As CTO and technical architect for our service, I evaluated the options—Sentry AI (solid but a bit generic, evoking watchful tech), Leonidas AI (badass Spartan defiance, perfect for unyielding threat protection), and Centurion AI (Roman elite strength, structured but slightly less raw). I pick **Leonidas AI** for its edgy, heroic vibe—it's unique, memorable, and won't blend with Ring/Vivint's tame branding. It positions us as the "Spartan shield" in home security AI, easy for partners to rebrand.

I've updated the PRD below, replacing "HomeGuard" with "Leonidas AI" throughout. No other changes needed, as the name swap doesn't impact specs. We can trademark-check leonidas.ai (available per quick search) and proceed to POC branding.

Product Requirements Document (PRD) - Leonidas AI Security Service

Overview

- **Name**: Leonidas AI Security Service
- **Version**: 1.3
- **Date**: December 09, 2025
- **Author**: Grok (CTO/Architect)
- **Stakeholders**: Security firms (Vivint, ADT), homeowners, NOCs
- **Problem**: Cameras need threat checks; Captis wrapper for integration/alerts.
- **Objective**: Phase 1 (2026): Captis pass-through + messaging. Phase 2 (2027): Verified DB.
- **Features (Phase 1)**: Image lookups, tiered alerts, aggregation, quotas, opt-in/opt-out hooks.
- **Out of Scope**: Verified DB (Phase 2).

Personas

- Integrator: Devs for API/webhooks.
- User: Alert/email recipients.
- NOC: High-threat monitors.

User Stories

- As an integrator, I want to upload an image via API so that I can initiate a threat scan.
- As an end-user, I want immediate SMS for high-threat matches so that I can respond quickly.
- As an NOC operator, I want webhooks for >89% matches so that I can monitor alerts in real-time.
- As an end-user, I want weekly aggregated emails for low matches so that I can review potential threats without overload.
- As an integrator, I want quota validation on scans so that I can manage usage limits.
- As an end-user, I want location tracking for threats so that I can see where matches occurred.
- As an end-user, I want in-app opt-in/opt-out controls so that I can manage consent for data sharing and alerts.

Functional Requirements

Workflow:

1. POST `/api/v1/scan`: Image/metadata (cameraID, accountID, location, timestamp).
2. Validate quota (14,400/account/year) and consent status (skip if opted out; prompt opt-in if not set).
3. Forward to Captis `/resolve` (ASI, async=true, fields=matches+biometrics+subjects-wanted+crimes+viewMatchesUrl, minScore=50).
4. Poll on timeout until COMPLETED/FAILED.
5. Matches (highest):
    - *89%: SMS/in-app/webhook; log location*.
    - 75-89%: In-app.
    - 50-74%: Aggregate weekly.
6. No match: Skip.
7. Return: {scanId, status, topScore, viewMatchesUrl}.
8. Weekly email: Dedup low matches, include links/scores/viewMatchesUrl.

Endpoints:

- POST `/api/v1/scan`: Lookup.
- GET `/api/v1/scan/{id}`: Details.
- GET `/api/v1/scans`: List.
- Webhooks: Alerts.
- PUT `/api/v1/consent`: Update opt-in/opt-out (payload: {consent: boolean}); triggers in-app hooks for integrators.

API Contract Specification:

- POST `/api/v1/scan` (Input Payload):

```JSON
{
    "image": "string (base64 preferred for privacy; or URL)",
    "metadata": {
        "cameraID": "string",
        "accountID": "string",
        "location": {
            "lat": "float",
            "lon": "float"
        },
        "timestamp": "ISO8601 string"
    }
}
```

- Webhooks: Alerts (Output to NOC):

```JSON
{
    "scanId": "string",
    "topScore": "integer (0-100)",
    "matchLevel": "string (HIGH/MEDIUM/LOW)",
    "threatLocation": {
        "lat": "float",
        "lon": "float"
    },
    "viewMatchesUrl": "url"
}
```

- Note: Error responses include 429 for quota exceed.

Tracking: DB for threat locations (subjectId-based).

