--- FILE: Docs/InstallationGuide.md ---

Installation Guide - Contest Log Analyzer

Version: 0.30.30-Beta Date: 2025-08-05

--- Revision History ---

[0.30.30-Beta] - 2025-08-05

- Updated environment variable from CONTEST_DATA_DIR to CONTEST_LOGS_REPORTS.

- Added Git workflow instructions for developers and beta testers.

[0.30.0-Beta] - 2025-08-05

- Initial release of Version 0.30.0-Beta.

- Standardized all project files to a common baseline version.

This project uses a two-branch Git workflow to separate ongoing development from stable beta releases.

- `main` branch: Contains the latest stable version for beta testers.
- `develop` branch: Used for all new coding and features, which may be unstable.

This guide is divided into two sections: one for **Beta Testers** to install and update the software, and one for the **Developer** to manage the workflow.

Instructions for Beta Testers

These instructions will help you install the latest stable beta release of the Contest Log Analyzer.

Step 1: Clone the Repository

Before you begin, you will need to have **Git** installed on your system to download the project's source code. Open your command prompt or terminal, navigate to the directory where you want to store the project, and run the following command to download the software:

```
git clone https://github.com/kd4d/Contest-Log-Analyzer.git "Contest-Log-Analyzer"
cd "Contest-Log-Analyzer"
```

Step 2: Create and Activate a Conda Environment

This project uses `conda` for environment and package management. It is highly recommended to create a dedicated environment to avoid conflicts with other Python projects.

```
conda create --name contest-analyzer python=3.11 -y
conda activate contest-analyzer
```

Step 3: Install Dependencies

Once the environment is activated, install the required libraries using the following commands.

```
conda update --all -y
conda install pandas matplotlib seaborn -y
```

Step 4: Set Up the Environment Variable

The program requires the `CONTEST_LOGS_REPORTS` environment variable to be set to the root directory containing your `Logs`, `data`, and `reports` subdirectories.

- **Windows (Temporary, for the current command prompt session):**

  ```
  set CONTEST_LOGS_REPORTS="C:\path\to\your\Contest-Log-Analyzer"
  ```

- **macOS/Linux (Temporary, for the current terminal session):**

  ```
  export CONTEST_LOGS_REPORTS="/path/to/your/Contest-Log-Analyzer"
  ```

Step 5: Download Required Data Files

Create a `data` directory inside your `Contest-Log-Analyzer` project folder if it doesn't exist. Place the necessary data files in this directory.

- **Required for all contests:** `cty.dat` (available from country-files.com)
- **Required for ARRL DX:** `ARRLDXmults.dat`
- **Required for ARRL SS:** `SweepstakesSections.dat`

Step 6: Updating to a New Version

When a new stable version is announced, navigate to your project directory in your terminal or command prompt and run this single command:

```
git pull
```

This will download and apply all the updates for the latest beta release.

Developer Guide

This section outlines the development and release workflow.

One-Time Setup: Creating the `develop` Branch

If you haven't already, create the `develop` branch from the `main` branch. This will be your primary branch for all new work.

```
# From your repository's root directory
git checkout -b develop
git push -u origin develop
```

Standard Development Workflow

All new features, bug fixes, and other code changes should be committed to the `develop` branch.

1. **Ensure you are on the `develop` branch:**

   ```
   git checkout develop
   ```

2. **Do your work:** Modify, add, and delete files as needed.
3. **Commit your changes:**

   ```
   # Add your changes to the staging area
   git add .

   # Commit the changes with a descriptive message
   git commit -m "A brief description of the work you did"

   # Push your commits to the remote develop branch
   git push origin develop
   ```

Publishing a New Beta Release to Testers

When the code in the `develop` branch is stable, tested, and ready for a beta release, you merge it into the `main` branch. This makes it available to your testers.

1.

1. **Update the Revision History:** Before merging, update the `Revision History` section at the top of this `InstallationGuide.md` file with the new version number and changes.
2. **Switch to the `main` branch:**

   ```
   git checkout main
   ```

3. **Ensure `main` is up-to-date:**

   ```
   git pull origin main
   ```

4. **Merge `develop` into `main`:** This brings all the stable, new code into the release branch.

   ```
   git merge develop
   ```

5. **Push the `main` branch:** This publishes the new release.

   ```
   git push origin main
   ```

6. **Switch back to `develop` to continue working:**

   ```
   git checkout develop
   ```

After this process, you can inform your beta testers that a new version is available. They can then run `git pull` to get the changes.