## **Outline Desing of Document Comparator**

Contents present in **base** **line doc** (i.e. text, images, tables, signatures, templates, positions of text in page)

Contents present **in** **target** **files** (i.e. text, images, tables, signatures, templates, positions of text in page)

Contents present both **in** **target** **files** and **base line files** (i.e. text, images, tables, signatures, templates, positions of text in page) highlighting differences.

Visual comparison

Target doc image highlighting diff

Baseline doc image highlighting diff

Search a keyword to show it is present in target with all occurrences.

After uploading both files

Note: The files in both baseline and target can be .doc/.pdf/.png/.jpg/.html/.xml

Upload Target file.

Target Documents

Document Comparator

Upload Base line file.

Base line Documents

Meta data comparison

Search a string for its font properties.

Download Text report.

View Text report.

## SUMMARY:

The **"Document Comparator Tool"** HTML file is a comprehensive web-based application designed to **upload, analyse, and compare two documents**—a baseline and a target—across multiple dimensions. Here's a summary of its key features and functionality:

**🔍 Core Features**

1. **Document Upload Interface**
   * Supports **PDF, DOC, DOCX, and TXT** formats.
   * Allows uploading of a **baseline** and a **target** document.
   * Drag-and-drop functionality is supported.
2. **Search Functionality**
   * Users can **search for specific text** within the target document.
   * Results include **highlighted matches** and **contextual previews**.
3. **Comparison Capabilities**
   * **Visual Preview Comparison**:
     + Generates side-by-side image previews of each page.
     + Highlights **visual differences** using pixel-level analysis.
   * **Text Content Comparison**:
     + Uses the **Diff Match Patch** algorithm to show added, removed, and unchanged text.
     + Displays **text similarity percentage**.
   * **Document Metadata Comparison**:
     + Compares properties like **page count, font usage, and structure**.
     + Highlights differences in **fonts, font sizes, and styles**.
4. **Font Property Search**
   * Users can search for a string and retrieve its **font size, style, and colour** from both documents.
5. **Report Generation**
   * Generates a **printable report** summarizing the comparison results.

**🧠 Technical Highlights**

* Uses **PDF.js** for PDF parsing.
* Uses **Mammoth.js** for DOCX parsing.
* Uses **Tesseract.js** for OCR fallback when text extraction fails.
* Implements **canvas-based rendering** for visual previews.
* Responsive and styled with **modern CSS** for a clean UI.

## System Design (Both HLD & LLD)

### 📐 High-Level Design (HLD)

**🎯 Purpose**

The Document Comparator Tool is a **client-side web application** that allows users to upload and compare two documents (baseline and target) across multiple dimensions—text, visual layout, and metadata.

**🧱 Architecture Overview**

* **Frontend**: HTML, CSS, JavaScript
* **Libraries**:
  + **PDF.js** – for PDF parsing and rendering
  + **Mammoth.js** – for DOCX parsing
  + **Tesseract.js** – for OCR fallback
  + **Diff Match Patch** – for text comparison

**🧩 Major Components**

1. **Document Upload Interface**
2. **Document Parsers**
3. **Comparison Engines**
   * Text Comparison
   * Visual Comparison
   * Metadata Comparison
4. **Font Property Search**
5. **Report Generator**

**🔄 Data Flow**

User Uploads Documents

↓

Document Parsers Extract Content

↓

Comparison Engines Analyse Differences

↓

Results Displayed in UI + Report Generated

### 🔧 Low-Level Design (LLD)

**1. Document Upload Interface**

* Accepts .pdf, .docx, .txt
* Uses <input type="file"> and FileReader to load files into memory

**2. PDF.js Parser**

* Extracts:
  + Text content
  + Page previews (canvas)
  + Metadata (title, author, fonts)
* Fallbacks to OCR if text extraction fails

**3. Mammoth.js Parser**

* Converts DOCX to plain text
* Simulates pagination for visual preview

**4. Tesseract.js OCR**

* Used when PDF text extraction fails
* Converts page images to text

**5. Text Comparison Engine**

* Uses diff\_match\_patch. diff\_main()
* Highlights:
  + Additions (green)
  + Deletions (red)
* Calculates match percentage

**6. Visual Comparison Engine**

* Renders pages to canvas
* Compares pixel blocks
* Highlights differences with bounding boxes

