PrivacyScrub V4: Technical Design Document (TDD)

Author: Greg Burns **Version:** 4.2 (Final Design Document) **Date:** November 20, 2025

1. Introduction and Project Goals

This TDD outlines the technical blueprint for PrivacyScrub V4. The document is designed to achieve **100% compliance** with the V4 SRS, leveraging a local, GPU-accelerated, multi-model architecture for video processing.

1.1 V4 Implementation Mandate

The design adheres to these core architectural principles:

- **Local Model Fusion:** Eliminate network latency by hosting the entire multi-model stack (Faces, Plates, Text, Logos) locally on the worker.
- **Compliance Enforcement:** Rigorously enforces non-negotiable profile constraints (GDPR, HIPAA) within the configuration logic ($\text{get\_config\_for\_profile}$).
- **Stability:** Implements explicit fixes for the known V3 pipeline bugs (read-only array and stuck status).

2. System Architecture

The V4 architecture is a high-performance, distributed, event-driven model on GCP, optimized for high concurrency.

2.1 Worker Service: V4 Multi-Model Detection Stack

The worker container hosts the entire model stack and the processing pipeline, fulfilling FR-DET-02.

- **Backbone / General:** RT-DETR or YOLOv8 (Local). Provides general object context (Persons, Vehicles).
- **Face Detector:** Dedicated Face model (Local). Provides high-accuracy detection of faces.
- **Plate Detector:** Dedicated Plate model (Local). Provides high-accuracy license plate identification.

- **Text Detector:** Local $\text{Scene Text Detector}$ (DBNet/CRAFT) plus OCR Recognizer. Extracts bounding boxes and content for sensitive text.
- **Logo Detector:** Local Logo Classifier (e.g., fine-tuned YOLO head). Identifies high-value brand marks.
- **Fusion Layer:** Custom Python logic combining bounding box results, resolving conflicts, and applying ROI filtering before anonymization.

2.2 Job Lifecycle and Endpoints

The API surface defines the contract for consuming the asynchronous worker service.

- **Endpoints:**
    - $\text{POST /v1/anonymize-video}$: Initiates job.
    - $\text{GET /v1/jobs/\{job\_id\}}$: Returns the full status object.
    - $\text{DELETE /v1/jobs/\{job\_id\}}$: Sets $\text{status}=\text{CANCELLED}$ (FR-VID-07).
- **Status Transitions (FR-VID-01):** The worker implements explicit state changes to guarantee accurate client feedback:
    - $\text{QUEUED} \rightarrow \text{CHUNKING} \rightarrow \text{PROCESSING} \rightarrow \text{STITCHING} \rightarrow \text{COMPLETED}$.

3. Functional Requirements and Implementation Details

3.1 Job Lifecycle API Endpoints (FR-VID-01, FR-VID-07, FR-VID-08)

The following public endpoints expose the asynchronous video processing lifecycle:

- $\text{POST /v1/anonymize-video}$
    - Validates the uploaded video (type/size).
    - Uploads to GCS under $\text{input/\{job\_id\}/original.mp4}$.
    - Creates a Firestore job document with status = "QUEUED" and captured configuration.
    - Enqueues the $\text{/internal/split-video}$ task.
- $\text{GET /v1/jobs/\{job\_id\}}$
    - Returns a status object containing:
        - status ($\text{QUEUED}, \text{CHUNKING}, \text{PROCESSING}, \text{STITCHING}, \text{COMPLETED}, \text{FAILED}, \text{CANCELLED}$)
        - chunks_total, chunks_completed (for progress)
        - output_url (if COMPLETED or coordinates-only manifest ready)
        - error_message (if FAILED).
- $\text{DELETE /v1/jobs/\{job\_id\}}$
    - Sets $\text{status = "CANCELLED"}$ for the job.

    - Workers MUST check for CANCELLED before starting heavy work and abort gracefully if set.

3.2 Compliance Profile Enforcement (FR-CP-02)

The configuration logic strictly enforces profile requirements, overriding user inputs where necessary.

