```
import tkinter as tk
from tkinter import Label, PhotoImage
from datetime import datetime
from PIL import Image,ImageTk
import pygame
def create_main_app():
    window_page.destroy()
    global root
    root = tk.Tk()
    root.title("Switch Control Panel")
    root.geometry("1000x700")
    root.resizable(width='False',height='False')
    root.configure(bg="gray")
    pygame.mixer.init()
    switches = [False] * 8
    current_time = datetime.now().time()
    scheduled_commands = {}
    def get_time(time_str):
        try:
            time_value = datetime.strptime(time_str, '%H:%M')
            return time_value.time()
        except ValueError:
            return None
    def print_switch_state(switches, switch_number):
        return "Switch {}: {}".format(switch_number, 'ON' if switches[switch_number - 1] else 'OFF')
    def print_all_switch_states(switches):
        return '\n'.join("Switch {}: {}".format(i + 1, 'ON' if state else 'OFF') for i, state in
enumerate(switches))
    def process_all_on_command(switches):
        for i in range(len(switches)):
            switches[i] = True
```

```
    return "All switches turned ON."
def process_all_off_command(switches):
    for i in range(len(switches)):
        switches[i] = False
    return "All switches turned OFF."

  def schedule_command(scheduled_commands, switch_numbers, time):
    for switch_number in switch_numbers:
        scheduled_commands.setdefault(time, []).append(switch_number)

  def execute_scheduled_commands(switches, current_time, scheduled_commands):
    switch_numbers = scheduled_commands.pop(current_time, [])
    for switch_number in switch_numbers:
        switches[switch_number - 1] = True
        print("Switch {} will turn ON at {}".format(switch_number, current_time.strftime('%H:%M')))

    if switch_numbers:
        response = input("Do you want to execute these switches for every day? (Y/N):
").strip().upper()
        if response == "Y":
            for switch_number in switch_numbers:
                switches[switch_number - 1] = True
            print("Scheduled switches executed for every day.")
        else:
            print("Scheduled switches not executed for every day.")
  global command
  def process_time_command(command, switches, current_time, scheduled_commands):
    time_str = command.strip().split()[1]
    time_value = get_time(time_str)
    if time_value is None:
        return "Invalid time format. Please enter the time in HH:MM format."
```

```
    time = time_value
    execute_scheduled_commands(switches, time, scheduled_commands)
    return "It is now {}.".format(time.strftime('%H:%M'))

def process_on_command(command, switches):
    command_parts = command.strip().split()
    if len(command_parts) < 2 or command_parts[1].upper() != "ON":
        return "Invalid command. Please specify switch number(s) to turn ON."
    for switch_str in command_parts[0].split(","):
        try:
            switch_number = int(switch_str)
            switches[switch_number - 1] = True
        except ValueError:
            return "Invalid switch number: {}".format(switch_str)
    return "Switch(es) {}: ON.".format(command_parts[0])

def process_off_command(command, switches):
    command_parts = command.strip().split()
    if len(command_parts) < 2 or command_parts[1].upper() != "OFF":
        return "Invalid command. Please specify switch number(s) to turn OFF."
    for switch_str in command_parts[0].split(","):
        try:
            switch_number = int(switch_str)
            switches[switch_number - 1] = False
        except ValueError:
            return "Invalid switch number: {}".format(switch_str)
    return "Switch(es) {}: OFF.".format(command_parts[0])

def process_schedule_command(command, current_time, scheduled_commands):
    command_parts = command.strip().split()
```

```
    switch_numbers = [int(s) for s in command_parts[0].split(",")]
    time_str = command_parts[3]
    time_value = get_time(time_str)
    if time_value is None:
        return "Invalid time format. Please enter the time in HH:MM format."

    schedule_command(scheduled_commands, switch_numbers, time_value)
    response = input("Do you want to execute these switches for every day? (Y/N): ").strip().upper()
    if response == "Y":
        return "Switches {} will turn ON at {} and are scheduled for every
day.".format(command_parts[0], time_value.strftime('%H:%M'))
    else:
        return "Switches {} will turn ON at {} and are not scheduled for every
day.".format(command_parts[0], time_value.strftime('%H:%M'))

  def process_command(command, switches, current_time, scheduled_commands):
    if not command:
        return "Invalid command. Please try again."

    command_parts = command.strip().split()

    if command.startswith("TIME"):
        return process_time_command(command, switches, current_time, scheduled_commands)

    if command == "ALL ON":
        return process_all_on_command(switches)

    if command == "ALL OFF":
        return process_all_off_command(switches)
    if command.startswith("?ALL"):
        return print_all_switch_states(switches)
```

```
    if command.startswith("?") and command[1:].isdigit():
        switch_number = int(command[1:])
        return print_switch_state(switches, switch_number)

    if command == "STOP":
        return "SIMULATION END"

    if len(command_parts) == 2 and command_parts[1] == "ON":
        return process_on_command(command, switches)

    if len(command_parts) == 2 and command_parts[1] == "OFF":
        return process_off_command(command, switches)

    if len(command_parts) == 4 and command_parts[1] == "ON" and command_parts[2] == "AT":
        return process_schedule_command(command, current_time, scheduled_commands)

    return "Invalid command. Please try again."

