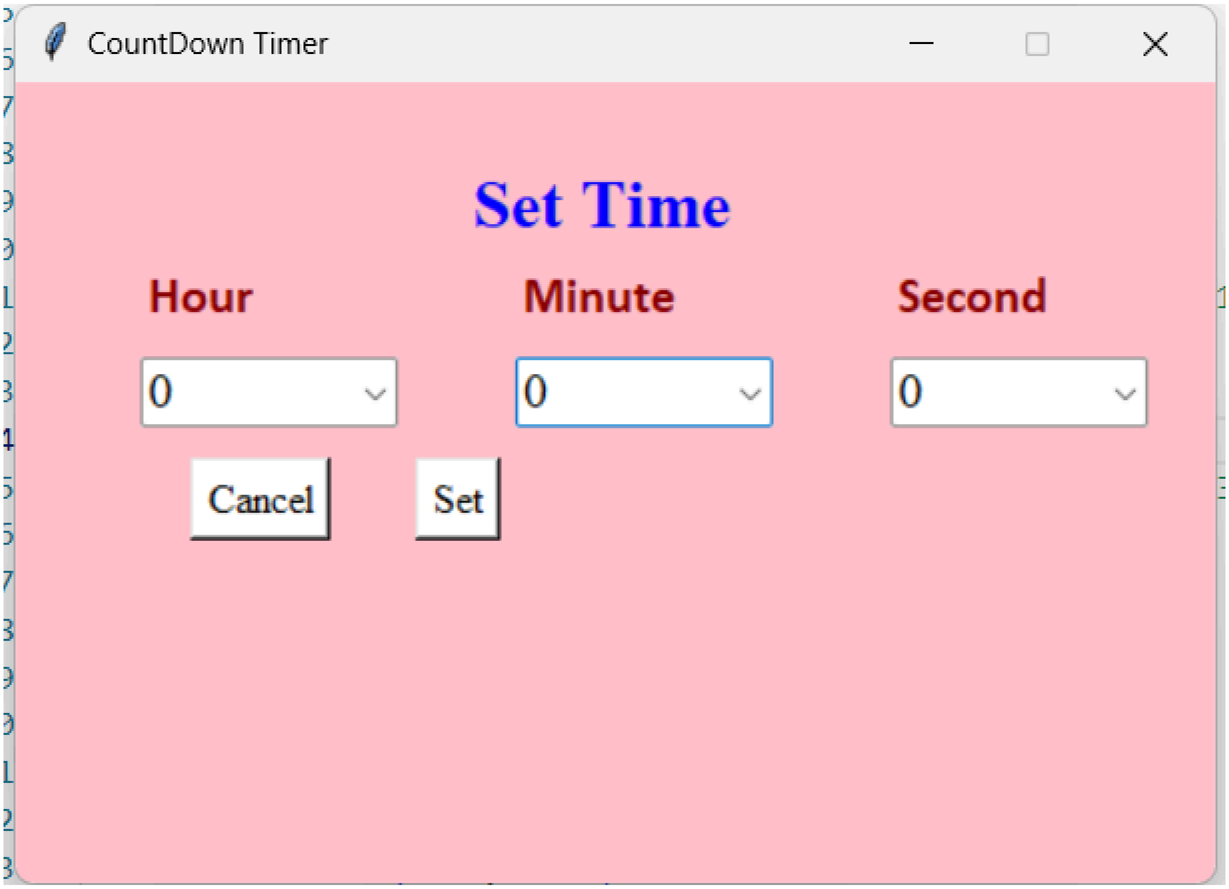
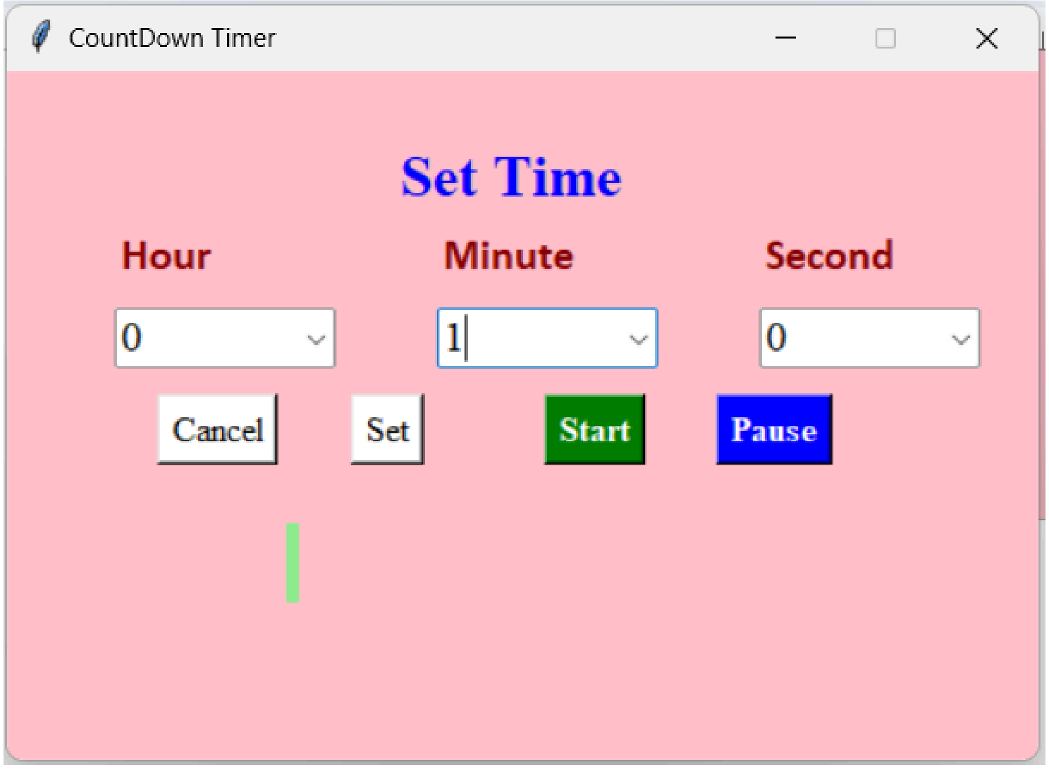
```
# Import the time module
import time
from tkinter import *
import multiprocessing

from tkinter import ttk, messagebox
from threading import *


# Hour list
hour_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 20, 21, 22, 23, 24]

# Minute List
min_sec_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 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,
]

# Creating a CounDown Class
class CountDown:
    def __init__(self, root):
        self.window = root
        self.window.geometry("480x320+0+0")
        self.window.title('CountDown Timer')
        # Tkinter window background color
        self.window.configure(bg='pink')
        # Fixing the Window length constant
        self.window.resizable(width = False, height = False)

        # Declaring a variable to pause the countdown time
        self.pause = False

        # The Start and Pause buttons are placed
        # inside this frame
        self.button_frame = Frame(self.window, bg="pink",
        width=240, height=40)
        self.button_frame.place(x=230, y=150)
        # This frame is used to show the countdown time label
        self.time_frame = Frame(self.window, bg="pink",  width=480,
height=120).place(x=0, y=210)

        # Tkinter Labels
        time_label = Label(self.window, text="Set Time",
        font=("times new roman",20, "bold"), bg='pink',fg='blue')
        time_label.place(x=180, y=30)

        hour_label = Label(self.window, text="Hour",
        font=("Calibri",15,'bold'), bg='pink', fg='Dark red')
        hour_label.place(x=50, y=70)

        minute_label = Label(self.window, text="Minute",
        font=("Calibri",15,'bold'), bg='pink', fg='Dark red')
        minute_label.place(x=200, y=70)

        second_label = Label(self.window, text="Second",
        font=("Calibri",15,'bold'), bg='pink', fg='Dark red')
        second_label.place(x=350, y=70)
        # ===========================================

        # Tkinter Comboboxes
        # Combobox for hours
        self.hour = IntVar()
        self.hour_combobox = ttk.Combobox(self.window, width=8,
        height=10, textvariable=self.hour,
        font=("times new roman",15))
        self.hour_combobox['values'] = hour_list
        self.hour_combobox.current(0)
        self.hour_combobox.place(x=50,y=110)

        # Combobox for minutes
        self.minute = IntVar()
        self.minute_combobox = ttk.Combobox(self.window, width=8,
        height=10, textvariable=self.minute,
        font=("times new roman",15))
        self.minute_combobox['values'] = min_sec_list
        self.minute_combobox.current(0)
        self.minute_combobox.place(x=200,y=110)
        # Combobox for seconds
        self.second = IntVar()
        self.second_combobox = ttk.Combobox(self.window, width=8,
        height=10, textvariable=self.second,
        font=("times new roman",15))
        self.second_combobox['values'] = min_sec_list
        self.second_combobox.current(0)
        self.second_combobox.place(x=350,y=110)
        # ===========================================

        # Tkinter Buttons
        # Cancel button
        cancel_button = Button(self.window, text='Cancel',
        font=('times new roman',12), bg="white", fg="black",
        command=self.Cancel)
        cancel_button.place(x=70, y=150)

        # Set Time Button
        # When the user will press this button