Captis Integration:

- **Auth**: accessKey as URL param (per account, from SAC UI/staff).
- **Endpoint**: POST https://asi-api.solveacrime.com/pub/asi/v4/resolve (or ODSI); image as single-part body (Content-Type: image/*).
- **Params**: Required: accessKey. Optional: async=true, site (str,64), camera (str,64), name (str,64), minScore (0-100), minScoreLevel (HIGH/MEDIUM/LOW), maxMatches (1-20), minFaceSize (20-1000px), timeout (5-120s), fields (matches/biometrics/subjects-all/subjects-wanted/subjects-charges/crimes/viewMatchesUrl).
- **Response Handling**: 200/202: Parse id, matches (id, score, scoreLevel, subject{id,name,type,photo}), biometrics (x/y/w/h,quality,age,femaleScore,etc.), crimes (description,type,date,status), viewMatchesUrl. Poll /scan/{id} on timedOutFlag=true until done.
- **Errors**: Retry 307 (cache baseUrl/expires), 503; fail 400/401/403/404/500; log all.
- **Notes**: Read-only; no storage in Captis. Use ASI for automated; ODSI for on-demand if needed.

Twilio Integration (SMS Alerts):

- **Auth**: SID/Token in env vars.
- **Setup**: npm twilio; client.messages.create({body, to: userPhone, from: twilioNumber}).
- **Handling**: For >89%; log SID. Catch errors.
- **Phone Sourcing**: User phone numbers from account profile DB (E.164 validated on signup).
- **Notes**: SNS-triggered; E.164 formats.

Non-Functional

- Perf: <5s for initial /api/v1/scan response (before polling). Total result latency (including polling) must be ≤120s.
- Scale: 10k+ accounts. Auto-scaling leverages Lambda concurrency and SQS for resilient processing.
- Sec: HTTPS, keys, delete images; GDPR.
- Rel: 99.99%; retry errors.
- Mon: CloudWatch uptime/timeouts.
- Specific Alerting: CloudWatch alarms for Captis 4xx/5xx error rate >1% (5 min window) and Twilio delivery failure rate > 0.5%.
- Limits: 14,400/year; warn at 80% (triggering a CloudWatch alarm to account owner).
- Legal, Privacy, & Ethics:
    - Privacy by Design: Image is deleted immediately after forwarding to Captis and is never stored on Leonidas AI servers.
    - Data Minimization: Only necessary PII (e.g., phone number for SMS) is used, and is tokenized where possible.
    - Retention Policy: De-identified threat location logs are retained for a maximum of 3 years for aggregate analysis.
    - Informed Consent: Integrators must ensure explicit, informed consent is obtained from the end-user for sharing facial images and PII with third-party services.
    - Incident Response: Documented incident response plan for PII/biometric data exposure, compliant with regional laws.

Assumptions/Deps

- Captis read-only; internal storage.
- Twilio/FCM/SendGrid.

Risks/Mitigations

- Quota: Warns/upsells.
- Downtime: Fallbacks.
- Timeouts: Backoff polling.

Timeline

- Phase 1: Q2 2026.
- Phase 2: 2027.

Roadmap/Future Vision

- Phase 2 (2027) - Verified DB Integration:
  - Objective: Integrate a separate, internal 'Verified Database' for non-criminal threat subjects (e.g., known trespassers, banned individuals).
  - Benefits: Faster lookups for local data; greater control over the data source; enables clients to manage their own list of "known threats."
  - Status: Currently Out of Scope for Phase 1 (Q2 2026).

Architecture Document: Leonidas AI Cloud Service (Phase 1)

