COUNTDOWN TIMER

Suraj Singh

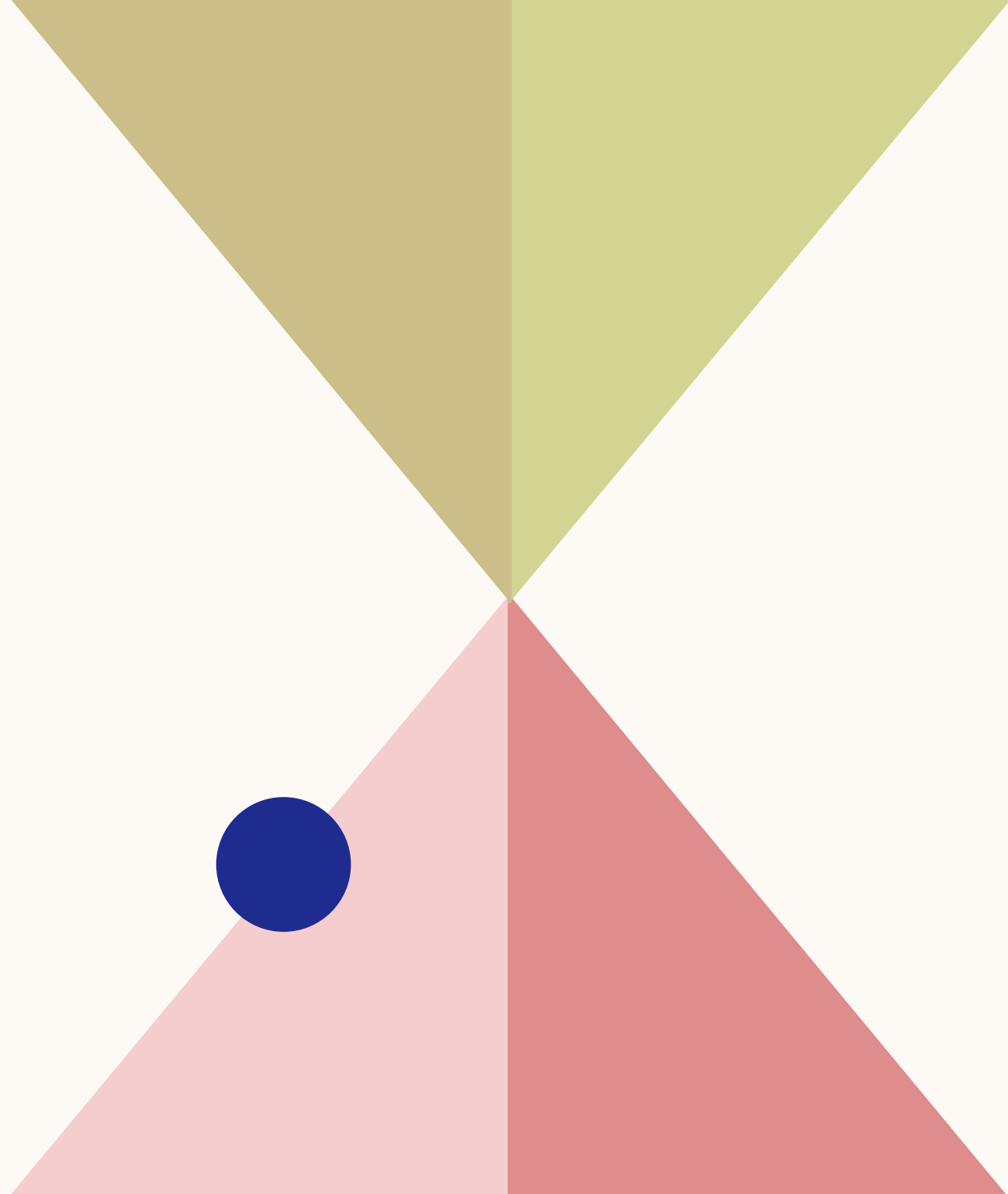
AGENDA

Introduction

Primary goals

code with major functions

Working images



INTRODUCTION

The aim is to make countdown timer in python so it enables user to countdown to zero. With features like

. Start

. Pause

. Resume

.Stop

.Reset

PRIMARY GOALS

To create a countdown timer which can efficiently countdown to zero, with essential features.

