

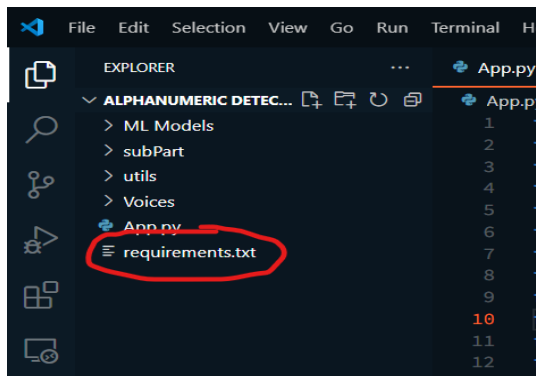
## MANUAL GUIDE

### Before running the program:

1. First, install Python version 3.10 (recommended), and later
2. Install Ide, either PyCharm or Visual Studio Code (recommended)
3. Your device should have a good CPU.

### How to Download Dependencies (Visual Studio Code):

1. Open the program folder, then find the requirements.txt

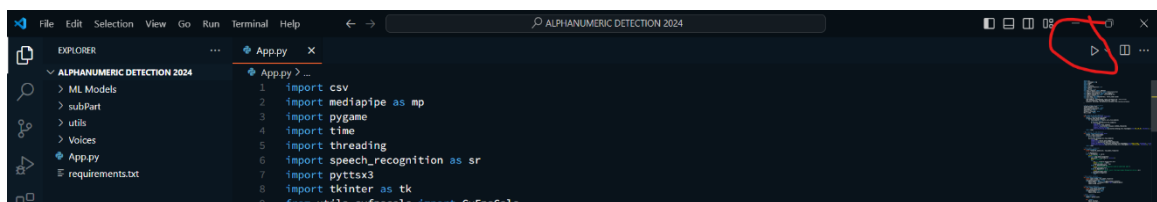


2. Open the terminal by pressing “ctrl + `” on your keyboard, then copy and paste this command into your terminal: “pip install -r requirements.txt” and then hit enter.
3. Just wait until all dependencies are downloaded and completed. Then, you can run the program 🎉.

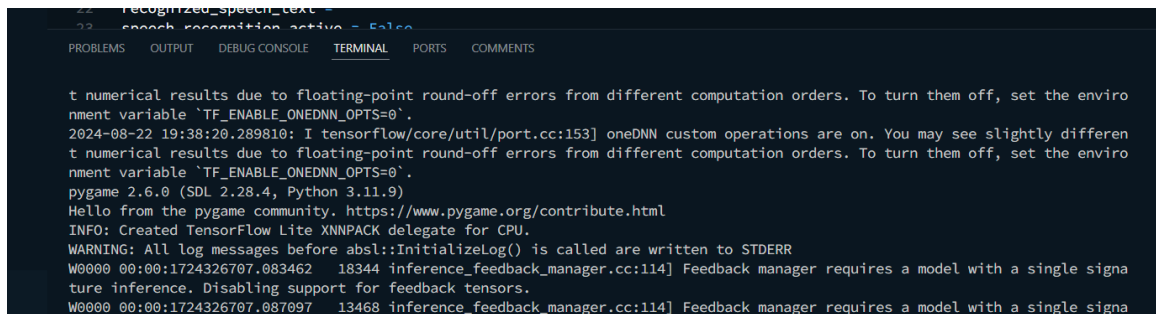
**Note:** There are two methods for starting the program. As part of our thesis requirements, we needed to implement a prototype, hence the separate codes. One code is for using the user's device only, and the other is for running the program with the prototype included.

### Run the program for Device user only:

1. Find the App.py file, then open that file. You will see a bunch of code. Do not do anything to ensure it will not get any error, but if you know what you are doing, it's fine 👍. Hit the play button in the top right.

