Group Leader — Complete Responsibilities Manual

> **Purpose:** This document is an exhaustive, no-excuses guide to everything the group leader must do for the Cameroon Weather App project. It consolidates all previously discussed plans, scaffold requirements, weekly milestones, integration rules, communication protocols, QA checklists, and deployment steps. Give this to the leader and they will have everything needed to run the team like a small, humane software factory.

Executive Summary

The Group Leader is the project architect, integration engineer, and collaboration facilitator. Your job is not merely to assign tasks — it is to remove blockers before they exist, define contracts other members can code to, enforce standards, and deliver the final, polished product. This manual lists every concrete responsibility, deliverable, and the exact acceptance criteria the leader must produce and verify.

---

1. Pre-development Setup (Week 0)

1.1 Repository & Project Scaffold (Deliver by Day 0)

- Create the GitHub repository and set default branch to `main`.
- Add `.gitignore` and a minimal README with project name and short goals.
- Create initial folder structure:
  - `/src` or root with `index.html`, `styles.css`, `script.js` (placeholders)
  - `/assets` for icons/images
  - `/docs` for spec, contracts, and meeting notes
  - `/tests` for shared test fixtures/mock data
- Commit scaffold with message `chore: project scaffold`

Acceptance Criteria: Repo exists, README explains purpose, scaffold files are in place, and repository access is set for the team.

1.2 Access & Permissions

- Add team members as collaborators or use an organization with access controls.
- Create a `CONTRIBUTING.md` explaining branch rules and commit message format.
- Ensure repository visibility matches project policy (private/public) and that CI tokens (if any) are secured.

Acceptance Criteria: All members have correct repository access; `CONTRIBUTING.md` committed.

1.3 Naming Conventions & Coding Standards

- Publish a one-page style guide in `/docs` covering:
  - HTML ID/class naming (kebab-case, semantic prefixed ids like `weather-container`)
  - CSS class scheme for dynamic states (`.bg-sunny`, `.bg-rainy`)
  - JS function naming and exported function signatures
  - JSON data schema for API responses
- Decide on simple linting rules if desired (optional).

Acceptance Criteria: `docs/style-guide.md` present with concrete examples.

1.4 Mock Data & API Contract

- Design a mock JSON payload schema and store a canonical file in `/tests/mock-weather.json`.
- Include field names, types, units, and example values (e.g., `temperature: number °C`).
- Define error response structure for invalid city and network failure.

Acceptance Criteria: `mock-weather.json` exists and is referenced in the README.

1.5 Development Environment Instructions

- Document how to run the project locally (open `index.html`, or simple HTTP server command).
- Provide recommended tools (VSCode, Git client) and any required environment variables (if using a real API key).

Acceptance Criteria: `docs/dev-setup.md` with explicit steps.

---

2. Interface Contracts (The Leader-as-API)

The leader must declare the exact interfaces each module exposes and consumes. Ship these as a single source of truth in `/docs/contracts.md`.

2.1 HTML Contract

- List required DOM element IDs and their purpose, for example:
  - `#city-input` (string input)
  - `#search-btn` (button)
  - `#weather-container` (root container)

    - `#city-name`, `#temperature`, `#humidity`, `#wind-speed`, `#weather-icon`, `#error-message`, `#loading`
- For each element include expected content/format (text, numeric with units, CSS class toggles).

Acceptance Criteria: `docs/contracts.md` contains HTML element list and semantics.

2.2 CSS Contract

- Define CSS classes that JS will toggle, such as:
    - `.bg-sunny`, `.bg-rainy`, `.bg-cloudy`, `.bg-night`
    - `.hidden`, `.visible`, `.loading`, `.error`
- Specify animations/transitions expectations (e.g., `transition: background 0.5s ease`).

Acceptance Criteria: CSS contract documented and sample class block included.

2.3 JS / API Contract

- Publish the function signatures and data shapes:
    - `getWeather(city: string) -> Promise<WeatherResponse | ErrorResponse>`
    - `updateUI(weatherData: WeatherResponse) -> void`
- Provide field mapping for the `WeatherResponse` schema (temperature, humidity, wind, condition, iconKey).

