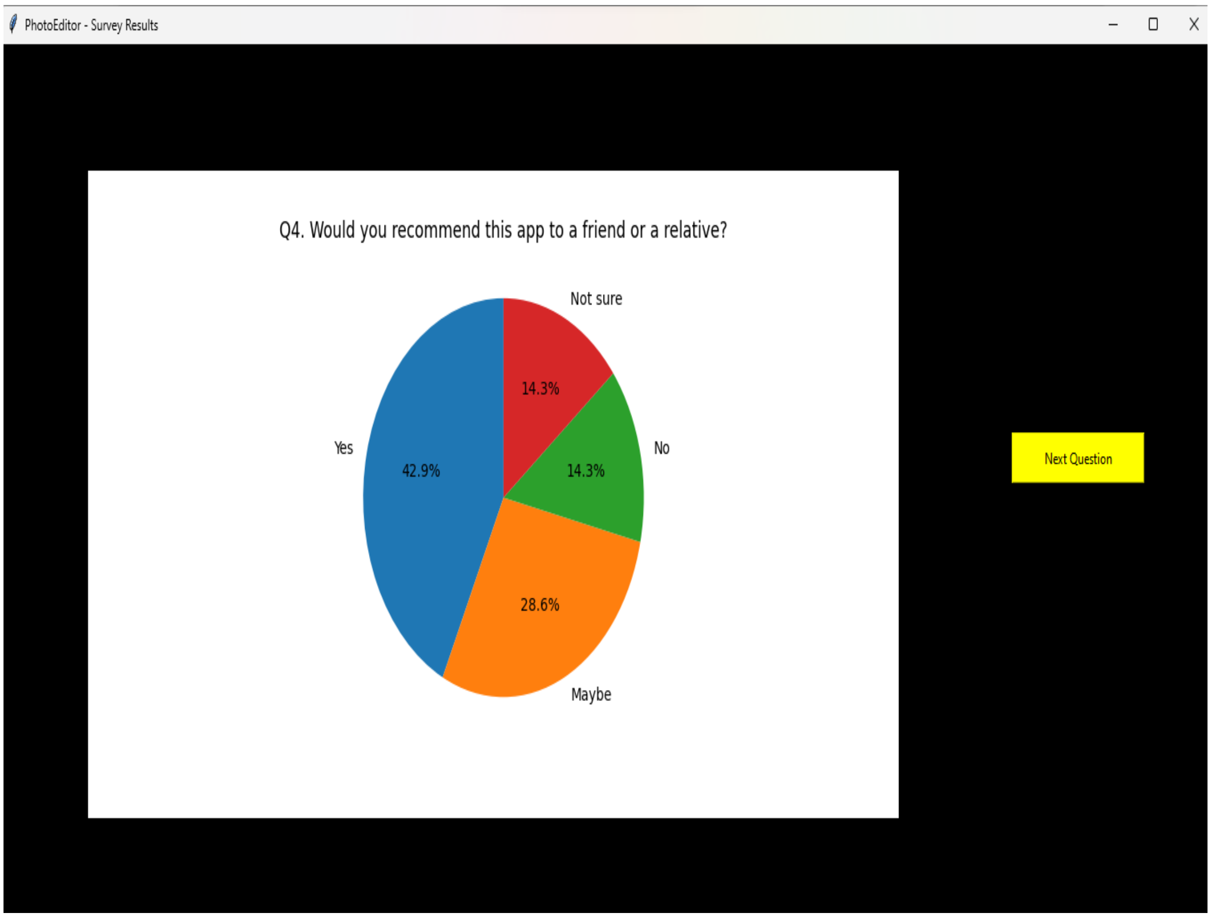
Question 3: Which set of functionalities did you find more useful?

