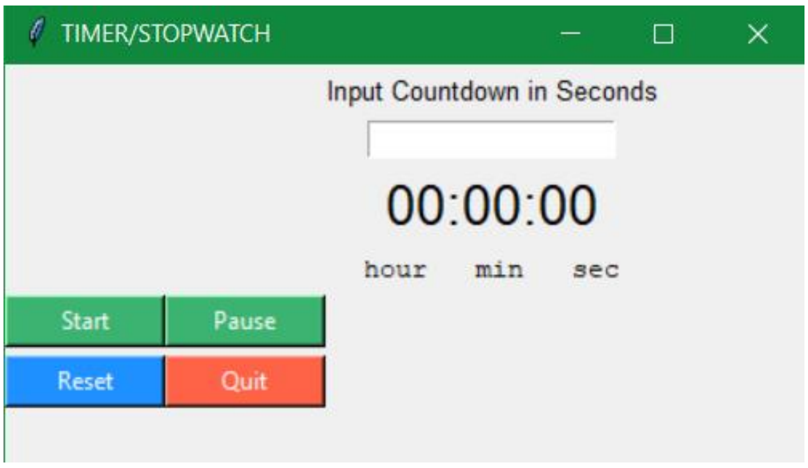
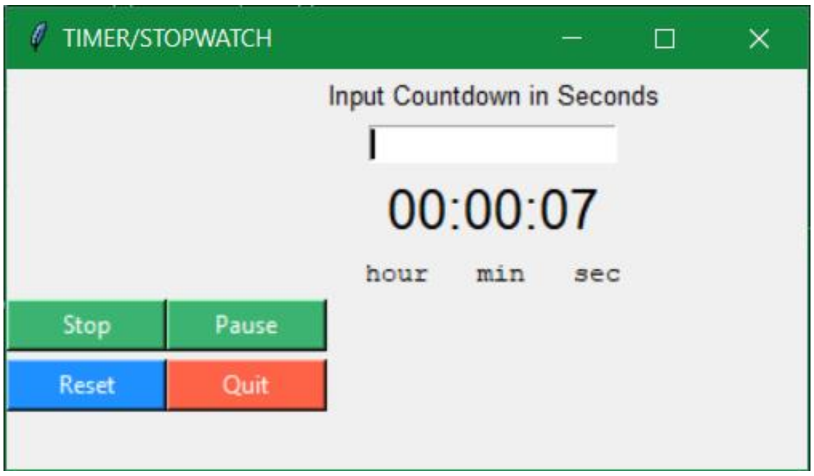
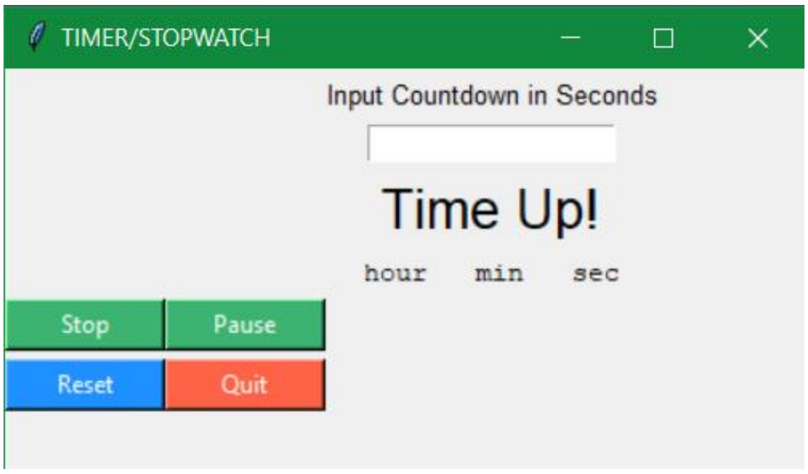
```
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 10 21:34:53 2023

@author: AABID HUSSAIN DAR
"""

import tkinter as t
import tkinter.messagebox




class Application(t.Frame):
    def __init__(self, master, *args, **kwargs):
        t.Frame.__init__(self, master, *args, **kwargs)
        self.master = master
        self.running = False
        self.time = 0
        self.hours = 0
        self.mins = 0
        self.secs = 0
        self.build_interface()

    def build_interface(self):
        self.clock = t.Label(self, text="Input Countdown in Seconds", font=("monospace", 10), width=20)
        self.clock.grid(row=0, column=6, stick="S", pady=2)

        self.time_entry = t.Entry(self)
        self.time_entry.grid(row=1, column=6, pady=2)
        self.clock = t.Label(self, text="00:00:00", font=("monospace", 20), width=10)
        self.clock.grid(row=2, column=6, stick="S", pady=2)


        self.time_label = t.Label(self, text="hour   min   sec", font=("courier", 10), width=15)
        self.time_label.grid(row=3, column=6, sticky="N")


        self.power_button = t.Button(self, text="Start", command=lambda: self.start(),
background="mediumseagreen", foreground="white", width=10)
        self.power_button.grid(row=5, column=3, sticky="NE", pady=2)


        self.pause_button = t.Button(self, text="Pause", command=lambda: self.pause(),
background="mediumseagreen", foreground="white", width=10)
        self.pause_button.grid(row = 5,column=4, sticky = "NW", pady=2)



        self.reset_button = t.Button(self, text="Reset", command=lambda: self.reset(),
background="dodgerblue", foreground="white", width=10)
        self.reset_button.grid(row=6, column=3, sticky="NW", pady=2)


        self.quit_button = t.Button(self, text="Quit", command=lambda: self.quit(),
background="tomato", foreground="white", width=10)
        self.quit_button.grid(row=6, column=4, sticky="NE", pady=2)



        self.master.bind("<Return>", lambda x: self.start())
        self.time_entry.bind("<Key>", lambda v: self.update())


    def calculate(self):
        """time calculation"""
        self.hours = self.time // 3600
        self.mins = (self.time // 60) % 60
        self.secs = self.time % 60
        return "{:02d}:{:02d}:{:02d}".format(self.hours, self.mins, self.secs)


    def update(self):
        """validation"""
        self.time = int(self.time_entry.get())
        try:
            self.clock.configure(text=self.calculate())
        except:
            self.clock.configure(text="00:00:00")


    def timer(self):
        """display time"""
        if self.running:
            t.Label(self, bg="grey")
            if self.time <= 0:
                self.clock = t.Label(self, text="Time Up!", font=("monospace", 20), width=10)
                self.clock.grid(row=2, column=6, stick="S", pady=2)
            else:
                self.clock.configure(text=self.calculate())
                self.time -= 1
                self.after(1000, self.timer)


    def start(self):
        """start timer"""
        try:
            self.time = int(self.time_entry.get())
            self.time_entry.delete(0, 'end')
        except:
            self.time = self.time
        self.power_button.configure(text="Stop", command=lambda: self.stop())
        self.master.bind("<Return>", lambda x: self.stop())
```

