PrivacyScrub – Software Requirements Specification (V4)

1. Introduction

1.1 Purpose

This document defines the **V4 Software Requirements** for **PrivacyScrub**, an API-first media anonymization platform that detects and obscures sensitive information in images and videos to support privacy and regulatory compliance (GDPR, CCPA, HIPAA Safe Harbor–inspired) for user-generated and professional media.

V4:

- **Includes and refines all requirements from V1–V3**, including: * Image & video anonymization. * Asynchronous job handling, chunking, and stitching. * Local inference (no per-frame external API calls). * GPU acceleration and hardware video codecs. * Compliance profiles and evaluation pipeline.
- Introduces a **modern, multi-model anonymization stack** and world-class MLOps/production guarantees.

This SRS is intended for:

- Backend engineers (FastAPI, async workflows, GCP).
- ML engineers (detection models, evaluation, active learning).
- DevOps/SRE/MLOps (deployment, observability, GPU infra).
- Security, privacy, and compliance stakeholders.

1.2 Scope

PrivacyScrub V4 provides:

- API-first anonymization for **images and videos**.
- Detection and redaction of: * Faces * License plates * Logos / brand marks * On-screen text (signs, jerseys, documents, name labels) * EXIF and container metadata (e.g., GPS, device IDs, author fields) * Other visual PII/PHI relevant to GDPR/CCPA/HIPAA principles
- Multiple anonymization modes: * Blur * Pixelate * Black box (solid mask)
- Asynchronous video processing with chunking and parallel processing.
- Local, GPU-accelerated inference stack (no frame-level HTTP calls).

- Compliance profiles: * `NONE` * `GDPR` * `CCPA` * `HIPAA_SAFE_HARBOR`
- Evaluation pipeline with public benchmarks and internal test suites.
- Streamlit (or similar) UI as a reference client for the APIs.

Out of scope for V4, but considered future work:

- Audio anonymization (voice redaction, beeps, etc.).
- Full document anonymization beyond text within images/videos.
- On-premise and air-gapped deployments (architecture should not preclude them).

1.3 Definitions and Acronyms

- **PII** – Personally Identifiable Information
- **PHI** – Protected Health Information
- **GDPR** – General Data Protection Regulation (EU)
- **CCPA** – California Consumer Privacy Act
- **HIPAA Safe Harbor** – HIPAA de-identification method based on removing or generalizing 18 identifiers
- **ROI** – Region of Interest (rectangular region in image/video frames)
- **RT-DETR** – Real-Time Detection Transformer (or similar efficient general object detector)
- **DBNet / CRAFT / EAST** – State-of-the-art scene text detection architectures
- **Logos in the Wild** – Example logo detection dataset for logo anonymization

2. System Overview

2.1 Product Vision

PrivacyScrub allows users—from journalists to real estate agents to marketers—to:

> “Upload media → configure compliance/profile → receive anonymized media or detection masks → publish confidently without exposing unnecessary PII/PHI or brand conflicts.”

V4’s vision is to be a **world-class anonymization engine**, with:

- High recall and precision for critical PII/PHI elements.
- Real-time or faster-than-real-time video throughput in GPU deployments.
- Compliance-aware defaults via profiles.
- Measurable performance and quality via standardized evaluation.

2.2 Architecture Perspective

Core components:

1. **API Service (FastAPI on Cloud Run/GKE)** - REST endpoints for image & video anonymization. - Job creation, job status, cancellation endpoints. - Stateless; uses durable backing services for state.
2. **Job Orchestrator** - Cloud Tasks (or equivalent) for chunk-based video processing. - Firestore (or equivalent) for job metadata: - Status - Chunk information - Options snapshot - Metrics and errors
3. **Worker Service (GPU-capable)** - Local inference engine with a **multi-model stack**: - RT-DETR (or similar) for general object detection (people, vehicles, etc.). - Dedicated face detection model (e.g., RetinaFace/YOLO-face). - Dedicated license plate detection model. - Scene text detection model (DBNet/CRAFT/EAST style) plus text recognizer. - Logo detection model fine-tuned on a logo dataset (e.g., Logos in the Wild). - Hardware-accelerated video decode/encode (NVDEC/NVENC or equivalent). - Anonymization (blur/pixelate/black box) applied frame-wise.
4. **Storage (Google Cloud Storage)** - Temporary storage of: - Original inputs - Chunked intermediates - Final anonymized outputs - Strict lifecycle policies and TTL-based deletion.
5. **Model Registry & Configuration** - Versioned records of model artifacts and config: - Model names and hashes - Configs per target type (faces, plates, logos, text) - Integration with evaluation pipeline and deployment gating.
6. **Frontend UI (Streamlit)** - Not authoritative for application logic. - Convenience client for demos, QA, and manual validation. - Uses only documented API endpoints.

