# SkyAware API Design

The Backend, driven primarily by **Sawaneh (Senior Backend)** and **Hassan (Full Stack)**, is the central nervous system of **SkyAware**. It is responsible for orchestrating all data flows, managing the serverless infrastructure on GCP, and ensuring the Frontend (Saul) has a fast, stable API to consume.

## I. Backend Deep Dive: Sawaneh & Hassan Mandate

The Backend operates on **Node.js (for the API)** and interfaces with Python/GCP services for data processing.

### A. Core Architecture & GCP Management (Sawaneh's Domain)

| Area | Detail / Task | Deliverable | GCP Service / Tool |
|---|---|---|---|
| **1. Cloud Orchestration** | Set up the core GCP project, deploy the Node.js API to a scalable service, and manage the communication between all services. | Fully deployed, auto-scaling Node.js API. Configured Service Accounts. | **GCP Cloud Run (Node.js API)**, **IAM** |
| **2. API Contract Enforcement** | Define, document, and enforce the JSON data structure for all endpoints. Integrate all other services/functions into these endpoints. | Stable, predictable JSON response from all routes. | Node.js (Express/Hapi) |
| **3. TEMPO Data Integration** | Implement the logic to connect the /api/tempo_grid endpoint to the **GCP Cloud Storage** bucket where **Omar's** processed data sits. | /api/tempo_grid returns the correct, secure URL link to the GeoJSON/Raster file. | Node.js, GCP Storage Client |
| **4. Alert Logic** | Implement the server-side logic to determine when a user's *forecast* exceeds the health risk threshold (e.g.,AQI>150 ) and simulate a notification flag. | A functional check routine that runs periodically or on-demand. | Node.js/Cloud Run |

## B. Data Ingestion & API Endpoints (Hassan's Domain)

| Area | Detail / Task | Deliverable | Tools / Dependencies |
|---|---|---|---|
| **1. Ground Data Ingestion** | Write clean, error-handled Node.js functions to fetch real-time ground AQI (EPA AirNow) and weather data (OpenWeatherMap). | Two modular, reusable Node.js functions: fetchEpaAqi(lat, lon) and fetchWeather(lat, lon). | Node.js (axios, API Keys), **Sawaneh** (Integration) |
| **2. Historical Data Collection (ML Prep)** | Write a one-off script to gather 30-60 days of historical AQI and weather data for target cities (needed for Omar's ML training). | Clean, formatted CSV/JSON file of historical features. | Node.js/Python, **Omar** (Input Format) |
| **3. Current AQI Endpoint** | Integrate the real-time ground/weather functions into the /api/current_aqi route, adding basic caching and error handling. | /api/current_aqi is live, fast, and stable. | Node.js API |
| **4. Validation Logic** | Implement the comparison logic within the /api/current_aqi route: finding the nearest ground station and comparing its reading to the TEMPO-derived reading. | current_aqi_epa vs. current_aqi_tempo_derived fields populated. | Node.js |

## II. Backend Endpoint Specifications (The Contract)

The following APIs are the *only* interfaces the Frontend (Saul) will need to consume.

| Endpoint | Purpose | Responsible | Data Flow Logic |
|---|---|---|---|
| **/api/current_aqi?lat=...&lon=...** | Provide all real-time data for the map view. | **Hassan / Sawaneh** | 1. **Hassan** fetches EPA/Weather. 2. **Sawaneh** applies Health Advice Logic. 3. **Sawaneh** returns JSON. |
| **/api/forecast?lat=...&lon=...** | Provide the predicted AQI timeline. | **Sawaneh / Omar** | 1. **Sawaneh** gathers current features (Temp, Wind, Current AQI, TEMPO AQI). 2. **Sawaneh** sends features to **Omar's** ML model API. 3. **Sawaneh** receives prediction, formats it with health advice, and returns JSON. |
| **/api/tempo_grid** | Provide the map visualization data. | **Sawaneh / Omar** | **Sawaneh** queries **GCP Cloud Storage** for the latest GeoJSON/Raster file link (created by **Omar**) and returns that link to the client. |

## III. Backend Critical Dependencies

| Dependency (Input Required) | Source Team Member | Format / Detail Needed |
|---|---|---|
| **TEMPO Processing Pipeline** | **Omar** (AI/ML) | Confirmation that the **Python Cloud Function** is deployed and successfully writing the GeoJSON/Raster file to the GCP Storage bucket hourly. |
| **ML Inference Endpoint URL** | **Omar** (AI/ML) | The live, secure URL for the deployed ML model (e.g., a **GCP Cloud Run** endpoint) that can accept features and return a prediction. |
| **EPA/Weather Functions** | **Hassan** (Full Stack) | The final, tested Node.js functions for fetching all necessary external data. |
| **Health Advice Logic** | **Ebrima** (Team Lead) | The mapping of AQI ranges to specific health advice strings (e.g.,101−150→to "Limit outdoor exertion"). |