Backend Machine Learning Pipeline
Automated Interview Analysis

System Architecture Document

February 16, 2026

Contents

1 Purpose and Scope **3**

2 Input Data Contract **3**
2.1 Assumptions . . . 3

3 Technology Stack **3**
3.1 Programming Language . . . 3
3.2 Audio Processing . . . 3
3.3 Video Processing . . . 4
3.4 Language Models . . . 4
3.5 Data Handling . . . 4

4 Pipeline Overview **4**

5 Stage 0: Canonical Time Base **4**
5.1 Objective . . . 4
5.2 Process . . . 4
5.3 Internal Metadata . . . 5

6 Stage 1: Signal Extraction **5**
6.1 Candidate Audio . . . 5
6.2 Interviewer Audio . . . 5
6.3 Candidate Video . . . 5
6.4 Outputs . . . 5

7 Stage 2: Temporal Grouping **6**
7.1 Speaking Segmentation . . . 6
7.2 Question–Answer Mapping . . . 6
7.3 Outputs . . . 6

8 Stage 3: Behavioral Metrics **6**
8.1 Audio-Based Metrics . . . 6
8.2 Video-Based Metrics . . . 6
8.3 Output . . . 6

9 Stage 4: Semantic Relevance Scoring **7**
9.1 LLM Role . . . 7
9.2 Constraints . . . 7
9.3 Output . . . 7

10 Stage 5: JD-Conditioned Aggregation **7**
10.1 JD Decomposition 7
10.2 Chronological Scoring 7
10.3 Output 7

11 Final Output **8**

12 Explicit Non-Goals **8**

13 Conclusion **8**

1 Purpose and Scope

This document defines the backend machine learning pipeline for an automated interview analysis system.

The interview user interface (UI) is assumed to be complete and provides pre-separated modalities:

- Candidate audio only
- Interviewer audio only
- Candidate video only (no audio)

The backend processes these inputs to produce structured, time-aligned, explainable evaluation outputs.

No model training is performed. All inference relies on pretrained models and deterministic logic.

2 Input Data Contract

Each interview session provides the following files:

```
candidate_audio.wav
candidate_video.mp4
interviewer_audio.wav
job_description.txt
```

2.1 Assumptions

- Speaker separation is already complete
- Candidate video contains only the candidate
- All files belong to a single interview session
- Audio can be aligned to the video timeline

The candidate video timeline is treated as the canonical time base.

3 Technology Stack

3.1 Programming Language

- Python 3.11

3.2 Audio Processing

- ffmpeg
- librosa
- webrtcvad
- faster-whisper

3.3 Video Processing

- opencv-python
- mediapipe

3.4 Language Models

- Local LLM: Qwen 2.5 3B
- Sentence-BERT–compatible embedding model

3.5 Data Handling

- numpy, scipy
- pydantic
- JSON for all intermediate and final artifacts

4 Pipeline Overview

```
Input Files
  ↓
Stage 0: Canonical Time Base
  ↓
Stage 1: Signal Extraction
  ↓
Stage 2: Temporal Grouping
  ↓
Stage 3: Behavioral Metrics
  ↓
Stage 4: Semantic Relevance Scoring
  ↓
Stage 5: JD-Conditioned Aggregation
  ↓
Final Output (output.json)
```

Each stage produces explicit artifacts consumed by the next stage.

5 Stage 0: Canonical Time Base

5.1 Objective

Unify all modalities onto a single authoritative timeline.

5.2 Process

- Extract FPS and duration from candidate video
- Normalize audio timestamps
- Align audio streams to video time

5.3 Internal Metadata


```
{
  "timebase": "video",
  "fps": 30,
  "duration_sec": 1832.4
}
```


6 Stage 1: Signal Extraction

This stage performs measurement only. No semantic interpretation.

6.1 Candidate Audio

Extracted features (timestamped):

- RMS energy
- Fundamental frequency (pitch)
- Pitch variance
- Speech rate
- Pause duration
- Voice activity detection

Additionally:

- Speech-to-text transcription with word-level timestamps

6.2 Interviewer Audio

- Speech-to-text transcription with timestamps

6.3 Candidate Video

Frames sampled at every 10th frame.

Per-frame features:

- Face presence
- Face bounding box size
- Eye gaze direction
- Head pose (yaw, pitch, roll)
- Facial landmark movement

6.4 Outputs

- candidate_audio_raw.json
- interviewer_transcript.json
- candidate_video_raw.json

7 Stage 2: Temporal Grouping

7.1 Speaking Segmentation

Candidate voice activity defines speaking vs non-speaking intervals.


```
{
  "segment_id": "S4",
  "type": "speaking",
  "start": 112.3,
  "end": 129.7,
  "video_frames": [3370, 3380, 3390]
}
```


7.2 Question–Answer Mapping

Rules:

- Interviewer speech defines questions
- Subsequent candidate speaking defines answers
- Short silences are merged

7.3 Outputs

- speaking_segments.json
- qa_pairs.json

8 Stage 3: Behavioral Metrics

Derived metrics are computed per answer or per speaking segment.

8.1 Audio-Based Metrics

- Confidence proxy
- Fluency score
- Stress proxy
- Consistency / evasiveness proxy

8.2 Video-Based Metrics

- Face presence ratio
- Eye contact stability
- Head movement entropy
- Micro-movement intensity

8.3 Output

- candidate_behavior_metrics.json

9 Stage 4: Semantic Relevance Scoring

9.1 LLM Role

The LLM evaluates semantic alignment only.

Inputs:

- Question text
- Answer text
- Relevant job description excerpt

9.2 Constraints

- Low temperature
- Forced structured output
- No free-form prose

9.3 Output

```
{
  "qa_id": "QA7",
  "relevance": 0.84,
  "coverage": 0.79,
  "off_topic": false
}
```

Stored as:

- relevance_scores.json

10 Stage 5: JD-Conditioned Aggregation

10.1 JD Decomposition

The job description is parsed into:

- Required skills
- Soft skills
- Weight vectors

10.2 Chronological Scoring

Scores are updated incrementally per QA pair:

- Skill confidence
- Communication effectiveness
- Behavioral consistency

10.3 Output

- candidate_score_timeline.json

11 Final Output

The final backend artifact is:

`output.json`

It contains:

- Aggregated scores
- Chronological performance evolution
- Evidence-backed explanations

12 Explicit Non-Goals

The system does not attempt:

- Emotion detection
- Lie detection
- Psychological diagnosis
- Human replacement

All outputs are evidence-based proxies.

13 Conclusion

This backend architecture is modular, deterministic, explainable, and auditable. Strict separation between measurement, interpretation, and aggregation prevents hallucination and ensures reliability.