def process_command_gui():

    command = command_entry.get()
    response = process_command(command, switches, current_time, scheduled_commands)
    result_text.config(state=tk.NORMAL)
    result_text.delete(1.0, tk.END)
    result_text.insert(tk.END, response)
    result_text.config(state=tk.DISABLED)
def play_sound():
    sound=pygame.mixer.Sound('click.mp3')
    sound.play()
img0=PhotoImage(file='img0.png')
label1 = tk.Label(root,image=img0)
```

```
    label1.place(x=0,y=0)



    global command_entry
    command_entry = tk.Entry(root, width=40)
    command_entry.pack(pady=20)
    command_entry.place(x=700, y=580)

    command_button = tk.Button(root, text="Run Command",
command=lambda:[process_command_gui(), display_image(),play_sound()])
    command_button.place(x=800, y=610)

    result_text = tk.Text(root, width=30, height=3, state=tk.DISABLED)
    result_text.place(x=700,y=650)

    exit_button=tk.Button(root,text='Exit',command=lambda:Exit_win())
    exit_button.place(x = 950, y = 670)
    def Exit_win():
        global pic
        win3=tk.Tk()
        win3.title('??')
        win3.geometry('300x100')
        label2=tk.Label(win3,text='Are you sure you want to exit?')
        label2.pack()
        c1=tk.Checkbutton(win3,text='yes',command=lambda:[root.destroy(),win3.destroy()])
        c2=tk.Checkbutton(win3,text='No',command=lambda:win3.destroy())
        c1.pack()
        c2.pack()

    def display_image():
```

```
        global
img1,img2,img3,img4,img5,img6,img7,img8,img8,img9,img10,img11,img12,img13,img13,img14,img
15,img16,img0
        img0=PhotoImage(file='img0.png')
        img1= PhotoImage(file="room1-1on.png")
        img2=PhotoImage(file="room1-1off.png")
        img3=PhotoImage(file="room2-1on.png")
        img4=PhotoImage(file="room2-1off.png")
        img5=PhotoImage(file="room3-1on.png")
        img6=PhotoImage(file="room3-2on.png")
        img7=PhotoImage(file="room3-1,2on.png")
        img8=PhotoImage(file="room3-1,2off.png")
        img9=PhotoImage(file="room4-1on.png")
        img10=PhotoImage(file="room4-2on.png")
        img11=PhotoImage(file="room4-1,2on.png")
        img12=PhotoImage(file="room4-1,2off.png")
        img13=PhotoImage(file="room5-1on.png")
        img14=PhotoImage(file="room5-2on.png")
        img15=PhotoImage(file="room5-1,2on.png")
        img16=PhotoImage(file="room5-1,2off.png")

        command = command_entry.get().upper()
        label1 = tk.Label(root,image=img0)
        label1.place(x=0,y=0)
        for widget in root.winfo_children():
          if isinstance(widget, tk.Label):
             widget.destroy()
        if command == '1 ON':
             label1 = tk.Label(root, image=img1)
        elif command == '1 OFF':
             label1 = tk.Label(root, image=img2)
        elif command == '2 ON':
```

```
        label1 = tk.Label(root, image=img3)
    elif command == '2 OFF':
        label1 = tk.Label(root, image=img4)
    elif command == '3 ON' :
        label1 = tk.Label(root, image=img5)
    elif command == '3 OFF' :
        label1 = tk.Label(root, image=img8)
    elif command == '4 ON':
        label1 = tk.Label(root, image=img6)
    elif command == '4 OFF':
        label1 = tk.Label(root, image=img8)
    elif command == '3,4 OFF':
        label1 = tk.Label(root, image=img8)
    elif command == '3,4 ON':
        label1 = tk.Label(root, image=img7)
    elif command == '5 ON':
        label1 = tk.Label(root, image=img9)
    elif command == '5 OFF':
        label1 = tk.Label(root, image=img12)
    elif command == '5,6 OFF':
        label1 = tk.Label(root, image=img12)
    elif command == '6 ON':
        label1 = tk.Label(root, image=img11)
    elif command == '6 OFF':
        label1 = tk.Label(root, image=img9)
    elif command == '5,6 ON':
        label1 = tk.Label(root, image=img11)
    elif command == '7 ON':
        label1 = tk.Label(root, image=img13)
    elif command == '7 OFF':
        label1 = tk.Label(root, image=img16)
```

```
        elif command == '7,8 OFF':
            label1 = tk.Label(root, image=img16)
        elif command == '8 ON':
            label1 = tk.Label(root, image=img14)
        elif command == '8 OFF':
            label1 = tk.Label(root, image=img16)
        elif command ==  '7,8 ON':
            label1 = tk.Label(root, image=img15)
        else:
           process_command(command)

        if label1:
           label1.place(x=0, y=0)

    root.mainloop()

window_page = tk.Tk()
window_page.title("Welcome to Switch Control")
window_page.geometry("800x549")
background_image = PhotoImage(file="front.png")
background_label = tk.Label(window_page, image=background_image)
background_label.place(x=0, y=0, relwidth=1, relheight=1)
login_label = tk.Label(window_page,height=1,text='login',font=("Arial", 16))
login_label.place(relx=0.4, rely=0.62, anchor=tk.CENTER)
Entry1=tk.Entry(width=15)
Entry1.place(relx=0.45,rely=0.6)
enter_button = tk.Button(window_page, text="Enter", command=create_main_app, bg="white",
fg="black", font=("Arial", 20))
enter_button.place(relx=0.5, rely=0.8, anchor=tk.CENTER)

window_page.mainloop()
```