        # the 'Start' and 'Pause' button will
        # show inside the 'self.button_frame' frame
        set_button = Button(self.window, text='Set',
        font=('times new roman',12), bg="white", fg="black",
        command=self.Get_Time)
        set_button.place(x=160, y=150)

    # It will destroy the window
    def Cancel(self):
        self.pause = True
        self.window.destroy()

    # When the set button is pressed, this
    # function gets called
    def Get_Time(self):
        self.time_display = Label(self.time_frame,
        font=('times new roman', 20 , "bold"),
        bg = 'light green', fg = 'Red')
        self.time_display.place(x=130, y=210)

        try:
            # Total amount of time in seconds
            h = (int(self.hour_combobox.get())*3600)
            m = (int(self.minute_combobox.get())*60)
            s = (int(self.second_combobox.get()))
            self.time_left = h + m + s

            # If the user try to set the default time(0:0:0) then
            # a warning message will display
            if s == 0 and m == 0 and h == 0:
                messagebox.showwarning('Warning!',\
                'Please select a right time to set')
            else:
                # Start Button
                start_button = Button(self.button_frame, text='Start',
font=('times new roman',12,'bold'), bg="green", fg="white",
                command=self.Threading)
                start_button.place(x=20, y=0)

                # Pause Button
                pause_button = Button(self.button_frame, text='Pause',
font=('times new roman',12,'bold'), bg="blue", fg="white",
                command=self.pause_time)
                pause_button.place(x=100, y=0)
        except Exception as es:
            messagebox.showerror("Error!",  f"Error due to {es}")

    # Creating a thread to run the show_time function
    def Threading(self):
        # Killing a thread through "daemon=True" isn't a good idea
        self.x = Thread(target=self.start_time, daemon=True)
        self.x.start()

    # It wil clear all the widgets inside the
    # 'self.button_frame' frame(Start and Pause buttons)
    def Clear_Screen(self):
        for widget in self.button_frame.winfo_children():
            widget.destroy()

    def pause_time(self):
        self.pause = True
        mins, secs = divmod(self.time_left, 60)
        hours = 0
        if mins > 60:
            # hour minute
            hours, mins = divmod(mins, 60)

        self.time_display.config(text=f"Time Left: {hours}: {mins}:
{secs}")
        self.time_display.update()


    # When the Start button will be pressed then,
    # this "show_time" function will get called.
    def start_time(self):
        self.pause = False
        while self.time_left > 0:
            mins, secs = divmod(self.time_left, 60)

            hours = 0
            if mins > 60:
                # hour minute
                hours, mins = divmod(mins, 60)

            self.time_display.config(text=f"Time Left: {hours}: {mins}:
{secs}")
            self.time_display.update()
            # sleep function: for 1 second
            time.sleep(1)
            self.time_left = self.time_left -1

            # if the pause button is pressed,
            # the while loop will break
            if self.pause == True:
                break

if __name__ == "__main__":
    root = Tk()
    # Creating a CountDown class object
    obj = CountDown(root)
```

```
root.mainloop()
```