Options: Rotate and Mirror, Brightness and Contrast, Colour Inversion, Greyscale



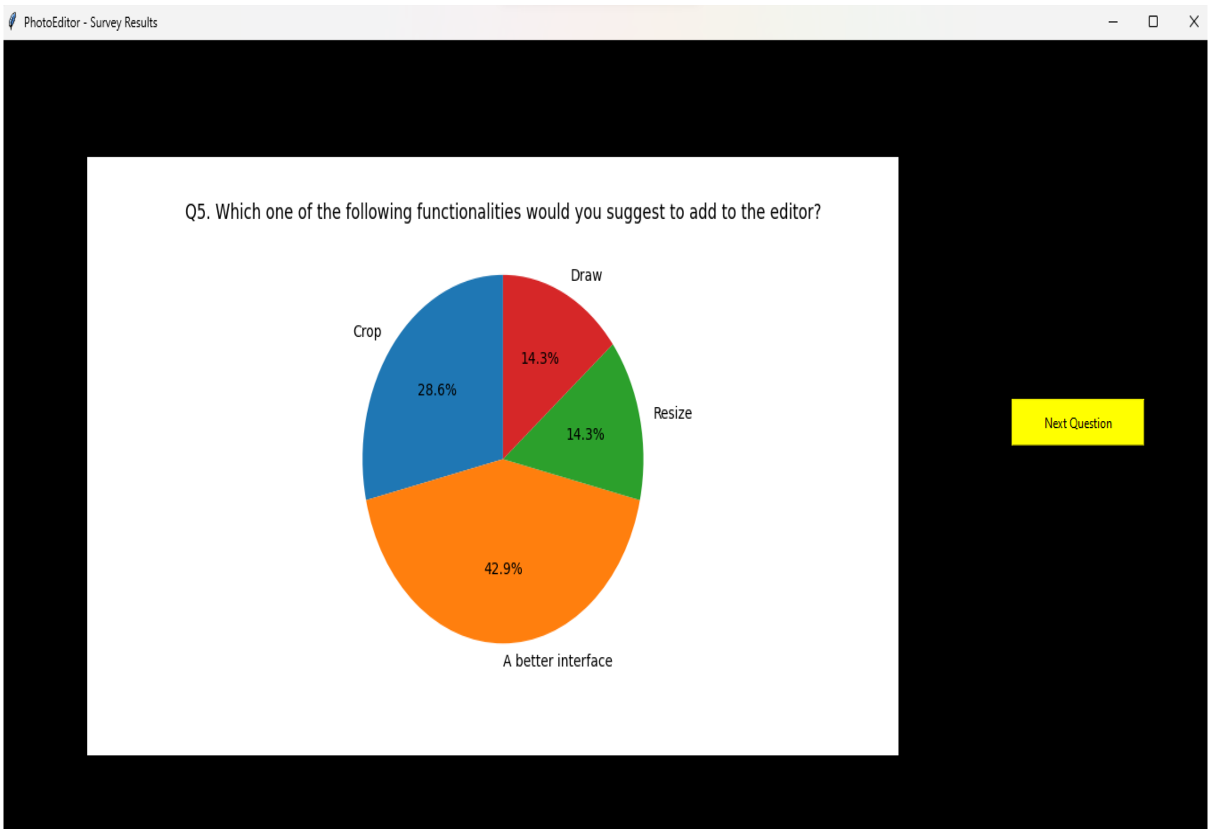
Question 4: Would you recommend this app to a friend or a relative?

Options: Yes, No, Maybe, Not sure



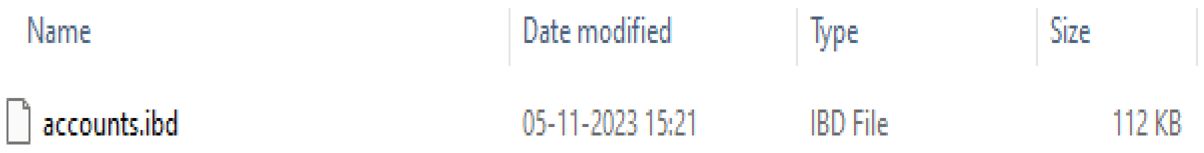
Question 5: Which one of the functionalities would you suggest to add to the editor?

Options: Crop, Resize, Draw, A better interface


PhotoEditor - Survey Results
Q5. Which one of the following functionalities would you suggest to add to the editor?
Draw
14.3%
Crop
28.6%
Resize
14.3%
42.9%
A better interface
Next Question

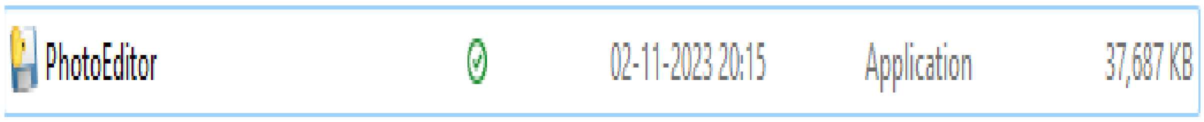
## FILES PART OF THIS PROJECT

accounts table in database ‘12ceditor’; filename: accounts.ibd



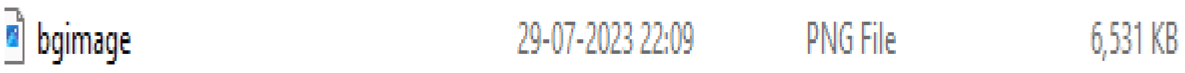
| Name | Date modified | Type | Size |
|---|---|---|---|
| accounts.ibd | 05-11-2023 15:21 | IBD File | 112 KB |

Executable file named PhotoEditor.exe



| | | | |
|---|---|---|---|
| PhotoEditor | 02-11-2023 20:15 | Application | 37,687 KB |

Background image of login page : bgimage.png



| | | | |
|---|---|---|---|
| bgimage | 29-07-2023 22:09 | PNG File | 6,531 KB |

## BIBLIOGRAPHY:

https://en.m.wikipedia.org

https://www.britannica.com

https://www.geeksforgeeks.com

https://www.w3schools.com

https://www.tutorialspoint.com

https://www.stackoverflow.com

In addition to these digital sources,

The book COMPUTER SCIENCE WITH PYTHON by SUMITA ARORA was also referred for this project.