



ÜSKÜDAR UNIVERSITY ★ FACULTY OF ENGINEERING AND NATURAL SCIENCES

Real Time Reduce Noise by Reading Lips

B.Sc. Thesis by: Hamza Çeçen

Thesis Supervisor: Dr. Kristin Surpuhi Benli

SOFTWARE ENGINEERING DEPARTMENT

June/2024

Contents

1. Introduction:	5
1.1 Purpose of the Thesis:	5
1.2 Background	6
1.3 Hypothesis	6
1.4 Objectives	6
2. MATERIALS AND METHODS	7
2.1 Libraries:	7
2.2 Database:	7
2.3 Image Processing:	10
2.4 PyAudio:	13
2.5 Socket:	13
2.6 Speech Recognition	15
3. RESULTS AND DISSCUSSION	17
3.1 Application of Work	17
3.2 Results:	22
3.3 Discussion:	22
4. CHALLENGES AND FUTURE IMPROVEMENTS	23
5. CONCLUSION	25
6. REFERENCES	26

Figure 1 - Project	5
Figure 2 - Libraries	7
Figure 3 - MongoDB.....	7
Figure 4 - Datas for User Interface	8
Figure 5 - Connect to MongoDB Codes	8
Figure 6 - Code Structure: Aligning User Interface with MongoDB	9
Figure 7 - Facial Landmark.....	10
Figure 8 - Code of Landmark	10
Figure 9 - Display of Current Distance of the Lip.....	11
Figure 10 - Code of Drawing Circle in Lip.....	11
Figure 11 - Result of Drawing Lip.....	11
Figure 12 - Code for Talking.....	12
Figure 13 - Code for Not Talking	12
Figure 14 - Reduce Noise Schema	13
Figure 15 - Code Structure: Non-Stationary Reduce Noise	13
Figure 16 - Socket	14
Figure 17 - Code Structure: Socket Client.....	14
Figure 18 - Speech Recognition Schema	15
Figure 19 - Code Structure: Transform from Speech to Text.....	15
Figure 20 - Transformed Text in the Label	16
Figure 21 - Filling boxes	17
Figure 22 - Error: Not selected the role.....	17
Figure 23 - Error: Wrong username or password.....	17
Figure 24 - The Main Window	18
Figure 25 - If not talking	19
Figure 26 - If mouth closed.....	19
Figure 27 - If it is talking	20
Figure 28 - Client and Server Connecting.....	20
Figure 29 - Stopped Camera and Microphone	21
Figure 30 - Recordings and note in the file	21
Figure 31 - Logger Note	21
Figure 32 - Current Project Schema	22
Figure 33 - Expecting Project Schema (Fail)	23
Figure 34 - Expecting Code: To put reduce noise class.....	24
Figure 35 - Result of expecting code (fail)	24

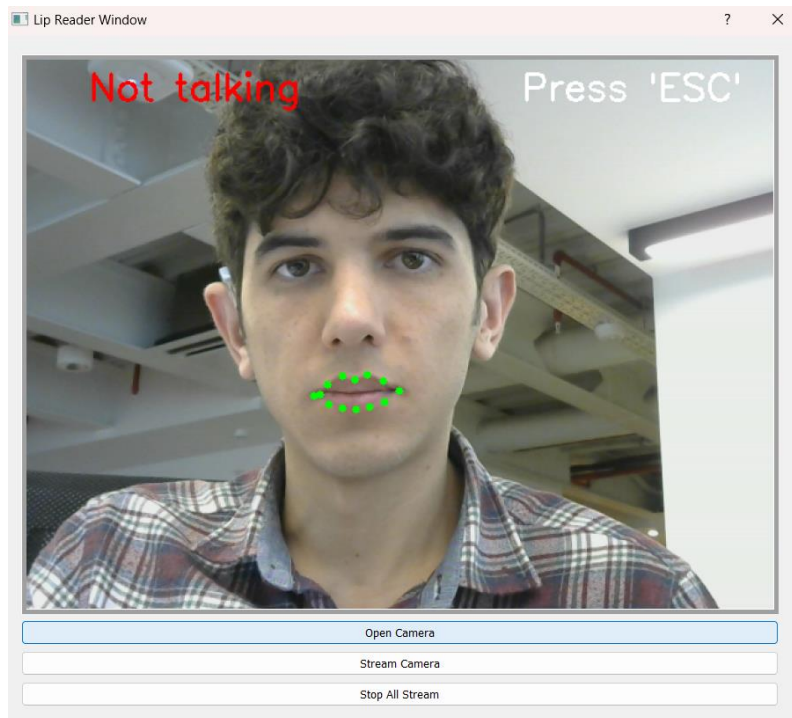


Figure 1 - Project

1. Introduction:

Abstract:

The goal of this project is to build an image processing system that can accurately detect movement of the mouth and start to reduce noise when we speak in the background.

Problem:

According to the WHO, more than 1.5 billion people or approximately 20% of the global population live with hearing loss. Of those 1.5 billion people, 430 million of them have disabling hearing loss and suffer from deafness or hard-of-hearing. When we look at the data on the number of hearing impaired people in our country; The number of disabled people who are alive and registered in the National Disability Data System created by the Ministry of Family and Social Services; It is seen that there are 2,511,950 people, 1,414,643 of whom are men and 1,097,307 of whom are women, and 179,867 of them are hearing impaired. Thanks to advances in technology, hearing aids continue to improve. However, some parts are still underdeveloped. For example, a deaf person who has hearing aids has difficulty talking to the other party in a noisy environment. Given the this problem, I propose using computer vision and audio library to help the deaf and hard-of-hearing in noise environment by creating an algorithm that can reduce noise by lip movements.

1.1 Purpose of the Thesis:

The primary objective of this thesis is to develop an innovative image processing system capable of accurately detecting and tracking mouth movements while simultaneously reducing background noise during speech. By leveraging advanced computational

techniques, the proposed system aims to enhance the clarity and intelligibility of audio signals in noisy environments, thereby improving overall communication experiences for users.

1.2 Background

Despite improvements in hearing aid technology, challenges persist for individuals with hearing impairments, especially in noisy environments. For instance, background noise often hinders communication even with hearing aids. To address this issue, this thesis proposes using computer vision and audio library technologies to create an algorithm that reduces noise based on lip movements, aiming to improve communication for the deaf and hard-of-hearing in noisy environments.

