B.M.S. COLLEGE OF ENGINEERING

(Autonomous college under VTU)

Bull Temple Rd, Basavanagudi, Bengaluru, Karnataka 560019
2023-2025
Department of Computer Applications

Progress Report is submitted for Team AAT work in the subject

“PYTHON PROGRAMMING”
(22MCA1PCPY)

By

1. **BR Shreesha**
2. **Amit S**
3. **Manjunath Pradeep Gaonkar**

Under the Guidance

Prof. R.V. Raghavendra Rao
(Assistant Professor)

Alarm Clock

Code:

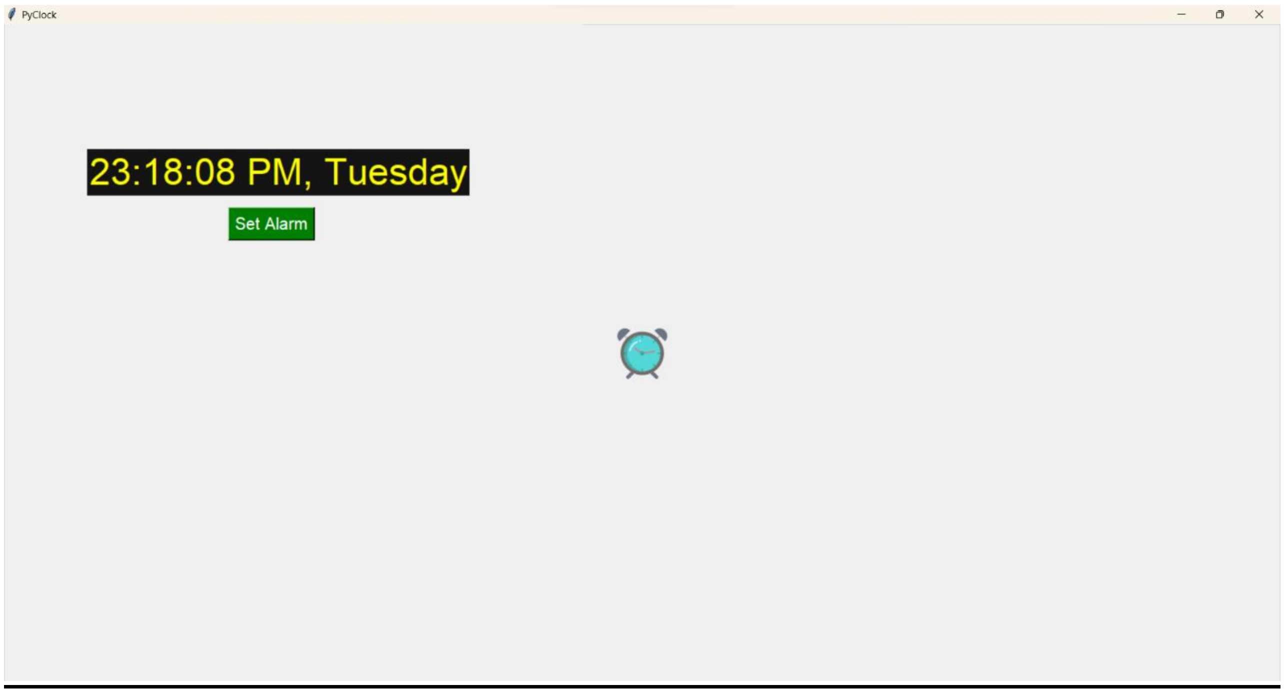
```
import time
from tkinter import *
from PIL import ImageTk
from tkinter import ttk, messagebox
from playsound import playsound
import multiprocessing
from datetime import datetime
from threading import *

hours_list = ['00', '01', '02', '03', '04', '05', '06', '07',
              '08', '09', '10', '11', '12', '13', '14', '15',
              '16', '17', '18', '19', '20', '21', '22', '23', '24']

minutes_list = ['00', '01', '02', '03', '04', '05', '06', '07',
                '08', '09', '10', '11', '12', '13', '14', '15',
                '16', '17', '18', '19', '20', '21', '22', '23',
                '24', '25', '26', '27', '28', '29', '30', '31',
                '32', '33', '34', '35', '36', '37', '38', '39',
                '40', '41', '42', '43', '44', '45', '46', '47',
                '48', '49', '50', '51', '52', '53', '54', '55',
                '56', '57', '58', '59']

ringtones_list = ['deewana','sound']

ringtones_path = {
    'deewana': '..\..\Redmi Note 7s\Download\Dewana Kar Raha Hai Tera Roop Sunehra.mp3',
    'sound':'sound.wav'
}

class AlarmClock:
    def __init__(self, root):
        self.window = root
        self.window.geometry("680x420+0+0")
        self.window.title("PyClock")
        self.window.resizable(width = True, height = True)

        # Background image of the first window.
        self.bg_image = ImageTk.PhotoImage(file="alarm.png")
        self.background = Label(self.window, image=self.bg_image)
        self.background.place(x=0,y=0,relwidth=1,relheight=1)

        # Display Label shows the current time in the
        # first window
        self.display = Label(self.window, font=('Helvetica', 34),
        bg = 'gray8', fg = 'yellow')
        self.display.place(x=100,y=150)

        self.show_time()

        set_button = Button(self.window, text="Set Alarm",
        font=('Helvetica',15), bg="green", fg="white",
        command=self.set_alarm)
        set_button.place(x=270, y=220)

      # Method to show the current time in the first window
    def show_time(self):
        current_time = time.strftime('%H:%M:%S %p, %A')
        # Placing the time format level.
        self.display.config(text = current_time)
        self.display.after(100, self.show_time)