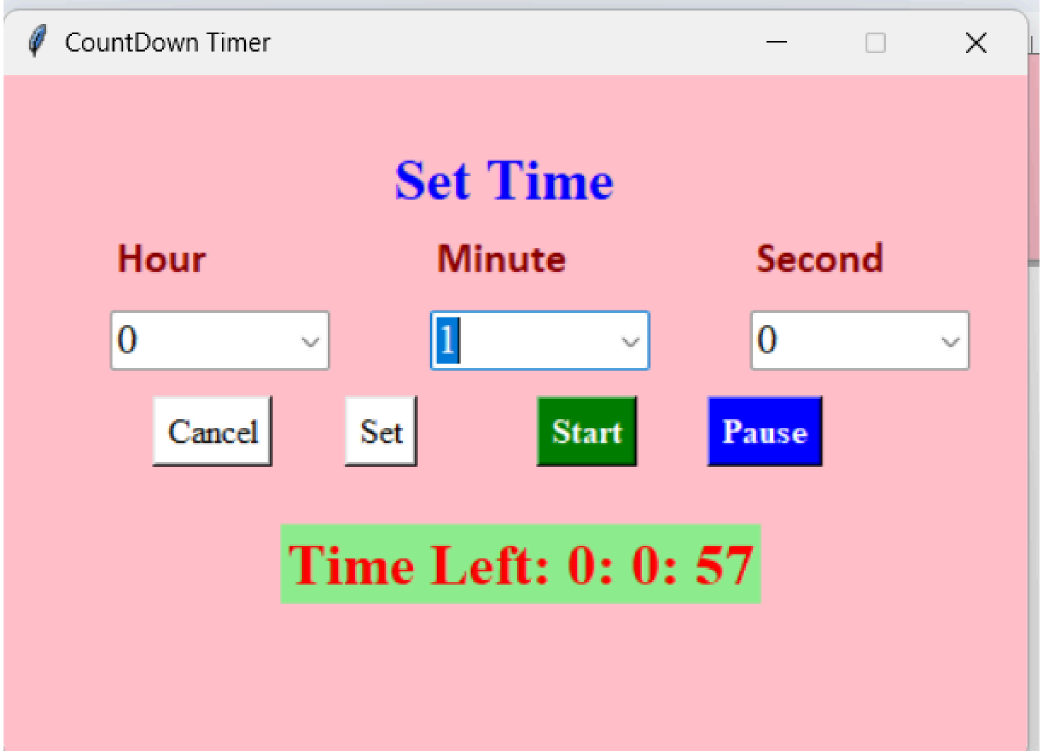




Hour 0

Minute 1

Second 0



Hour 0

Minute 1

Second 0

Time Left: 0: 0: 57

Submitted by :
Navyasri Nannapaneni