User Manual for Log Analysis Flask Application

Industrial Project - Katarzyna Góźdź, Jan Michalak, Wiktor Zborowski

December 19, 2024

Contents

1 Introduction

This application is designed to process, analyze, and visualize log data. It provides a web interface for configuring analysis parameters, viewing metrics, generating visualizations, and downloading comprehensive reports.

2 Features

- **File Processing:** At the beginning as an input we should provide the application .log files with specific naming convections. The app will convert it to the csv files which will be necessary for the next analysis steps.

- **Data Analysis:** Computes error metrics such as MAE, MSE, RMSE, MAPE, Max Error, and Std Error.

- **Visualizations:** Generates histograms, line plots, and boxplots for error distributions across different degrees.

- **Web Interface:** Provides routes for processing logs, configuring parameters, viewing data, and downloading reports.

- **Report Generation:** Allows downloading analysis results in PDF format.

3 Prerequisites

Before installing and running the application, ensure that your system meets the following requirements:

- **Operating System:** Windows, macOS, or Linux.

- **Python:** Version 3.10 or higher.

- **Most important python Packages:**

  - Flask
  - pandas
  - numpy
  - matplotlib
  - tabulate

- **Additional Tools:**

  - A web browser (e.g., Chrome, Firefox).

4 Installation

4.1 Clone the Repository

First, clone the application's repository to your local machine:

```
1 git clone https://github.com/Torekas/log-analysis-flask-app.git
2 cd log-analysis-flask-app
```

4.2 Set Up a Virtual Environment

It's recommended to use a virtual environment to manage dependencies:

```
1 python -m venv venv
```

Activate the virtual environment:

- **Windows:**

```
1     venv\Scripts\activate
2
```

- **macOS/Linux:**

```
1     source venv/bin/activate
2
```

4.3 Install Dependencies

Install the required Python packages using `pip`:

```
1 pip install -r requirements.txt
```

Ensure that your `requirements.txt` includes:

```
1 Flask
2 pandas
3 numpy
4 matplotlib
5 tabulate
6 requests
```

5 Configuration

5.1 File Naming Conventions

The application converts all .log files in the specified directory to the csv files with the same name as the .log files. After this process application require specific name of the csv files so it is better to name the .log file in the given way as below:

The following regular expression is used to match a specific file name structure:

$$Putty\_(Big|Small|BigSmall)\_(\backslash d+)cm\_initf\_115200\_(\backslash d+)\_degree(?:\_(\backslash d+))?\backslash.csv$$

5.2 Regex Explanation

1. **Putty_**: Matches the literal string "Putty_".

2. **(Big—Small—BigSmall)**: Matches one of the options: "Big", "Small", or "BigSmall".

3. **\d+**: Matches one or more digits (a number).

4. **cm**: Matches the literal string "cm".

5. **_initf_115200_**: Matches the literal string "_initf_115200_".

6. **\d+**: Matches another numeric value (one or more digits).

7. **_degree**: Matches the literal string "_degree".

8. **(?:_(\d+))?**:
   - **(?: ...)**: A non-capturing group.
   - **_(\d+)**: Matches an optional underscore followed by one or more digits.
   - **?**: Makes the entire non-capturing group optional.

9. **\.csv**: Matches the literal string ".csv".

Here are some example filenames that match the given regular expression:

- `Putty_Big_100cm_initf_115200_0_degree.csv`

- `Putty_Small_150cm_initf_115200_45_degree_1.csv`

- `Putty_BigSmall_200cm_initf_115200_90_degree.csv`

5.3 Mapping Degrees to File Indices

During the measurements for the big boards we tried many angles so to not make a mess in the namings convenctions we use following numbers starting from the 1 and then map. The application maps file indices to specific degree angles as follows:

File Index	Degree
1	0°
2	45°
3	90°
4	135°
5	180°
6	225°
7	270°
8	315°

Table 1: Mapping of File Indices to Degrees

But the file does not have to have this index in the name. If it is alone the tool will process it without any errors. It is only necessary when we have many files with the same name.