1.3 Hypothesis

It is hypothesized that by integrating real-time mouth movement detection with noise reduction algorithms, the proposed image processing system will effectively enhance speech clarity and intelligibility in noisy environments. It is further hypothesized that the system's performance will be superior to existing methods, thereby offering a promising solution for improving communication accessibility for individuals with hearing impairments.

1.4 Objectives

The objectives of this thesis are as follows:

- To design and implement an image processing system to detecting mouth movements.
- To develop algorithms for reducing background noise during speech.
- To evaluate the performance of the proposed system through quantitative analysis and comparison with existing methods.
- To assess the practical feasibility and usability of the developed system in real-world scenarios.
- To contribute to the advancement of audiovisual communication technology by providing a novel solution for improving speech clarity in noisy environments.

2. MATERIALS AND METHODS

2.1 Libraries:

I imported several libraries to facilitate the development of the proposed image processing system. Specifically, OpenCV was utilized for the precise detection of lip movements, while PyAudio played a crucial role in effectively reducing background noise during speech. Additionally, I utilized complementary libraries such as numpy and wave to support various data processing tasks essential to the project. Furthermore, I used PyQt5 in creating an intuitive interfaces for seamless interaction with the developed system.



Figure 2 - Libraries

2.2 Database:

The initial interface of the system features a user authentication module where users are required to input their username and password. This information is then matched with the database stored in MongoDB. Upon successful authentication, users gain access to the main interface, serving as the central hub for interacting with the system's functionalities.

Even though my databases are similar to SQL databases, I preferred MongoDB, which is a NoSQL database, because I have experience with this database.



Figure 3 - MongoDB

There is no sign-up for the user interface to work. Therefore, the username, password, and role information are entered manually in MongoDB.

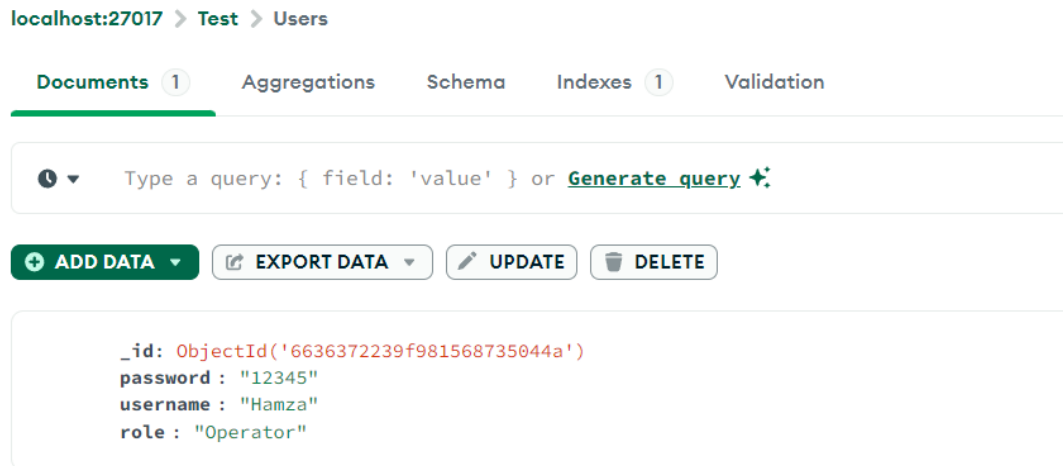


Figure 4 - Datas for User Interface

When the code is executed, it automatically connects to MongoDB. Here is the code:

```
def Connect_To_MongoDB(self):
    self.client = MongoClient('mongodb://localhost:27017/')

    self.db = self.client['Test']
    self.collection = self.db['Users']

    # Write the information to the logger
    logging = Logger.Log()
    logging.info("Connected to MongoDB.")
```

Figure 5 - Connect to MongoDB Codes

After selecting the role and filling in the username and password, if they match the records in MongoDB, this interface will close and the main window will open. Here is the code:

```

1 usage
def enter(self):
    selected_role = self.cmbChoice.currentText()

    # If the role box is empty
    if selected_role == "":
        message.message_user_role_error()
        logging.info("It showed that User need to select role type.")
    else: # If the role box is selected
        username_document = self.collection.find_one({'role': selected_role})
        if username_document:
            username = username_document.get('username')
            password = username_document.get('password')
            # print(f"User name: {username}, Password: {password}")
            input_username = self.txtUsername.toPlainText()
            input_password = self.txtPassword.toPlainText()
            if input_username == username:
                if input_password == password:
                    logging.info("Username and password are correct. Main window is opened!")
                    self.close()
                    self.mainWindow = mainView.MyApp()
                    self.mainWindow.show()
            else:
                logging.info("User not found.")
                message.user_not_found()

```

Figure 6 - Code Structure: Aligning User Interface with MongoDB

2.3 Image Processing:

In this stage of the project, the focus is on lip segmentation, a critical step towards translating lip movements into words. The process involves isolating just the lips and the surrounding area from the video stream. To achieve this, I defined constant values for the width and height of the lip segment. After successfully segmenting out the lips, padding is applied to the segmented image to ensure it matches the predefined constant dimensions.

Here's an overview of the steps involved:

- 1- Utilizing the computer's webcam and OpenCV library to stream live video footage of the speaker.
- 2- Calculating the distance between the top and bottom lips, crucial for accurate lip segmentation.

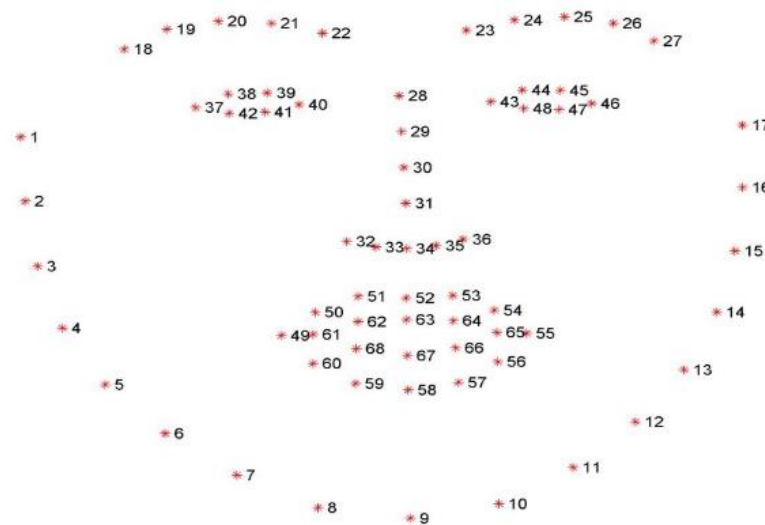


Figure 7 - Facial Landmark

```
# Calculate the distance between the upper and lower lip landmarks
mouth_top = (landmarks.part(51).x, landmarks.part(51).y)
mouth_bottom = (landmarks.part(57).x, landmarks.part(57).y)
lip_distance = math.hypot(mouth_bottom[0] - mouth_top[0], mouth_bottom[1] - mouth_top[1])
```

