Cameroon Weather App - Team Roles and Responsibilities

A. Full Summary of the Entire Project

The group must build a **weather web application** that uses a public **weather API** to show real-time weather information for cities (mainly in Cameroon).

1. Functional Requirements

- Fetch real-time weather data from an API.
- Search bar for city names.
- Display weather details:
  - Temperature
  - Humidity
  - Wind speed
  - Weather condition (sun, rain, clouds)
- Dynamic icons based on weather type.
- Background that changes depending on weather conditions.
- Error screen if the user enters an invalid city or if the API fails.
- Loading indicator during API calls.

2. Technical Requirements

- Single-page application (SPA).
- HTML for structure, CSS for styling, JS for interactivity.
- `fetch()` and `async/await` for API calls.
- DOM manipulation to update UI dynamically.
- Error handling for failed requests, invalid API keys, or network issues.

3. What is NOT Included

- No backend.
- No database.
- No deployment responsibilities for team members (leader handles that).
- No extra frameworks unless allowed (depends on teacher, usually pure HTML/CSS/JS).

B. Split Into 5 Clear Work Sections

1. Member 1 — UI Layout & Base HTML Structure

Tasks:

- Create HTML skeleton:
  - Search bar
  - Weather info container
  - Icons container
  - Error message container
  - Loading indicator area
- Ensure IDs/classes are correctly named for JS access.
- Provide placeholder elements for:
  - temperature
  - humidity
  - wind speed
  - weather icon
  - city name

Deliverables:

- `index.html` complete layout.
- Clean and valid structure ready for styling and JS connection.

---

2. Member 2 — Styling & Responsive CSS

Tasks:

- Create the CSS file(s).
- Style:
  - Dashboard layout
  - Weather widgets/cards
  - Search bar styling
  - Background visuals
  - Error and loading states
- Define background classes:
  - `sunny`
  - `rainy`
  - `cloudy`
  - `night`

- Ensure responsive design for phone/tablet/laptop.

Deliverables:

- `styles.css` visually polished and responsive.
- Background classes ready for JS toggling.

---

3. Member 3 — API Integration (Fetch Logic)

Tasks:

- Set up the API call using `fetch()`.
- Handle valid responses:
  - parse JSON
  - extract temperature, humidity, wind speed, condition
- Detect and handle errors:
  - Invalid city
  - Bad API key
  - Network failure
- Implement reusable function:

```
async function getWeather(city) { ... }
```

Deliverables:

- `api.js` or part of `script.js` containing:
  - `fetch()` logic
  - JSON parsing
  - error detection

---

4. Member 4 — DOM Manipulation & Dynamic UI Updates

Tasks:

- Receive data from API module and update:
  - temperature text
  - humidity text
  - wind speed text
  - city name
- Set the appropriate weather icon.
- Switch background classes depending on weather.
- Control loading state visibility.

- Control error message visibility.

Deliverables:

- JS functions such as:

```
function updateUI(data) { ... }

function showError(message) { ... }
```

---

5. Member 5 — Search Function, User Interaction, and App Logic

Tasks:

- Handle search bar input.
- Listen to "Enter" key or search button click.
- Call the API module using the user's input.
- Trigger loading state before calling API.
- Trigger UI update after receiving data.
- Trigger error screen if needed.
- Coordinate API + UI modules.

Deliverables:

- Main `script.js` logic tying all components.
- Fully working search functionality.

D. Summary Table

Member	Main Responsibility	Deliverables
1	HTML Layout	`index.html` skeleton with IDs/classes
2	CSS Styling	`styles.css` with responsive layout and backgrounds
3	API Integration	`api.js` with `getWeather()` function and error handling
4	DOM & UI Updates	JS functions: `updateUI()`, `showError()`, `showLoading()`
5	App Logic & Search	`script.js` linking search input → API → UI updates