SPECIFICATIONS

- Program uses following function/library and platforms:-
- 1. Python 3.10.5(64- bit)
- 2. tkinter python library
- 3. time module
- 4. threading module

OVERVIEW

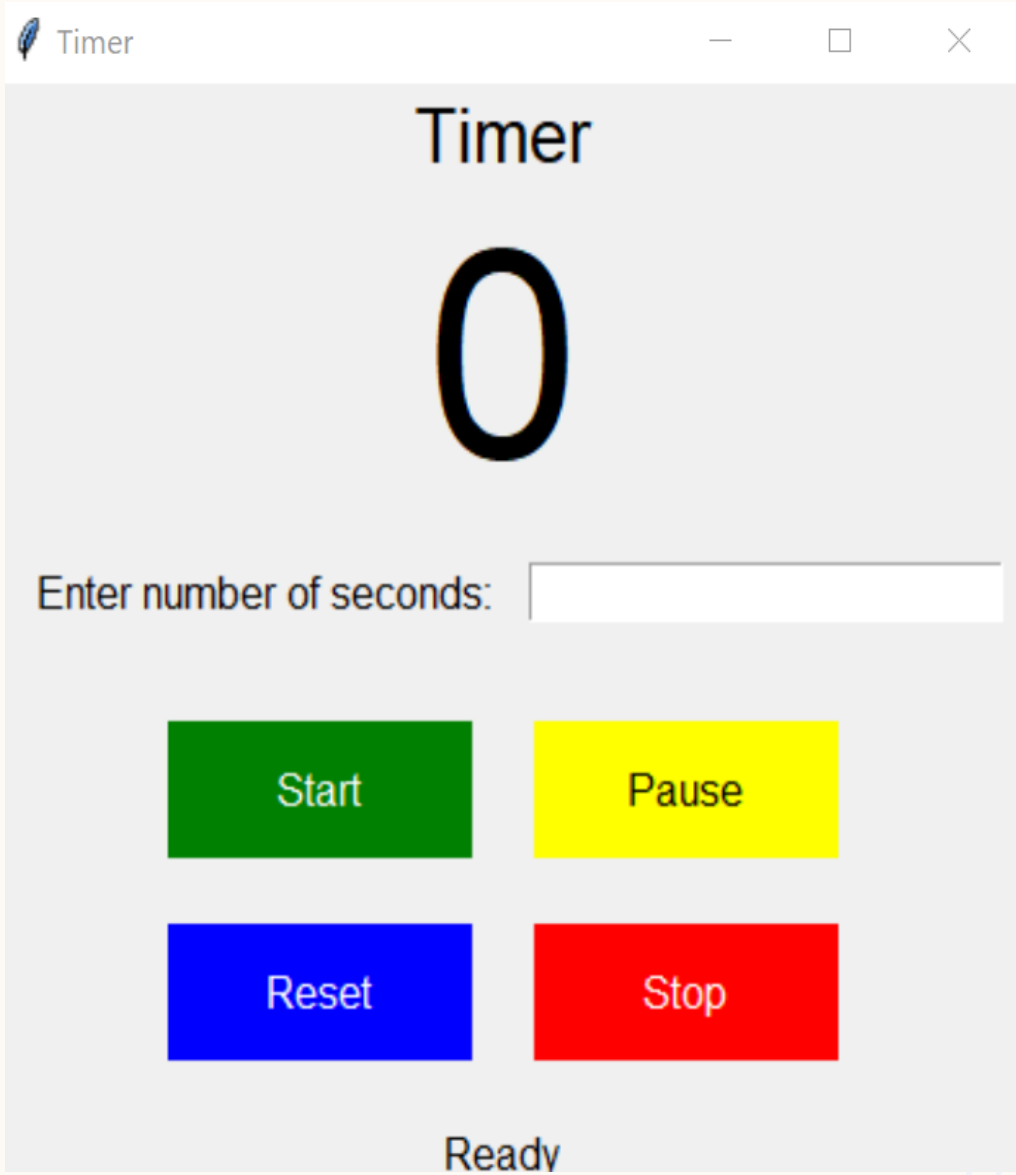
The program uses the built in Tkinter library of python for the graphical user interface of the program.
It uses the Tk object of the tkinter library to create the main window with the size 600x400 pixels.
It also uses different widgets provided by tkinter library to render different elements of the program.
For example lable, entry, button widgets.