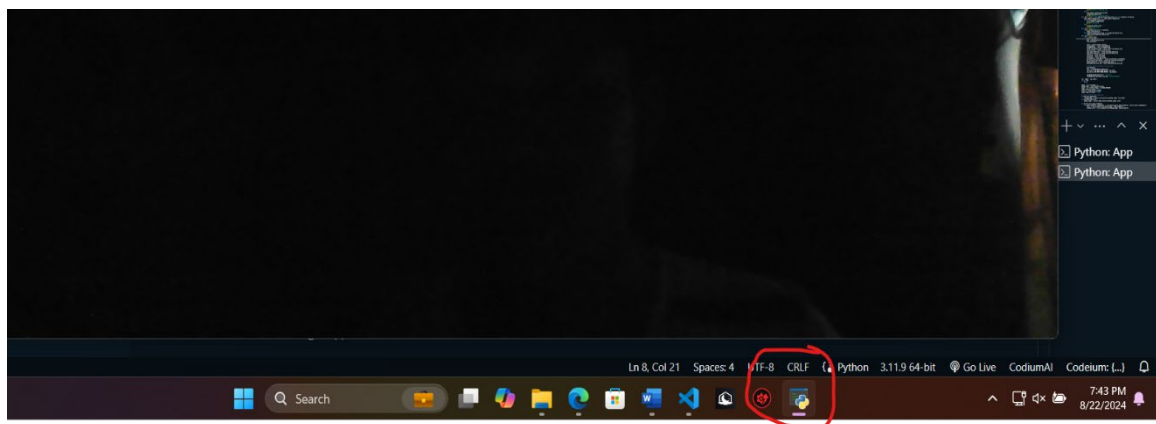


2. You may encounter a message like the one in the Visual Studio code terminal. Don't be concerned; it's simply a suggestion from the framework I'm using, and it won't impact the model's performance or prevent the code from functioning



```
22 recognized_speech_text =  
23 speech_recognition_active = False  
  
t numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.  
2024-08-22 19:38:20.289810: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.  
pygame 2.6.0 (SDL 2.28.4, Python 3.11.9)  
Hello from the pygame community. https://www.pygame.org/contribute.html  
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.  
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR  
W0000 00:00:1724326707.083462 18344 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback tensors.  
W0000 00:00:1724326707.087097 13468 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback tensors.
```

3. The OpenCV frame will appear. Please wait, as the loading time depends on your CPU speed and how quickly the model and other components in the code are loaded. The screenshot below shows that the program has loaded and OpenCV is running.

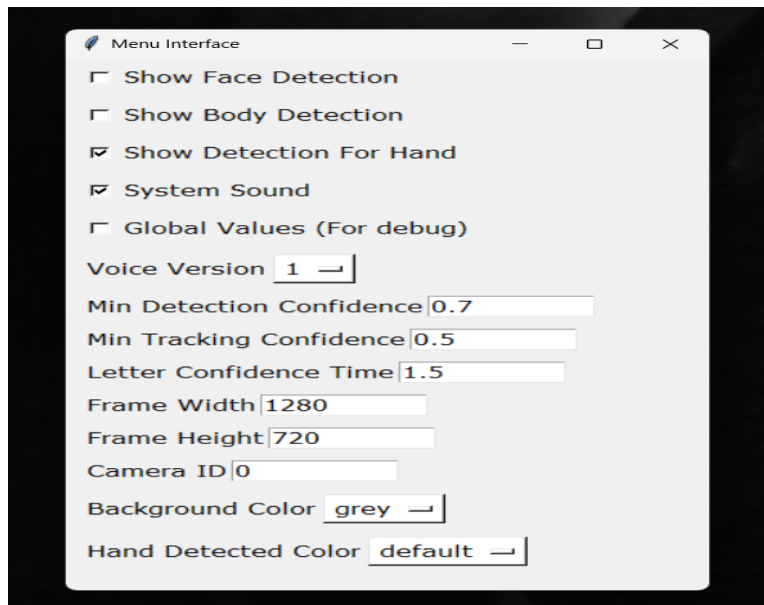


4. You're all set! You can now try our latest model for sign language recognition, specifically designed for A-Z, 0-10, and the 'I Love You' gesture. 😊

## To Customize the program settings:

### 1. Menu Settings

To access the program's menu settings, first ensure it is running. Then, simply press the 'm' key on your keyboard to open the menu interface. You can exit the menu by clicking the exit button or pressing 'm' again. Feel free to experiment with changing the values; any changes will take effect immediately. However, please note that your changes will not be saved. If you rerun the program, the settings will revert to their default values. Here is an image of the menu for your reference.



## 2. Change the default setting values.

If you wish to modify the default values, locate the "utils" folder and navigate to the "config.py" file. From there, you can customize the values to your liking."