- **GDPR:** $\text{confidence\_threshold} \ge 0.6$, $\text{target\_faces} = \text{True}$, $\text{target\_text} = \text{True}$, $\text{strip\_metadata} = \text{True}$.
- **HIPAA_SAFE_HARBOR:** $\text{confidence\_threshold} \ge 0.7$, $\text{mode} = \text{BLACK\_BOX}$, $\text{strip\_metadata} = \text{True}$, $\text{target\_all} = \text{True}$.
- **CCPA:** $\text{confidence\_threshold} \ge 0.55$, $\text{target\_faces} = \text{True}$, $\text{target\_plates} = \text{True}$, $\text{strip\_metadata} = \text{True}$.
- **NONE:** Default profile. $\text{target\_all} = \text{True}$, $\text{mode} = \text{BLUR}$, $\text{confidence\_threshold} = 0.5$, $\text{strip\_metadata} = \text{True}$.

3.3 Detection Interface and Data Output

- **Image API Wiring (FR-IMG-01):** The $\text{/v1/anonymize-image}$ endpoint accepts all Form fields for per-target toggles ($\text{target\_faces}, \text{target\_plates}, \text{target\_logos}, \text{target\_text}$) and mode, applying profile constraints first.
- **EXIF / Metadata Strip (FR-IMG-03):** For images, when $\text{strip\_metadata} = \text{True}$ (e.g., under GDPR), the $\text{/v1/anonymize-image}$ implementation MUST re-encode the output using a clean Pillow save path, ensuring all EXIF and other metadata are removed from the returned file.
- **Coordinates-Only Mode (FR-COORD-01):**
    - **Images (**POST**):** Returns a $\text{JSON payload directly}$ with detections.
    - **Videos (**GET**):** The worker writes a JSON manifest to GCS with per-frame detections; $\text{GET /v1/jobs/\{job\_id\}}$ returns the GCS link to this manifest in output_url.
- **ROI Handling (FR-IMG-01):** The $\text{Model Fusion Layer}$ filters all detections to ensure only bounding boxes that intersect the optional ROI rectangle are considered for anonymization in both images and video frames.

3.4 Reference UI (Streamlit)

The Streamlit UI acts as a thin client over the public API.

- **Images:** Provides configuration controls (targets, mode, profile, ROI) and renders a before/after preview.

- **Videos:** Calls $\text{POST /v1/anonymize-video}$ for submission, then polls $\text{GET /v1/jobs/\{job\_id\}}$ to display status, chunk progress, and the final output_url.
- **QA View:** An internal QA mode renders original vs. anonymized frames side-by-side with detection overlays for review (FR-HIL-01).

4. Operational and MLOps Requirements

4.1 Performance and GPU Acceleration (NFR-PERF-01)

- **Frame Mutability (FR-DET-07):** The frame_processor MUST create a writeable copy immediately ($\text{frame = np.array(frame, copy=True)}$) to eliminate the V3 "read-only array" bug.
- **Hardware Codecs:** The worker utilizes NVDEC/NVENC (via ffmpeg) for hardware video decode and encode to achieve the $\ge 1\text{x}$ real-time SLO.
- **Resource Allocation:** Cloud Run is configured with 4 GiB memory and 2 vCPUs to support the multi-model stack.

4.2 Security and Observability

- **Logging Hygiene (NFR-SEC-03):** Logs MUST be scrubbed to avoid storing raw frames or PII. Only job IDs, technical metrics, and redacted error messages are permitted.
- **Metrics and CI/CD Gating (FR-EVAL-01 / FR-EVAL-03):**
    - An automated evaluation suite runs against benchmark datasets ($\text{faces, plates, logos, text}$) to compute metrics ($\text{F1, mAP}$).
    - CI/CD promotion is gated; new models MUST pass configured metric thresholds.
- **Drift Monitoring:** Detection statistics ($\text{confidences, targets-per-frame}$) are periodically tracked and compared against baselines to flag potential model degradation.
- **Data Retention (NFR-SEC-04):** GCS bucket lifecycle rules MUST enforce TTL deletion of all media files.
- **Multi-Tenancy:** The API gateway enforces per-tenant quotas and rate limits, and metrics are segmented by tenant identifier.