SOLUTION WITH CODE

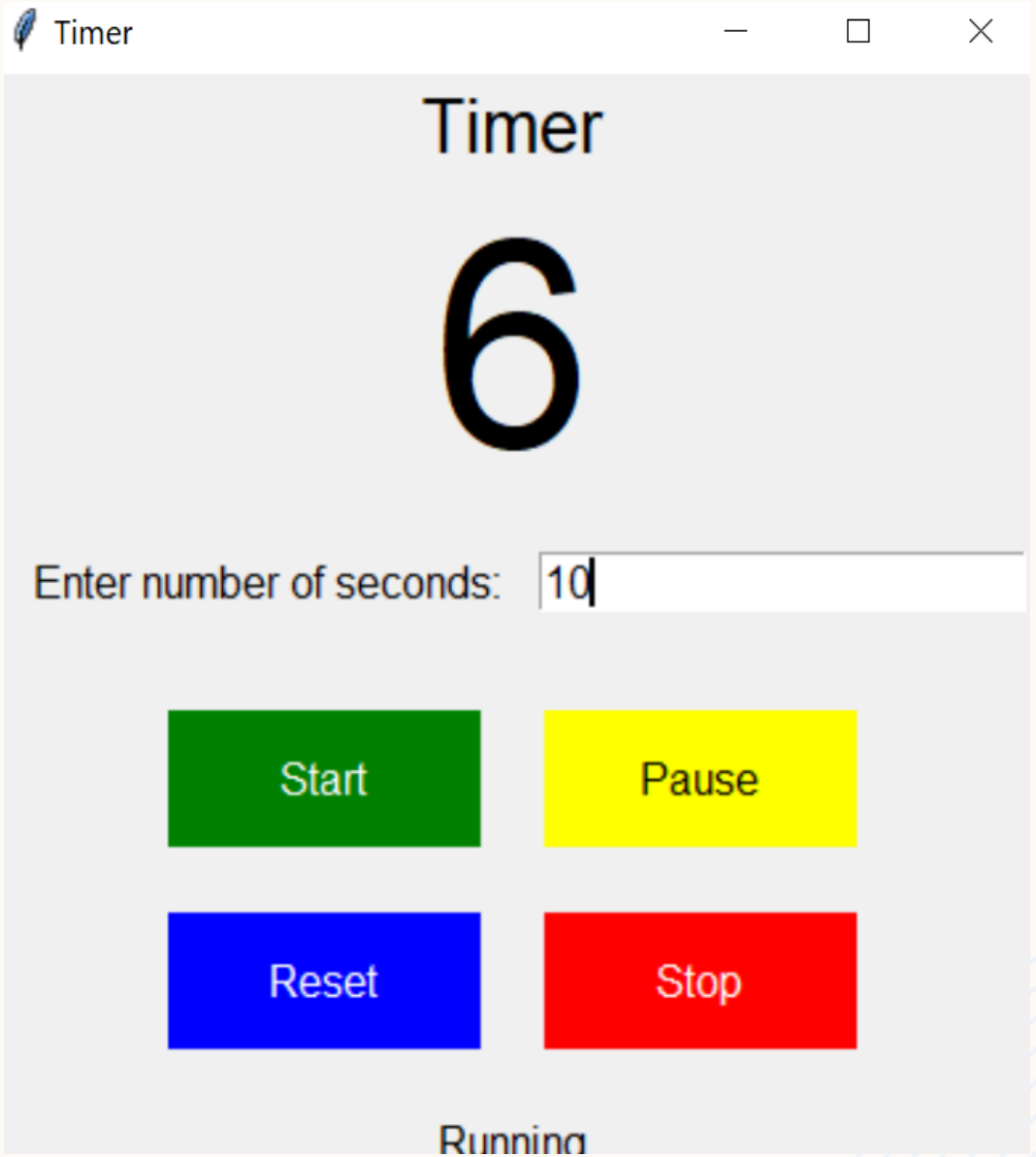
Suraj Singh

Code:-

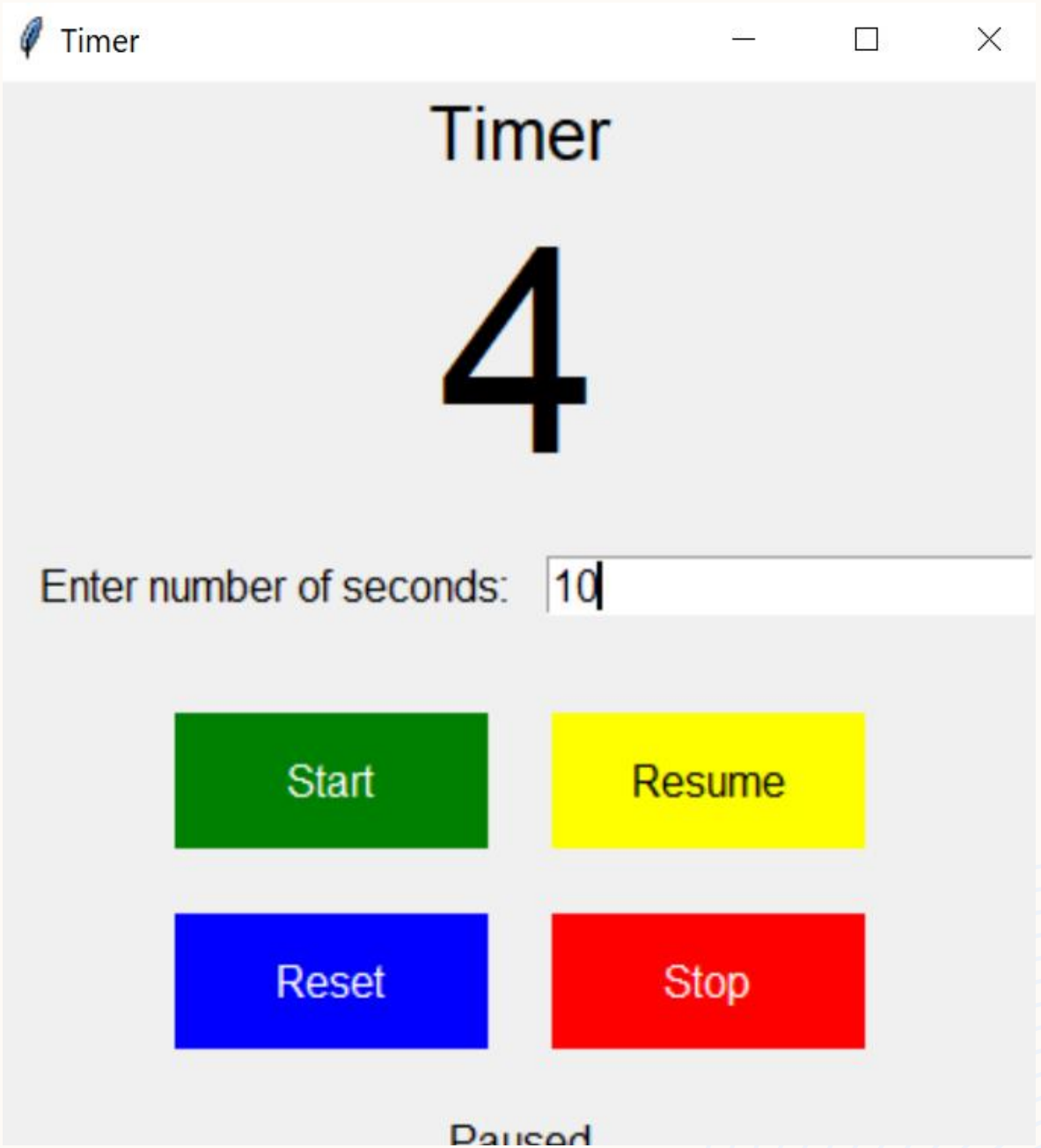
```
from tkinter import *
import time
from threading import Thread

class Timer:
    def __init__(self, label, status_label):
        self.label = label
        self.limit = None
        self.isRunning = False
        self.status_label = status_label
        self.isReset = False
        self.isStopped = False
        self.isPaused = False

    def start(self):
        t = Thread(target=self.__start)
        t.start()

    def __start(self):
        self.status_label.config(fg="black")
        if not self.limit or self.isRunning:
            return
        self.isRunning = True
        while self.limit >= 0 if self.limit else self.isRunning:
            if(not self.isRunning):
                break
            self.label.config(text=self.limit)
            self.limit -= 1
            time.sleep(1)
        self.isRunning = False
        fg = "black"
        status = "Time's up!"
        if self.isPaused:
            status = "Paused"
        elif self.isStopped:
            status = "Stopped"
        elif self.isReset:
            status = "Reset"
        else:
            fg = "red"
        self.status_label.config(text=status)
        self.status_label.config(fg=fg)
        if not self.isPaused:
            self.limit = None
        self.isReset = False
        self.isStopped = False
        self.isPaused = False

    def pause(self):
        self.isRunning = False
```



