DIGITAL STOPWATCH

WITH PYTHON

OUR NAME

ELIN

ANNA CALISTA EVANGELIN

GIGA

GIGA FELICIA ANANTA

RAHMA

ANISA RAHMAYANI AYUNINGRUM

BACKGROUND

Stopwatch or also known as a timer is a tool used to measure time. The shape of this tool is similar to a watch.

The function of the stopwatch itself is to measure time for certain purposes, both for educational purposes, competitions, performances, research and others. Another stopwatch function is a stopclock feature. This feature functions as a time delay without affecting the process of measuring time.

BACKGROUND

What happens if the stopwatch is in the form of a software? Which can be used on laptops and smart phones.

In making this software we use the Python programming language, a high-level programming language that can execute a number of multi-purpose instructions directly (interpretatively) with the Object Oriented Programming method and also uses dynamic semantics to provide a level of readability of the syntax.

PURPOSE OF MAKING

1. To find out how the stopwatch works
2. To know the implementation of stopwatch in digital system

PROCESS OF MAKING

Materials Used:

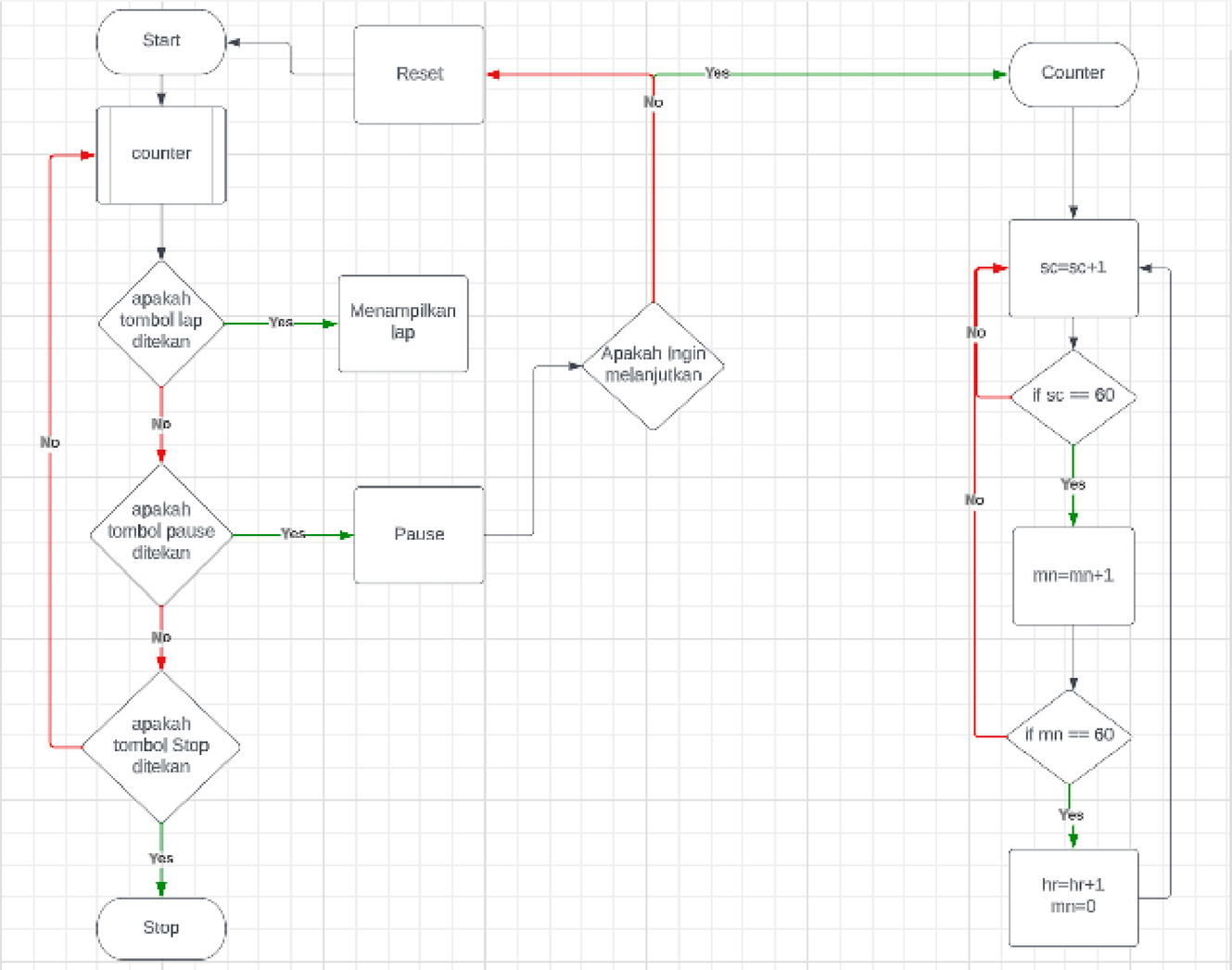
1. Laptops
2. Lucid chart
3. Visual Studio Code (VSCode)