3. Functional Requirements

All requirements from V1–V3 are retained and superseded where necessary. V4 adds a multi-model stack, improved compliance, and production hardening.

3.1 Targets & Compliance Profiles

3.1.1 Target Categories

Category 1 – Core PII (Always Supported)

- Faces (full and identifiable partial faces).
- License plates (all regional formats).
- EXIF/metadata: * GPS coordinates * Device IDs / camera model and serial * Timestamps (where appropriate) * Author / creator fields
- Household-level visual identifiers: * Nameplates on doors/mailboxes * Name badges (where detectable)

Category 2 – Toggleable High-Value Targets

- Logos / brand marks: * Clothing logos * Store signs * Product packaging

- On-screen text: * Street signs   * Store names   * Jersey names and numbers   * Document text
- Financial information: * Credit/debit card numbers   * Bank account numbers
- Signatures: * Handwritten or digital signatures
- Tattoos / strong unique identifying marks
- Other high-risk text: * Email addresses   * Phone numbers   * Physical addresses

Category 3 – Advanced / Location-Based

- Specific addresses and location details from visible signs.
- Highly distinctive building facades, when combined with strong location cues.
- Contextual identifiers: * Hospital or clinic names (medical context)   * School names in footage featuring minors.

For V4, Category 3 is handled via a combination of text recognition, metadata removal, and object detection. Full “scene uniqueness” modeling remains a future enhancement.

3.1.2 Compliance Profiles

```
compliance_profile: "NONE" | "GDPR" | "CCPA" | "HIPAA_SAFE_HARBOR"
```

NONE

- Defaults: * $\text{faces} = \text{true}$ * $\text{plates} = \text{true}$ * $\text{logos} = \text{true}$ * $\text{text} = \text{true}$
- $\text{mode} = \text{blur}$
- $\text{confidence\_threshold} = 0.5$
- $\text{strip\_metadata} = \text{true}$
- User may toggle any target on or off, and change mode/threshold.

GDPR

- Force-enable: * Faces   * License plates   * Text related to names and addresses
- $\text{strip\_metadata} = \text{true}$ (especially GPS and device IDs).
- Minimum $\text{confidence\_threshold}$ (e.g., $\ge 0.6$).
- Default $\text{mode} = \text{blur}$.
- User cannot disable core PII targets; may enable additional ones (logos, tattoos, etc.).

CCPA

- Force-enable: * Faces   * License plates   * Text relating to household-level identifiers (addresses, family names)
- $\text{strip\_metadata} = \text{true}$.
- Similar or slightly higher thresholds than default.
- User cannot disable core PII targets; may enable additional ones.

HIPAA_SAFE_HARBOR

- Force-enable: * Faces (full-face and comparable images) * License plates / vehicle identifiers * Text that contains: * Names * Contact info * Account numbers
- $\text{strip\_metadata} = \text{true}$ (GPS and other potential identifiers).
- Default $\text{mode} = \text{black\_box}$ or strong $\text{pixelate}$.
- Minimum $\text{confidence\_threshold}$ (e.g., $\ge 0.7$).
- Documented as a technical aid only; no legal guarantee of compliance.

FR-CP-01: Both image and video paths MUST apply $\text{get\_config\_for\_profile}$ and enforce these constraints.

FR-CP-02: User options MAY further restrict targets (turn off non-mandatory ones), but MUST NOT disable profile-mandated targets.

3.2 Image Anonymization (Synchronous)

Endpoint: $\text{POST /v1/anonymize-image}$

FR-IMG-01: Accepts:

- File: $\text{image/jpeg}$ or $\text{image/png}$ as $\text{multipart/form-data}$.
- Options via form fields or nested JSON, including at least: * $\text{targets.faces}: \text{bool}$ * $\text{targets.plates}: \text{bool}$ * $\text{targets.logos}: \text{bool}$ * $\text{targets.text}: \text{bool}$ * $\text{mode}: \text{"blur" | "pixelate" | "black\_box"}$ * $\text{confidence\_threshold}: \text{float}$ * $\text{coordinates\_only}: \text{bool}$ * $\text{compliance\_profile}: \text{string}$ * Optional $\text{roi}: [x1, y1, x2, y2]$ in normalized or pixel coordinates.

