**CSV Comparison Tool - High-Level Scope Document**

**1. Project Overview**

The **CSV Comparison Tool** is a Python-based utility designed to compare two CSV files. It identifies discrepancies such as mismatched values, missing or extra rows, missing or extra columns, and duplicated rows. The tool generates a comprehensive **HTML report** that summarizes the differences, making it easy for users to validate and correct their datasets.

**2. Project Objectives**

* **CSV File Comparison**: Compare two CSV files, highlighting discrepancies between the data.
* **Data Integrity Validation**: Ensure the primary key columns are consistent and the data is formatted correctly.
* **HTML Report Generation**: Provide an easy-to-understand summary report in HTML format that includes mismatched data, missing/extra rows, missing/extra columns, and duplicate rows.
* **Error Handling**: Notify users about missing files, missing columns, or data discrepancies with clear error messages.
* **Ease of Configuration**: Allow users to easily configure file paths, primary key columns, and output file location using a config.ini file.

**3. Key Features and Functionalities**

**3.1 Configuration Setup**

* **File Paths**: Users configure paths for the input CSV files and output report.
* **Primary Key Columns**: Users specify which columns will be used as the primary key for comparison.
* **Customizable Output File**: The output file location and format are configurable.

**3.2 File Validation**

* **Existence Check**: The tool verifies the existence of the specified files before proceeding.
* **Primary Key Check**: It ensures that all primary key columns are present in both files.

**3.3 Data Comparison**

* **Row Comparison**: Compares data in the rows of the two CSV files based on the primary key.
* **Column Comparison**: Flags any differences in corresponding columns.
* **Missing/Extra Rows**: Identifies rows that are present in one file but missing in the other.
* **Duplicate Rows**: Detects duplicate rows in either of the files.

**3.4 Report Generation**

* **Mismatch Summary**: Details any mismatched values between the two CSV files.
* **Missing/Extra Columns**: Lists columns that are present in one file but not the other.
* **Duplicate Row Detection**: Identifies rows that are duplicated within each file.
* **Email Integration**: Provides an option to send the comparison report via email.

**3.5 Error Handling**

* **Graceful Error Handling**: The tool will handle issues such as missing files, missing columns, and invalid data formats with detailed error messages.

**4. Scope Inclusions**

**4.1 Supported Input Files**

* Only **CSV files** are supported for input.

**4.2 Primary Key Columns**

* Users can specify which columns to use as the **primary key**.

**4.3 Data Comparison Types**

* **Row Comparison**: Compares data based on primary key.
* **Column Comparison**: Flags mismatches between files.
* **Missing/Extra Rows**: Identifies missing or extra rows in either file.
* **Duplicate Rows**: Identifies duplicate rows in either file.

**4.4 Report Generation**

* The tool generates an **HTML report** which includes:
  + **Mismatch Summary**: List of mismatches based on the primary key and columns.
  + **Summary Tables**: Lists missing columns, rows, and duplicates.
  + **Email Sending Option**: Provides a button to send the report via email.

**5. Out of Scope**

* **File Formats**: The tool supports only **CSV files**.
* **Real-Time Comparison**: The tool does not support real-time updates or synchronization.
* **Graphical User Interface (GUI)**: The tool is a command-line utility, with no GUI.
* **Advanced Data Transformation**: The tool does not perform data manipulation beyond basic comparison.

**6. Assumptions and Dependencies**

**6.1 Environment Requirements**

* **Python Version**: Python 3.x is required to run the tool.
* **Libraries**: The following Python libraries are needed:
  + **pandas**
  + **configparser**
  + **os**
  + **webbrowser**

**6.2 Permissions**

* Users must have **read and write permissions** for the input files and the output report location.

**6.3 Input File Format**

* The input files must be in **CSV format** with consistent headers.

**7. Constraints**

**7.1 File Size**

* The tool is optimized for medium-sized files but may experience performance issues with extremely large datasets.

**7.2 File Comparison Limitations**

* The tool only compares **two CSV files** at a time; it does not support multi-file comparison.

**8. Dependencies**

* **Python Libraries**:
  + **pandas**: For reading and comparing CSV files.
  + **configparser**: For reading configuration files.
  + **os**: For file existence checks.
  + **webbrowser**: For opening the generated report in a web browser.

**9. Next Steps**

**9.1 Tool Implementation**

* Implement the core CSV comparison logic.
* Develop error-handling mechanisms.

**9.2 Report Generation**

* Develop the HTML report formatting and summary generation functionality.

**9.3 Testing**

* Conduct thorough testing to ensure the tool handles edge cases (missing columns, large files, etc.).

**9.4 Deployment**

* Package the tool for distribution, including the necessary dependencies.

**Appendix: Sample HTML Report Structure**

**9.1 HTML Report Overview**

The HTML report will include the following sections:

* **Report Information**: Basic details about the comparison (files compared, primary key columns).
* **Mismatch Summary**: Displays rows where values differ between the two files.
* **Missing Columns**: Lists columns missing from one file.
* **Missing/Extra Rows**: Displays rows that are missing/extra between the files.
* **Duplicate Rows**: Identifies duplicate rows in the CSV files.
* **Actionable Report**: An email button to share the report.