Tools used:

1. Qt Designer
2. Python Programming Language

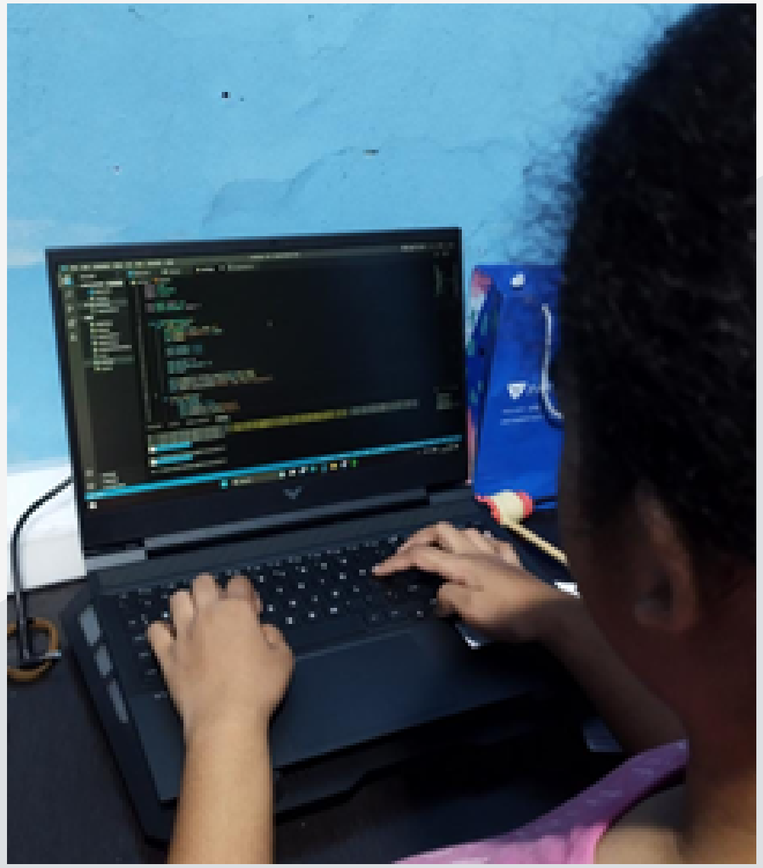
Before make an application. First, we make a flow of the program use a flowchart. After created the flowchart we made the User Interface(UI) to looks more attractive. Next, we started create the code use the Python programming language.

THESE ARE THE PROCESS OF MAKING THE DIGITAL STOPWATCH APPLICATION THAT WE MADE.



The Flowchard, that we made on Lucid Chart

THESE ARE THE PROCESS OF MAKING THE DIGITAL STOPWATCH APPLICATION THAT WE MADE.



