PerfectForm

Memorial University of Newfoundland

Computer Science 4301

Course Project Final Report

Divya Murali Krishnan (201823978),

Tarun Subramanian(201746641),

Jacob Critch (201743861)

Table of contents

Introduction

The objective of this project was to design a Python application utilizing computer vision techniques for analyzing a side-profile video of an individual doing exercise. The primary goal was to develop a tool capable of determining the posture of the subject to prevent injury and optimize workout results. For this purpose, MediaPipe Pose was employed, which is a high-fidelity body pose tracking solution that can extract 33 3D landmarks and a background segmentation mask from RGB frames. Additionally, OpenCV, an open-source computer vision library, was utilized for image and video processing.

Code and Implementation

Dependencies and requirements

The project was developed in Python, utilizing numpy, MediaPipe, OpenCV, PyQT5 and python's math library. This program will run in almost any device as long as it is connected to a camera.

Project structure

Figure 1: UML diagram for the project

The application is centered around an OpenCV VideoCapture feed loop, which takes input from the device's connected camera(s). Each frame is fed into several functions (derived from MediaPipe) which process and draw on the frame using CV functions. We developed functions to evaluate the offset distance, body posture inclination, and to dispatch alerts for poor body posture. The primary approach was to leverage the body posture detection main loop, which we modified for our specific requirements. We obtained the coordinates of the body posture landmarks, which enabled us to determine the offset distance and inclination of the subject's body posture. Our code is comprised of 3 files:

1. app.py: This script is a PyQt5-based graphical user interface for a simple application called "PerfectForm". The application allows users to select a mode (Squat or Curl), choose a camera from available cameras, and start or stop the camera. The script imports main module, which is assumed to contain the functions startCam(), stopCam(), setMode(), and setCamera().

Figure 2: UML diagram for app.py

2. Main.py: This script contains the logic for setting up the camera, processing the images using the MediaPipe library, and checking the form for the user's chosen exercise mode (Squat or Curl). The script imports a formchecker module, which is assumed to contain the function squats(). The main.py script provides functions to start and stop the camera, set the exercise mode, and set the camera index.

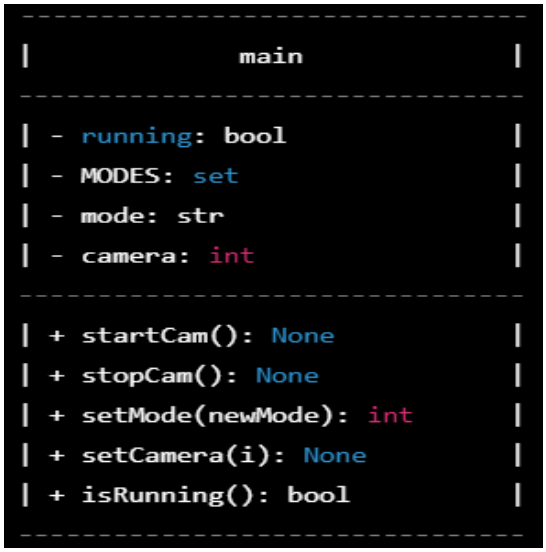

Figure 3: UML diagram for main.py

3. Formchecker.py: Using MediaPipe libraries for pose and hand landmark detections, this script contains utility functions to calculate angles, palm normal vectors, and checks if the user should start the exercise based on their hand position. It then checks if the person is standing sideways and in the correct starting position, as well as evaluates their posture during the descent, ascent, and bottom of a squat.

