AI-Powered Smart Gate & Security Command Center

Background

Modern gated communities, campuses, and commercial properties rely on **multiple disconnected systems** for:

- Visitor approvals
- Gate-level verification
- Security monitoring
- Incident reporting

This fragmentation leads to delayed threat detection, poor visibility, and manual dependency on guards and facility staff.

Your challenge is to build a **unified AI-powered security MVP** that demonstrates how **visitor management, facial recognition, and incident intelligence** can work together in a single platform.

Objective

Build a **working, end-to-end MVP** that enables:

- Digital visitor pre-approval
- **Face-based identity verification at entry points**
- Watchlist-based threat alerts
- Incident logging and tracking
- A **centralized security command dashboard**

The solution should feel realistic, usable, and extensible beyond the hackathon.

Core Functional Requirements

1. Visitor Pre-Approval

- A resident or authorized user should be able to digitally pre-approve a visitor.
- The visitor's identity must be captured in advance (photo/selfie or equivalent).
- Approved visitors should be time-bound and trackable.

2. Gate Entry & Identity Verification

- Security personnel should be able to verify visitors at the gate.
- Verification must include **facial recognition** (using any AI service or library).
- The system should return a clear outcome:
  - Allowed
  - Denied
  - Needs manual verification
- All entry attempts must be logged.

3. Watchlist / Alert Detection

- The system should maintain a watchlist of flagged individuals.
- During gate verification:
  - If a visitor matches the watchlist beyond a defined confidence level,
  - The system must immediately raise a **high-severity alert**.
- Alerts should be visible to admins in real time.

4. Incident & Case Management

- Security or residents should be able to log incidents.
- Incidents should support:
  - Categories

    - Severity levels
    - Evidence upload (optional)
- Each incident must move through a clear lifecycle (open → resolved).

5. Unified Security Command Dashboard

A single dashboard must provide visibility into:

- Visitor activity
- Gate entry logs
- Watchlist alerts
- Active and resolved incidents

This dashboard is the **primary demo screen**.

Constraints & Guidelines

- Facial recognition **is mandatory**
- Teams are free to use:
    - AWS, Azure, Gemini, open-source models, or any AI APIs
- Payments are **out of scope**
- Live CCTV streaming is **out of scope**
- IoT / boom barrier integrations are **out of scope**
- Mobile apps are **optional** (web-first is sufficient)

Hackathon Expectations

- Build time: **8–10 hours**
- Team size: **3–4 members**

- Focus on:
    - Functional completeness
    - Clean flow
    - Real-world applicability
- Mock data is acceptable where needed, but **flows must work end-to-end**

Evaluation Criteria

Teams will be judged on:

1. **End-to-end functionality**
2. Effective use of **facial recognition**
3. Quality of alerting and incident handling
4. Dashboard clarity and usability
5. Thoughtfulness toward real-world scalability

Continuation Beyond Hackathon

This problem is designed so that:

- The same solution can be **extended post-hackathon**
- Advanced AI, analytics, surveillance intelligence, and authority integrations can be layered later

Think of this as building the **foundation of a real security platform**, not a throwaway prototype.

Final Note

You are not expected to build everything perfectly.
You are expected to **think like system builders**, prioritize intelligently, and demonstrate how AI can meaningfully improve security operations.