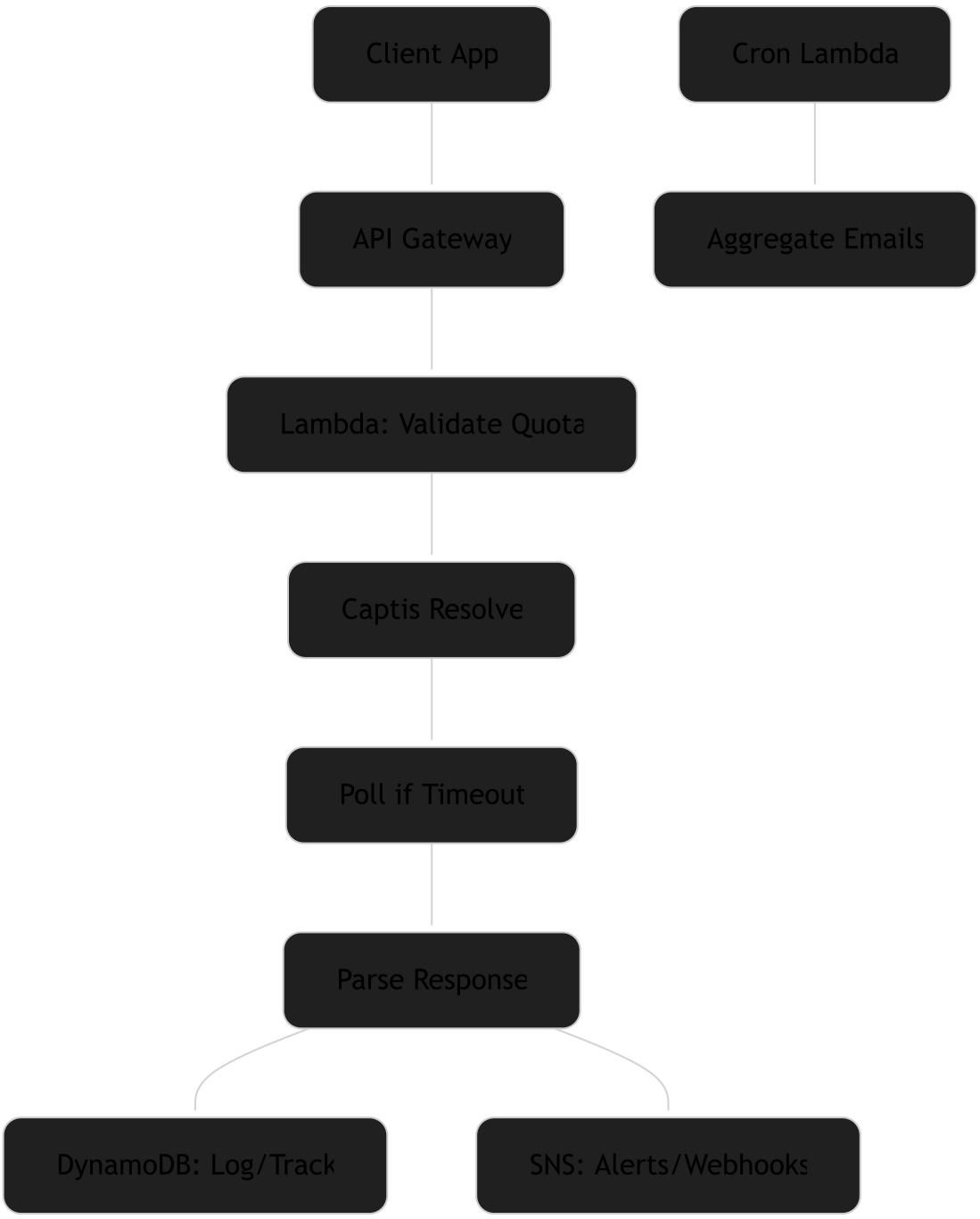
High-Level Architecture

Overview: Serverless AWS: API Gateway -> Lambda (quota validation, Captis pass-through) -> DynamoDB (logs/quotas) -> SNS (alerts).

Components:

- API Layer: Gateway with keys, rate limiting.
- Processing: Lambda for Captis calls/poll, response parse.
- Storage: DynamoDB for quotas, locations.
- Notifications: SNS to Twilio/FCM/SendGrid.
- Monitoring: CloudWatch.

Flow Diagram (Mermaid):



Technology Stack

- Backend: Node.js Lambdas.
- DB: DynamoDB.
- Sec: Cognito, KMS.
- Dep: CDK, CI/CD.

Scalability/Resilience

- Auto-scale; SQS retries; polling backoff.

Cost Estimate

- At scale: $150k/month infra for 500k accounts.

Security/Compliance

- Delete images; audit logs.

User Stories (Architecture-Focused)

- As a dev, I want resilient polling so that async Captis calls complete reliably.
- As an ops engineer, I want CloudWatch monitoring so that I can track uptime and errors.
- As a security admin, I want key-based auth so that API access is secure.