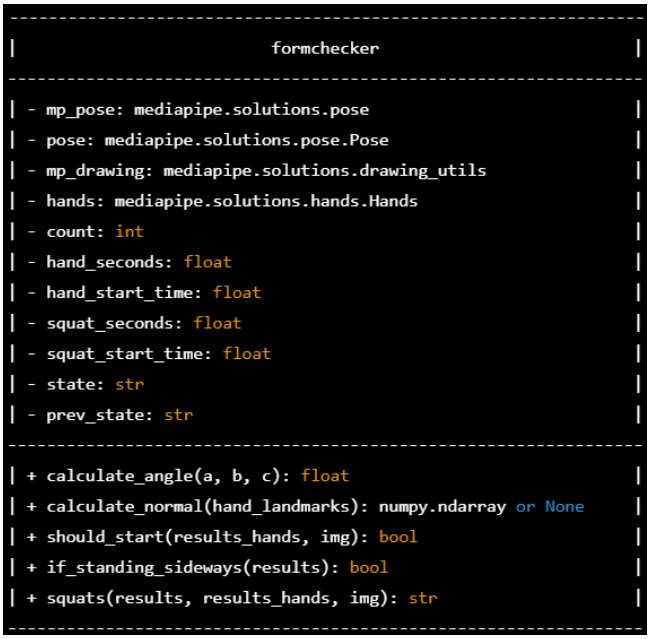

Figure 4: UML diagram for formchecker.py

Flow of control

The main execution starts in main.py. It initializes and sets up the user interface, as well as the video stream using OpenCV. In the main loop, main.py reads frames from the video stream and processes them. For each frame, it calls the process_frame() function in mediapipe_pose.py. The process_frame() function in mediapipe_pose.py uses the MediaPipe library to detect the pose and hand landmarks in the input frame. It returns the processed frame, pose landmarks, and hand landmarks to main.py.

Back in main.py, it calls the squats() function in formchecker.py with the pose landmarks, hand landmarks, and the input frame as arguments.

The squats() function in formchecker.py performs several checks and calculations to analyze the user's squat form. It calls helper functions like if_standing_sideways(), should_start(), and calculate_angle() from the same file, and interacts with mediapipe_pose.py to access the pose landmarks. Based on the results of the checks and calculations, the squats() function updates the global state variables representing the current squat state (starting, descending, bottom, ascending, or idk). It also updates the squat count and provides feedback on the user's form, if necessary. The squats() function then returns the feedback status to main.py, which displays the status and squat count on the frame using OpenCV's cv2.putText() function.

Finally, main.py shows the processed frame with the overlaid pose landmarks, hand landmarks, and feedback to the user. This process repeats for each frame in the video stream.

In summary, the flow of control starts in main.py and moves back and forth between formchecker.py and mediapipe_pose.py to process the frames, analyze the user's squat form, and provide real-time feedback.

Figure 5 shows the sequence diagram of our code when the app is initiated and set to the default device camera, with squat chosen as our work out.