**7. Metadata Comparison Engine**

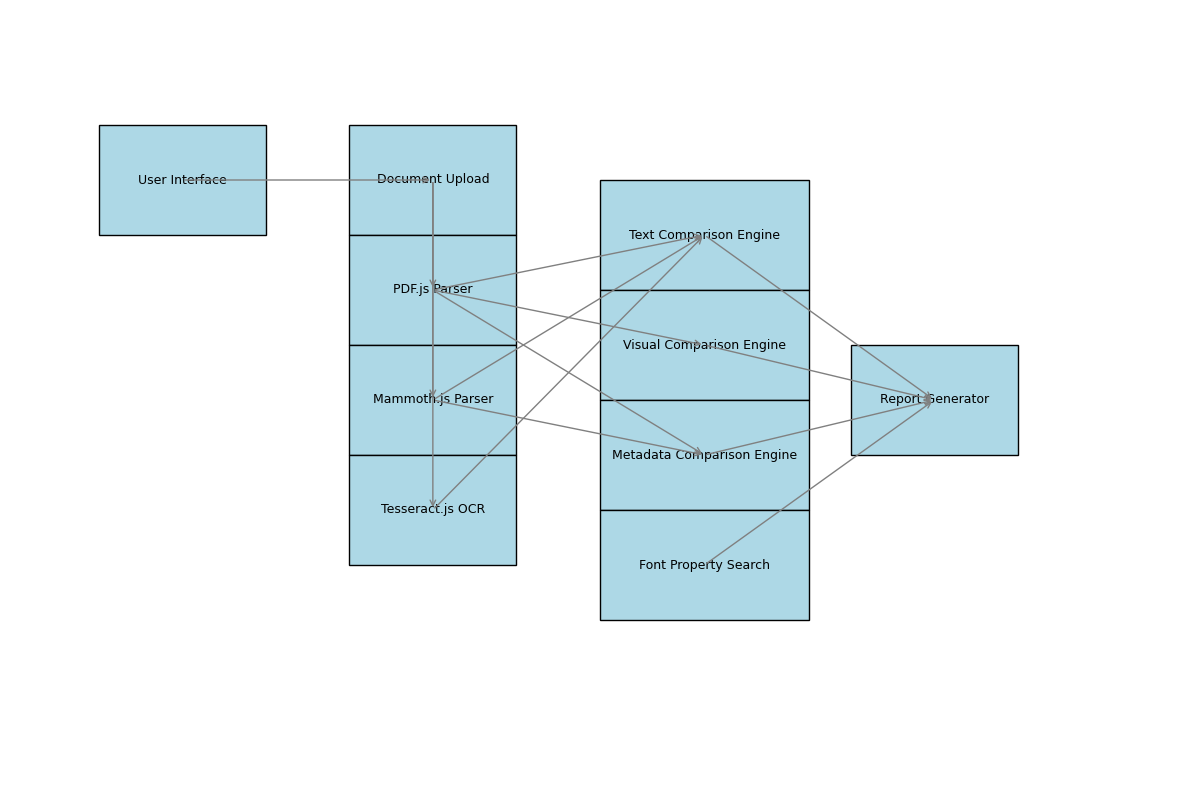
* Compares:
  + Page count
  + Font usage
  + Average font size
* Displays differences with visual indicators

**8. Font Property Search**

* Searches for a string
* Displays:
  + Font size
  + Font family
  + Color
  + Bold/Italic

**9. Report Generator**

* Compiles:
  + Text diffs
  + Visual diffs
  + Metadata summary
* Opens a new tab with printable HTML



## Security Considerations of the Application.

Based on the HTML file you provided, here’s a breakdown of the **security implications** when uploading documents to this tool:

**🔐 Security Considerations**

1. **Local-Only Processing (Client-Side)**
   * The tool uses **JavaScript libraries** like:
     + PDF.js for PDFs
     + Mammoth.js for DOCX
     + Tesseract.js for OCR
   * These libraries run **entirely in the browser**, meaning:
     + **No files are uploaded to a server**.
     + All processing (text extraction, comparison, rendering) happens **locally on your device**.
2. **No External File Transmission**
   * There are **no network requests** in the code that send your documents to any external server.
   * The only external scripts loaded are **public libraries** from trusted CDNs (e.g., jsDelivr, Cloudflare).
3. **Temporary In-Memory Use**
   * Files are read into memory using FileReader or arrayBuffer, processed, and then discarded when the page is refreshed or closed.
   * No persistent storage (like localStorage or cookies) is used to retain document content.
4. **No Authentication or Logging**
   * The tool does not include any user tracking, login, or analytics scripts.
   * This reduces the risk of data leakage or surveillance.

**✅ Conclusion**

If we are using this tool **locally** (e.g., opening the HTML file in our browser), our documents are **not exposed to the internet** and remain **secure**.

## Limitations

The **Document Comparator Tool**, while powerful and user-friendly, does have some limitations—especially considering its client-side, browser-based nature. Here's a breakdown:

**⚠️ Functional Limitations**

**1. File Size and Performance**

* Large documents (especially PDFs with many pages or images) may cause:
  + **Slow performance**
  + **Browser memory issues**
  + **Freezing or crashing** on low-end devices

**2. OCR Accuracy (Tesseract.js)**

* OCR fallback is used when text extraction fails.
* Limitations include:
  + **Lower accuracy** for poor-quality scans
  + **No layout preservation**
  + **Slower processing**

**3. Limited Format Support**

* Only supports:
  + .pdf, .doc, .docx, .txt
* Does **not support**:
  + .odt, .rtf, .xls, .ppt, or image files directly

**4. No Real-Time Collaboration**

* It’s a **single-user tool** with no multi-user or cloud sync features.

**🔐 Security & Privacy Limitations**

**5. No Encryption or Access Control**

* Files are processed in-browser, but:
  + There’s **no encryption** of data in memory
  + No user authentication or access control

**6. Browser Dependency**

* Relies on modern browser features (e.g., canvas, FileReader, Web Workers)
* May not work properly on:
  + **Older browsers**
  + **Mobile browsers**

**🧪 Comparison Limitations**

**7. Visual Comparison Is Approximate**

* Canvas-based rendering is a **simplified preview**
* Not a pixel-perfect representation of original document layout

**8. Text Comparison Ignores Formatting**

* Only compares **raw text**
* Ignores:
  + Font size
  + Bold/italic
  + Paragraph structure

**9. Metadata Comparison Is Basic**

* Compares only a few fields:
  + Page count
  + Fonts
  + Author/Title (if available)
* Does not handle embedded metadata or digital signatures

**Things to include more:**

1. Multiple-document comparison.
2. Report for image and meta data results.
3. Supporting even for word(.docs).
4. Authentication/Authorization (username and password login).