CityGuardian: Forecast → Propose → Simulate → Approve (Challenge 1)

Summary: We will predict cell-level surges around major events, have an agent propose a safe mitigation, test it in a digital twin, (only after human approval) apply it, and show the before/after on a simple 3D city view.

Problem & impact: During games, storms, or incidents, individual cells can approach saturation, which will hurt call setup success and latency for first responders. Proactive optimization, respecting public-safety priority/preemption and Band-14 capacity, will keep service stable exactly where it's needed most.

Solution: Say there is a "Game Night" surge → forecast says Cell S7 ≈ 92% in 10 min → agent proposes move 15% of S7's load to neighbors S5/S6 → digital twin shows hotspot variance ↓, p95 latency ↓, CSSR stable → operator clicks Approve → 3D map cools from red to yellow/green with a before/after card.

Technical approach

1) Data / Forecast (predict +10 min load per cell).
We will simulate/ingest per-cell KPIs every 10s: utilization (% of radio capacity), p95_latency_ms (95th-percentile delay), call_success_rate (CSSR), handover_fail_rate, neighbors. Features: rolling means/trends, time features (hour/weekend/event), and neighbor pressure.

A small gradient-boosted tree (e.g., SageMaker XGBoost) will learn to predict traffic_load +10 min (time-shifted target). **Evaluation**: mean error on held-out windows; alarm when forecasted load crosses a threshold.

2) Agent & guardrails (human-in-the-loop).
The agent will choose one whitelisted, reversible action based on forecast + neighbor headroom, using a simple scoring rule (e.g.: "expected hotspot relief / risk"):
1. **LB_SHIFT**: move 10–20% edge users from hot cells to specific neighbors.
2. **TILT_NUDGE**: tiny (simulated) tilt/power adjustment to rebalance edges.
3. **PRIORITY_OK**: ensure responder traffic retains top priority (sim flag).
4. **DEPLOYABLE_RECOMMEND**: suggest a temporary site when a cluster stays hot.

The agent will output an action object (type, parameters, bounds, rollback). Also, nothing will be applied without operator approval.

3) Digital twin (approve/reject gate).
A small simulator recomputes KPIs as if the action were applied: adjust loads (e.g., −15% on S7, +7.5% on S5/S6), update utilization, map utilization→latency with a smooth curve, and update CSSR/handover based on congestion.
Approve only if ALL are true: (1) Hot-cell utilization drops ≥10% (2) Cluster p95 latency decreases (3) CSSR not worse (ideally ↑) (4) No neighbor ends >85% utilization

Else: Reject (or try a smaller action).
Evaluation: blocked/failed setups ↓ (CSSR ↑), hotspot variance ↓, p95 latency ↓, time-to-mitigation, and zero unsafe actions applied.

4) UI (3D viewer).
A low-poly Three.js grid: color = utilization. A panel shows *Forecast → Agent proposal → Twin result → Approve*. A tiny timeline replays before/after during the surge. Designed to remain useful when deployables are in the field (text/metrics first, minimal bandwidth).