AI/ML work for SkyAware

I. AI/ML Deep Dive: Omar's Mandate

A. Pipeline 1: TEMPO Data Processing (The Scientific Engineering Task)

This pipeline converts raw satellite data into a visualization-ready format, demonstrating the scalable cloud computing requirement.

Step	Detail	Output / Deliverable	Tools / GCP Service
1. Data Ingestion & Subsetting	**Omar** uses the NASA EDL/Harmony-py credentials to programmatically request the latest hourly TEMPO L2/L3 NRT product (e.g.,NO2). The request should immediately subset the data to North America to minimize file size.	The latest hour's TEMPO NetCDF4/HDF file.	Python (harmony-py, netCDF4, xarray)
2. AQI Conversion (Scientific Logic)	**Omar** loads the TEMPO data grid. Implements the **EPA AQI Conversion Formula** (provided by Ebrima) to transform the raw pollutant concentration (mol/cm2) into the final 0-500 AQI number grid.	A processed Xarray DataArray with AQI values.	Python (xarray, Pandas)
3. Format & Size Optimization	The AQI grid is too large for the browser. **Omar** must convert it to a lightweight, map-ready format (e.g., a simplified GeoJSON for vector display or a compressed GeoTIFF for raster display). This must be highly optimized.	A lightweight **GeoJSON** or **GeoTIFF file**.	Python (geopandas, rasterio or similar)
4. Cloud Storage & Scheduling	The GeoJSON/GeoTIFF is uploaded to the **GCP Cloud Storage Bucket** (managed by Sawaneh). The entire Python script is deployed as a **GCP Cloud Function** triggered hourly by **Cloud Scheduler**.	Automated hourly data refresh on GCP Storage.	GCP Cloud Function, Cloud Storage, Cloud Scheduler

---

B. Pipeline 2: AQI Forecasting (The Machine Learning Task)

This pipeline leverages historical and real-time data to predict future air quality, fulfilling the "Forecasting" requirement.

1. Model Choice & Features

- **Model: XGBoost Regressor** (for its speed, robustness, and ability to handle non-linear relationships between inputs).
- **Target Variable:** Max AQI for the next 24 hours (a single integer).
- **Input Features (Hassan's contribution is critical here):**
    - Time Series: **Historical AQI** (24hr, 48hr moving averages).
    - Weather: **Temperature, Wind Speed, Humidity** (current or forecasted).
    - *Novel Feature:* **Current TEMPO-derived AQI/Pollutant Level** (This is the critical new feature that TEMPO enables).
    - *Advanced (if time):* Planetary Boundary Layer (PBL) Height (from MERRA-2/OpenWeatherMap).
-

2. Training and Deployment

Step	Detail	Deliverable	Tools / GCP Service
1. Data Preparation	**Omar** takes the historical data (CSV/JSON) from **Hassan**, cleans it, engineers time-series features (lags, rolling means), and splits it into training/testing sets.	A clean Pandas DataFrame ready for XGBoost.fit().	Python (Pandas)
2. Model Training & Validation	**Omar** trains the XGBoost model. Validation involves calculating a quick performance metric (e.g.,R2 or RMSE) to demonstrate the model's credibility.	A trained, serialized model file (e.g., .pkl or ONNX).	Python (XGBoost, Scikit-learn)
3. Model Serving	The trained model is served as an inference API. This is a lightweight **Flask/FastAPI** microservice deployed to a separate **GCP Cloud Run** instance.	A dedicated /predict API endpoint that accepts JSON features and returns a predicted AQI integer.	GCP Cloud Run, Python (FastAPI/Flask)

4. API Integration	**Sawaneh's** Node.js API connects to this /predict endpoint when the /api/forecast route is hit.	Seamless communication between the Node.js API and the Python ML service.	Sawaneh (Node.js) ↔\leftrightarrow↔ Omar (FastAPI/GCP)

II. Omar's Critical Dependencies

Omar's work is the most complex and relies heavily on parallel work streams.

Dependency (Input Required)	Source Team Member	Format / Detail Needed
EPA AQI Formula & Breakpoints	**Ebrima** (Team Lead)	The exact scientific formulas for NO2 / O3 to AQI conversion.
GCP Credentials & Bucket Name	**Sawaneh** (Sr. Backend)	The Service Account Key and the name of the **GCP Cloud Storage** bucket for storing processed TEMPO data.
Historical Training Data	**Hassan** (Full Stack)	A clean, time-indexed CSV or JSON file containing historical AQI and Weather data for the target locations.
TEMPO Product ID	**Ebrima/Self** (Resource Search)	The exact product ID for the Near Real-Time TEMPO L2/L3 data to be used with harmony-py.
API Call to ML Model	**Sawaneh** (Sr. Backend)	The exact JSON structure (input features) that the deployed ML model will expect for inference.