In [1]:
```
pip install openai-whisper watchdog ffmpeg-python
```

```
   ------------ --------------------------- 9.5/30.3 MB 5.2 MB/s eta 0:0
0:04
   ------------ --------------------------- 9.5/30.3 MB 5.0 MB/s eta 0:0
0:05
   ------------ --------------------------- 9.7/30.3 MB 5.1 MB/s eta 0:0
0:05
   ------------ --------------------------- 9.8/30.3 MB 4.9 MB/s eta 0:0
0:05
   ------------- -------------------------- 10.2/30.3 MB 5.1 MB/s eta 0:0
0:04
   ------------- -------------------------- 10.6/30.3 MB 5.1 MB/s eta 0:0
0:04
   ------------- -------------------------- 10.8/30.3 MB 5.1 MB/s eta 0:0
0:04
   -------------- ------------------------- 11.5/30.3 MB 5.5 MB/s eta 0:0
0:04
   -------------- ------------------------- 11.8/30.3 MB 5.5 MB/s eta 0:0
0:04
   --------------- ------------------------ 12.2/30.3 MB 5.8 MB/s eta 0:0
0:04
```

In [2]:
```
# import libraries
```

In [4]:
```
import os
import whisper
import time
import json
from pathlib import Path
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
import ffmpeg
```

In [5]:
```python
# Configuration
WATCH_FOLDER = "media"
LOG_FILE = "processed_files.json"
MODEL = whisper.load_model("base")  # Change to "small", "medium", or "large"

# Load or Initialize Processed Files Log
def load_processed_files():
    if os.path.exists(LOG_FILE):
        with open(LOG_FILE, "r") as file:
            return json.load(file)
    return {}

def save_processed_files(processed_files):
    with open(LOG_FILE, "w") as file:
        json.dump(processed_files, file)

processed_files = load_processed_files()
```

```
100%|████████████████████████████████████████| 139M/139M [00:20<00:00, 7.22MiB/s]
```

In [8]:
```python
# Transcribe Function
def transcribe_audio(file_path):
    """Transcribe audio file using Whisper and save the text output."""
    print(f"Processing: {file_path}")
    output_text_file = file_path.with_suffix(".txt")


    if str(file_path) in processed_files:
        print(f"Skipping {file_path}, already processed.")
        return
    result = MODEL.transcribe(str(file_path))

    # Save transcript
    with open(output_text_file, "w", encoding="utf-8") as f:
        f.write(result["text"])

    processed_files[str(file_path)] = True
    save_processed_files(processed_files)
    print(f"Saved transcription: {output_text_file}")
```

In [*]:
```
# Convert Video to Audio
def extract_audio(video_path):
    """Convert video file to audio (WAV format) for transcription."""
    audio_path = video_path.with_suffix(".wav")
    if audio_path.exists():
        return audio_path

    (
        ffmpeg.input(str(video_path))
        .output(str(audio_path), format="wav", acodec="pcm_s16le", ac=1, ar="
        .run(overwrite_output=True)
    )
    return audio_path

# Scan & Process Existing Files
def scan_directory(directory):
    """Scan directory recursively and transcribe audio/video files."""
    audio_formats = {".mp3", ".wav", ".aac", ".m4a"}
    video_formats = {".mp4", ".mkv", ".mov", ".flv"}

    for file_path in Path(directory).rglob("*"):
        if file_path.suffix.lower() in audio_formats:
            transcribe_audio(file_path)
        elif file_path.suffix.lower() in video_formats:
            audio_file = extract_audio(file_path)
            transcribe_audio(audio_file)

# Real-Time Monitoring
class FileHandler(FileSystemEventHandler):
    """Handles new files in the watched directory."""
    def on_created(self, event):
        if event.is_directory:
            return

        file_path = Path(event.src_path)
        print(f"New file detected: {file_path}")

        if file_path.suffix.lower() in {".mp3", ".wav", ".aac", ".m4a"}:
            transcribe_audio(file_path)
        elif file_path.suffix.lower() in {".mp4", ".mkv", ".mov", ".flv"}:
            audio_file = extract_audio(file_path)
            transcribe_audio(audio_file)

def start_monitoring():
    """Start watching the directory for new files."""
    event_handler = FileHandler()
    observer = Observer()
    observer.schedule(event_handler, WATCH_FOLDER, recursive=True)
    observer.start()
    print(f"Monitoring started on {WATCH_FOLDER}...")

    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()
    observer.join()
```

```
if __name__ == "__main__":

    Path(WATCH_FOLDER).mkdir(parents=True, exist_ok=True)

    print("Scanning existing files...")
    scan_directory(WATCH_FOLDER)

    start_monitoring()
```

```
Scanning existing files...
Monitoring started on media...
```

How It Works

Scans existing files in media/ and its subdirectories.
Processes audio and video files (extracting audio from video if needed).
Saves transcriptions in the same folder as the media file.
Keeps a log (processed_files.json) to avoid redundant processing.
Continuously monitors the folder for new files using watchdog.