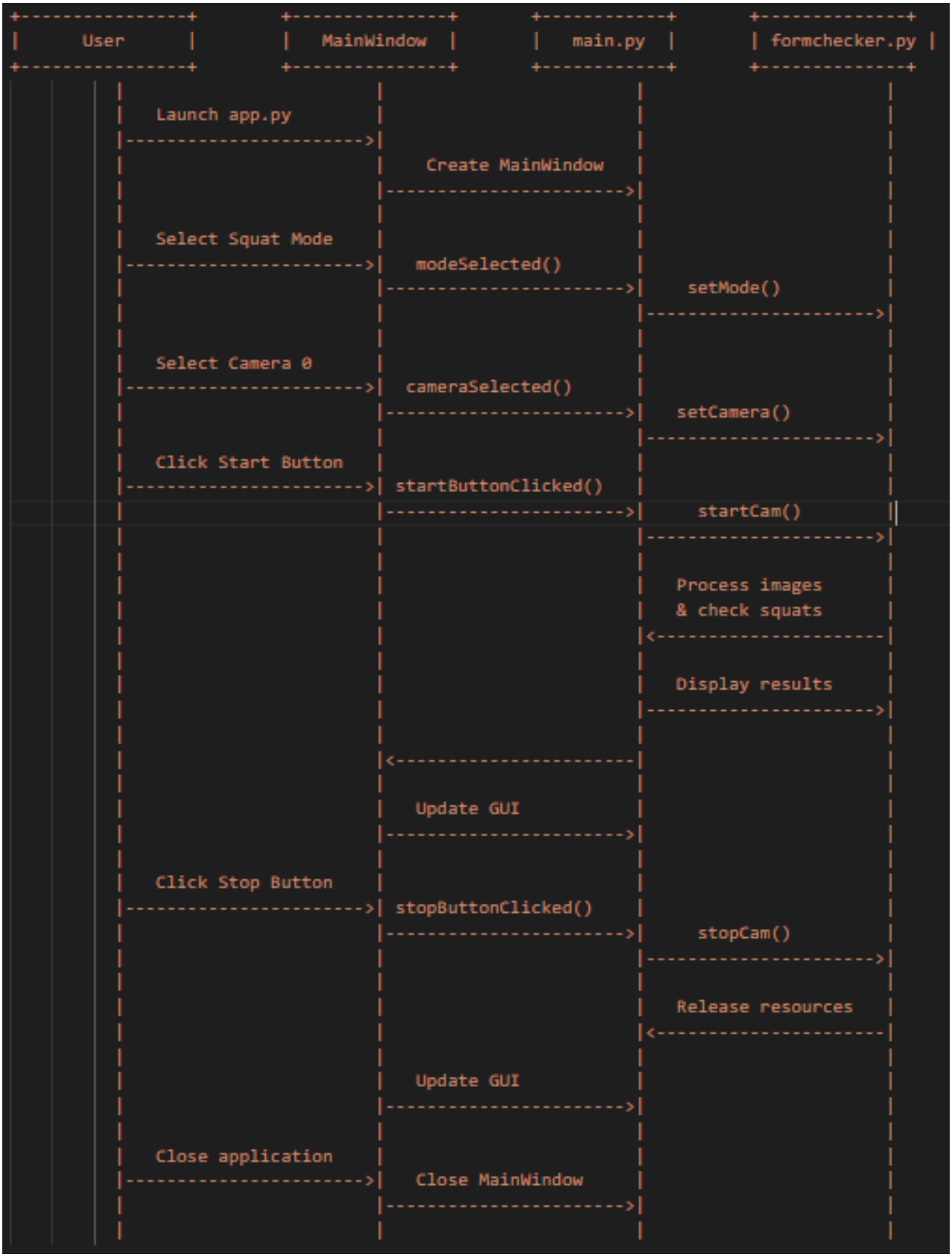

Figure 5: Sequence diagram when app is initiated using default camera on squat mode

Within the function that checks how the squats is done, the flow of control is as follows:
The squats() function is called, which first calls should_start() function to check if user wants to start the workout. It then calls if_sideways() to check if the person is standing sideways. If both these conditions are satisfied, it then checks and assigns squat states to indicate whether the person is starting a squat, descencding, at the bottom of the squat or ascending. We have a variable, 'count' that counts the number of repetitions of the exercise. In each state, the program checks knee, back, shoulder and feet posture using mediapipe pose landmarks. It also checks if the person is holding the squat in the 'bottom' state for a minimum of 3 seconds.