Material used to work on PJBL projects

THESE ARE THE PROCESS OF MAKING THE DIGITAL STOPWATCH APPLICATION THAT WE MADE.

wtch.py > MyGUI

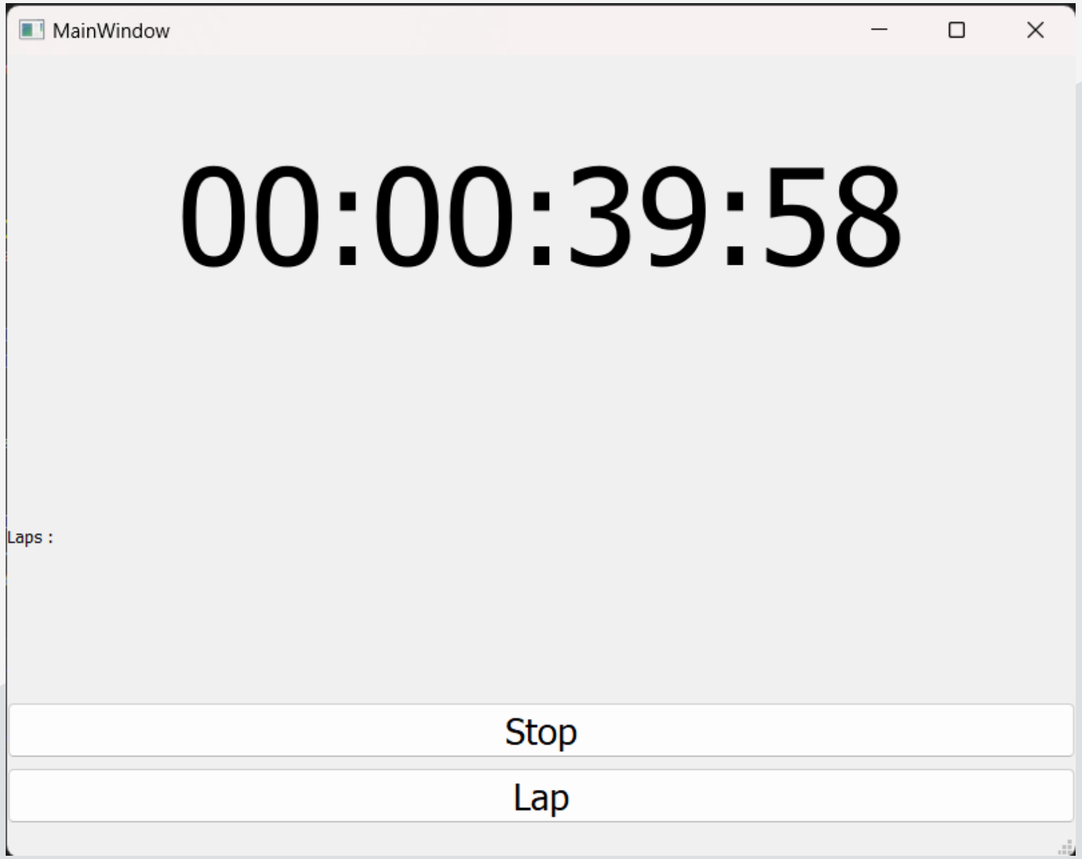
```python
import time
import threading
import typing

from PyQt5 import uic
from PyQt5.QtWidgets import *


class MyGUI(QMainWindow):

    def __init__(self):
        super(MyGUI, self).__init__()
        uic.loadUi("stopwatch.ui", self)
        self.show()

        self.running = False
        self.started = False

        self.passed = 0
        self.previous_passed = 0
        self.lap=1

        self.pushButton.clicked.connect(self.start_stop)
        self.pushButton_2.clicked.connect(self.lap_reset)
        self.label2.setStyleSheet("border: 10px solid transparent")

    def start_stop(self):
        if self.running:
            self.running = False
            self.pushButton.setText("Resume")
            self.pushButton_2.setText("Reset")
        else :
            self.running = True
            self.pushButton.setText("Stop")
            self.pushButton_2.setText("Lap")
            self.pushButton_2.setEnabled(True)
            threading.Thread(target=self.stopwatch).start()

    def lap_reset(self):
        if self.running :
            self.label1.setText(self.label1.text() + f"(Lap {self.lap} - Passed: {self.format_time_string(self.passed)}"
                                                     f" - Difference: {self.format_time_string(self.passed - self.previous_passed)}\n")
            self.lap += 1
            self.previous_passed = self.passed
```