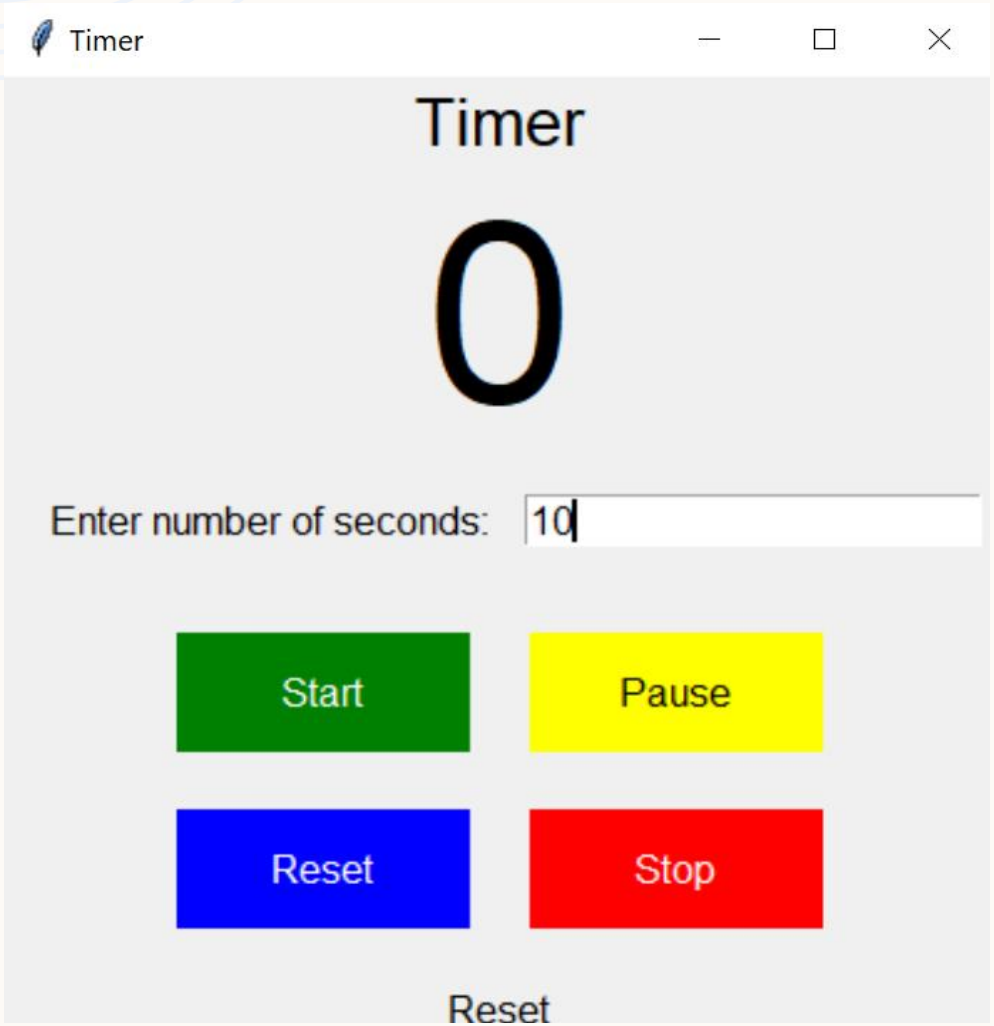
```
def main():
    # Creating Tk class for main GUI window
    root = Tk()
    root.title("Timer")
    root.geometry('400x400')
    label = Label(root, text="Timer", font=("Helvetica", 20))
    label.pack()
    countdown = Label(root, text="0", font=("Helvetica", 80))
    countdown.pack()
    entry_frame = Frame(root)
    enter_number_label = Label(entry_frame, text="Enter number of seconds:",
font=("Helvetica", 12), padx=12, pady=12)
    enter_number_label.pack(side=LEFT)
    enter_number = Entry(entry_frame, font=("Helvetica", 12))
    enter_number.pack(side=RIGHT)
    enter_number.focus()
    entry_frame.pack()
    button_frame = Frame(root, padx=12, pady=12)
    button_col_1_frame = Frame(button_frame, relief='flat')
    button_col_2_frame = Frame(button_frame, relief='flat')
    start_button = Button(button_col_1_frame, text="Start", font=("Helvetica", 12),
width=12, height=2, relief='flat', bg='green', fg='white')
    start_button.pack(side=LEFT, anchor=CENTER, pady=12, padx=12)
    pause_button = Button(button_col_1_frame, text="Pause", font=("Helvetica",
12), width=12,height=2, relief='flat', bg='yellow', fg='black')
    pause_button.pack(side=LEFT, anchor=CENTER, pady=12, padx=12)
    reset_button = Button(button_col_2_frame, text="Reset", font=("Helvetica",
12), width=12,height=2, relief='flat', bg='blue', fg='white')
    reset_button.pack(side=LEFT, anchor=CENTER, pady=12, padx=12)
    stop_button = Button(button_col_2_frame, text="Stop", font=("Helvetica", 12),
width=12,height=2, relief='flat', bg='red', fg='white')
    stop_button.pack(side=LEFT, anchor=CENTER, pady=12, padx=12)
    button_col_1_frame.pack(fill=X)
    button_col_2_frame.pack(fill=X)
    button_frame.pack()
    status_label = Label(root, text="Ready", font=("Helvetica", 12), padx=12,
pady=12)
    status_label.pack()

    t = Timer(countdown, status_label)
```