### Setting Explanation:

1. **showLandMarks:** This variable is a boolean value (True or False) that determines whether to display landmarks on the detected hand or not. If set to True, it will draw circles at key points on the hand, such as the wrist, fingertips, and joints. These landmarks can be useful for understanding the hand's posture and movement.
2. **soundDetected:** This boolean variable controls the sound output. If set to True, the program may play sounds or provide audio feedback when certain events occur, such as hand detection or gesture recognition.
3. **letter\_confidence\_time:** This variable sets the time threshold (in seconds) for letter confidence. It determines how long the program should wait after detecting a gesture before recognizing it as a valid input. This can help improve gesture recognition accuracy by ignoring brief or unintentional movements.
4. **voice\_version:** This integer variable indicates the version of the voice model used for speech recognition or synthesis. Different versions may offer improved accuracy or additional features.

5. **show\_face\_detection** and **show\_body\_detection**: These boolean variables control whether to visually display face detection and body detection results, respectively. If set to True, the program will draw bounding boxes or markers on the video feed to indicate detected faces or bodies.
6. **camera\_id**: This integer variable specifies the ID of the camera device to be used. Different cameras connected to the system may have different IDs. Setting this variable allows you to select the desired camera as the input source.
7. **cap\_width** and **cap\_height**: These integer variables set the width and height of the video frame, respectively, in pixels. You can adjust these values to change the resolution or aspect ratio of the captured video feed.
8. **min\_detection\_confidence** and **min\_tracking\_confidence**: These floating-point variables set the minimum confidence thresholds for object detection and tracking, respectively. Confidence values typically range from 0.0 to 1.0, where higher values indicate higher certainty. You can adjust these thresholds to control the trade-off between accuracy and false positives in detection and tracking.
9. **show\_terminal\_output**: This boolean variable controls whether to display terminal output. The program will print the global variables or setting values if set to True.
10. **background\_color\_option** and **background\_color**: These variables control the background color of the display or user interface. `background_color_option` seems to be a string that specifies a predefined color option, such as "grey." `background_color` is an RGB tuple (three integer values) that directly sets the background color.
11. **hand\_detected\_color\_bg** and **hand\_detected\_color\_option**: These variables are like the background color settings but specifically relate to the color used when a hand is detected. `hand_detected_color_bg` is an RGB tuple, and `hand_detected_color_option` seems to be a string that selects a predefined color option.

The program I made also offers an additional feature that allows you to construct sentences. It recognizes gestures, constructs sentences, and includes text-to-speech and speech-to-text conversion functionalities. However, the logic for this functionality still needs some fixing and better implementation 🙌 .

## **Additional Features:**

**Note:** The program should be running; this feature is only available for letters (right hand). These features are experimental, so you may encounter bugs and logical issues. The speak-to-speech feature is likely more unstable and has more bugs than the letter-to-sentence function.

1. On your keyboard, click the letter “d” to run the function for sentence construction.

### **How to use this function:**

1. Think first about what short sentence you want to construct and then remember the words of that sentence, then next the letter of each word.
2. keep your hand gesture and wait for 1.5 seconds (the time will depend on the letter confidence value); the program will save that letter on the word label.
3. If you saved the wrong letter in the word label, don't worry, I got you. Click the letter “f” on your keyboard to delete the current letter in the word label. It is only accessible for word labels, not for sentence labels.
4. If you see that you already have one word on the word label, now do space by making a hand gesture for space (just make the like hand gesture in your right hand 👉 ) to put the word in the sentence label.
5. Just continue the process until you are satisfied with your sentence. Then here you go, you've finally constructed a sentence 😊 .
6. You can also click the letter “r on your keyboard. Your device will read your constructed sentences.

**To exit:** Just click again the letter “d”

2. Click the letter “s” on your keyboard to text speech functionality.

### **How to use this function:**

1. On your keyboard, click the letter 'l' to start manual listening, then speak into the microphone on your device. When you are done talking, click the letter 'l' again to stop listening and proceed to recognize your speech. The sentences will appear on the screen.

2. If you want automatic listening, click 'a'. It will automatically use the same process as manual listening in the previous step. There's no need to click again. Just speak into the microphone on your device, and it will automatically perform the text-to-speech function.

**To exit:** Just click again the letter “s”

This manual was written by me, the developer of this program, along with my apprentice, who contributed significantly throughout the entire process of building the prototype from the start, including assisting with code and writing the manuscript