Figure 8 - Code of Landmark

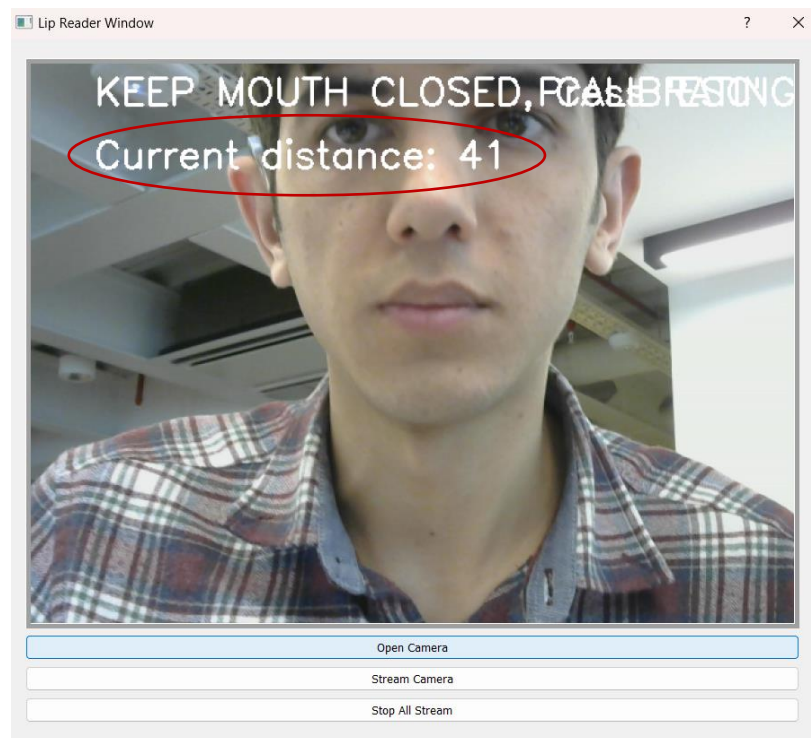


Figure 9 - Display of Current Distance of the Lip

- 3- Implementing algorithms to detect and track lip movements within the video stream. This involves analysing changes in lip position and shape over successive frames to identify movements.

```
# if user enters custom lip distance or script finishes calibrating
if (determining_lip_distance != 0 and LIP_DISTANCE_THRESHOLD != None):

    # Draw a circle around the mouth
    for n in range(48, 61):
        x = landmarks.part(n).x
        y = landmarks.part(n).y
        cv2.circle(img=frame, center=(x, y), radius=3, color=(0, 255, 0), thickness=-1)
```

Figure 10 - Code of Drawing Circle in Lip

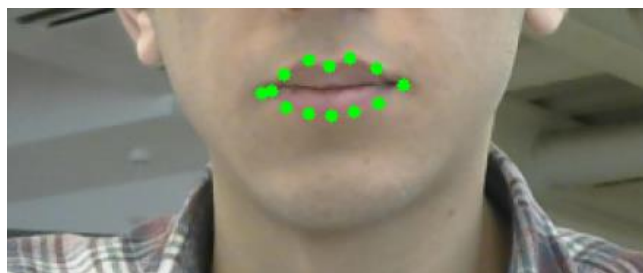


Figure 11 - Result of Drawing Lip

- 4- It reads the lip movements and determines whether the person is speaking by measuring the average distance. For example, let's assume my lip average is 41 cm if the average lip distance is more than 41 cm, then it indicates that the person is speaking. However, if the average distance remains unchanged, then it indicates that the person is not speaking.

```
if lip_distance > LIP_DISTANCE_THRESHOLD: # person is talking
    cv2.putText(frame, "Talking", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
```

Figure 12 - Code for Talking

```
else:
    if time.time() - self.mouth_open_timer > 5:
        cv2.putText(frame, "Not talking", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, RED, 2)
        #test_audio_thread.stop()
        #audio_start.stop()
    else:
        cv2.putText(frame, "Mouth closed", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, YELLOW, 2)
    #cv2.putText(frame, "Not talking", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, RED, 2)
```

Figure 13 - Code for Not Talking

By following these steps, the system can effectively segment and track lip movements in real-time, laying the groundwork for further analysis and interpretation of speech patterns.

2.4 PyAudio:

In conjunction with webcam functionality, the microphone is simultaneously activated to capture speech. This function enables real-time noise reduction during speech communication, particularly in scenarios where background noise may interfere with clarity. To achieve this, I used the non-stationary noise reduce module. This module is specifically designed to analyse and reduce non-stationary noise, making it an ideal choice for enhancing speech intelligibility in dynamic acoustic environments. By integrating PyAudio with non-stationary noise reduction techniques, the system ensures that speech signals remain clear and discernible, even amidst challenging auditory conditions.

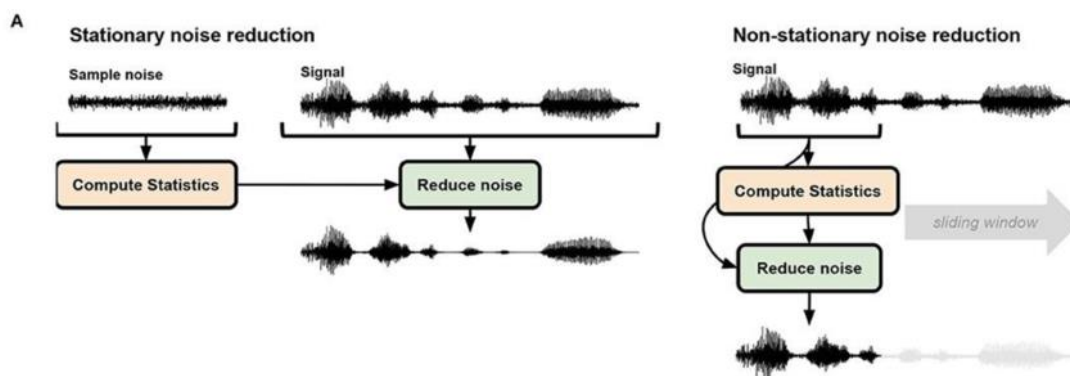


Figure 14 - Reduce Noise Schema

For choosing non-stationary noise reduction, it will reduce noise in real-time by taking input from the microphone. Here is the code:

```
1 usage
def record(self):
    self.stream.start_stream()
    while (self.open == True):
        data = self.stream.read(self.frames_per_buffer)
        np_data = np.frombuffer(data, dtype=np.int16)

        reduced_noise = nr.reduce_noise(y=np_data, sr=self.rate, thresh_n_mult_nonstationary=3, stationary=False)
        updated_data = reduced_noise * 5
        self.audio_frames.append(updated_data) # data olarak reduce_noise koydum
```