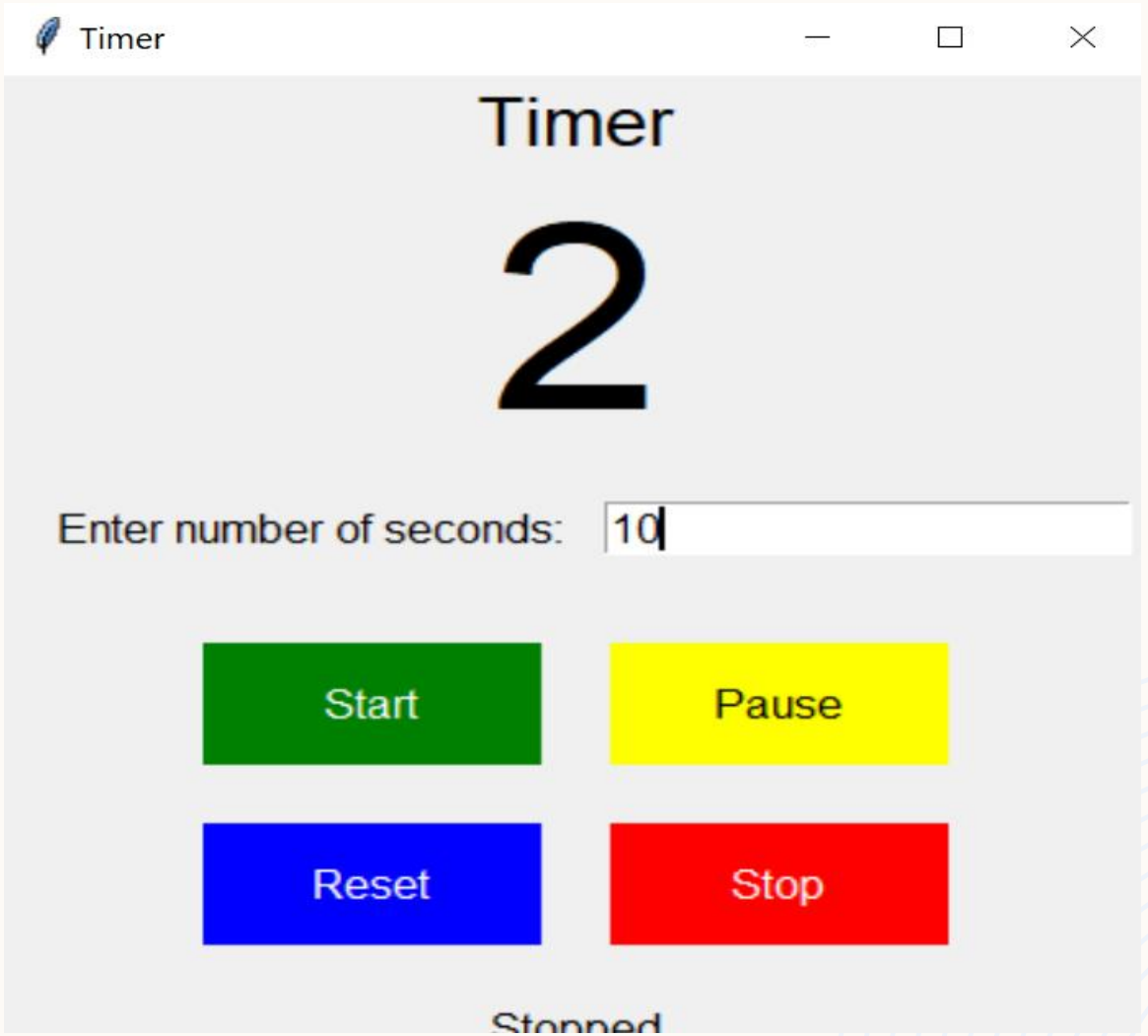


```
def start_timer():
    if not t.limit:
        try:
            t.limit = int(enter_number.get())
        except ValueError:
            status_label.config(fg="red")
            status_label.config(text="Enter a valid number")
            time.sleep(1)
            status_label.config(text="Ready")
            status_label.config(fg="black")
    t.start()
    status_label.config(text="Running")

def pause_timer():
    if not t.limit:
        return
    if t.isRunning:
        t.isPaused = True
        t.pause()
        pause_button.config(text="Resume")
        status_label.config(text="Paused")
    else:
        pause_button.config(text="Pause")
        t.isPaused = False
        t.start()
        status_label.config(text="Running")

def reset_timer():
    t.isReset = True
    t.isRunning = False
    countdown.config(text="0")
    pause_button.config(text="Pause")
    status_label.config(fg="black")
    status_label.config(text="Reset")

def stop_timer():
    t.isStopped = True
    t.isRunning = False
    pause_button.config(text="Pause")
    status_label.config(text="Stopped")
```



```
    start_button.config(command=start_timer)
    pause_button.config(command=pause_timer)
    stop_button.config(command=stop_timer)
    reset_button.config(command=reset_timer)

    root.mainloop()

if __name__ == '__main__':
    main()
```



PROJECT LINK

https://github.com/Suraj-singh048/countdowntimer

THANK YOU

Suraj Singh