Architect Handoff: The "Solo Audit" Sprint

Target Version: 0.126.x-Beta **Date:** 2025-12-18 **Context:** Mid-Sprint. Design complete; Implementation pending.

---

1. Executive Summary

We are refining the Web Dashboard to support **Single Log Uploads** ("Solo Mode"). The previous architecture implicitly assumed a competitive scenario (2+ logs), resulting in broken features (empty comparison tabs) and misleading labels ("Missed Multipliers") when a user uploaded only their own log.

Immediate Goal: Implement logic to detect `is_solo` state and adapt the UI to show **Descriptive Analytics** (what happened) rather than **Comparative Analytics** (who won).

2. Critical Design Decisions (Do Not Revisit)

A. Rejection of "Efficiency Metrics"

We explicitly **rejected** adding "Run vs. S&P Efficiency" scores or "Band Change Efficiency" metrics.

- **Reasoning:** Serious operators often use **SO2R** (Single Op 2 Radio) or **2BSIQ**.
- **The Trap:** In an SO2R environment, S&P QSOs are often made "in the gaps" of a Run on a second radio. Calculating an "S&P Rate" based on wall-clock time would falsely flag this high-skill activity as "inefficient."
- **Mandate:** We do not have enough telemetry (Focus Time, Radio ID) to judge efficiency. Stick to **Descriptive** plots.

B. Adoption of "Correlation Plots"

Instead of judging efficiency, we will visualize it via **Correlation Analysis**.

- **Report:** `plot_correlation_analysis.py` (Scatter Plot).
- **Axes:** X=`Run %`, Y=`Hourly Rate` (Left) and `New Mults` (Right).
- **Granularity: Hourly dots** (up to 48 points for CQ WW).
- **Filtering: None.** We explicitly decided to **keep the "noise" at the origin** (low rate/low run %) because it accurately represents off-times or messy M/M in-band activity. The user is smart enough to interpret the cluster at (0,0).

C. Phasing of Modularity

We acknowledged that the dashboard templates need refactoring to support other contests (WAE QTCs, Field Day). However, we decided to **defer** this

"Widget/Modular" refactor to **Phase 2**.

- **Constraint:** Do not attempt to refactor `dashboard.html` into widgets in this sprint. Focus only on the `if is_solo` conditionals.

3. Implementation Specifications

A. New Report Module

File: `contest_tools/reports/plot_correlation_analysis.py`

- **Type:** Plotly (Interactive + Static PNG).
- **Data Source:** `TimeSeriesAggregator` (Cumulative streams -> `.diff()` -> Hourly Deltas).
- **Logic:**
  - `Run %` = `Run_Delta` / `Total_Delta`.
  - `New Mults` = `Total_Mults_Delta`.
- **Visualization:** Two subplots (side-by-side). Scatter markers.

B. View Logic (`views.py`)

- `qso_dashboard`:
  - Calculate `is_solo = (len(callsigns) == 1)`.
  - If `is_solo`, force `matchups = []` to suppress the pairwise selector logic.
  - Register the new `correlation_file` in the context.
- `multiplier_dashboard`:
  - If `is_solo`, do **not** scan for `missed_multipliers_*.txt` (they won't exist).
  - Instead, scan for `multiplier_summary_*.txt` and map them to the display slot.

C. Template Logic

- `qso_dashboard.html`:
  - Add **"Correlations"** tab.
  - If `is_solo`: Hide "Pairwise Strategy" tab. Make "Correlations" the active default.
- `multiplier_dashboard.html`:
  - If `is_solo`: Change card header from "Missed Multipliers" (Red) to "Multiplier Matrix" (Blue/Info).

4. Next Actions for Architect

1. **Ingest:** Read `AIAgentWorkflow.md` and the Project Bundle.
2. **Verify:** Check `TimeSeriesAggregator.py` to confirm `run_qso_count` and `total_mults` streams are available for the new report.
3. **Execute:** Generate the **Builder Execution Kit** containing:

- `contest_tools/reports/plot_correlation_analysis.py`
- `web_app/analyzer/views.py` (Refactored)
- `web_app/analyzer/templates/analyzer/qso_dashboard.html` (Tabs updated)
- `web_app/analyzer/templates/analyzer/multiplier_dashboard.html` (Cards updated)