6 Running the Application

6.1 Start the Flask Server

To run the application, execute the following command within the project directory:

```
1 python app.py
```

By default, Flask runs on `http://127.0.0.1:5000`.

6.2 Accessing the Web Interface

Open your web browser and navigate to:

$$\text{http://127.0.0.1:5000}$$

This will load the application's homepage, where you can interact with various functionalities.

7 Application Usage

7.1 Processing Log Files

1. In the upper right corner navigate to the `/process_logs` route or access it via the web interface.

2. We have to provide the absolute path for our logs folder, the absolute path for our output folder and the absolute path of the place where we want to store our databse file to gather all tables in one place.

3. Click on the button to initiate log processing.

4. The application will parse all LOG files in the specified directory, convert them to csv files and gather all tables into .db file.

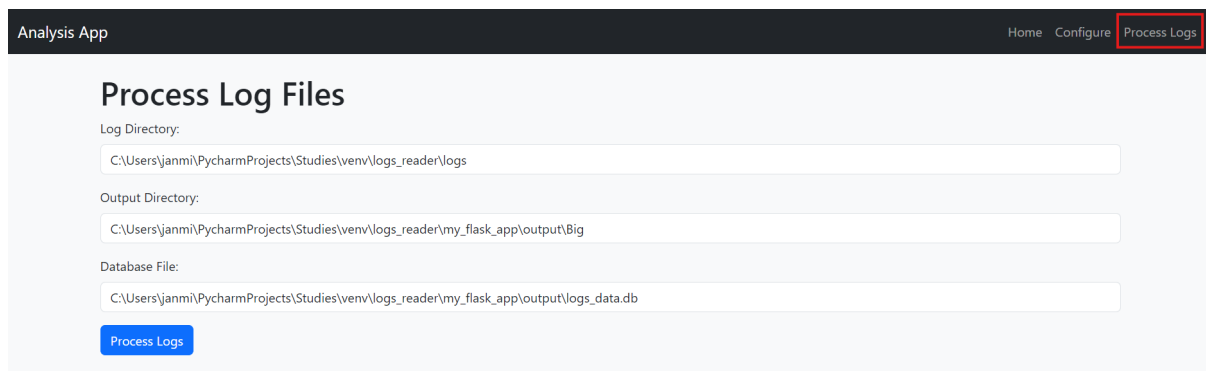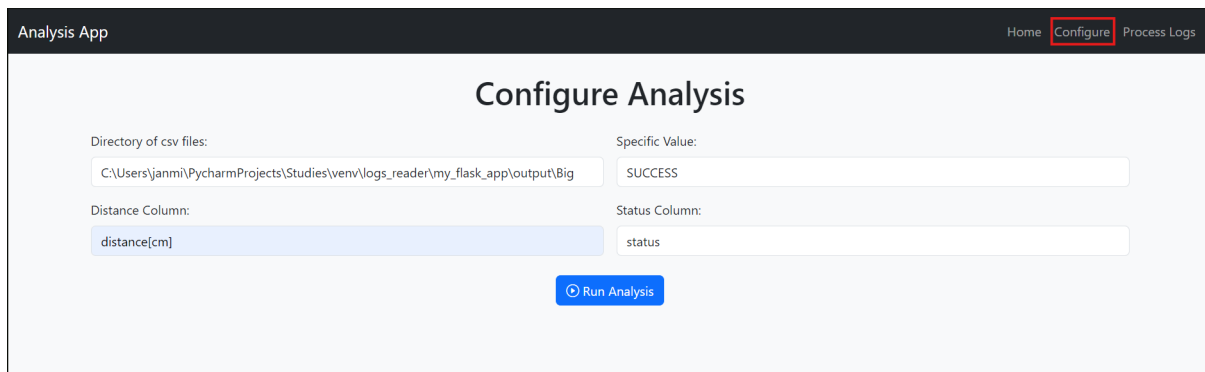5. Upon completion, a success or failure message will be displayed.

