```
pip install soundcard soundfile lameenc customtkinter

import threading
import soundcard as sc
import soundfile as sf
import lameenc
import customtkinter as ctk
import argparse
import sys
import os
from datetime import datetime

class StreamCapturePro:
    def __init__(self):
        self.is_recording = False
        self.sample_rate = 44100
        self.bitrate = 320

    def encode_to_mp3(self, wav_path):
        """Converts the recorded WAV to a 320kbps MP3 using
LAME."""
        mp3_path = wav_path.replace(".wav", ".mp3")
        data, sr = sf.read(wav_path)

        # Convert float audio to 16-bit PCM for the encoder
        pcm_data = (data * 32767).astype('int16').tobytes()

        encoder = lameenc.Encoder()
        encoder.set_bit_rate(self.bitrate)
        encoder.set_in_sample_rate(sr)
```

```
        encoder.set_channels(2)
        encoder.set_quality(2) # High quality setting

        mp3_data = encoder.encode(pcm_data)
        mp3_data += encoder.flush()

        with open(mp3_path, "wb") as f:
            f.write(mp3_data)

        # Clean up the temporary WAV file
        os.remove(wav_path)
        return mp3_path

    def record_loop(self, filename, duration=None):
        temp_wav = "temp_capture.wav"
        speaker = sc.default_speaker()

        with speaker.recorder(samplerate=self.sample_rate) as
mic:
            with sf.SoundFile(temp_wav, mode='w',
samplerate=self.sample_rate, channels=2) as file:
                if duration: # CLI timed mode
                    data = mic.record(numframes=self.sample_rate *
duration)
                    file.write(data)
                else: # GUI manual mode
                    while self.is_recording:
                        chunk = mic.record(numframes=2048)
                        file.write(chunk)
```

```
        final_file = self.encode_to_mp3(temp_wav)
        print(f"Finished! Saved as: {final_file}")

# --- GUI CODE ---
class App(ctk.CTk):
    def __init__(self, logic):
        super().__init__()
        self.logic = logic
        self.title("Aria-Grade Stream Recorder")
        self.geometry("400x250")

        self.lbl = ctk.CTkLabel(self, text="System Audio
Capturer", font=("Impact", 24))
        self.lbl.pack(pady=20)

        self.status = ctk.CTkLabel(self, text="Ready to record at
320kbps", text_color="gray")
        self.status.pack()

        self.btn = ctk.CTkButton(self, text="START RECORDING",
font=("Arial", 14, "bold"),
                                 height=50, command=self.toggle)
        self.btn.pack(pady=30)

    def toggle(self):
        if not self.logic.is_recording:
            self.logic.is_recording = True
            timestamp =
datetime.now().strftime("%Y%m%d_%H%M%S")
            self.btn.configure(text="STOP & ENCODE",
```

```
fg_color="red")
            self.status.configure(text="● RECORDING LIVE
STREAM", text_color="red")
            threading.Thread(target=self.logic.record_loop,
args=(f"stream_{timestamp}.mp3",)).start()
        else:
            self.logic.is_recording = False
            self.btn.configure(text="START RECORDING",
fg_color=["#3a7ebf", "#1f538d"])
            self.status.configure(text="Encoding MP3... please
wait", text_color="orange")

if __name__ == "__main__":
    core = StreamCapturePro()
    if len(sys.argv) > 1:
        # Simple CLI Usage: python script.py 60 my_record.mp3
        core.record_loop("cli_out.mp3", duration=int(sys.argv[1]))
    else:
        app = App(core)
        app.mainloop()
```