wtch.py > MyGUI

```python
        else:
            self.pushButton.setText("Start")
            self.pushButton_2.setText("Lap")
            self.pushButton_2.setEnabled(False)
            self.label2.setText("00:00:00:00")
            self.label1.setText("Laps : ")
            self.lap = 1
            self.passed = 0
            self.previous_passed = 0

    def format_time_string(self, time_passed):
        secs = time_passed % 60
        mins = time_passed // 60
        hours = mins // 60
        return f"{int(hours):02d}:{int(mins):02d}:{int(secs):02d}:{int((self.passed %1) * 100):02d}"

    def stopwatch(self):
        start = time.time()
        if self.started:
            until_now = self.passed
        else:
            until_now = 0
            self.started = True

        while self.running:
            self.passed = time.time() - start + until_now
            self.label2.setText(self.format_time_string(self.passed))

def main():
    app = QApplication([])
    window = MyGUI()
    app.exec_()


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

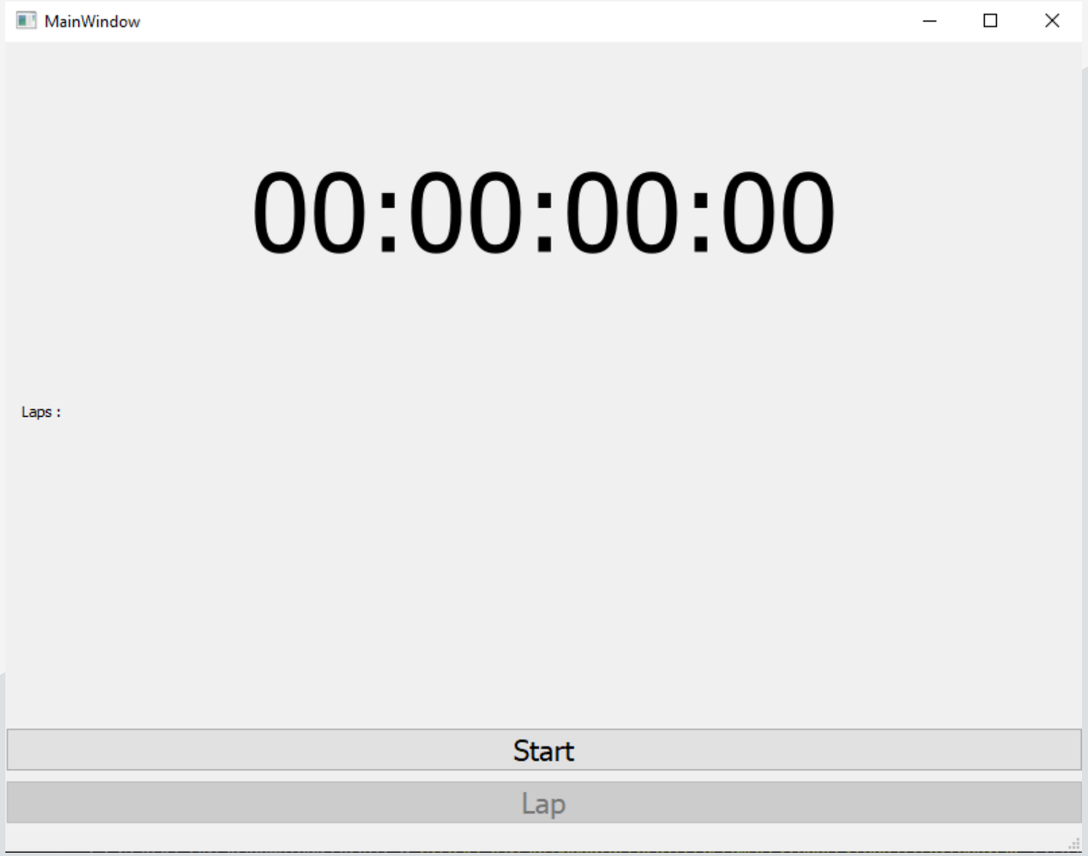
digital stopwatch coding code

RESULTS

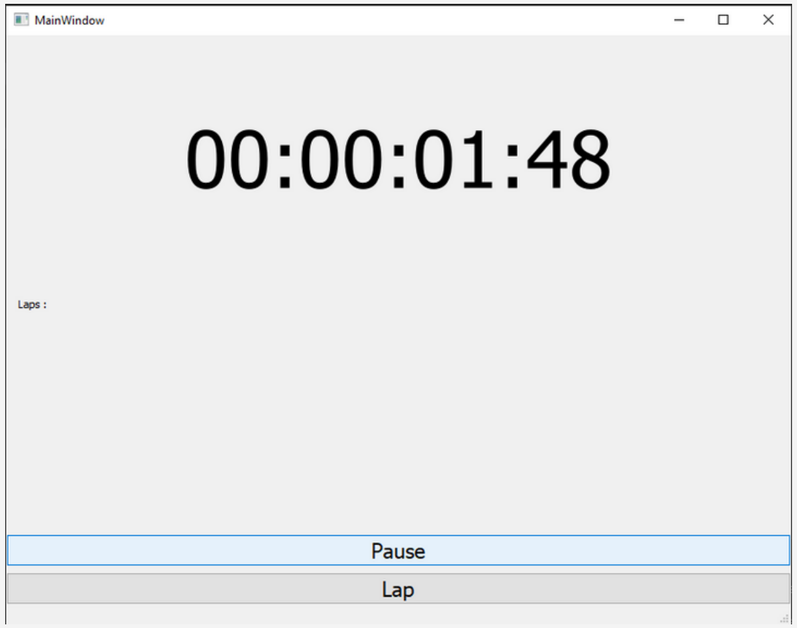


so this is the result of our digital stopwatch

Click the start button, to start the stopwatch

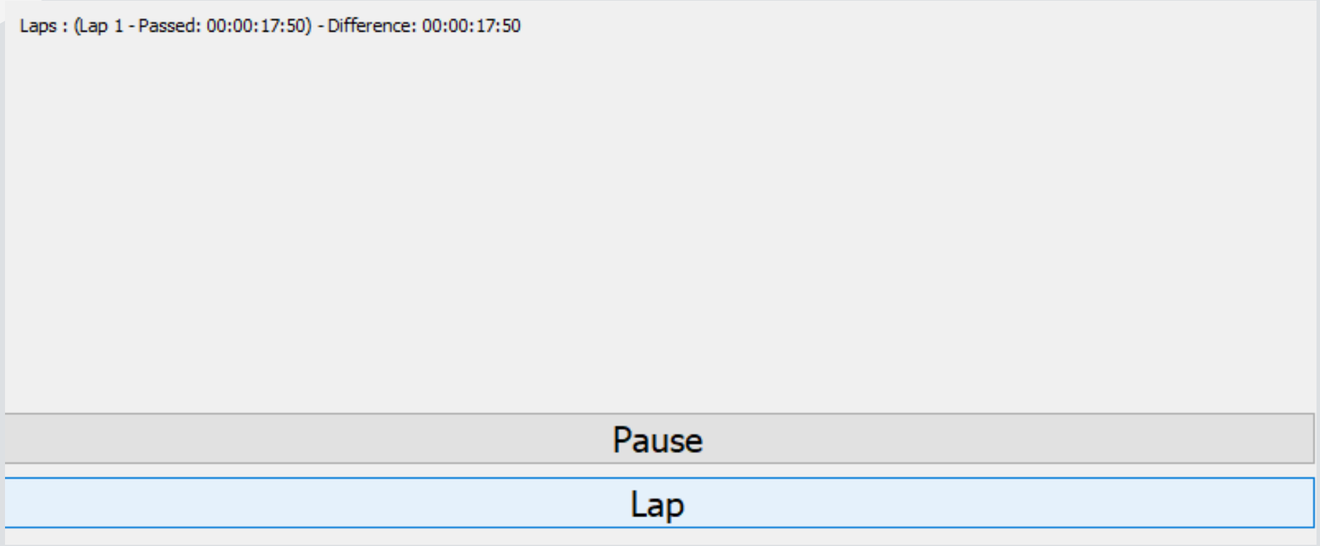


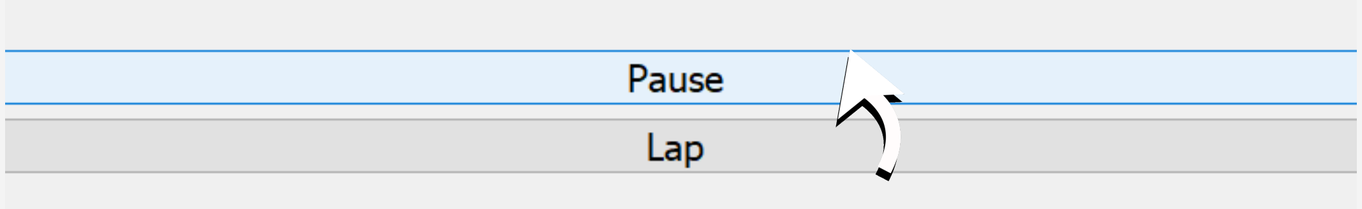
start button



After you start the stopwatch, the stopwatch will start counting. And the pause and lap buttons will appear.

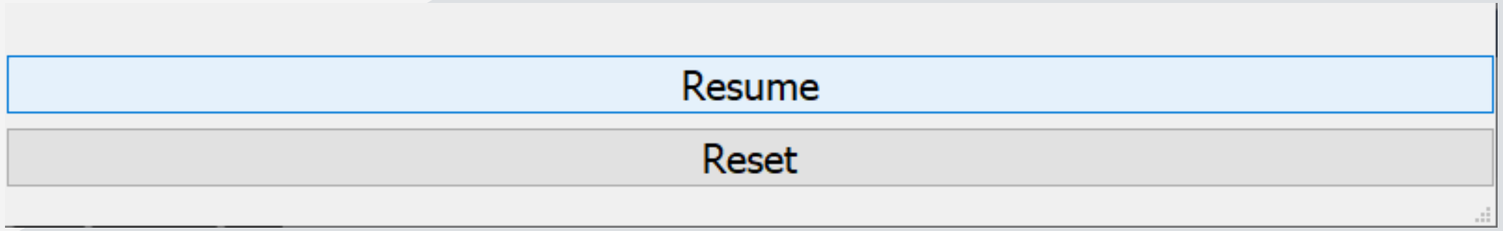