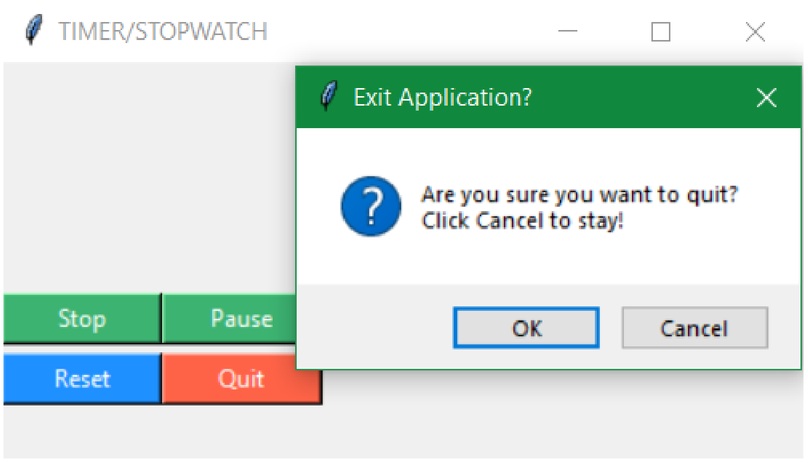
```
        self.running = True
        self.timer()

    def stop(self):
        """Stop timer"""
        self.power_button.configure(text="Start", command=lambda: self.start())
        self.master.bind("<Return>", lambda x: self.start())
        self.running = False

    def reset(self):
        """Resets the timer to 0."""
        self.power_button.configure(text="Start", command=lambda: self.start())
        self.master.bind("<Return>", lambda x: self.start())
        self.running = False
        self.time = 0
        self.clock["text"] = "00:00:00"

    def quit(self):
        """quit the window"""
        if t.messagebox.askokcancel("Exit Application?", "Are you sure you want to quit?\nClick Cancel
to stay!"):
            root.destroy()

    def pause(self):
        """Pause timer"""
        self.pause_button.configure(text="Resume", command=lambda: self.resume())
        self.master.bind("<Return>", lambda x: self.resume())
        if self.running == True:
            self.running = False
        self.timer()
    def resume(self):
      """Resume timer"""
      self.pause_button.configure(text="Pause", command=lambda: self.pause())
      self.master.bind("<Return>", lambda x: self.pause())
      if self.running == False:
        self.running = True
      self.timer()




if __name__ == "__main__":
    """Main loop of timer"""
    root = t.Tk()
    root.geometry("400x200")
    root.title("TIMER/STOPWATCH")
    Application(root).pack(side="top", fill="both", expand=True)
    root.mainloop()
```

SCREENSHOTS OF OUTPUT









Are you sure you want to quit?
Click Cancel to stay!