Acceptance Criteria: `docs/contracts.md` contains full JS/API contract with mock sample calls.

---

3. Branching Strategy & Git Workflow

3.1 Branch Naming & Use

- `main` - Always deployable, only maintained by leader or after approved PRs.
- Feature branches: `member1-html`, `member2-css`, `member3-api`, `member4-ui`, `member5-logic`.
- Hotfix branches: `hotfix/*` for urgent fixes.

3.2 Pull Request Rules

- Require descriptive PR titles and body describing changes and testing steps.
- Minimum one review from a different team member plus leader approval before merging to `main`.
- Squash merges to keep history tidy.

Acceptance Criteria: `CONTRIBUTING.md` documents the rules and GitHub branch protection (if available) is configured.

---

4. Weekly Integration Plan & Milestones

The leader must publish a weekly plan with daily checkpoints. Below is the canonical 3-week plan plus Week 0 prep.

Week 0: Preparation (Leader-focused)

- Scaffold repo, contracts, mock data, dev instructions.
- Schedule standup times and integration slots.
- Deliverable: Project scaffold committed, `docs/contracts.md` published.

Week 1: Foundation

- Verify Member 1 has pushed HTML skeleton by Day 2.
- Confirm Member 2 can start styling with skeleton.
- Ensure Member 3 publishes mock API and example responses by Day 4.
- Integration checkpoint at Day 7: confirm HTML + CSS placeholder styling.

Week 2: Core Integration

- Coordinate integration of Member 3's real API into Member 4's DOM code.
- Run daily mini-integration tests to discover mismatches early.
- Deliverable: Working search flow using real or mock API and DOM updates.

Week 3: Polish & Deployment

- Final integration, cross-device testing, accessibility checks.
- Prepare deployment (Netlify) and finalize README and demo script.
- Deliverable: Deployed app link, final merged `main`, documentation completed.

Acceptance Criteria for each week: a short integration report saved in `/docs/integration-weekX.md`.

---

5. Daily Standups & Communication Protocol

5.1 Standup Format (10–15 minutes max)

- What I did yesterday (name + 1 sentence)
- What I will do today (name + 1 sentence)
- Blockers (if any) — specify where leader intervention is required

5.2 Communication Tools

- Primary chat: choose 1 (Discord/Slack/WhatsApp) and use threads for topics.
- Version control: Git + GitHub PRs for code review.
- Meetings: short video calls for milestone demos.

5.3 Escalation Path

- If a blocker persists 24 hours, immediately raise during standup and the leader schedules a dedicated sync.

Deliverable: Leader publishes `docs/standup-format.md` and schedule.

---

6. Integration, QA & Testing Responsibilities (Leader-led)

6.1 Integration Testing Checklist

- Pull latest branch versions and merge into a `integration` branch for combined testing.
- Verify HTML IDs/classes match CSS & JS contracts.
- Confirm API payloads match the mock schema.
- Run manual flows: search valid city, search invalid city, offline/no-network simulation, slow API response.

6.2 Code Review Checklist (Leader enforces)

- Does the PR follow naming conventions?
- Are IDs/classes semantically correct and accessible?
- Did the author include unit/integration test cases or manual test steps?
- Are edge cases and error states handled and documented?
- Accessibility checks: ARIA labels, color contrast, keyboard navigation.

6.3 Automated & Manual Tests

- Provide test fixtures in `/tests` and document how to run them.
- If no test runner, publish manual test cases as a checklist and run-through steps.

6.4 Performance & Optimization

- Leader runs basic performance checks: ensure images are optimized, CSS is not excessively large, avoid layout thrashing on updates.

Deliverable: `docs/qa-checklist.md` and integration branch test reports.

---

7. Accessibility & UX Standards (Leader ensures compliance)

7.1 Minimum Accessibility Checklist

- All interactive elements must be keyboard reachable.
- Inputs have `label` or `aria-label`.
- Color contrasts meet WCAG AA minimum.
- Loading and error states are screen-reader friendly.

