Developer Guides

Below are the custom guides for each of your six team members.

👤 Role: Team Lead / Support & Content

Member: Ebrima S. Jallow

🎯 Your Primary Goals

1. Keep the team on track with the project roadmap and eliminate blockers.
2. Craft the final "pitch" (story, presentation, demo flow).
3. Ensure all scientific and health content is accurate and cited correctly.

📋 Your Action Guide

Setup Phase (Project Start):

GitHub: Create the repository. Give everyone write access. Create a README.md with the project title and team list.
Project Board: Set up a simple Trello/GitHub Project board with columns: To Do, In Progress, Blocked, Done. Create cards for all "Phase 1" tasks.
GCP: Confirm the GCP project is created, billing is enabled, and **Sawaneh** has the necessary admin permissions.
Design/UX: With **Hawa**, sketch the wireframes for the three main screens (Home/Map, Forecast, Educational) and share them with **Saul**.

Execution Phase (Content & Support):

AQI Formulas: Locate the **US EPA AQI Breakpoint Table** and the conversion equation (specifically for NO2 and/or O3). Document it clearly in a shared file and hand it to **Omar** (AI/ML) and **Sawaneh** (Backend) for implementation.
Educational Content: With **Hawa**, write the final text for the **Educational Mode**. Keep it simple:

- What is TEMPO?
- What is NO_2/Ozone?
- Why is it dangerous? (e.g., Respiratory issues, asthma triggers).

Health Advice Logic: Define the standard health advice for each AQI bucket (e.g., AQI 101-150: "Sensitive groups should reduce prolonged outdoor exertion"). Hand this logic to **Sawaneh** for the API.

Citation Check: Ensure **Saul** and **Hawa** have placed the required "Data provided by NASA TEMPO and EPA AirNow" citations on the appropriate screens.

Final Phase (Pitch):

Presentation: Build the slide deck. Focus on the story: **Problem (Dirty Air** →to→ **NASA TEMPO (The Solution/Data)** →to→ **SkyAware (The Actionable Tool)**.

Demo Flow: Choreograph the live demo with **Saul** to ensure it highlights the best features and avoids any known bugs.

🤝 Who You Need

- **Sawaneh:** For all GCP account and deployment status updates.
- **Hawa:** To co-create the educational content and wireframes.
- **Omar:** To confirm which specific TEMPO pollutant is being used (for content accuracy).

🔗 Resources You Need

- **EPA AQI Technical Document:** [Search for "EPA AQI Technical Assistance Document"]
- **NASA TEMPO Mission Website:** For facts and branding inspiration.
- **GCP Console:** For project administration.

👤 Role: Senior Frontend Developer

Member: Saul Zayn

🎯 Your Primary Goals

1. Build a fast, mobile-responsive Next.js application shell.
2. Implement a high-performance interactive map (Mapbox GL JS) that can render a custom TEMPO data layer.
3. Create a clean, informative UI for the forecast and educational content.

📋 Your Action Guide

Setup Phase:

Next.js Init: Initialize the Next.js project in **Ebrima's** repository. Set up the basic page structure: / (Home/Map), /forecast, /learn.
Mapbox Setup: Install Mapbox GL JS (or react-map-gl). Create a MapComponent.js. Use your API key to render a base map.
Styling: Implement the basic styling (colors/fonts) agreed upon with **Ebrima/Hawa**.

Execution Phase (Map & UI):

- **Ground Stations:** Consume the /api/current_aqi endpoint (from **Hassan/Sawaneh**) to place markers on the map for the 5-10 supported cities.
  TEMPO Layer (CRITICAL): Consume the /api/tempo_grid endpoint (from **Sawaneh**).
  - Retrieve the data_url (GeoJSON/Raster form GCP Storage).
  - Add it as a source and layer in Mapbox GL JS.
  - Use the min_val/max_val to apply a color scale (e.g., transparent green to opaque red).

Forecast UI: Build a component to display the 3-day forecast data from /api/forecast (from **Sawaneh**) in a clean, readable format (e.g., a simple card or chart).
Responsive Check: Test constantly on mobile viewports. The map must be usable on a phone.

Final Phase:

Optimization: Ensure the TEMPO layer loads and renders quickly. (Work with **Omar** if the file size is too large).
Demo Prep: Rehearse the demo flow with **Ebrima**.