Figure 1: An example how to set up values

7.2 Configuring Analysis Parameters

1. In the upper right corner navigate to the `/configuer` route through the web interface.

2. Fill in the required fields:

   - **Directory of csv files:** Absolute Path to the directory containing csv files.
   - **Specific Value:** The value to filter measurements (e.g., a status indicator). In the Big boards there is `SUCCESS` and in the Small one there is `Ok`
   - **Distance Column:** Name of the column representing distances in your data.In the Big boards there is `distance[cm]` and in the Small one there is `D_cm`
   - **Status Column:** Name of the column representing statuses in your data. In the Big boards there is `status` and in the Small one there is `Status`. It is case sensitive

3. Submit the form to save the configuration.

4. The application will initialize the `Orchestrator` with the provided parameters and run the analysis.

5. You will be redirected to the homepage upon successful configuration.

6. You can upload any number of files. The amount is not restricted. So for example you can upload 3 files and only three points will be calculated.
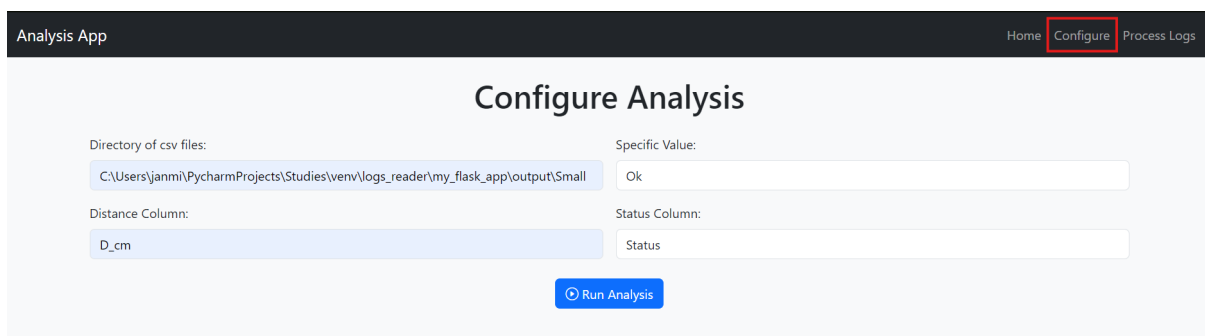
Figure 2: An example how to set up values for analysis

Figure 3: An example how to set up values for analysis

7.3 Viewing Analysis Results

1. On the homepage, select a specific `point_id` corresponding to a particular combination of size, distance, and angle.

2. The application will display:

   - **Metrics Table:** Displays error metrics such as MAE, MSE, RMSE, MAPE, Max Error, and Std Error for each degree.
   - **Chart Image:** Visualizes the error distributions through histograms, line plots, and boxplots.

3. To view the chart, ensure that the image is correctly loaded from the `/plot_chart` route.
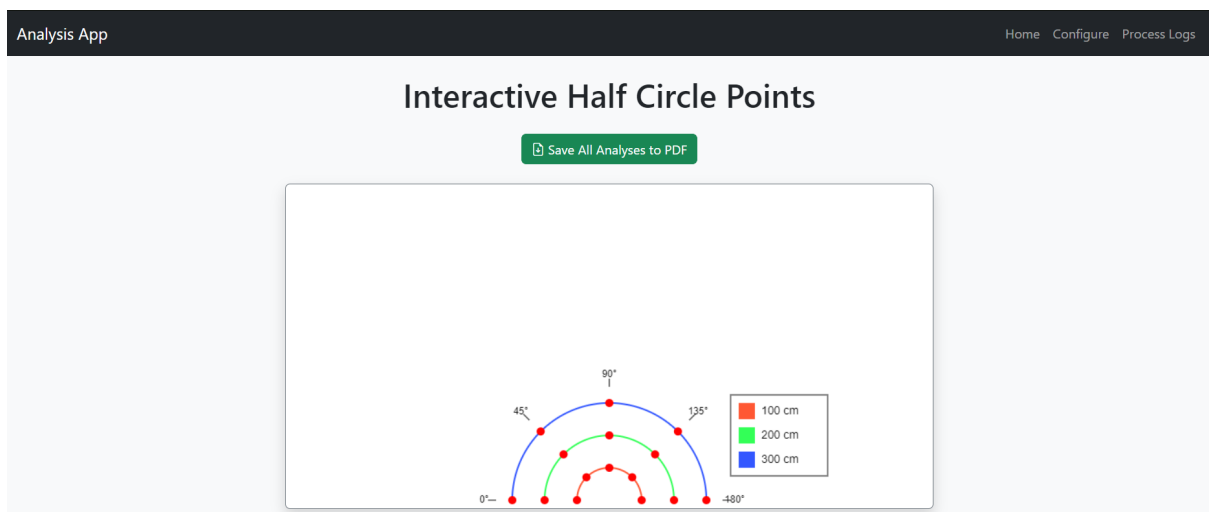