    def set_alarm(self):
        self.alarm_window = Tk()
        self.alarm_window.title("Set Alarm")
        self.alarm_window.geometry("680x420+200+200")

        # Hour Label
        hours_label = Label(self.alarm_window, text="Hours",
        font=("times new roman",20))
        hours_label.place(x=150, y=50)

        #  Minute Label
        minute_label = Label(self.alarm_window, text="Minutes",
        font=("times new roman",20))
        minute_label.place(x=450, y=50)

        # Hour Combobox
        self.hour_var = StringVar()
        self.hour_combo = ttk.Combobox(self.alarm_window,
        width=10, height=10, textvariable=self.hour_var,
        font=("times new roman",15))
        self.hour_combo['values'] = hours_list
        self.hour_combo.current(0)
        self.hour_combo.place(x=150,y=90)

        # Minute Combobox
        self.minute_var = StringVar()
        self.minute_combo = ttk.Combobox(self.alarm_window,
        width=10, height=10, textvariable=self.minute_var,
        font=("times new roman",15))
```

```
self.minute_combo['values'] = minutes_list
self.minute_combo.current(0)
self.minute_combo.place(x=450,y=90)

# Ringtone Label.
ringtone_label = Label(self.alarm_window, text="Ringtones",
font=("times new roman",20))
ringtone_label.place(x=150, y=130)

# Ringtone Combobox(Choose the ringtone).
self.ringtone_var = StringVar()
self.ringtone_combo = ttk.Combobox(self.alarm_window,
width=15, height=10, textvariable=self.ringtone_var,
font=("times new roman",15))
self.ringtone_combo['values'] = ringtones_list
self.ringtone_combo.current(0)
self.ringtone_combo.place(x=150,y=170)

# Create an entry for setting a message
message_label = Label(self.alarm_window, text="Message",
font=("times new roman",20))
message_label.place(x=150, y=210)

self.message_var = StringVar()
self.message_entry = Entry(self.alarm_window,
textvariable=self.message_var, font=("times new roman",14), width=30)
self.message_entry.insert(0, 'Wake Up')
self.message_entry.place(x=150, y=250)

# Test Button: For testing the ringtone music.
test_button = Button(self.alarm_window, text='Test',
font=('Helvetica',15), bg="white", fg="black", command=self.preview_alarm)
test_button.place(x=150, y=300)

# The Cancel Button: For cancel the alarm.
cancel_button = Button(self.alarm_window,
text='Cancel', font=('Helvetica',15), bg="white",
fg="black", command=self.alarm_window.destroy)
cancel_button.place(x=390, y=300)

# The Start Button: For set the alarm time
start_button = Button(self.alarm_window, text='Start',
font=('Helvetica',15), bg="green", fg="white", command=self._threading)
start_button.place(x=490, y=300)

self.alarm_window.mainloop()
```