7.2 UX Expectations

- Loading state is visible and prevents duplicate requests.
- Error messages are helpful and actionable.
- Search input is forgiving (trim whitespace) and case-insensitive.

Deliverable: Accessibility report in `/docs/accessibility.md`.

---

8. Final Merging & Deployment (Leader owns)

8.1 Final Merge Steps

- Create a `release` branch from the latest merged `integration` state.
- Run integration checklist and QA pass.
- Fix critical issues; freeze features.
- Merge `release` into `main` with leader-approved PR.

8.2 Deployment (Netlify recommended)

- Connect repo to Netlify or chosen host.
- Set environment variables for API key securely.
- Configure build/deploy settings (for simple static app, default build is fine).

8.3 Post-Deployment Checks

- Smoke test the deployed site with the same checklist used locally.
- Document the deployment URL in README and `docs/deployment.md`.

Deliverable: Live deployed app URL and deployment notes.

---

9. Documentation & Presentation (Leader-led)

9.1 README Contents

- Project title and short description
- How to run locally

- How to deploy
- Contributors and roles
- Link to deployed site

9.2 Demo Script & Presentation

- Prepare 5–7 minute demo script covering core flow, edge cases, and how to run locally.
- Provide a slide with assignments and contributions for the teacher.

Deliverable: `README.md`, `docs/demo-script.md`, and slide notes.

---

10. Conflict Resolution & Soft Skills

10.1 Handling Missed Deadlines

- If a member misses Week 1 objective, leader assigns buffer tasks (documentation, extra test cases) to keep them productive.

10.2 Dealing with Merge Conflicts

- Leader handles complicated merges; members must rebase onto `main` before opening PRs if conflicts are known.

10.3 Maintaining Morale

- Keep meetings short and focused.
- Call out good work publicly in the chat.
- Keep tone professional and constructive.

---

11. Deliverable Templates (Leader must supply)

Place the following templates in `/docs/templates`: - Pull Request template (title, summary, testing steps) - Issue template for bugs and improvements - Weekly integration report template - Standup note template

Acceptance Criteria: Templates present and used by team.

---

12. Leader Daily Checklist (Practical To-dos)

Before standup: - Pull latest branches and check integration status. - Review open PRs and assign reviewers. - Confirm schedule for any planned merges.

After standup: - Address reported blockers immediately (book a sync, provide resources). - Merge low-risk PRs after review. - Update integration report.

End of day: - Ensure tests/integration builds are green. - Prepare next day's plan and pairings if necessary.

---

13. Acceptance Criteria — What 'Done' Means

The leader must ensure that before final submission: - The app runs locally and on the deployed URL with no console errors. - All core flows pass manual QA (valid city, invalid city, offline behavior). - Documentation is complete and readable. - Codebase follows the agreed conventions and contracts.

If all the above are met, the project is ready for submission.

Appendix A — Example Mock JSON Schema (Canonical)

(Leader provides this file in `/tests/mock-weather.json`)

```
{
  "city": "Yaounde",
  "temperature": 28.5,
  "humidity": 82,
  "windSpeed": 10.5,
  "condition": "Rain",
  "iconKey": "rain"
}
```

Include error examples:

```
{ "error": "city_not_found", "message": "City not found" }
```

Appendix B — Quick Troubleshooting Guide for Common Blockers

1. **API key missing** — leader confirms environment variables and shares secure process for loading them.
2. **IDs mismatch** — leader runs a one-minute DOM check comparing `index.html` against `docs/contracts.md`.
3. **Merge conflicts** — leader performs rebase and resolves conflicts, documents the resolution.

Closing Notes

This manual is deliberately exhaustive. A competent leader uses this as both a checklist and a script. If you perform each step, the team will not stall, integration will be manageable, and the final submission will be solid enough to impress graders while being realistically buildable by five students over three weeks.

Document created by assistant.