Figure 15 - Code Structure: Non-Stationary Reduce Noise

2.5 Socket:

During this project, the need arose to establish connections with external devices or computers, leading to the adoption of socket programming or RTSP protocol. To enable the connection of another computer's webcam, I implemented a client-server architecture using sockets. This approach facilitates the seamless transmission of video data between the server, which hosts the webcam feed, and the client, which receives and processes the video stream. By leveraging socket communication, the system enables the integration of multiple webcams from remote devices, enhancing the versatility and scalability of the overall solution.

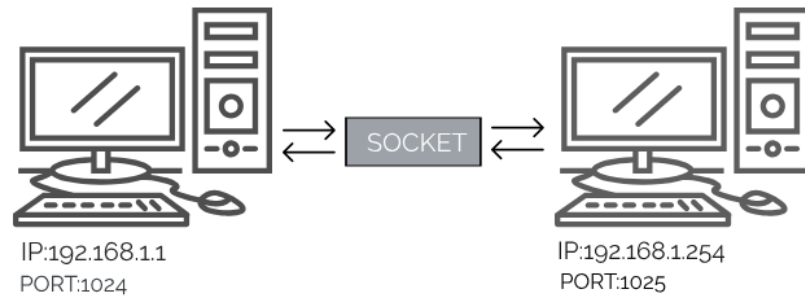


Figure 16 - Socket

```
def open_stream(self):  
    # Socket Create  
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
    host_name = socket.gethostname()  
    host_ip = socket.gethostbyname(host_name)  
    print('HOST IP:', host_ip)  
    log_host = "HOST IP: " + host_ip  
    logger.info(log_host)  
    port = 9999  
    socket_address = (host_ip, port)  
  
    # Socket Bind  
    server_socket.bind(socket_address)  
  
    # Socket Listen  
    server_socket.listen(5)  
    print("LISTENING AT:", socket_address)
```

Figure 17 - Code Structure: Socket Client

2.6 Speech Recognition

The final feature of this project is the implementation of speech recognition to generate subtitles for spoken words. This functionality aims to provide a textual representation of speech, enhancing communication accessibility.

When the microphone is activated, it records the voice while simultaneously reducing background noise. The cleaned audio is then saved to a file. Upon pressing the "Stop All Cam" button, the recorded voice file is processed, converting the speech to text using a speech recognition algorithm. This transcribed text is subsequently displayed in a label within the user interface, providing real-time subtitles for the spoken content. This feature enhances the accessibility and usability of the system, making it easier for users to understand and engage in communication, especially in noisy environments.

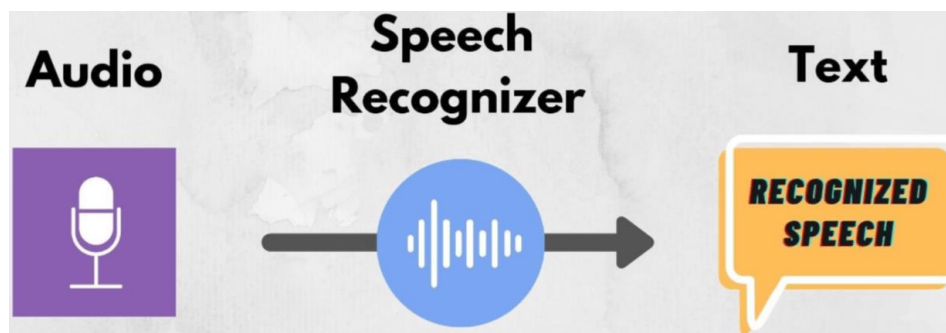


Figure 18 - Speech Recognition Schema

```
def start(self):
    audio_file_path = "Outputs/temp_audio.wav"
    sub_thread = threading.Thread(target=self.start_to_listen, args=(audio_file_path,))
    sub_thread.start()

def speech_to_text(self, audio_data):
    recognizer = sr.Recognizer()
    try:
        text = recognizer.recognize_google(audio_data)
        return text
    except sr.UnknownValueError:
        print("Google Speech Recognition could not understand audio")
        logger.info("Google Speech Recognition could not understand audio")
        return None
    except sr.RequestError as e:
        print("Could not request results from Google Speech Recognition service; {0}".format(e))
        return None
```

Figure 19 - Code Structure: Transform from Speech to Text

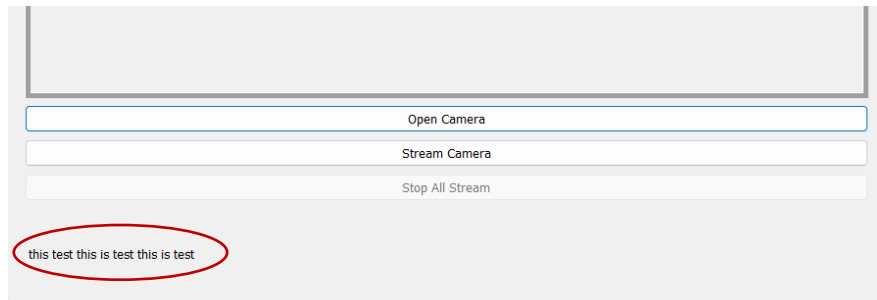
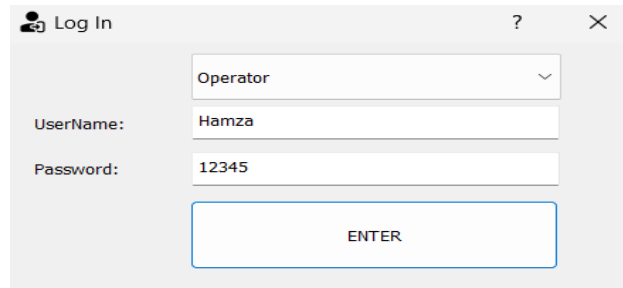


Figure 20 - Transformed Text in the Label

3. RESULTS AND DISSCUSSION

3.1 Application of Work

1. Filling username, password and select role.



The screenshot shows a 'Log In' window with a title bar containing a user icon, the text 'Log In', and standard window controls. Inside the window, there is a dropdown menu for 'Role' with 'Operator' selected. Below it are text input fields for 'UserName:' containing 'Hamza' and 'Password:' containing '12345'. At the bottom is a large button labeled 'ENTER'.

