****

**ARICH INFOTECH PVT. LTD.**

**Document Information**

| **Document Name** | **High Level Design (HLD)** |
| --- | --- |
| **File Comparator** |
| **Version No.** | **1.1** |
| **Release Date** | **18 - 04 -2024** |
| **Author** | **DINESH V** |

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Summary of change** |
| **1.1** | **18 – 04 -2024** | **Dinesh V** | - |
|  |  |  |  |
|  |  |  |  |

**Contents:**

**1.** Document Information ……………………………………………………………….....1

1.1**.** Document History…………………………………………………………………....1

**2.** Introduction......................................................................................................................3

2.1. Purpose of the Document ........................................................................................3

2.2. Objectives................................................................................................................3

2.3. Scope of HLD .........................................................................................................3

## 3. System Standards

3.1 Naming Standards…………………………………………………………………3

3.2 Coding Standard…………………………………………………………………..4

**4.** System Overview..............................................................................................................5

**5.** Drivers and Requirements................................................................................................6

5.1. Drivers.......................................................................................................................6

5.2. Requirements.............................................................................................................6

**6.** Process Flow diagram......................................................................................................7

**7.** Flow chart........................................................................................................................9

8. Future enhancements ……………………………………………………………..…..10

**2. Introduction**

* 1. **Purpose of the Document**

The purpose of this plan is to

* Identify core modules/sub-systems of the system.
* Identify components, state, and communication mechanisms between different Sub -systems and identify the external interface.
* Identify various usage scenarios.

**2.2 Objectives**

* Develop a user-friendly software tool.
* Enable users to efficiently compare two files.
* Support text, CSV, or JSON file formats.
* Provide a graphical user interface (GUI) for intuitive interaction.
* Implement robust file handling functionalities for selecting and displaying file contents.
* Utilize an effective comparison algorithm to identify differences.
* Incorporate a highlighting mechanism to visually distinguish discrepancies.
* Implement quit functionality for a seamless user experience.
* Deploy the software as an executable or installer for widespread accessibility.
* Commit to ongoing maintenance to address issues, bugs, or updates.
* Aim for usability, effectiveness, and longevity of the software

**2.3 Scope of HLD**

* This HLD covers all areas of system.

## 3. System Standards

* 1. **Naming Standards**
* **Class Names:**
* Use Camel Case convention.
* Example: FileComparatorApp
* **Function and Method Names:**
* Use lowercase with words separated by underscores.
* Be descriptive about the function's purpose.
* Example: choose\_file\_1
* **Variable Names:**
* Use lowercase with words separated by underscores.
* Be descriptive about the variable's purpose.
* Example: file1\_name.
* **Constants:**
* Use uppercase with words separated by underscores.
* Constants are typically defined at the top of the file or in a separate module.
* Example: PEP\_8
  1. **Coding Standard**

| **Category** | Standard |
| --- | --- |
| **PEP 8 Compliance** | Mostly adhered to, ensure consistency in spacing, line lengths, etc. |
| **Naming Conventions** | * **Class names:** Camel Case * **Function and method names:** lowercase with underscores * **Variable names:** lowercase with underscores * **Constants:** upper case with underscores |
| **Comments** | Include comments where necessary for clarity and explanation of complex logic. |
| **Modularity** | Code should be modular with separate functions for different tasks. |
| **Error Handling** | Include error handling, such as checking if files are selected before comparison. |
| **User Interface Design** | Clear and intuitive UI design with appropriate use of labels, buttons, and scrolled text areas**.** |
| **File Handling** | Files Should open using the with statement for proper resource management. |
| **Use of Libraries** | Effective utilization of tkinter for GUI development and itertools for iteration. |
| **Consistent Layout** | Consistent and organized layout of widgets in the UI. |
| **Color Coding** | Implement color coding to highlight differences in file content. |

**4. System Overview**

This System is a file comparison tool designed to provide users with a simple and efficient way to compare two files in text, CSV, or JSON format. The system consists of a graphical user interface (GUI) and underlying functionality for file handling, comparison, and highlighting of differences. Here's an overview of the key components and their interactions:

* **Graphical User Interface (GUI):**
* The GUI provides a user-friendly interface for interacting with the software.
* It includes components such as text boxes for displaying file contents, buttons for file selection and comparison, and a menu bar for navigation and functionality access.
* **File Handling:**
* The system allows users to select two files using the File Choose button.
* Upon selection, the contents of each file are displayed in separate text boxes (File 1 Contents and File 2 Contents).
* File handling functionalities ensure that only text (txt), comma-separated values (CSV), or JavaScript Object Notation (JSON) files can be selected.
* **Comparison Algorithm:**
* Once the files are selected, the system utilizes a comparison algorithm to analyse their contents and identify differences.
* The algorithm compares the text, CSV, or JSON structures of the files and highlights any discrepancies.
* **Highlighting Differences:**
* Differences between the two files are highlighted in red to make them visually distinct to the user.
* This highlighting mechanism allows users to quickly identify variations between the files and understand their significance.
* **User Interaction:**
  + Users interact with the system by selecting files, initiating the comparison process, and viewing the highlighted differences.
  + The GUI provides feedback to the user, indicating the progress of file selection, comparison, and highlighting.
* **Quit Functionality:**
  + Users can exit the software either by clicking the close mark provided in the window or selecting the "Exit" option from the menu bar.
  + This functionality ensures a smooth user experience and proper termination of the application.
* **Deployment and Maintenance:**
  + The system is deployed as an executable file or installer for easy distribution and installation on users' machines.
  + Ongoing maintenance and support are provided to address any issues, bugs, or updates that may arise post-deployment.
  + User feedback and feature requests are considered for future updates and enhancements to the software, ensuring its continued usability and effectiveness.

**5. Drivers and Requirements**

* 1. **Drivers:**
* User Need:
  + - Understanding the need for a file comparison tool among users, such as developers, data analysts, or anyone working with text, CSV, or JSON files who require an efficient way to identify differences.
* Market Demand:
  + - Assessing the market demand for such a tool, including potential competitors and existing solutions, to determine how the file comparator can differentiate itself and meet specific user requirements.
* Functionality Requirements:
  + - Identifying the essential functionalities users expect from a file comparison tool, such as support for various file formats, accurate comparison algorithms, intuitive GUI, and efficient highlighting of differences.
* Technical Feasibility:
  + - Evaluating the technical feasibility of implementing the required features, considering factors like programming languages, libraries, and platforms suitable for GUI development, file handling, and comparison algorithms.
* User Experience (UX) Design:
  + - Planning the GUI layout, user interactions, and feedback mechanisms to ensure a smooth and intuitive user experience, minimizing user frustration and maximizing usability.