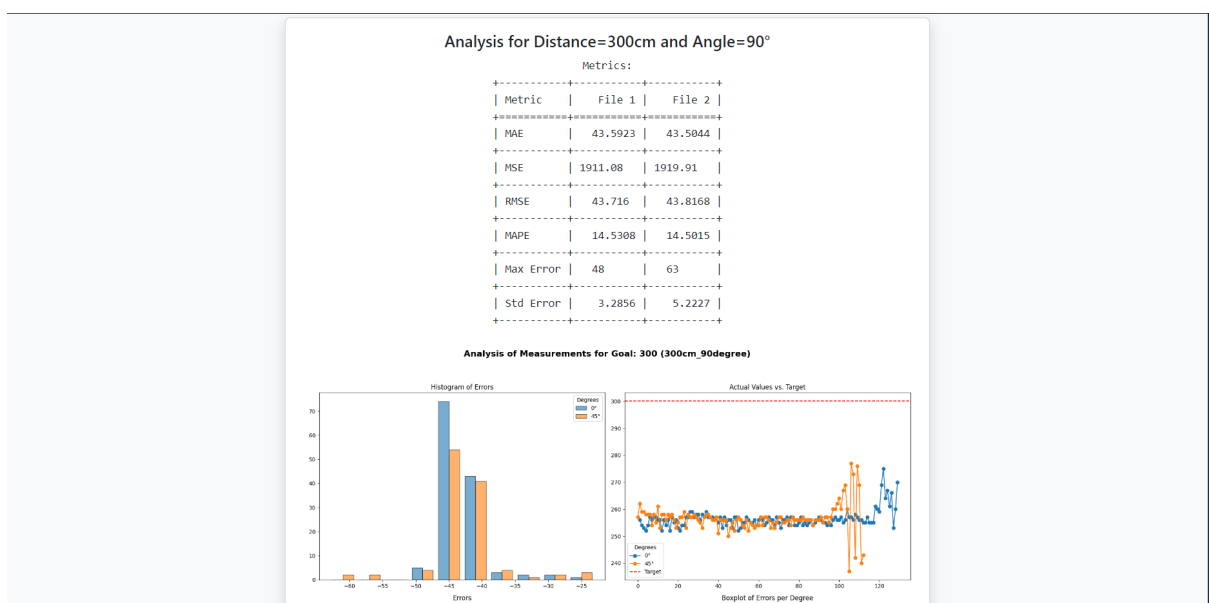Figure 4: An example how to set up values for analysis

Figure 5: An example how to set up values for analysis

7.4 Downloading Analysis Reports

1. Access the `/download_pdf` route via the web interface.

2. Click on the button to download a PDF report containing all analysis results, including metrics and visualizations.

8 Troubleshooting

8.1 Common Issues

8.1.1 Empty Metrics Table

- **Cause:** No data matches the specified `specific_value` filter.

- **Solution:** Verify that the `specific_value` is correctly set and that your data contains matching entries.

8.1.2 File Not Loaded

- **Cause:** Files do not follow the expected naming convention or have unsupported extensions.

- **Solution:** Ensure that all log files adhere to the naming patterns:

  ```
  Putty_Small_<distance>cm_initf_115200_<angle>_degree_<index>.<csv|log>
  Putty_Big_<distance>cm_initf_115200_<angle>_degree_<index>.<csv|log>
  ```

8.2 Checking Logs

Monitor the Flask server console output for any error messages or warnings during file processing and analysis.