Project Phase I Progress Report

Group 17

Student:	Ku Ka Yan	1155159909
	Ng Yu Chun	1155157839
	Ku Wai Kuen	1155160201
	Yau Wai Chun	1155143258

Introduction

Our group has developed an audio processing application that allows users to perform various operations on audio streams, such as recording, playback, editing, and conversion. This report provides an overview of the system, describes the implementation details, and outlines the contributions of each group member.

System Overview

Our application is mainly programmed using python and includes a graphical user interface (GUI).

Tested python version: Python 3.10.0

List of third-party libraries:

- numpy
- scipy
- noisereduce
- pyadudio
- librosa
- matplotlib
- speech_recognition
- Pydub
- tkinter

Before running the application, make sure all the required libraries are properly installed. To run the application, navigate to the project folder and enter the command:

```
python main.py
```

Implementation Details

1. Basic requirements

1.1 Graphical user interface:

A graphical user interface using *tkinter* library is provided. It consists of two main components: a control panel and a function panel. The control panel allows users to browse the audio files and perform other actions related to playing the audio.

1.2 Sound Recording & Saving:

Our application supports audio recording using the computer's built-in microphone or external devices, which can be explicitly selected on the top panel. The recorded audio streams are encoded and saved as WAV. The implementation is done in `audio_recorder.py`. We have implemented the required functionality to write the fmt-chunk and the data-chunk in WAV files using *pyaudio* and *wave libraries* for input device control and file read write only.

1.3 Sound Playback with Speed Control:

Our application includes a sound player that reads and parses selected WAV files from the recording explorer. It implements the decoder for WAV files to enable playback of the audio streams in `audio_player.py`. Additionally, our sound player allows users to adjust the playback speed to 2.0x or 0.5x, providing flexibility in audio playback. Additionally, a volume slider and seek bar is provided at the bottom left of the window.

1.4 Audio Trim:

We have implemented a basic audio editor within our application. It supports trimming and cutting of recorded audio files. Users can modify the start and end times of the audio, insert a new recording or overwrite sections with fresh recordings using the sliders. The implementation is mainly done in the `page_record.py`.

2. Enhanced Features

2.1 Background Noise Removal:

To ensure high-quality recordings with clear human voices, we implemented noise reduction using *noisereduce* library to remove background noise from audio streams. We have adopted non-stationary noise reduction using Per-Channel Energy Normalization [1]. The implementation is done in `audio_noise_remover.py`.

2.2 Audio to Text Conversion:

We integrated a google speech recognition function into our application, using speech_recognition library and implemented in `audio_text_converyer.py`. Currently, the audio-to-text function does not support music audio or audio with severe background noise

2.3 Visualization:

To provide users with a visual representation of the captured sound, we incorporated a waveform visualization feature in `page_noise_removal.py`. In this part, we used *librosa* library to plot the waveform of the audio. This feature displays the amplitude of the audio stream, giving users a useful indicator of the audio content. Users can see the audio visualization in the noise removal page after clicking the play button.

Contributions

Ku Ka Yan: Documentation, graphical user interface and testing
Ng Yu Chun: Sound recording, saving, editing and playback
Ku Wai Kuen: Audio visualization, background noise removal.
Yau Wai Chun: Documentation, audio to text, testing.

Conclusion

In conclusion, our audio stream processing application fulfills the basic requirements by providing essential operations such as recording, playback, and audio editing. Additionally, we have incorporated enhanced features including audio editing, background noise removal, audio to text conversion, and visualization. Each group member made significant contributions to the project, focusing on different aspects of implementation. Through this project, we have gained valuable experience in audio processing and software development.

References

[1] "noisereduce: Noise reduction using Spectral Gating in python," PyPI. https://pypi.org/project/noisereduce/

[2] "Librosa.display.waveshow□," librosa.display.waveshow - librosa 0.10.1 documentation, https://librosa.org/doc/main/generated/librosa.display.waveshow.html.