Limitations and Future Improvements

Current Limitations

1. **Vector Search Accuracy (Text-Based Embeddings Only)**
   The system uses text embeddings (`all-MiniLM-L6-v2`) derived from:
   - video object labels + summaries
   - audio transcriptions

   While this enables a **unified cross-media search**, it is an approximation of true visual/audio similarity. For videos, similarity is inferred from detected object semantics rather than raw visual features. This can occasionally lead to semantically “close” but visually unrelated results, especially in small datasets.

2. **In-Memory / Brute-Force Vector Search**
   Vector similarity is computed in Python using cosine similarity over embeddings loaded from SQLite. This approach is:
   - simple and transparent
   - sufficient for small to medium datasets

   However, it does not scale efficiently to very large numbers of media files, as similarity computation is O(n) per query.

3. **CPU-Only Media Processing**
   All processing is optimized for CPU usage:
   - MobileNet-SSD for object detection
   - Whisper-tiny for transcription

   This ensures portability but limits throughput and accuracy compared to GPU accelerated or larger models. Processing large videos or long audio files can be time-consuming.

4. **Single-Node Queue and Workers**
   The unified processing queue is implemented in process (single node).

   While it supports retries, progress updates, and error handling, it is not distributed and therefore limited in horizontal scalability.

5. **SQLite as Primary Storage**
   SQLite is used for simplicity and ease of deployment. While appropriate for the assignment scope, it is not ideal for:
   - high write concurrency
   - very large datasets
   - multi-instance deployments

Future Improvements

1. **Improved Vector Search Infrastructure**
   With more resources, the vector search layer could be upgraded to:
   - FAISS (CPU/GPU) for fast approximate nearest-neighbor search
   - a dedicated vector database

   This would significantly improve search performance and scalability while preserving the existing embedding approach.

   This would enable more accurate **visual similarity** search and reduce semantic drift.

2. **GPU Acceleration and Model Upgrades**
   With GPU availability:
   - Replace MobileNet-SSD with more accurate detectors (YOLOv8, etc)
   - Use larger Whisper models for improved transcription accuracy
   - Process more frames per video for higher recall

   The current design cleanly separates pipelines, making such upgrades straightforward.

3. **Distributed Queue and Worker Scaling**
   The job queue could be externalized to:
   - Redis
   - RabbitMQ
   - cloud-native task queues

   This would allow:

   - multiple workers
   - parallel media processing
   - better fault isolation
4. **Streaming and Chunked Processing**
   For very large media files:
   - process video/audio in chunks
   - stream frames or audio segments incrementally
   - provide finer-grained progress updates

   This would improve responsiveness and reduce memory pressure.

5. **Enhanced Frontend UX**
   Possible improvements include:
   - side-by-side comparison for reference search
   - richer visual overlays for detected objects
   - pagination for large result sets

Summary

The current implementation prioritizes **clarity, correctness, and CPU-friendly execution**, aligning with the constraints of a take-home assignment. The architecture cleanly separates concerns (frontend, backend, queue, pipelines, search), making it well-positioned for future scaling in performance, accuracy, and infrastructure sophistication.