Figure 21 - Filling boxes

2. If we do not select role or fill wrong username or password, it will show message boxes.

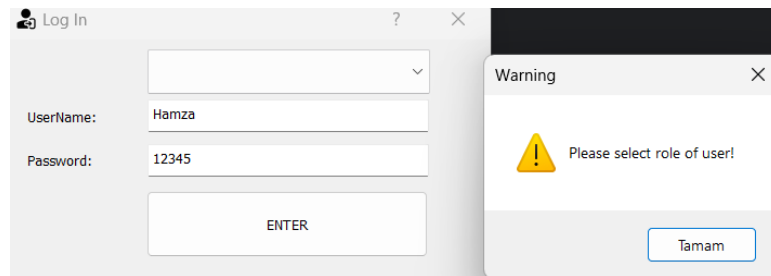


Figure 22 - Error: Not selected the role

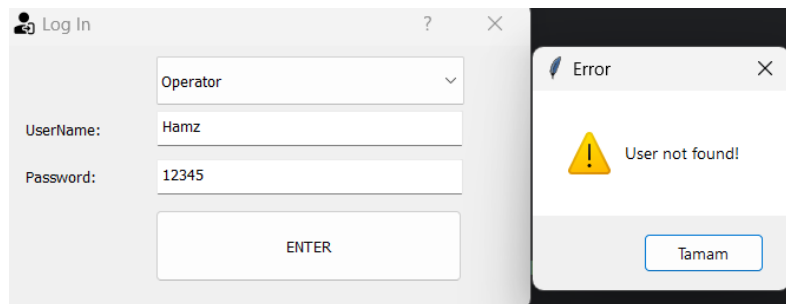


Figure 23 - Error: Wrong username or password

3. After filling true things, user interface will be closed, and main interface will be opened.

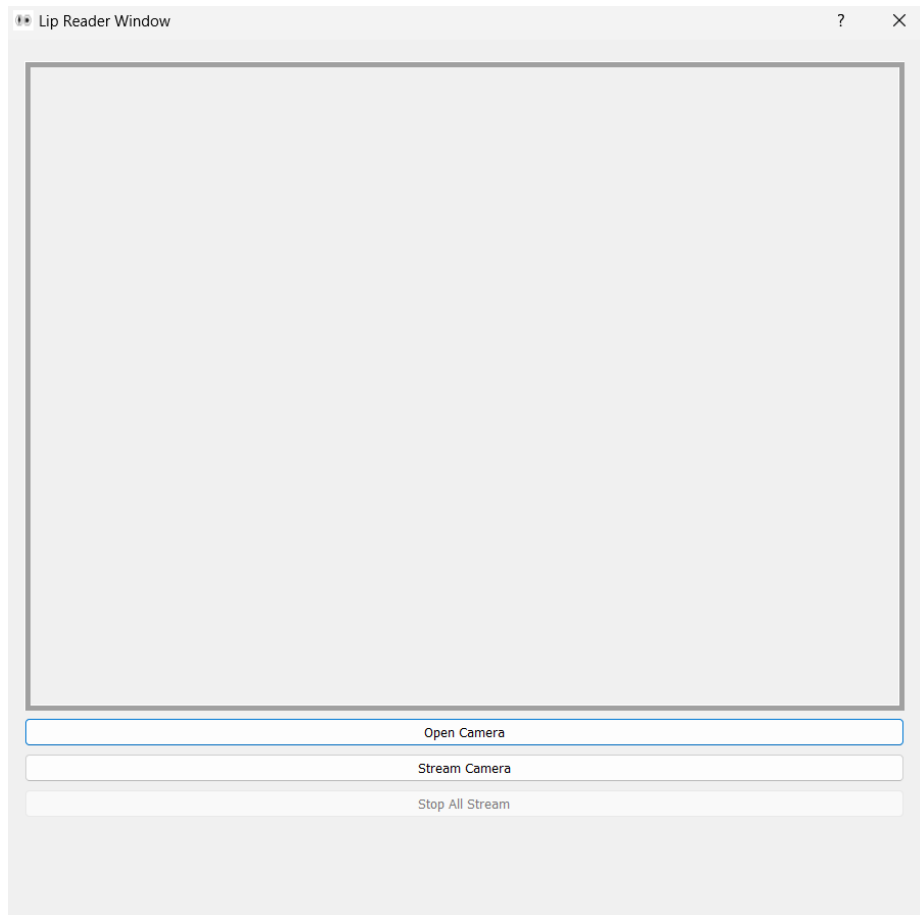


Figure 24 - The Main Window

4. There are two buttons to open the camera. If you press the "Open Camera" button, the camera and microphone will be activated and start recording.
 - If you are not talking, it will be say "Not Talking" in the right corner.

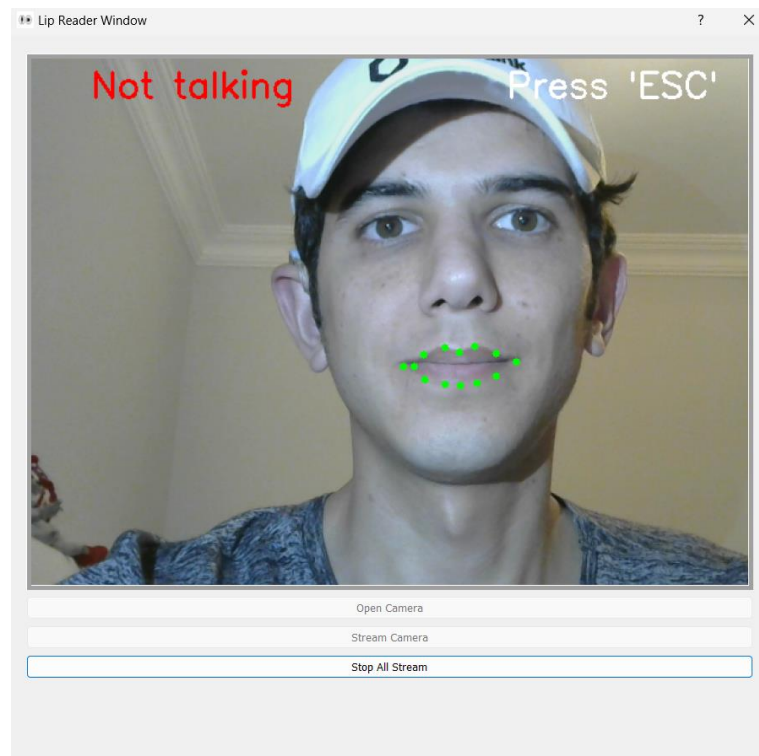


Figure 25 - If not talking

- If you are talking but your mouth is closed, then it will display "Mouth Closed".