🤝 **Who You Need**

- **Ebrima:** For wireframes and styling guidelines.
- **Sawaneh:** For the finalized API endpoint URLs and data structure (the "API Contract").
- **Omar:** If the TEMPO map layer (GeoJSON/Raster) is too large or not rendering correctly.

🔗 **Resources You Need**

- **Mapbox GL JS Documentation:** Especially the sections on **Sources (GeoJSON/Image)** and **Layers (Fill/Raster)**.
- **Mapbox API Key:** (You should have this).
- **Next.js Documentation.**

👤 Role: Senior Backend Developer

Member: Sawaneh

🎯 **Your Primary Goals**

1. Architect and manage the GCP environment (Cloud Run, Cloud Functions, Cloud Storage).
2. Build and deploy the Node.js (Express/Hapi) API that serves all data to the frontend.
3. Enforce the "API Contract" and ensure data flows correctly between components.

📋 Your Action Guide

Setup Phase (GCP & Node.js):

GCP Service Account: Create a dedicated Service Account in **Ebrima's** GCP project with permissions for Cloud Run Invoker, Cloud Functions Invoker, and Cloud Storage Admin. Share the credentials securely with **Omar**.
Cloud Storage: Create a public-read Bucket (e.g., skyaware-tempo-data) for **Omar** to store processed TEMPO map files.
Node.js API: Initialize the Node.js server project. Create placeholder routes for /api/current_aqi, /api/forecast, and /api/tempo_grid.

Execution Phase (Integration & Logic):

Integrate Ground Data: Integrate **Hassan's** data fetching functions into the /api/current_aqi endpoint. Implement caching (e.g., node-cache) to prevent hitting EPA/Weather APIs too often.
Integrate TEMPO: Configure the /api/tempo_grid endpoint to return the URL of the latest file in the **GCP Cloud Storage bucket** (generated by **Omar**).
Integrate Forecast: Connect the /api/forecast endpoint to **Omar's** deployed ML model/service. Pass the required location data and format the response according to the "API Contract".
Health Advice Logic: Implement the logic provided by **Ebrima** to map AQI numbers to health advice strings in the API response.

Final Phase (Deployment):

Deploy API: Deploy the Node.js API to **GCP Cloud Run**.
Alert Logic: Implement a simple background check (or a dedicated endpoint) to determine if a simulated user's location has a high forecasted AQI and generate an alert object.

🤝 Who You Need

Ebrima: For GCP access and the AQI →to→ Health Advice mapping logic.

- **Hassan:** For the working functions to fetch EPA/Weather data.
- **Omar:** For the GCP Cloud Storage bucket name (to serve TEMPO data) and the connection details for his ML model.

🔗 Resources You Need

- **GCP Console (Cloud Run, IAM, Storage).**
- **Node.js / Express Documentation.**

👤 Role: Full Stack Developer

Member: Hassan

🎯 Your Primary Goals

1. Write robust Node.js functions to fetch and clean data from external APIs (EPA AirNow, OpenWeatherMap).
2. Fetch and prepare the *historical* data required for the ML model.
3. Act as the bridge, helping connect the backend data to the frontend components.

📋 Your Action Guide

Execution Phase (Data Ingestion):

EPA AirNow (Real-time): Write a Node.js function using your API key to fetch current AQI for a list of target cities (lat/lon). Handle API errors gracefully. Hand this function to **Sawaneh**.
OpenWeatherMap (Real-time): Write a Node.js function using your API key to fetch current temp, wind, and humidity for the same cities. Hand this function to **Sawaneh**.
Historical Data (for ML): Write a script (Node.js or Python) to fetch **30-60 days of historical AQI and Weather data** for the target cities. Clean this data into a CSV/JSON format and give it to **Omar** for model training.

Execution Phase (Frontend Bridge):

Data Fetching (Next.js): Work with **Saul** to implement the data fetching logic in the Next.js components (e.g., using useEffect or getServerSideProps) to consume **Sawaneh's** API endpoints.
Validation Feature: Implement the logic/UI snippet that shows the comparison: "TEMPO Calculated AQI: X" vs. "EPA Ground AQI: Y".

Final Phase:

Optimization: Help **Sawaneh** ensure that your data fetching functions are efficient and correctly cached.

🤝 Who You Need

- **Sawaneh:** To know where to integrate your data fetching functions in the API.
- **Omar:** To know exactly what format/fields he needs for the historical training data.
- **Saul:** To help connect the API data to the frontend UI components.