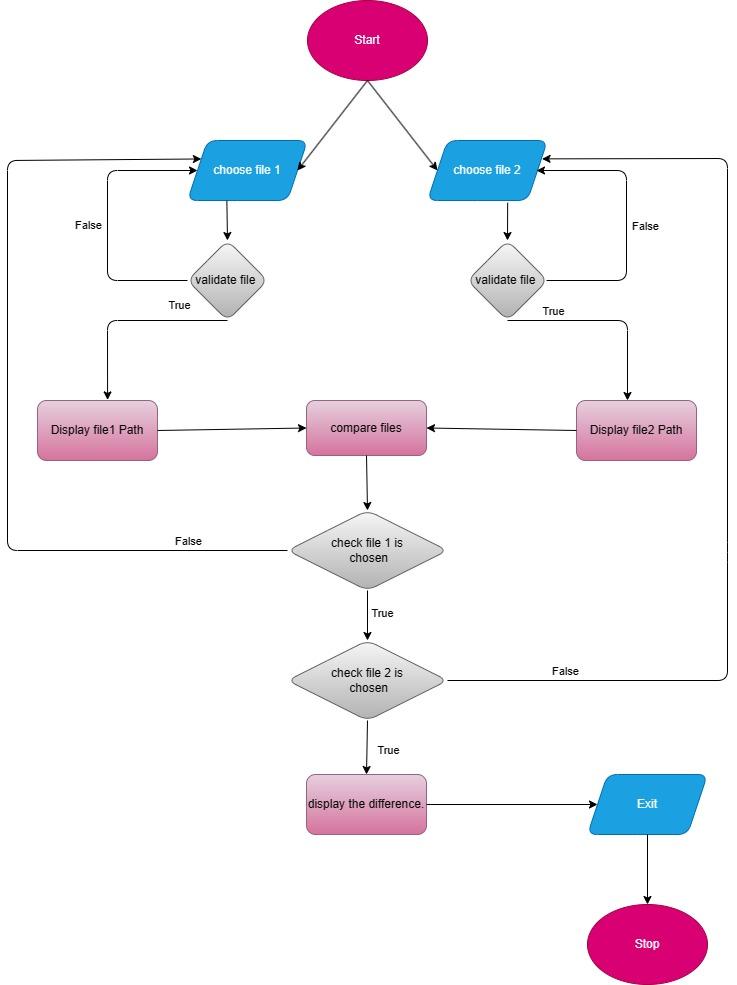
**5.2. Requirements**

* **Technical Requirements:**

Python Application

* **Hardware Requirements:**
* Processor (CPU): Any modern processor should suffice. Even older processors should handle this script without any issue.
* Memory (RAM): A minimum of 2GB RAM should be more than enough. Since the script is not memory-intensive, even systems with less RAM should handle it fine.
* Storage: The script itself doesn't require much storage space. However, you'll need space to store the files you're comparing.
* Graphics: Since you're using tkinter for GUI, any standard graphics card should work fine. Tkinter doesn't require high - end graphics processing.