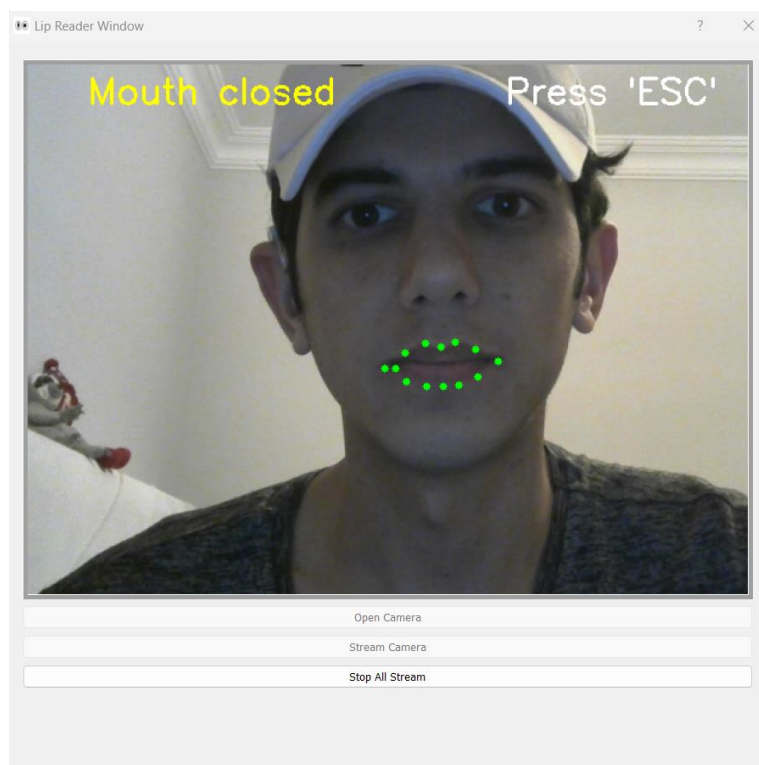


Figure 26 - If mouth closed

- If you are talking, it will display “Talking”.

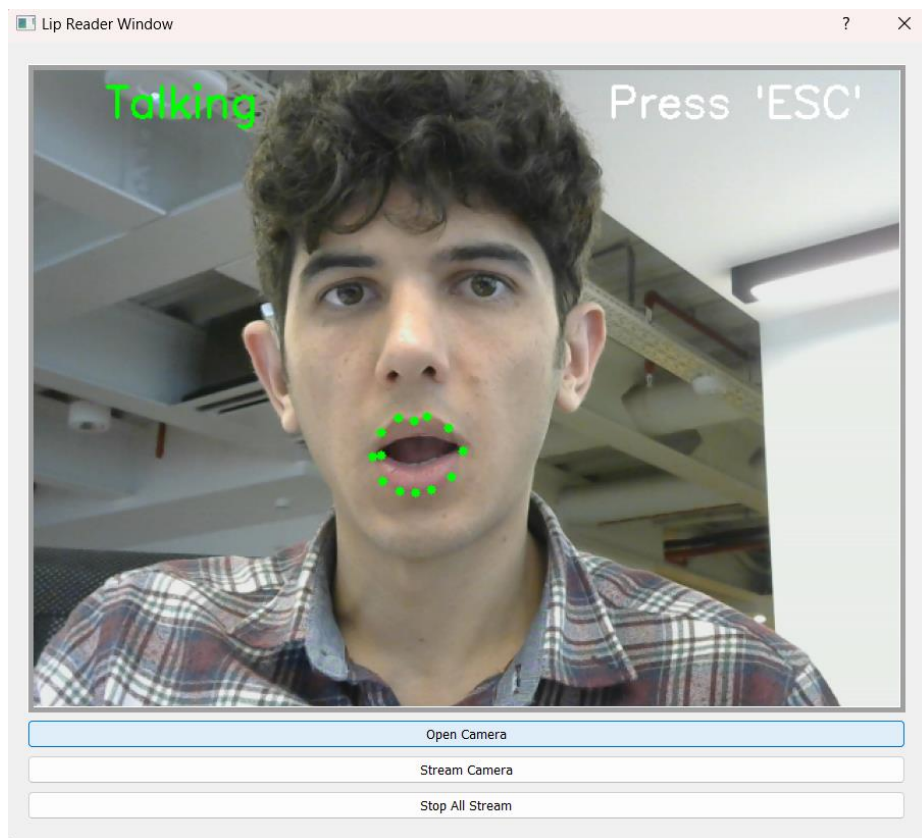


Figure 27 - If it is talking

5. However, if you press "Stream Camera," it will connect to the camera of another computer.

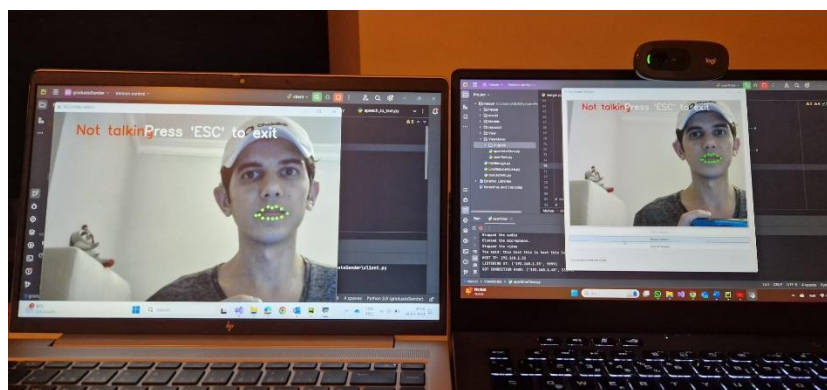


Figure 28 - Client and Server Connecting

6. When you press "Stop All Camera," both the camera and microphone will turn off. What you say will be subtitled using speech recognition from the recorded audio. Additionally, the video recording and audio recording will be merged into a regular movie.