FR-IMG-02: The endpoint MUST:

1. Parse the form fields / JSON.
2. Construct a $\text{PrivacyConfig}$ from the compliance profile.
3. Override $\text{PrivacyConfig}$ with user-specified options while respecting mandatory profile protections.
4. Run detection on the image using the V4 detection stack (see $\S 3.4$).
5. Apply anonymization within detected bounding boxes, respecting: - Selected targets - Mode - Confidence threshold - ROI
6. Return: - An anonymized image (binary) if $\text{coordinates\_only} = \text{false}$, or - A JSON structure describing detections if $\text{coordinates\_only} = \text{true}$.

FR-IMG-03: If $\text{strip\_metadata} = \text{true}$, output image MUST have all $\text{EXIF/metadata}$ removed.

FR-IMG-04: The API MUST return structured JSON error payloads for invalid input (type, size, corrupted file, invalid options).

3.3 Video Anonymization (Asynchronous, Chunked)

Endpoints:

- $\text{POST /v1/anonymize-video} \rightarrow$ submit a video for anonymization.
- $\text{GET /v1/jobs/\{job\_id\}} \rightarrow$ get job status and info.
- $\text{DELETE /v1/jobs/\{job\_id\}} \rightarrow$ cancel a job.

3.3.1 Job Lifecycle

FR-VID-01: Job statuses:

- $\text{QUEUED}$ – Job metadata created, video uploaded.
- $\text{CHUNKING}$ – Splitting video into chunks.
- $\text{PROCESSING}$ – Chunks being processed.
- $\text{STITCHING}$ – Processed chunks being combined.
- $\text{COMPLETED}$ – Final video available.
- $\text{FAILED}$ – Error occurred; error message should be provided.
- $\text{CANCELLED}$ – Job cancelled by client or system.

FR-VID-02: $\text{POST /v1/anonymize-video}$ MUST:

- Validate the uploaded video (e.g., $\text{video/mp4}$ $\text{v1}$).
- Upload it to GCS (e.g., $\text{input/\{job\_id\}/original.mp4}$).
- Create a Firestore job document with: * $\text{status = "QUEUED"}$   * $\text{chunks\_total} = 0$   * $\text{chunks\_completed} = 0$ * Options snapshot (including $\text{profile}$, $\text{targets}$, $\text{mode}$, $\text{threshold}$).
- Enqueue a split task (e.g., $\text{/internal/split-video}$).

3.3.2 Splitting & Chunk Processing

FR-VID-03: Split task ($\text{/internal/split-video}$) MUST:

- Inspect video duration.
- If $\text{duration} \le \text{MIN\_CHUNK\_DURATION\_SEC}$:   * Set $\text{chunks\_total} = 1$.   * Use original file as the single chunk.
- Else: * Use $\text{ffmpeg}$ to segment into $\text{N}$ chunks (default $\sim 5$ minutes).   * Store each chunk at $\text{input/\{job\_id\}/chunks/chunk\_\{i\}.mp4}$.   * Set $\text{chunks\_total} = \text{N}$.
- Set job $\text{status} = \text{"CHUNKING"}$.
- Enqueue one process-chunk task per chunk.

FR-VID-04: Process-chunk task ($\text{/internal/process-chunk}$) MUST:

- For the first started chunk, update job status from $\text{CHUNKING} \rightarrow \text{PROCESSING}$ (if not already updated).

- Download the chunk file from GCS to local disk.
- Run anonymization frame-wise using the V4 detection engine (see $\S 3.4$).
- Use a writeable frame copy to avoid “assignment destination is read-only” errors.
- Encode the processed chunk to $\text{output/\{job\_id\}/chunks/chunk\_\{i\}.mp4}$ with hardware acceleration if available.
- Upload processed chunk to GCS.
- Increment $\text{chunks\_completed}$ in Firestore.
- If $\text{chunks\_completed} \ge \text{chunks\_total}$ and no chunk has failed: * Enqueue $\text{/internal/stitch-video}$.

3.3.3 Stitching & Completion

FR-VID-05: Stitch task ($\text{/internal/stitch-video}$) MUST:

- Set job status to $\text{STITCHING}$.
- Concatenate processed chunks in correct order using $\text{ffmpeg}$.
- Strip metadata from final video output.
- Write final video to $\text{output/\{job\_id\}/final.mp4}$.
- Generate a signed URL or API-accessible download path.
- Update job: * $\text{status = "COMPLETED"}$ * $\text{output\_url} = <\text{generated\_url}>$

3.3.4 Error Handling & Cancellation

FR-VID-06: Any unhandled exception in split/process/stitch tasks MUST:

- Set job $\text{status = "FAILED"}$.
- Populate $\text{error\_message}$ with a concise description.