**6**. **Process Flow diagram**

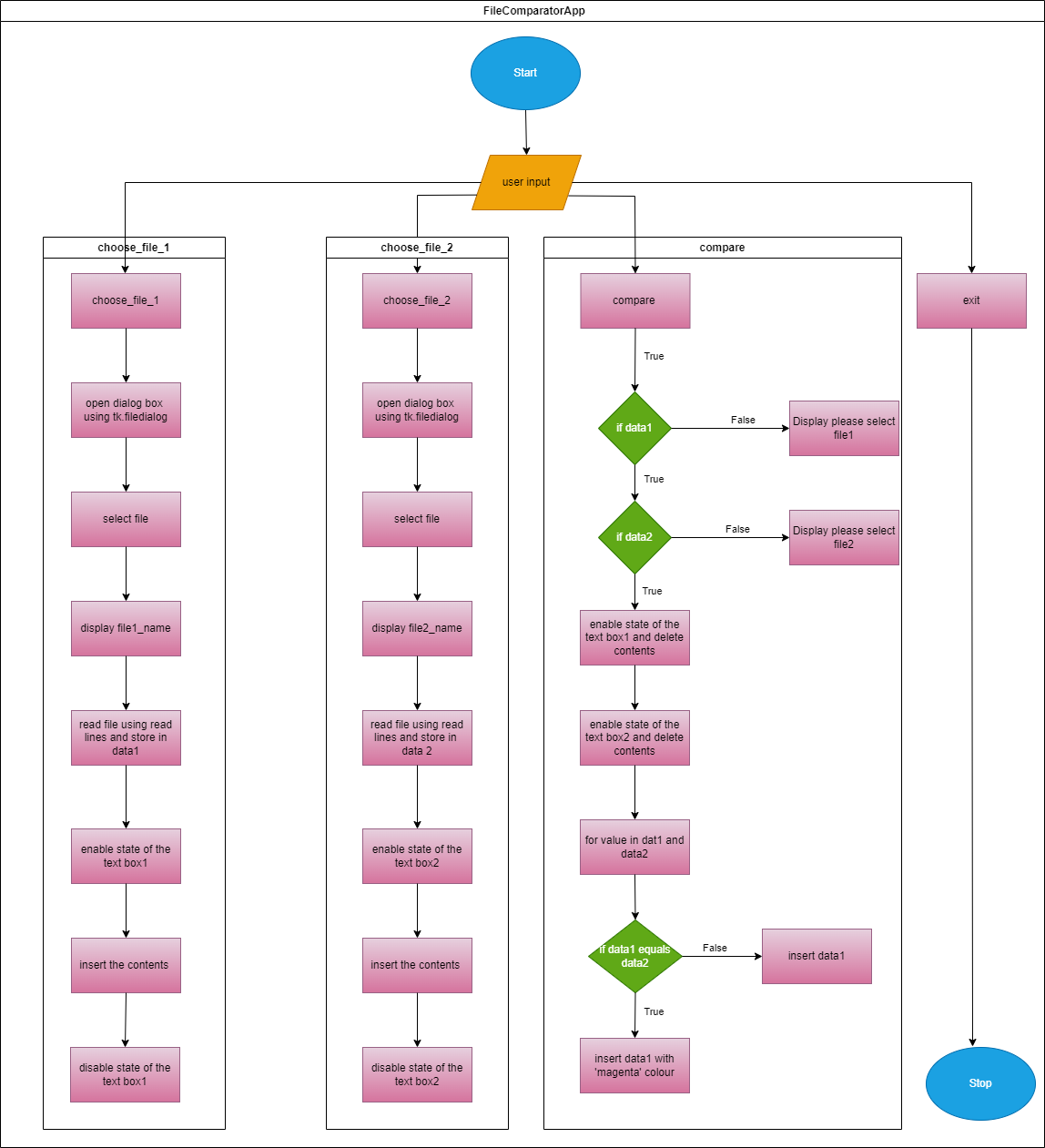


**Process Flow Diagram (PFD) Description:**

The process flow diagram outlines the steps involved in comparing two files and displaying their differences.

* **Main Components:**
  + **Start:** The process begins.
  + **Choose File 1:** The user selects the first file (File 1) using the “Choose File 1” button.
  + **Display File 1:** The application displays the contents of File 1 in a read-only text area.
  + **Choose File 2**: The user selects the second file (File 2) using the “Choose File 2” button.
  + **Display File 2**: The application displays the contents of File 2 in another read-only text area**.**
  + **Compare Files:** The user clicks the “Compare” button.
  + **Comparison Process**: The application compares the contents of both files and highlights any differences (e.g., added, modified, or deleted lines).
  + **Display Comparison Results:** The user can view the comparison results.
  + **End:** The process concludes.
* **Decision Points:**
  + If the user cancels file selection, the process stops.
  + If any issues occur during file comparison (e.g., file read errors), the application provides appropriate feedback.

**7 . Flow chart**



**8. Future enhancements**

1. User Feedback:

Implement a more user-friendly way to display differences, perhaps highlighting the specific lines where differences occur or providing a summary at the end of the comparison.

1. Error Handling:

Improve error handling to provide more informative messages to the user if file operations fail or if unexpected issues occur during the comparison process.

1. Performance Optimization:

Optimize the file comparison process for large files. Currently, simplementing a more memory-efficient approach, such as reading and comparing the files the entire contents of both files are loaded into memory, which may not be efficient for very large files.

1. Enhanced File Type Support:

Add support for more file types beyond just text, CSV, and JSON. Consider supporting popular binary formats or custom file formats.

1. Visual Enhancements:

Improve the visual appearance of the application by adding icons, themes, or custom styling to make it more visually appealing and intuitive for users.

1. Save Comparison Results:

Provide an option for users to save the comparison results to a file for later reference. This could be useful for documenting changes or sharing the results with others.

1. Integration with Version Control Systems:

If applicable, consider integrating the application with version control systems like Git to provide more advanced comparison features and support for version history.

1. Multi-Threaded Processing:

Explore the possibility of using multi-threading to improve the performance of the comparison process, especially when dealing with multiple large files simultaneously.

1. Localization:

Add support for multiple languages to make the application accessible to users from different regions.