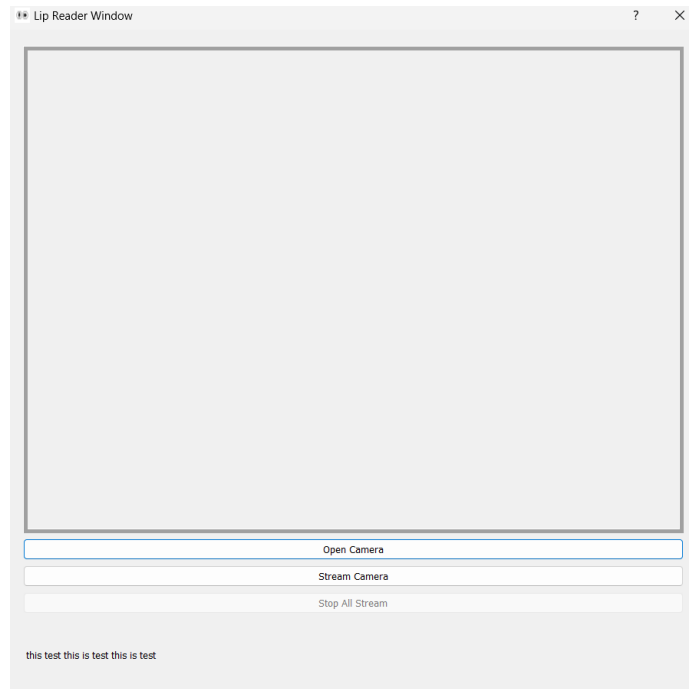


Figure 29 - Stopped Camera and Microphone

7. To view recordings such as MP3s, videos, and merged videos, you can see them here.

MyAppLogger	27.05.2024 22:42	Metin Belgesi	7 KB
temp_audio	27.05.2024 22:42	WAV Dosyası	1.801 KB
temp_video	27.05.2024 22:42	MOV Dosyası	2.653 KB
merged_output_video	19.05.2024 21:42	MOV Dosyası	592 KB

Figure 30 - Recordings and note in the file

8. Finally, if you want to see the movements for the project, you can click "MyAppLogger" to view the logs.

```

2024-05-19 19:24:58,431 - MyAppLogger - INFO - The interface is working
2024-05-19 19:52:47,779 - MyAppLogger - INFO - The interface is working
2024-05-19 19:54:53,326 - MyAppLogger - INFO - Cam and microphone are closed
2024-05-19 19:56:07,669 - MyAppLogger - INFO - The interface is working
2024-05-19 19:58:04,518 - MyAppLogger - INFO - Cam and microphone are closed
2024-05-19 20:08:53,479 - MyAppLogger - INFO - The interface is working
2024-05-19 20:10:42,806 - MyAppLogger - INFO - Camera is working!
2024-05-19 20:10:42,806 - MyAppLogger - INFO - Camera is working!
2024-05-19 20:10:53,346 - MyAppLogger - INFO - Cam and microphone are closed
2024-05-19 20:10:53,346 - MyAppLogger - INFO - Cam and microphone are closed
2024-05-19 20:10:54,625 - MyAppLogger - INFO - Google Speech Recognition could not understand audio
2024-05-19 20:10:54,625 - MyAppLogger - INFO - Google Speech Recognition could not understand audio
2024-05-19 20:10:54,625 - MyAppLogger - INFO - Google Speech Recognition could not understand audio
2024-05-19 21:29:17,377 - MyAppLogger - INFO - The interface is working
2024-05-19 21:29:17,378 - MyAppLogger - INFO - selambaba
2024-05-26 14:22:24,194 - MyAppLogger - INFO - It showed that User need to select role type.
2024-05-26 14:22:24,203 - MyAppLogger - INFO - User not found.
2024-05-26 14:26:26,952 - MyAppLogger - INFO - It showed that User need to select role type.
2024-05-26 14:26:26,957 - MyAppLogger - INFO - User not found.
2024-05-26 14:27:44,876 - MyAppLogger - INFO - It showed that User need to select role type.
2024-05-26 14:27:47,162 - MyAppLogger - INFO - Username and password are correct. Main window is opened!
2024-05-26 14:28:52,286 - MyAppLogger - INFO - It showed that User need to select role type.

```

Figure 31 - Logger Note

3.2 Results:

Upon initiating the webcam directly or establishing a connection via socket to access the webcam feed from another computer, the system successfully detects and tracks lip movements in real-time. Concurrently, noise reduction algorithms are activated to mitigate background noise, enhancing the clarity of speech signals. The integration of speech recognition functionality further enriches the system's capabilities by accurately transcribing spoken words into text format.

Overall, the combined efforts of webcam streaming, lip movement detection, noise reduction, and speech recognition culminate in a comprehensive solution for enhancing communication accessibility, particularly for individuals with hearing impairments. The successful execution of these functionalities demonstrates the efficacy and practical utility of the proposed image processing system, underscoring its potential to significantly improve the communication experience in noisy environments.

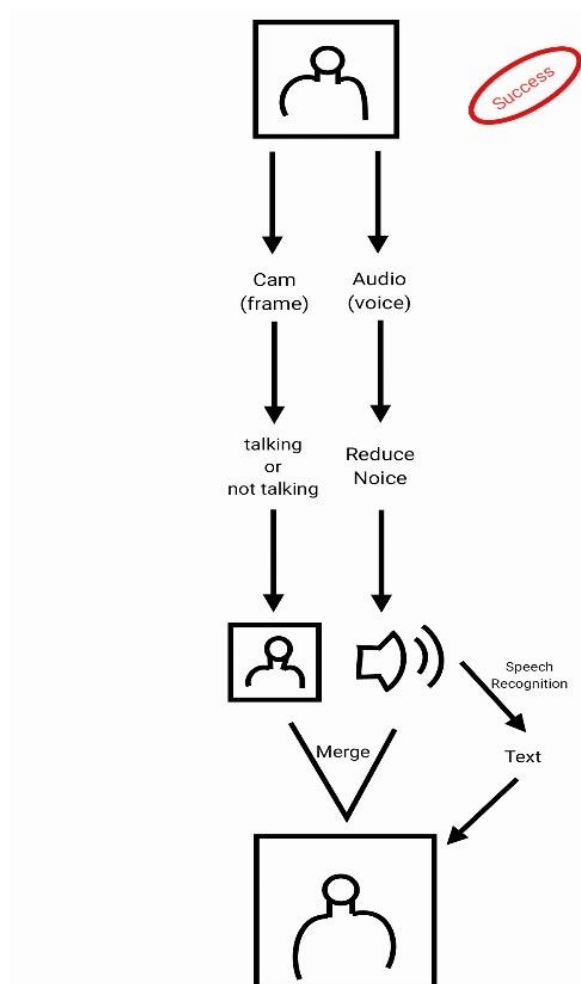


Figure 32 - Current Project Schema

3.3 Discussion:

Actually, detecting lip movements and reduce noise is not novel idea and has been done many times. However, I wanted to merge these two functions so that I could see results. I

faced many problems such as when detecting lip movements and happened “talking” and from there I wanted to start reducing noise. However, because of working different input device camera was freezing. Here I will give details.