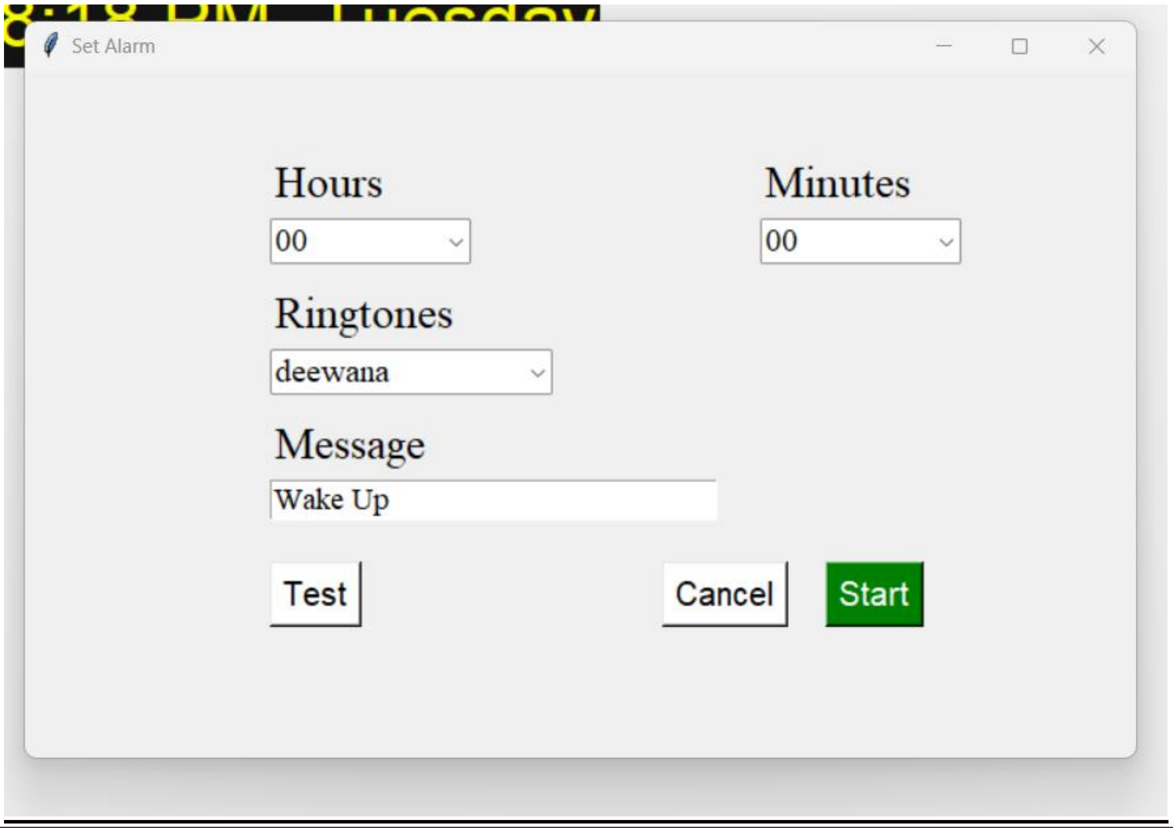
```
    def preview_alarm(self):
        process = multiprocessing.Process(target=playsound,
        args=(ringtones_path[self.ringtone_combo.get()],))
        process.start()
        messagebox.showinfo('Playing...', 'press ENTER to stop playing')
        process.terminate()

        # Method for creating a thread


    def _threading(self):
        x = Thread(target=self.save_alarm)
        x.start()

    def save_alarm(self):
        alarm_time = f"{self.hour_combo.get()}:{self.minute_combo.get()}"
        messagebox.showinfo("Alarm Set", f"Alarm set for {alarm_time}")
        sound_name = self.ringtone_combo.get()
        message = self.message_entry.get()
        found = False
        try:
            while True:
                # The current time is in 24 hour format
                current_time = datetime.now()
                # Converting the current time into hours and minutes
                current_time_format = current_time.strftime("%H:%M")
                if current_time_format == alarm_time:
                    process = multiprocessing.Process(target=playsound,
                    args=(ringtones_path[sound_name],))
                    process.start()
                    messagebox.showinfo("Alarm",f"{message}, It's {alarm_time}")
                    result = messagebox.askyesno("Custom Message", "Do you want to snooze for 5 minutes?")
                    if result:
                        print("User clicked 'Yes'")

                        process.terminate()


                        p = int(self.minute_combo.get())
                        p5 = str(p + 5)
                        alarm_time = f"{self.hour_combo.get()}:{p5}"
                        messagebox.showinfo("Alarm Set", f"Alarm set for {alarm_time}")
                        sound_name = self.ringtone_combo.get()
                        message = self.message_entry.get()
                        try:
                          while True:
```

```
            # The current time is in 24 hour format
                    current_time = datetime.now()
            # Converting the current time into hours and minutes
                    current_time_format = current_time.strftime("%H:%M")
                    if current_time_format == alarm_time:
                      process = multiprocessing.Process(target=playsound,
                      args=(ringtones_path[sound_name],))
                      process.start()
                      messagebox.showinfo("Alarm",f"{message}, It's {alarm_time}")
                      process.terminate()
                      found = True
                      break

                  if found:
                        break



                except Exception as es:
                    messagebox.showerror("Error!", f"Error due to {es}")




              else:
                  print("User clicked 'No'")
                  process.terminate()
                  break

         except Exception as es:
             messagebox.showerror("Error!", f"Error due to {es}")

   if __name__ == "__main__":
       root = Tk()
       obj = AlarmClock(root)
       root.mainloop()
```

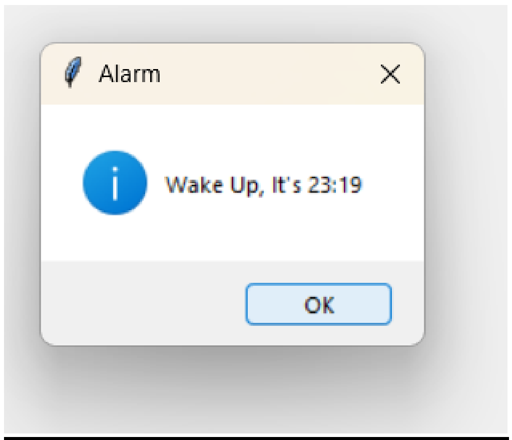
Sample output:



```
23:18:08 PM, Tuesday
```



Hours: 00
Minutes: 00
Ringtones: deewana
Message: Wake Up



Wake Up, It's 23:19

Completed:

1. Alarm can be set and cancelled.

Target:

1. Add a function that snoozes the alarm.