If you click the lap button, the time when you click the lap button will appear.

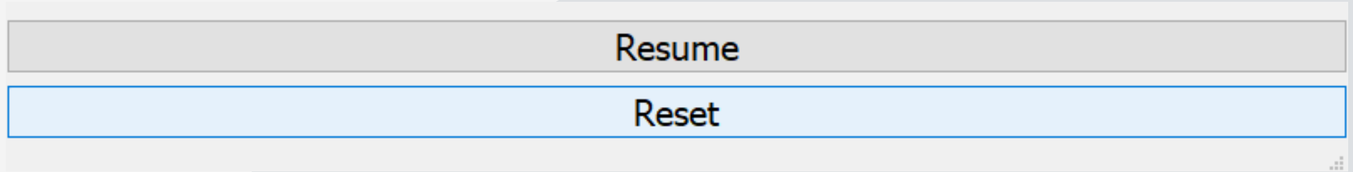




And if you click the 'pause' button, the stopwatch will stop, and show you in what second you pause the stopwatch

if you want to continue counting time, you can click the 'resume' button





you can also reset the stopwatch and start over, back from scratch

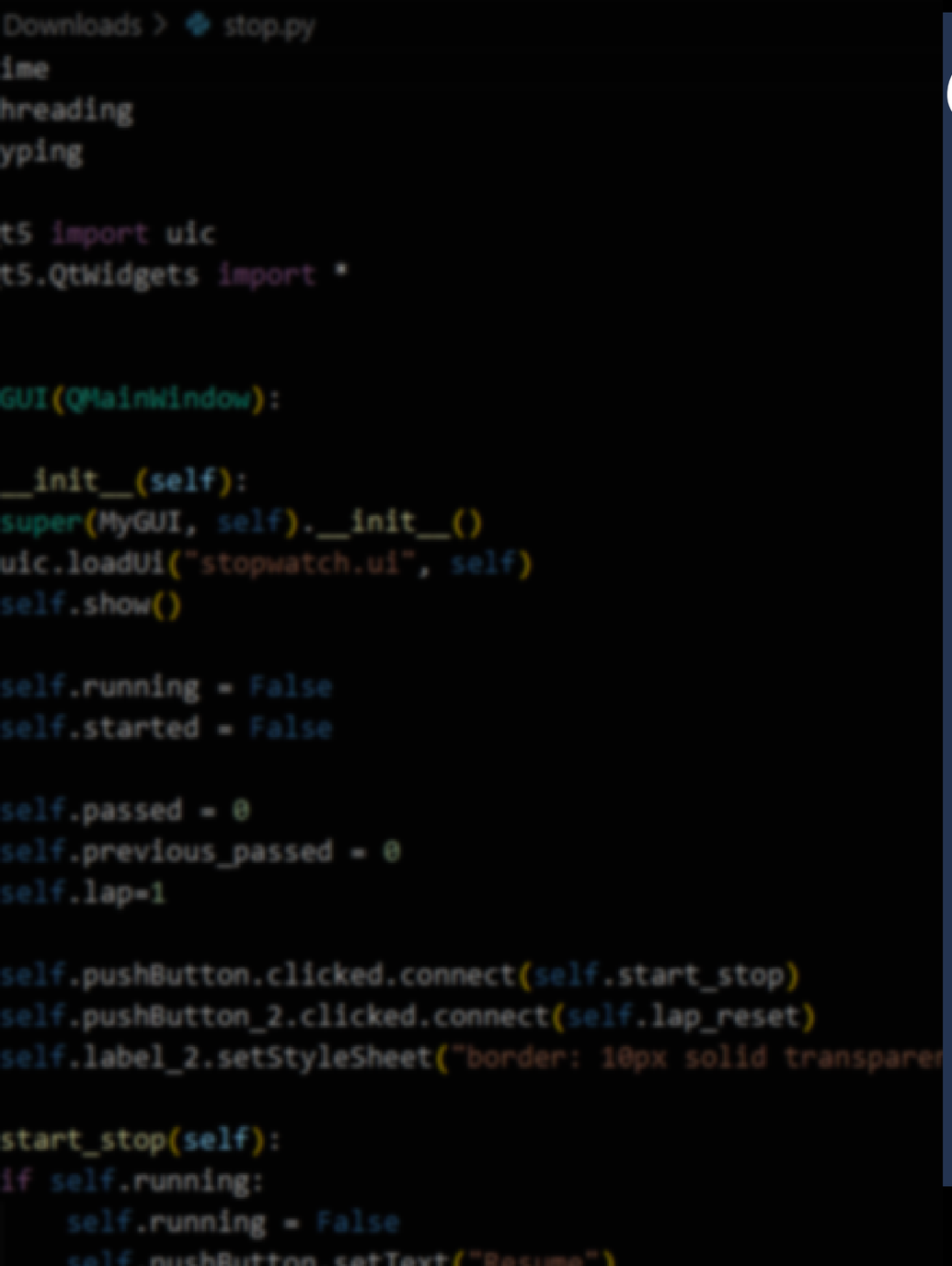
BENEFITS

Benefits for the developer:

1. Improving Application Development Skills
2. Understanding of Time Processing
3. Introduction to Supporting Features

Benefits for the user:

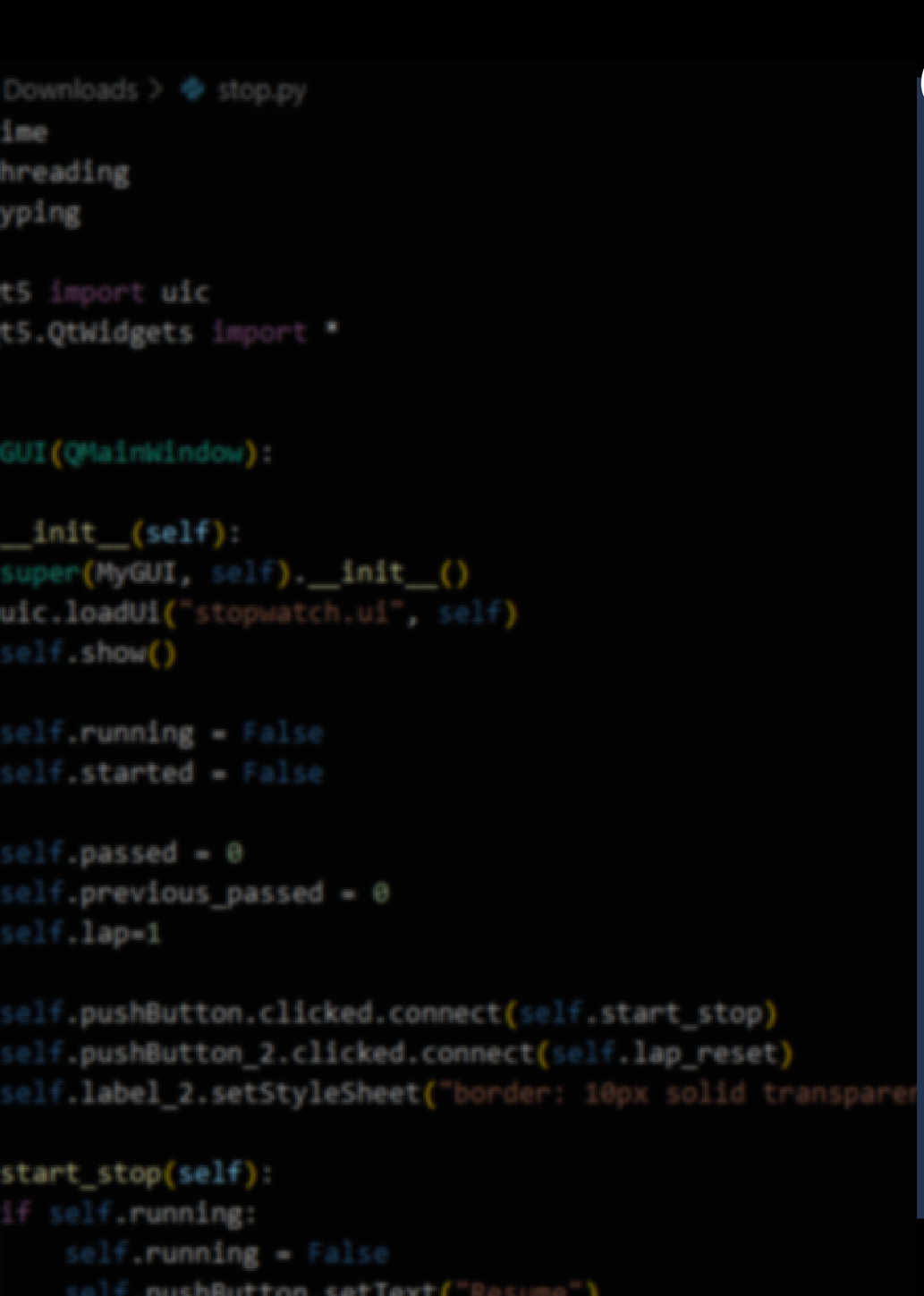
1. as a measure of time when exercising and cooking
2. to manage time
3. measure the right time



CONCLUSION

Here are some conclusions that can be drawn:

This digital stopwatch is used to measure time so that it is accurate and precise. It can start, stop and time precisely. And also has some features like, Lap, Pause, Resume and Reset. Can also calculate time in various units such as hours, minutes, seconds and milliseconds.



CONCLUSION

1. The Advantages

a.) Provides a clear and easy to understand stopwatch display.

b.) Can be executed properly and responsively, without any significant obstacles.

c.) The simple interface design makes using the application intuitive and easy to learn.

2. The Disadvantages

a.) This application does not yet provide a reminder or notification feature for alarms

b.) No option to save time history

c.) No countdown and alarm features are provided to allow users to set time precisely.

SUGGESTION

Based on the analysis above, there are several suggestions that can be given to improve this digital stopwatch application:

1. Added a reminder or notification feature for alarms, so users don't have to keep actively monitoring the time. With notifications, users can perform other activities and still get timely alerts.
2. Provides option to save previously measured time history. This will allow users to track and compare previously measured times, and access them again if needed.

SUGGESTION

3. Provides a wider range of customization options, such as custom alarm sound selection or stopwatch appearance. This will give users more choices and personalize the app experience.
4. Added countdown and alarm features to make it easier for users to count down.

THANK YOU