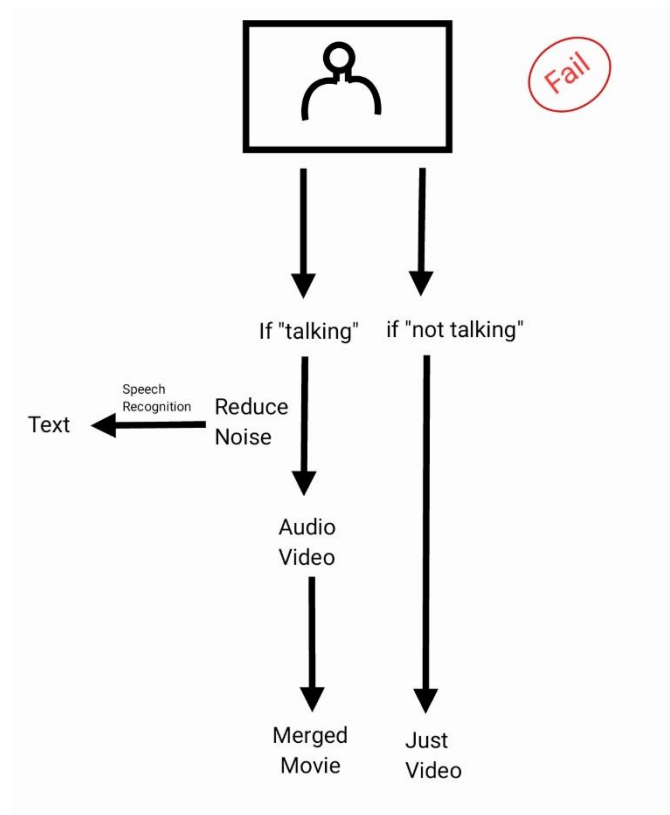


Figure 33 - Expecting Project Schema (Fail)

While the individual components of detecting lip movements and reducing noise are not novel concepts and have been explored extensively in previous research, the novelty of this project lies in the integration of these two functions within a unified system. The primary motivation behind this integration was to observe the synergistic effects of combining these functionalities and to evaluate the feasibility of their simultaneous implementation.

4. CHALLENGES AND FUTURE IMPROVEMENTS

PyAudio: When the system detected lip movements indicative of speech, noise reduction algorithms caused disruptions, such as freezing of the webcam feed due to conflicts in resource allocation between the camera and microphone. According to my researching, there is no existing function or library that allow microphone and camera do not work together, but it should be separated.

```

# if user enters custom lip distance or script finishes calibrating
if (determining_lip_distance != 0 and LIP_DISTANCE_THRESHOLD != None):

    # Draw a circle around the mouth
    for n in range(48, 61):
        x = landmarks.part(n).x
        y = landmarks.part(n).y
        cv2.circle(img=frame, center=(x, y), radius=3, color=(0, 255, 0), thickness=-1)

    RED = (0, 0, 255) # Not talking
    YELLOW = (0, 255, 255)

    # print(len(curr_word_frames))

    if lip_distance > LIP_DISTANCE_THRESHOLD: # person is talking
        cv2.putText(frame, "Talking", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        print("started to listen")
        audio_thread = audio_record.AudioRecorder()
        audio_thread.start()

```

Figure 34 - Expecting Code: To put reduce noise class

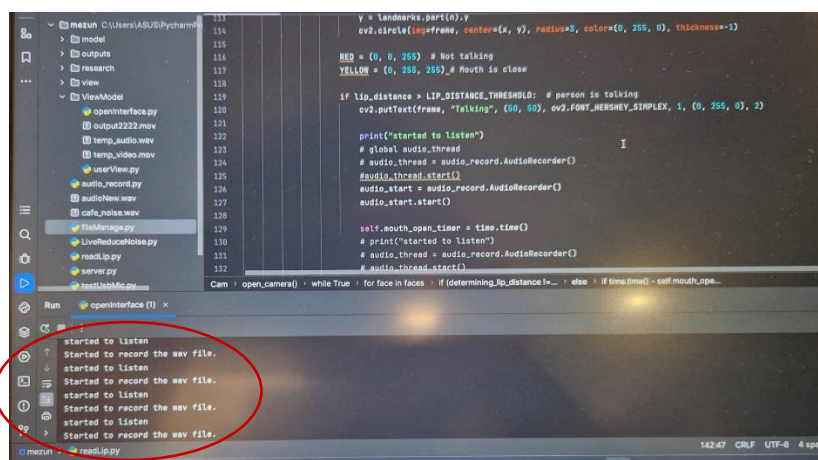


Figure 35 - Result of expecting code (fail)

5. CONCLUSION

This project uses image processing and PyAudio to create an algorithm that in the noise environment will be reduced noise by detecting lip movements. The user interface is facilitated through a database system. Each camera undergoes image processing techniques, including lip segmentation, while PyAudio employs non-stationary noise reduction methods. Libraries such as OpenCV, numpy, math, and PyAudio are utilized throughout the project. The resulting model is lightweight yet effective, capable of detecting lip movements and reducing noise with ease.

6. REFERENCES

- <https://stackoverflow.com/questions/61861229/how-to-mesure-the-distance-between-2-points-in-the-facial-landmarks-from-dlib-68>
- <https://www.geeksforgeeks.org/yawn-detection-using-opencv-and-dlib/>
- <https://github.com/samwestby/OpenCV-Python-Tutorial/tree/main>
- <https://www.geeksforgeeks.org/creating-a-camera-application-using-pyqt5/>
- <https://pyshine.com/Socket-programming-and-openc/>
- <https://github.com/timsainb/noisereduce>
- <https://github.com/Dhriti03/Noise-Reduction/blob/master/README.md>
- <https://www.geeksforgeeks.org/python-convert-speech-to-text-and-text-to-speech/>