Distributed Weather System - Team 7

Generated: 2025-11-28 23:05:19 UTC

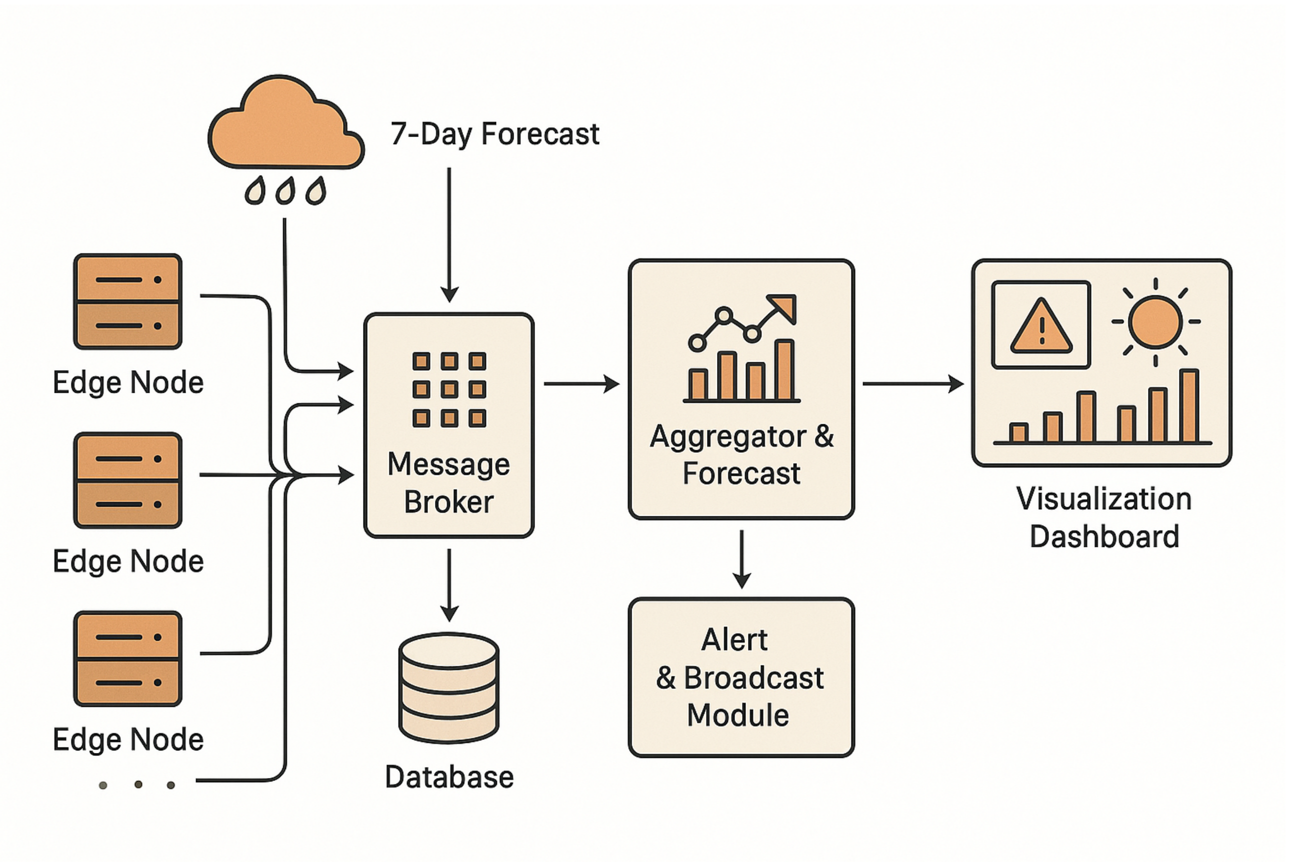
1) Team Information

Member	ID	Role	Feature
Dhruv Kalra	5143420	Backend Developer	Data Stream Handler
Janaranjanee Sendhil Kumar	-	Backend Developer	Alert & Broadcast Module
Dipal Nirmal	-	Frontend Developer	Visualization Dashboard
Minhajuddin Muhammad	5144183	Backend Developer	Aggregator & Forecast Engine
Yuvraj Singh Palh	5143317	Scrum Master	Edge Node Data Service

2) Introduction

Distributed weather intelligence platform that ingests edge observations via MQTT, stores them in TimescaleDB, computes rolling aggregates and forecasts, triggers alerts, and serves a rich visualization dashboard. Designed as a decoupled, scalable distributed system.

Architecture Overview



3) System Details

- Edge Nodes & Simulators: publish JSON to MQTT topics city/{city}/observations.
- Message Broker (Mosquitto): TCP + WebSocket listeners; routes observations and alerts.
- Database (Timescale/Postgres): hypertables observations, aggregates, alerts with indexes.

- Aggregator & Forecast (FastAPI): MQTT ingestion to DB; rolling aggregates (15m/1h); rules engine emits alerts; OpenWeather integration with caching; APIs: /health, /ready, /metrics, /cities/{id}/observations, /aggregates, /alerts, /forecast.
- Alerts: stored in DB and published to MQTT alerts/{city}.
- Visualization: frontend consumes API for observations, aggregates, forecast; alert panel removed per request.
- Backend Dashboard: architecture-style view showing Edge -> Broker -> DB -> Aggregator -> Visualization with live counts.

4) Pipeline (End-to-End)

- Ingest: Edge -> MQTT -> Aggregator writes to Timescale observations.
- Compute: Aggregation job writes windowed aggregates (15m/1h).
- Forecast: OpenWeather fetcher + API with caching.
- Alerts: Rules on aggregates -> alerts table + MQTT alerts/{city}.
- Serve: FastAPI REST; Frontend dashboards render current/aggregates/forecast.

5) How to Run

- Prereqs: Docker Desktop, Python 3.11 (for scripts).
- Start: docker compose -f infra/docker-compose.yml up --build
- Frontend: cd frontend && python -m http.server 3000 -> http://localhost:3000
- Backend dashboard: http://localhost:3000/backend-dashboard.html
- API key header: X-API-Key: devkey (change for prod).
- Seed data: python edge-sim/publisher.py --host localhost --port 1883 --city toronto --source edge-sim --interval 5
- Manual publish: docker compose -f infra/docker-compose.yml exec mosquitto mosquitto_pub -t city/toronto/observations -m "{...payload...}"
- Verify: curl -H "X-API-Key: devkey" http://localhost:8080/health; check /observations, /aggregates, /alerts, /forecast?lat=43.6532&lon=-79.3832

6) Data Contracts

- Observation: {city_id, source, observed_at, temp_c, humidity, wind_kph, pressure_hpa, rain_mm}
- Aggregate: city_id, bucket_start, bucket_width, temp_avg/min/max, humidity_avg, wind_avg
- Alert: {city_id, level, rule, message, triggered_at} (DB + MQTT alerts/{city})

7) Highlights

- MQTT decoupled ingestion; edge-friendly.
- Timescale optimized time-series storage.
- Rule-based alerts with MQTT broadcast.
- OpenWeather integration with caching.
- Health/readiness/metrics endpoints for ops.
- Demo-friendly: simulator, dashboards, start script.