Assignment: Build "SentinelSight" — AI Video Analytics Platform (2-Day Sprint)

Deadline

Submission deadline: Sunday, 11 January 2026 — 6:00 PM IST
Time budget: ~2 days total. Build as much as possible; prioritize a working end-to-end system.

---

1) Objective

Build a working MVP of a **multi-camera video analytics platform** that:

1. Ingests CCTV streams (at least 1–2 RTSP sources)
2. Runs computer vision inference (object detection + simple rules)
3. Generates **events/alerts**
4. Shows results in a **web dashboard** + exposes APIs
5. Is structured to scale (multi-camera, multi-model, modular pipelines)

This should follow best practices seen in modern platforms:

- **VMS + centralized multi-site camera management** (e.g., Milestone)
- **Analytics modules for review/respond/research** concepts (e.g., BriefCam's module framing)
- **AI alerts + accelerated search style experiences** (e.g., Avigilon)
- **Local-first privacy approach + zones + event messaging** (e.g., Frigate with MQTT/events and local processing)

---

2) Mandatory Scope (Must-Haves)

A. Video Ingestion

- Support **RTSP camera streams** (minimum: 1, target: 4+).
- Each stream is treated as a "Camera" entity with:
    - name, location tag, RTSP URL, status (online/offline), FPS (approx), last frame time
- Auto-reconnect on failure.

B. Inference & Analytics

Implement at least one of these model paths:

- **Object detection** (people/vehicle) using any standard model (YOLO family, Detectron, etc.)
- Optional: simple **tracking** (IDs across frames) if time permits

Analytics logic (minimum):

- **Zones** (draw polygon/rectangle zones per camera)
- **Rule engine (at least 2 rules)**:
    1. Intrusion: person enters restricted zone
    2. Loitering: person remains in zone > N seconds
- Generate events containing:
    - camera_id, timestamp, rule, object_type, confidence, bounding boxes, snapshot path

C. Event Store + API

- Persist events in a database (SQLite acceptable for MVP).
- Provide REST API endpoints:
    - GET /cameras
    - POST /cameras (add RTSP stream)
    - GET /events?camera_id=&from=&to=&rule=
    - GET /events/{id}
    - Optional: GET /health, GET /metrics

D. Dashboard (UI)

A simple web UI that shows:

- Camera list + status
- Live preview (best effort; even periodic frames is ok)
- Event feed (filter by camera/rule/time)
- Event details view with snapshot + metadata
- Zone editor (even a basic JSON input is acceptable; GUI drawing is bonus)

E. Packaging / Run Instructions

- Repo with clean structure

- README with setup steps
- Ideally dockerized (bonus but strongly preferred)

---

3) "Freedom to Add Features" — Suggested Enhancements (Choose as many as you can)

You are encouraged to research global products and add features beyond the MVP. Examples inspired by real systems:

VMS-like capabilities (Milestone-style)

- Multi-site grouping / tags ("Dubai – Site A – Floor 3")
- Role-based permissions (Admin/Operator/Viewer)
- Camera onboarding wizard, export/import camera configs

Analytics modules framing (BriefCam-style)

Build features that map to:

- **Review:** post-incident search/filtering
- **Respond:** real-time alerts
- **Research:** trends/heatmaps/counters over time

AI-Driven Operations (Avigilon-style)

- "Attention required" alert view (reduce information overload)
- Faster review: timeline scrub + event jump navigation

Local-first + event messaging (Frigate-style)

- MQTT / Webhook event publishing
- Config-driven rules (YAML/JSON)
- Clip recording around event (pre/post seconds)

Performance / scaling best practices (optional)

If you explore acceleration stacks like **OpenVINO** or **DeepStream**, document what you tried:

- Latency vs throughput settings (OpenVINO performance hints)
- Handling multiple RTSP streams / runtime stream management (DeepStream runtime stream patterns)

---

4) Architecture Guidance (Recommended)

You can implement any architecture, but it should be understandable and extensible.

Recommended MVP architecture

- ingestion-service: pulls RTSP, decodes frames
- inference-service: runs model and emits detections
- rules-engine: zones + rules → events
- api-service: stores and serves events/cameras
- ui: dashboard

For a 2-day build, this can be a single monorepo service, but **keep modules separated**.

---

5) Deliverables (What to Submit by the Deadline)

Required

1. **Git repo** with code
2. **README** including:
    - How to run
    - How to add a camera stream
    - How to test (even minimal)
    - Known limitations + next steps
3. **Demo script** (1 page max):
    - What to click
    - What to show in 3–5 minutes
4. **Research & Best Practices Notes (mandatory)**
    - 1–2 pages describing:
        - 3 global platforms you studied (e.g., Milestone/BriefCam/Avigilon/Frigate)
        - Features you adopted
        - Features you would add next + why

Optional (highly valued)

- Docker Compose
- Short screen recording (2–4 minutes)
- Postman collection / OpenAPI spec

---

6) Evaluation Criteria (How Your Work Will Be Scored)

1) Working End-to-End System — 35%

- Can we add a camera and see it “working”?
- Are events generated reliably?
- Does the UI reflect reality (camera status, event feed)?

2) Engineering Quality — 20%

- Clean architecture, modularity, readability
- Error handling, retries, logging
- Sensible defaults + configuration approach

3) Product Thinking & Feature Depth — 15%

- Features beyond MVP that clearly improve usability
- Operator-friendly UX decisions (filters, views, speed of reviewing incidents)
- Thoughtful event schema and search

4) Performance & Stability — 10%

- Handles multiple streams reasonably
- Doesn’t crash on a broken stream
- Basic metrics (FPS per camera, queue sizes, inference time) = strong bonus

5) Security, Privacy, Compliance Mindset — 10%

- Data minimization approach (store events/snapshots vs full video)
- Access control (even simple), audit trail (bonus)
- Clear notes on GDPR-safe patterns and what is not implemented

6) Research & Best Practices Adoption — 10%

- Clear evidence you studied real systems and extracted patterns

- Explicit mapping: “We adopted X because…”
- Mature roadmap thinking

Bonus Points (up to +15%)

- Dockerized deployment
- MQTT/webhooks integration
- Exportable event clips
- Heatmaps / trend charts
- Multi-tenant or role-based access control
- OpenAPI docs + decent test coverage

---

7) What to Highlight in the Final Discussion (Mandatory Talking Points)

1. What you built end-to-end and what’s still stubbed
2. Your architecture choices (and tradeoffs under 2-day constraints)
3. How you researched existing platforms and what you adopted
4. What you’d do next with 2 more weeks (scaling, multi-model, storage, search, permissions)