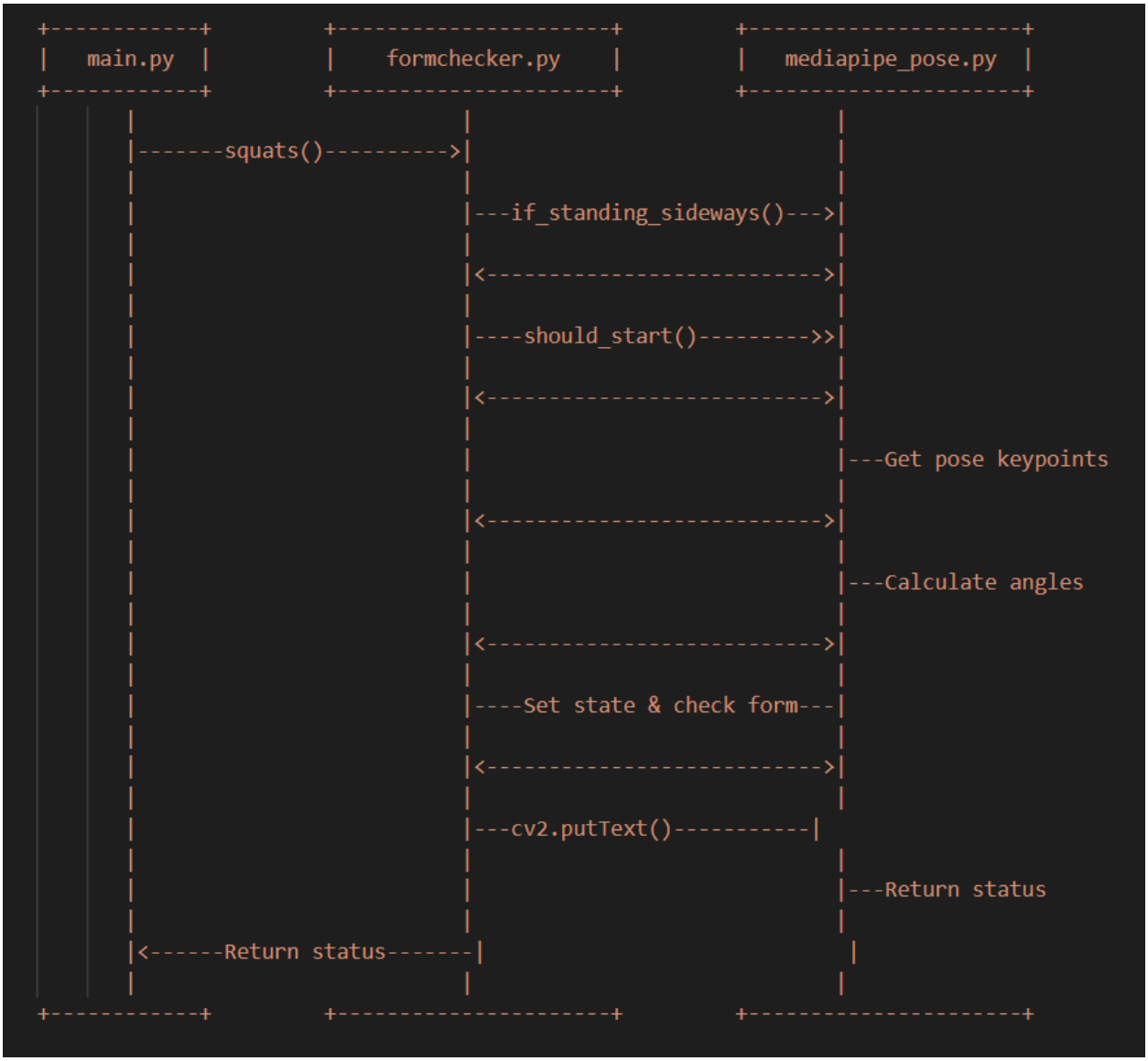

Figure 6: Sequence diagram showing flow of control within formchecker.py, when the squats() function is called

Results

Our Python application effectively detects the posture of someone working out by analyzing an uploaded video of a side-profile. The tool can be used as a workout aid to get better results and avoid injury.

At the moment, the code is able to check if the person is holding out their palm for 3 seconds and start the exercise check once this is satisfied.

