Technical Design Document (TDD): Teach by Doing (TbD) MVP - V2.0

Status: Approved for Build (Cloud MVP)

Author: Greg Burns

Date: November 22, 2025

Objective: Transition TbD to a fully asynchronous, scalable, and decoupled GCP architecture to eliminate Cloud Run timeout risks and establish the GCS transport layer.

---

1. System Overview and Architectural Shift

1.1 V2.0 Core Architecture: Decoupled Fan-Out

V2.0 adopts the **Dispatcher-Worker Pattern** to safely handle long-running video processing jobs on a serverless platform. This eliminates the risk of hitting the 60-minute Cloud Run timeout.

- **Dispatcher Service (Frontend):** Receives the simple GCS path and immediately queues the job (low latency, fast return).
- **Worker Service (Backend):** Subscribes to the queue, pulls the video, runs the heavy processing, and uploads the results.

1.2 Tech Stack

Component	**Technology**	**Role**
Transport	Google Cloud Pub/Sub	The asynchronous queue (decouples services).
Storage	Google Cloud Storage (GCS)	The reliable input/output bucket.
Compute	Google Cloud Run (Two Services)	Scalable, containerized execution environment.

Processing	Python/OpenCV/Tesseract	Execution of the V1 "Bridge Stack" logic.
Infrastructure	Terraform (Implied for future provisioning)	Ensures reproducible deployment.

---

2. Service Definitions

2.1 Service A: Dispatcher (Frontend API)

This service is the public-facing entry point.

Requirement	**Implementation Details**
Input Source (IR-01)	Expose a RESTful POST /submit endpoint. Accepts a JSON payload containing gcs_uri (FR-02) and pathway_name.
Logic	Validates the input URI. Publishes a standardized message to the Pub/Sub topic (tb-d-ingest-tasks).
Output (IR-02)	MUST immediately return a **HTTP 202 Accepted** status code with a Task ID (e.g., UUID).

GCP Hook: This service will be triggered by a direct HTTP request.

2.2 Service B: Worker (Backend Processing Engine)

This service runs the heavy lifting and must be resilient.

Requirement	**Implementation Details**
Trigger (IR-03)	Exposed as a Pub/Sub subscriber endpoint (not public HTTP).

Ingestion (FR-03, NFR-04)	**GCS SDK:** Uses the official google-cloud-storage Python SDK to download the video specified by the gcs_uri to the local /tmp directory. Must use a try...finally block to delete the file.
Processing (FR-04, NFR-03)	**Core V1 Pipeline:** Executes the existing PySceneDetect, OpenCV, and Tesseract logic defined in the V1 TDD. **Must be fully stateless.**
Output (FR-07)	The final Pathway.json payload is uploaded directly to a GCS results bucket.

GCP Hook: This service will be configured via an IAM policy to subscribe to the Pub/Sub topic.

---

3. Data Flow and Execution Logic

3.1 GCS Ingestion Flow (FR-03)

The key code change is abstracting the file source:

1. **Parse URI:** gcs_uri is parsed into bucket_name and blob_name.
2. **Download:** google-cloud-storage.download_blob_to_file is called.
3. **Local Path:** The processing pipeline uses the temporary local path (e.g., /tmp/video.mp4).
4. **Cleanup:** os.remove(local_path) is executed upon job completion/failure.

3.2 Pub/Sub Messaging Schema

The payload must carry all necessary information to the worker:

JSON
```
{
  "task_id": "UUID-V4",
  "client_id": "manufacturing-systems-inc",
  "gcs_uri": "gs://pad-raw-video/tbddemo_1.mp4",
  "output_bucket": "pad-results-processed"
}
```

3.3 State Management (NFR-03)

The system **MUST** be entirely stateless:

- **No local session files** between requests.
- **No database reads** during core processing loop (V3/MongoDB is out of scope).

- All configuration (like processing thresholds) is passed via the **Pub/Sub message** or environment variables.

---

4. Operational Requirements (Deployment)

4.1 Containerization (NFR-03, NFR-05)

- **Single Container Image:** Both the Dispatcher and Worker will use the same Docker image.
- **Entrypoint:** The service's CMD in the Dockerfile will be determined by a SERVICE_TYPE environment variable passed by Cloud Run (dispatcher or worker).

4.2 Security & IAM (NFR-06)

IAM roles are critical for security:

- **Dispatcher SA:** Needs permissions to **Publish** to the Pub/Sub topic.
- **Worker SA:** Needs permissions to **Subscribe** to the Pub/Sub topic and **Read/Write** to the GCS buckets.

4.3 Scalability (NFR-02)

- The Worker Service will be configured with a high maximum number of concurrent instances (e.g., max-instances=50) to handle large-scale simultaneous video processing tasks.

TDD Update Log - V2.1 (Post-Build)

4. Operational Requirements

4.1 Containerization & Runtime

- **Requirement Update:** The application router (`main.py`) MUST utilize **lazy loading** for service modules. The Dispatcher service must not import heavy dependencies (OpenCV, Scenedetect) at the top level to ensure fast cold starts and prevent memory crashes on standard Cloud Run instances.

2. Service Definitions

2.2 Service B: Worker

- **Retry Strategy:** The Worker MUST return a 200 OK status to Pub/Sub only upon successful processing. In the event of a processing failure (e.g., corrupt video), it SHOULD return a 500 error to trigger Pub/Sub's retry policy (exponential backoff), or explicitly acknowledge dead-letter messages.

1. System Overview

1.2 Tech Stack & Resources

- **Dispatcher Resources:** Minimum 2GiB Memory, 1 CPU (to support Python runtime overhead).
- **Worker Resources:** Minimum 2GiB Memory, 2 CPU.
- **Timeout:** Worker service timeout set to 3600 seconds (1 hour).

5. Security & Access (Refined)

5.1 Frontend Integration

- **Service Account:** External clients (like the Streamlit Test Console) require a Service Account Key with `roles/storage.objectAdmin` on both the Input and Output buckets to bypass strict bucket metadata permissions (`storage.buckets.get`) that are often restricted.