Real-time Speech Transcription POC – Working Document

Overview

- Browser microphone → Client (HTML/JS) → Backend WebSocket (`/ws`) → AWS Transcribe Streaming → live transcript → S3 on stop.

- On Stop: concatenates only final transcripts; uploads `YYYYMMDD_HHMMSS.txt` to S3 bucket.

Services and Ports

- Backend: FastAPI at http://localhost:8000 (WebSocket `/ws`, health `/health`)

- Client: Static HTML at http://localhost:8080/simple_client.html

WebSocket Protocol

- Client → Backend

- `{ "type":"audio", "payload":"" }`

- `{ "type":"control", "action":"stop" }`

- Backend → Client

- `{ "type":"transcript", "text":"...", "is_final":true|false }`

- `{ "type":"saved", "s3_key":"YYYYMMDD_HHMMSS.txt" }`

Environment Variables

- `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, `AWS_REGION`, `S3_BUCKET`

- Optional: `TRANSCRIBE_LANGUAGE_CODE` (default `en-US`), `LOG_LEVEL` (default `INFO`).

Docker Compose

- `backend` (port 8000) and `client` (port 8080) services; shared bridge network; backend has healthcheck; client waits for backend healthy.

Backend (FastAPI + AWS Transcribe)

- File: `backend/main.py`

- Key parts:

- Transcribe Streaming: `start_stream_transcription(language_code=..., media_sample_rate_hz=16000, media_encoding="pcm")`

- WebSocket `/ws`: accepts JSON messages; decodes base64 audio; feeds to AWS stream; on stop, ends stream, concatenates finals, uploads to S3.

- S3 save via `boto3.put_object` offloaded with `run_in_executor`.

- Sends back live/final transcripts and final `saved` notification.

Client (HTML/JavaScript)

- Files: `client/simple_client.html`, `client/serve_client.py`, `client/Dockerfile`

- Flow:

- Opens WebSocket to `ws://localhost:8000/ws`.

- Captures mic with Web Audio API at 16kHz, mono.

- Converts Float32 to PCM16 and base64; sends as `{type:"audio"}`.

- Displays live text and accumulates final text.

- On Stop: sends `{type:"control", action:"stop"}`; shows S3 key on save.

Terraform

- File: `terraform/main.tf`

- Provisions:

- S3 bucket (versioned, encrypted, public-blocked).

- IAM user with inline policy: `transcribe:StartStreamTranscription`, `s3:PutObject`, `s3:AbortMultipartUpload` for that bucket.

- Access keys outputs (sensitive).

Runbook

1. Terraform: `cd terraform && cp terraform.tfvars.example terraform.tfvars && terraform init && terraform apply`

2. `.env`: copy `env.example` $\rightarrow$ `.env` and fill outputs (keys, region, bucket)

3. Start: `docker compose up -d --build`

4. Use: open `http://localhost:8080/simple_client.html`, Start/Stop to see/save transcripts

Troubleshooting

- WebSocket fails: ensure backend healthy and ports free; env vars set.

- Mic issues: allow permissions; HTTPS required in production.

- S3 failures: check bucket name and IAM policy.

- CORS: backend allows `http://localhost:8501` by default; add `http://localhost:8080` if needed.

Production Notes

- Use HTTPS for client; tighten CORS; prefer IAM roles; add monitoring/logging; scale via ECS/EKS and ALB.