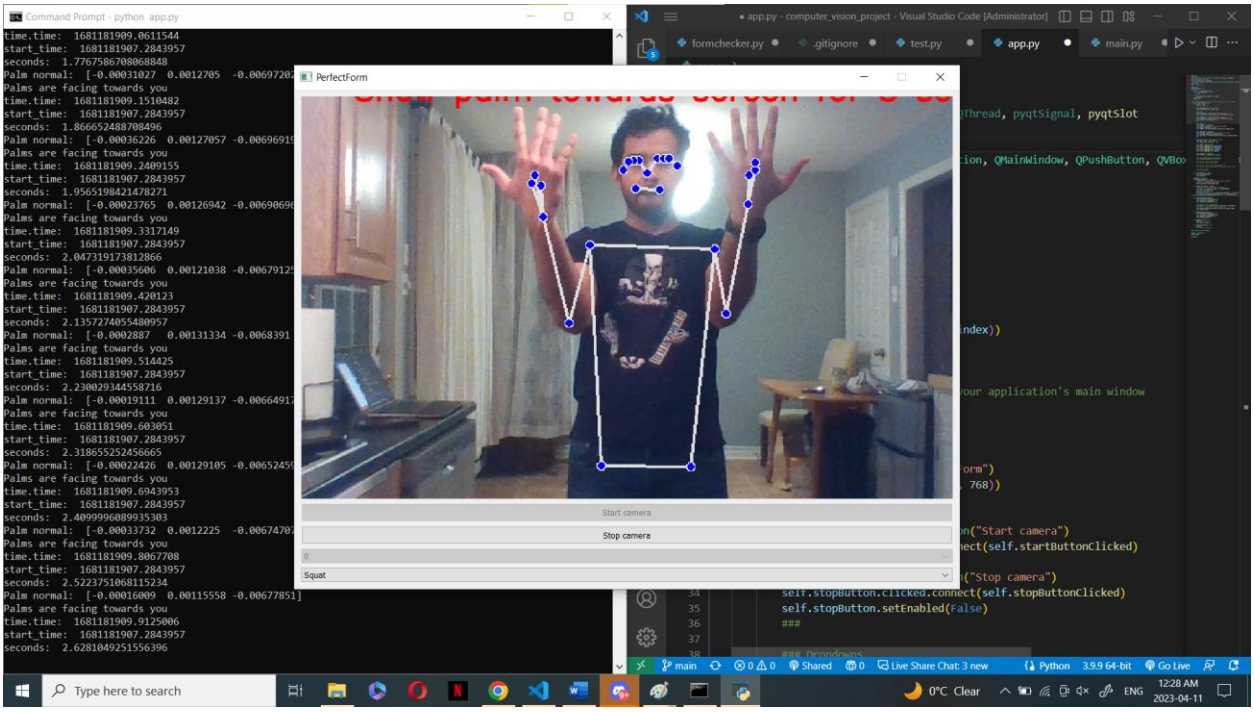

Figure 7: Program checks if hands are held either facing directly towards or away from camera for 3 seconds

The program then prompts the person to stand sideways.

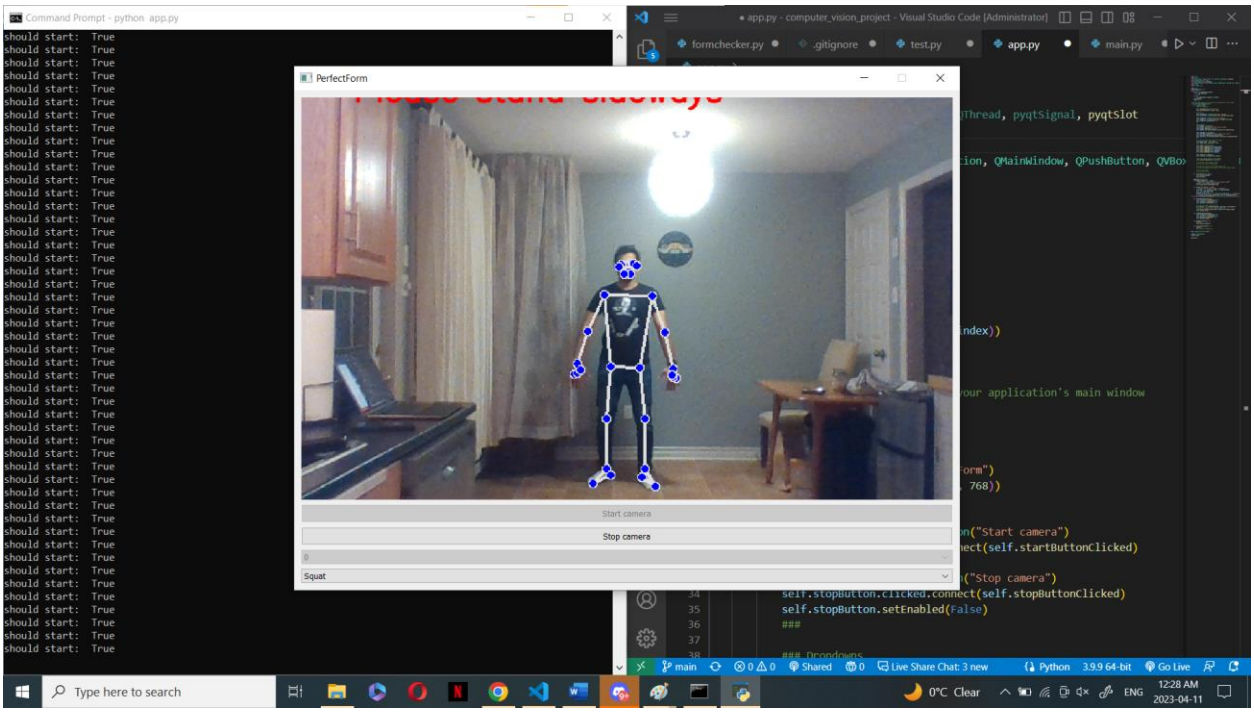

Figure 8: If person is not standing sideways, there is a prompt on the top of the screen that asks the person to stand facing sideways

Following this, is enters the 4 different states: starting, ascending, bottom and descending based on the knee angle and previous state.