🔗 Resources You Need

- **EPA AirNow API Documentation.**
- **OpenWeatherMap API Documentation.**
- Your API Keys for both.

👤 Role: AI/ML Engineer

Member: Omar

🎯 Your Primary Goals

1. Build the **Python Cloud Function** on GCP to access, process, and convert TEMPO data into a map-ready format.
2. Train a simple, reliable AQI forecasting model and deploy it for the API to use.

📋 Your Action Guide

Execution Phase (TEMPO Processing - Python/GCP):

TEMPO Access: Use harmony-py (or direct Earthdata access) in a Python script to fetch the latest **TEMPO NRT L2/L3 (`NO2` or O3)** file (NetCDF4/HDF).
Data Conversion (CRITICAL):

    - Use xarray/netCDF4 to open the file.
    - Extract the pollutant column data and the lat/lon grid.
    - Implement the **AQI Conversion Formula** (provided by **Ebrima**) to convert the raw values to AQI.
    - **Transform:** Convert this gridded AQI data into a **highly simplified GeoJSON** (polygons/points) or a **GeoTIFF/Raster**. Prioritize small file size!
- **GCP Integration:**
    - Set this script up as a **GCP Cloud Function (Python)**.
    - Configure the function to save the output (GeoJSON/Raster) to the **Cloud Storage Bucket** created by **Sawaneh**.
    - Set up **Cloud Scheduler** to trigger this function every hour.

Execution Phase (Forecasting Model):

Training: Use the historical data provided by **Hassan** (AQI + Weather) to train a simple **XGBoost** (or Linear Regression) model to predict the *next day's max AQI*. Don't overcomplicate the model. Focus on a working pipeline.
Serving: Deploy this model (e.g., as a separate Cloud Function or a simple Flask microservice on Cloud Run) so **Sawaneh's** Node.js API can send it current conditions and get a prediction back.

🤝 Who You Need

- **Ebrima:** For the AQI conversion formulas and to confirm the TEMPO product.
- **Sawaneh:** For GCP Service Account credentials and the Cloud Storage Bucket name.
- **Hassan:** For the historical training data (CSV/JSON).

🔗 Resources You Need

- **NASA Earthdata Login Credentials.**
- **NASA Harmony API Documentation / TEMPO Product Documentation.**
- **GCP Documentation:** Cloud Functions (Python) & Cloud Storage Client Libraries.
- **Python Libraries:** xarray, netCDF4, pandas, scikit-learn/xgboost, google-cloud-storage.

👤 Role: Support / Content

Member: Hawa Cham

🎯 Your Primary Goals

1. Ensure the application is bug-free and user-friendly (QA Testing).
2. Create and implement clear, accessible content for the **Educational Mode**.
3. Manage project documentation and ensure all challenge requirements (citations, etc.) are met.

📋 Your Action Guide

Execution Phase (Content & UI):

Educational Content: Collaborate with **Ebrima** to finalize the text and find/create simple visuals for the Educational Mode (Pollutant definitions, health impacts, "What is TEMPO?").
Implement Educational UI: Work in the Next.js project (with **Saul's** guidance) to build the /learn page and insert your finalized content and visuals. Ensure it looks good on mobile.

Execution Phase (QA & Documentation):

- **QA Testing (Ongoing):** Constantly test the app.
    - Do the map markers load?
    - Does the TEMPO layer show up?
    - Does the forecast make sense?
    - Are there broken links or typos?
    - Report all bugs to the relevant developer (**Saul**, **Sawaneh**, or **Hassan**) immediately.

Data Citation: Ensure the footer or a dedicated "About" section clearly cites: "Data provided by NASA TEMPO Mission and EPA AirNow."
README: Maintain the project's README.md. It must include: Project Title, Team Members/Roles, Tech Stack (GCP/Next.js/Node.js/Python), Data Sources used, and Instructions on how to run/deploy the project.

Final Phase:

Final Smoke Test: Do a final, complete run-through of the user journey before the final submission and demo.

🤝 **Who You Need**

- **Ebrima:** To finalize the content and citation requirements.
- **Saul:** For guidance on implementing your content into the Next.js UI.
- **All Developers:** To report bugs to and get the details for the README.

🔗 **Resources You Need**

- **Access to the GitHub Repository** (to edit README and Educational page).
- **NASA/EPA Websites** (for sourcing content and correct citation formats).