--- METADATA: FILE_COUNT=2 ---

--- FILE: ArchitectureRoadmap.md ---

ArchitectureRoadmap.md

Version: 2.4.0 **Date:** 2025-12-15 **Status:** Phase 4 (Analytics Refactor) - Step 4 (Public Log Fetcher) Ready

---

1. The Strategic Vision

Goal: Create a single analysis engine that powers both a Command-Line Interface (CLI) and a Web Interface (Django). **Core Constraint:** "Write Once, Render Everywhere." Logic must not be duplicated. **Deployment Model: Stateless & Portable.**

- No User Accounts. No User Database. Atomic Sessions.
- Containerized (Docker) to ensure "Laptop == Server".

2. The "Golden Path" Workflow

This defines the specific user experience and data flow for the Web Interface.

1. **The Three-Slot Model:** The analyzer presents **three generic input slots** (Log A, Log B, Log C).
2. **Identity Agnostic:** There is no concept of "My Log" vs. "Competitor."
3. **Source Agnostic:** Each slot can be filled by either:
   - **Direct Upload:** User uploads a local `.log` file.
   - **Public Fetch:** User selects a contest/year/callsign (Phase 4.2).
4. **Session Persistence:**
   - Files exist in `media/sessions/<session_key>/` for the duration of the analysis session.
   - Lazy Cleanup ensures old sessions are purged automatically.
   - **Dashboard Context:** Session state is persisted to `dashboard_context.json` to allow stateless navigation.

3. Architectural Decision Records (ADRs)

ADR-001: Data Abstraction Layer (DAL)

- **Decision:** All business logic exists in `data_aggregators/`. Returns **Pure Python Primitives**.

ADR-002: Unified Visualization (Client-Side Rendering)

- **Decision:** Replace Matplotlib with **Plotly**.

- **Animation:** Replace server-side MP4 generation with **Plotly HTML Animations**.

ADR-007: Shared Presentation Layer

- **Decision:** The Web App (Django) must point to the existing `contest_tools/templates`.
- **Reason:** Ensures CLI HTML reports and Web Views are bit-for-bit identical.

ADR-009: Session-Scoped Persistence

- **Decision:** Replace "Ephemeral I/O" (tempfile) with Session-Scoped Storage (`media/sessions/`).
- **Reason:** Required to support drill-down navigation and static asset serving (images/HTML) in the browser.

---

4. Master Transition Timeline

Phase 1: Data Decoupling (COMPLETE)

- **Status:** Complete. DAL is operational.

Phase 2: Visualization Standardization (COMPLETE)

- **Status:** Complete. Static charts migrated to Plotly.

Phase 2.5: Animation Modernization (COMPLETE)

- **Status:** Complete. `plot_interactive_animation.py` implemented.

Phase 3: The Web Pathfinder (COMPLETE)

- **Status:** Complete. Containerization, Django Bootstrap, and Validation successful.

Phase 4: The Strategy Board & Analytics Refactor (ACTIVE)

- **Status:** In Progress.
- **Goal:** Implement the "Top-Down" UI strategy ("The Arena", "The War Room").
- **Tasks:**
  - ☒ **Step 1: The Strategy Board:** Upgrade Dashboard table with "Run %" metrics.
  - ☒ **Step 2: The Drill-Down UI:** Implement the "Triptych" (Animation/Plots/Mults).

- ☒ **Step 3: Sub-Page Views:** Create Django views to wrap raw report files.
  - ☒ **Dashboard Hardening:** Implemented `dashboard_view` persistence to fix navigation loops.
  - ☒ **Semantic Routing:** Enforced strict lowercase paths in `views.py` for Linux compatibility.
  - ☒ **Visualization Upgrade:** Consolidated Cumulative Difference plots into single-chart superimposed layout.
- ☐ **Step 4: Public Log Fetcher:** (Phase 4.2) Implement scraper for public contest logs to fill slots 2 & 3.

--- FILE: ArchitectHandoff.md ---

Architect Handoff Notes

Date: 2025-12-15 **To:** Incoming Architect **Focus:** Phase 4, Step 4 (Public Log Fetcher)

Context & Status

The **Web Dashboard (Phase 4, Steps 1-3)** is now **Complete and Hardened**. We have just concluded an extensive Ad-Hoc debugging session that resolved critical usability and compatibility issues.

Key Architectural Updates (Session Persistence)

- **The "Back Button" Loop:** We resolved a navigation loop where the "Back to Dashboard" button sent users to the Upload form.
- **Solution:** We implemented **Session Context Persistence**. The `analyze_logs` view now dumps the dashboard context to `dashboard_context.json` within the session directory. A new view, `dashboard_view`, reloads this context.
- **Impact:** Future views MUST link to `dashboard_view` (passing `session_id`) rather than the generic `analyze` endpoint.

Visualization Enhancements

- **Cumulative Difference Plots:** We refactored `plot_cumulative_difference.py` from a 3-subplot layout to a **Single Superimposed Chart** (Total, Run, S&P, Unknown).
- **DAL Update:** `time_series.py` was updated (v1.7.0) to provide granular `sp_qsos` and `unknown_qsos` streams.

Platform Compatibility

- **Linux/Docker Paths:** We enforced `.lower()` on all URL path components in `views.py` to match the lowercase directories created by

`ReportGenerator`. **Strict case matching is now a hard requirement.**

Immediate Priorities

The foundation is solid. The next logical step is to populate the "Competitor" slots with real data without requiring manual uploads.

- **Target: Phase 4, Step 4 (Public Log Fetcher)**.
- **Goal:** Allow users to enter a Callsign/Contest/Year and automatically fetch the public log (if available) into Slot 2 or 3.
- **Mechanism:** Likely scraping 3830scores or public log archives (rules permitting).

Recommended Next Action

1. **Definitive State Initialization:** Upload the full bundle + this Kit to reset the token window.
2. **Begin Phase 4.2:** Analyze requirements for the Public Log Fetcher.