In each state, the posture is checked using the following parameters:

1. Feet must be shoulder width apart and parallel to each other.
2. Back and shoulders must be straight.
3. The angle between shoulder, hip and knee are constantly checked with varying thresholds depending on state.
4. The angle between hip, knee and ankle are constantly checked with varying thresholds depending on state.

If the user holds a good squat state for 3 seconds at ‘bottom’ state, it counts as 1 repetition.

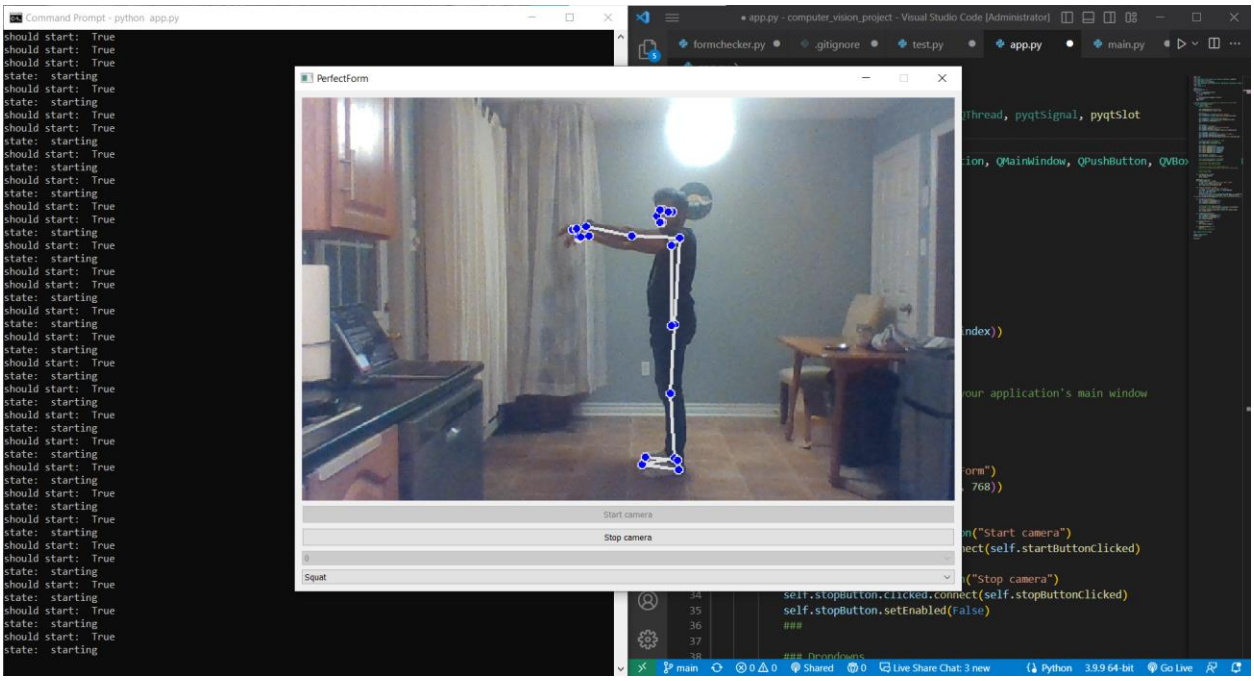

Figure 9: User standing sideways, ready to squat

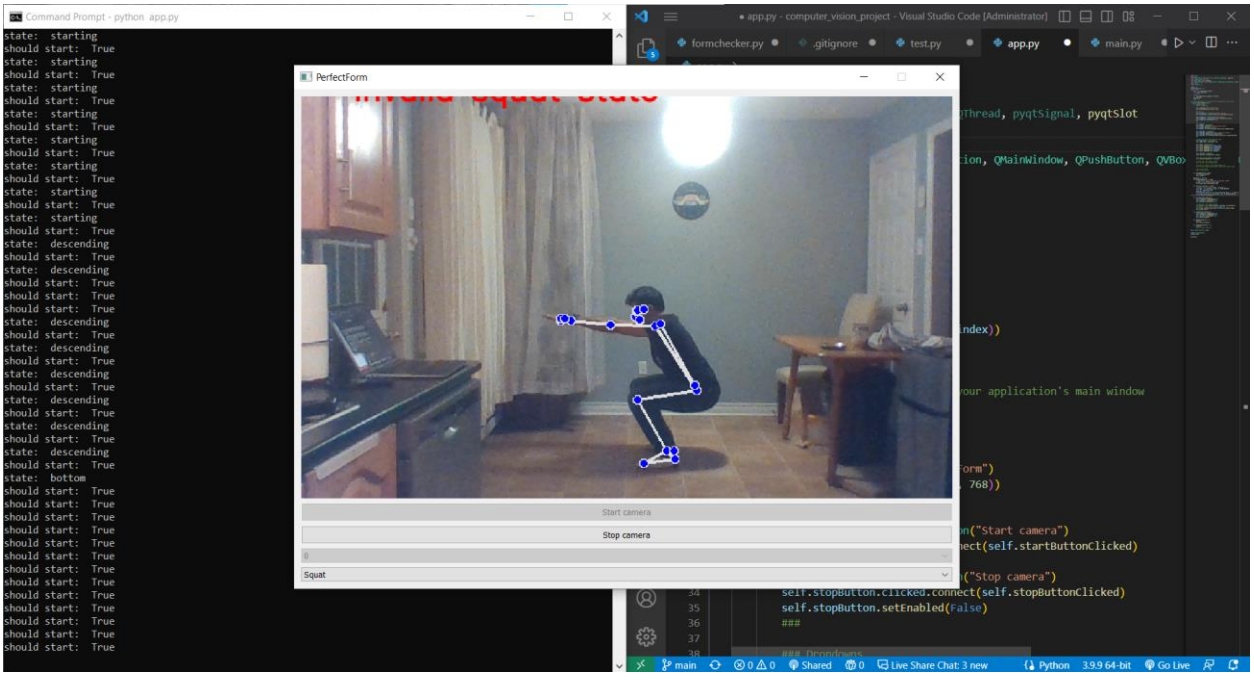

Figure 10: User in the squat position for 3 seconds, count is incremented

Future work

This app was intended to be used initially as a workout form checker, to allow the app's users to workout while ensuring that they're form is optimal to ensure best results and minimize injuries. Hence, the next step would be to expand on our library of exercises to accommodate for as many exercises as possible. We would also like to add a feature that allows users to upload their video, not just use real time feed. Further, we would like to personalize landmarks and angles to accommodate for different BMIs and body types. This app can also be extended towards exercised geared towards physiotherapy, to allow users to optimize their recovery during physiotherapy.

Outside the scope of this course, we would like to add features such as work-out and physiotherapy suggestions, etc. based on user's interest and requirement.

Challenges & Conclusions

The biggest challenge with developing this project was to handle real time inputs. Since the code was constantly in a while loop for as long as the camera was on, we had to be careful about exit conditions and entry conditions. Moreover, displaying messages, frame capture and analysis had to be done very carefully to ensure the highest number of frames read to have accurate outputs and calculations.

However, both MediaPipe and OpenCV have a very well documented user guide. The community of users was also wide and very helpful and hence we were able to successfully complete this project.

In conclusion, we have successfully developed a desktop Python application that utilizes computer vision techniques to analyze an uploaded video of a side-profile of someone working out. We used MediaPipe Pose and OpenCV as our primary packages and created several functions to calculate offset distance, body posture inclination, and to send alerts for poor body posture. This tool has the potential to be used as a workout aid to help users achieve better results and avoid injury.

References:

"MediaPipe Pose," Google Developers, https://mediapipe.dev/.

"Building a Body Posture Analysis System using MediaPipe," MediaPipe, https://learnopencv.com/building-a-body-posture-analysis-system-using-mediapipe/

"OpenCV: OpenCV-Python Tutorials," OpenCV, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html

“Getting Started With PyQt5”. PythonGUIs https://www.pythonguis.com/pyqt5-tutorial/#start