FR-VID-07: $\text{DELETE /v1/jobs/\{job\_id\}}$ MUST:

- Set job $\text{status = "CANCELLED"}$.
- Prevent scheduling of further tasks for that job.
- Workers MUST check for cancellation before heavy work and abort gracefully when safe.

3.3.5 Progress Reporting

FR-VID-08: $\text{GET /v1/jobs/\{job\_id\}}$ MUST return:

- $\text{job\_id}$
- $\text{status}$
- $\text{created\_at}$, $\text{updated\_at}$
- $\text{chunks\_total}$
- $\text{chunks\_completed}$
- Optional $\text{chunks\_failed}$

- Derived $\text{progress}$ ($\text{0.0 – 1.0}$) when relevant
- $\text{output\_url}$ and $\text{TTL}$ (if $\text{COMPLETED}$)
- $\text{error\_message}$ (if $\text{FAILED}$)

3.4 Detection & Anonymization Engine (V4 Model Stack)

V4 introduces a multi-model detection stack instead of a single $\text{YOLO-style}$ model.

3.4.1 Detection Interface

FR-DET-01: All detection MUST flow through a central interface:

```
def detect_frame(
    frame: np.ndarray,
    config: PrivacyConfig
) -> List[Detection]:
    """
    Returns a list of detections, each with:
      - type (face, plate, logo, text, tattoo, etc.)
      - bbox/poly coordinates
      - confidence
      - metadata (e.g., recognized text)
    """
```

FR-DET-02: The detector MUST orchestrate the following internal sub-models:

- **General Object Detector:** Model: $\text{RT-DETR}$ or equivalent real-time detector.
- **Face Detector:** Dedicated face detection model (e.g., $\text{RetinaFace/YOLO-face}$).
- **License Plate Detector:** Dedicated license plate detection model.
- **Scene Text Detection + OCR:** Text detection model ($\text{DBNet/CRAFT/EAST}$ style) plus text recognizer.
- **Logo Detection:** Logo-specific detector fine-tuned on a logo dataset.

FR-DET-03: Detection outputs MUST include:

- $\text{type}$ ($\text{e.g., "face", "plate", "logo", "text", "tattoo"}$)
- $\text{bbox}$ and/or $\text{poly}$ ($\text{coordinates}$)
- $\text{confidence}$
- Optional: $\text{text content}$ ($\text{for text detections}$)

FR-DET-07: Implementation MUST operate on a writeable frame array:

- At start of frame pipeline: $\text{frame = np.array(frame, copy=True)}$ or equivalent.
- Avoid in-place modifications to read-only views.

3.5 Coordinates-Only Mode & ROI

FR-COORD-01: If $\text{coordinates\_only} = \text{true}$:

- **Images (**$\text{POST}$**):** Returns a $\text{JSON payload directly}$ with detections.
- **Videos (**$\text{GET}$**):** The worker writes a $\text{JSON manifest}$ to $\text{GCS}$ with per-frame detections; $\text{GET /v1/jobs/\{job\_id\}}$ returns the $\text{GCS}$ link to this manifest in $\text{output\_url}$.

ROI Handling ($\text{FR-IMG-01}$ **/** $\text{V4 Detection}$**):**

- The detection functions accept an optional $\text{ROI}$ parameter.
- For **images** and **video frames**, only detections whose bounding boxes intersect the $\text{ROI}$ rectangle are considered for anonymization.

3.6 Reference UI (Streamlit)

The Streamlit UI acts as a thin client over the public API ($\text{FR-UI-01}$).

- **Images:** Provides upload controls and $\text{UI}$ elements for selecting targets, $\text{mode}$, $\text{profile}$, $\text{confidence}$, and optional $\text{ROI}$ inputs.
- **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 $\text{output\_url}$.
- **QA View (FR-HIL-01):** An internal $\text{QA}$ mode renders original vs. anonymized frames side-by-side with detection overlays for review. Reviewer adjustments are stored in an access-controlled dataset for future model retraining and analysis.

4. Non-Functional Requirements (NFR)

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

- **Hardware Codecs:** The worker utilizes $\text{NVDEC/NVENC}$ (via $\text{ffmpeg}$) for hardware video decode and encode to achieve the $\ge 1\text{x}$ real-time $\text{SLO}$.
- **Resource Allocation:** $\text{Cloud Run}$ is configured with $\text{4 GiB}$ memory and $\text{2 vCPUs}$ 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.
- **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.