--- FILE: ImplementationPlan.md ---

Implementation Plan - Phase 3: The Web Pathfinder

Version: 1.0.0 **Target:** 0.102.0-Beta

1. Executive Summary

This plan executes **Phase 3** of the Architecture Roadmap. It creates the infrastructure required to run the Contest Log Analyzer as a web application. It introduces Docker containerization and bootstraps a Django project structure (`web_app`) alongside the existing `contest_tools` library.

2. Architecture & Constraints

- **Statelessness:** The web tier is stateless. Persistence is ephemeral.
- **Shared Presentation Layer (ADR-007):** The Django `settings.py` will explicitly map the `TEMPLATES` directory to the existing `contest_tools/templates` to ensure the CLI and Web UI use identical rendering logic.
- **No FFmpeg:** The Docker image will typically utilize `python:3.11-slim` and intentionally excludes multimedia libraries, enforcing the modernization achieved in Phase 2.5.
- **Directory Structure:**

```
/ (Project Root)
   contest_tools/       (Existing Core Library)
   Dockerfile           (New)
   docker-compose.yml   (New)
   requirements.txt     (New)
   web_app/             (New Django Project Root)
       manage.py
       config/          (Django Settings Package)
           settings.py
           ...
```

3. Proposed Changes

3.1. Infrastructure Files

A. `Dockerfile`

- **Base Image:** `python:3.11-slim`
- **Directives:**
  - Set `PYTHONPATH` to `/app` to ensure `contest_tools` is importable.
  - Install dependencies from `requirements.txt`.
  - Explicitly exclude `ffmpeg` installation.

B. `docker-compose.yml`

- **Service:** `web`
- **Volume Mapping:** `.:/app` to enable hot-reloading of code changes during development.
- **Ports:** Maps container `8000` to host `8000`.

C. `requirements.txt`

- Consolidates core analysis libs (`pandas`, `plotly`) with web libs (`django`, `gunicorn`).

3.2. Django Bootstrap (`web_app/`)

A. `web_app/manage.py`

- Standard Django entry point.
- Configured to point to `config.settings`.

B. `web_app/config/settings.py`

- **Key Configuration:**
```
BASE_DIR = Path(__file__).resolve().parent.parent
# ...
TEMPLATES = [
    {
        # ...
        'DIRS': [BASE_DIR.parent / 'contest_tools' / 'templates'],
        # ...
    },
]
```
- This explicitly bridges the new Web App to the existing CLI templates.

C. Supporting Config Files

- `web_app/config/urls.py`: Basic URL routing.
- `web_app/config/wsgi.py`: WSGI entry point.
- `web_app/config/asgi.py`: ASGI entry point.
- `__init__.py` files to make directories packages.

4. Verification Plan

4.1. Automated Syntax Check

- All Python files will undergo syntax validation.

4.2. Architecture Compliance

- **Pre-Flight:** Verify `settings.py` template path resolution logic.
- **Post-Flight:** User will be instructed to run `docker-compose up --build` to verify the container builds and Django starts (Outputting ”Quit the server with CONTROL-C”).

---

5. Surgical Changes

File: `Dockerfile` (New)

```
# Dockerfile
# Version: 0.102.0-Beta
FROM python:3.11-slim

# Prevent Python from writing pyc files to disc
ENV PYTHONDONTWRITEBYTECODE 1
# Prevent Python from buffering stdout and stderr
ENV PYTHONUNBUFFERED 1
# Add the root directory to PYTHONPATH so web_app can import contest_tools
ENV PYTHONPATH /app

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["python", "web_app/manage.py", "runserver", "0.0.0.0:8000"]
```