# **Architectural Design Document**

## **OCR-Based Text Extraction and Comparison System**

### **1. Introduction**

This document outlines the architectural design of an OCR-based system that processes images, extracts text using Tesseract, and performs text similarity analysis. The system includes various modules, each responsible for different stages of the process, ensuring efficiency and accuracy.

### **2. System Overview**

The OCR system is designed to:

* Preprocess input images for better OCR accuracy.
* Extract text from images using Tesseract OCR.
* Compare extracted text with reference text using similarity algorithms.
* Store processed images, extracted text, and comparison results systematically.

### **3. System Architecture**

#### **3.1 High-Level Design**

The architecture follows a modular approach, ensuring scalability and maintainability.

#### **3.2 Module Breakdown**

##### **3.2.1 Input Handling Module**

* **Folder:** Input
* **Purpose:** Stores input images before preprocessing.
* **Example File:** ocr\_preprocessing\_input.png

##### **3.2.2 Image Preprocessing Module**

* **Folder:** ImageProcessing
* **Files:**
  + AdaptiveThreshold.cs
  + ConvertToGrayscale.cs
  + Deskew.cs
  + GlobalThresholding.cs
  + SaturationAdjustment.cs
  + ShiftImage.cs
* **Purpose:**
  + Converts images to grayscale.
  + Applies adaptive/global thresholding.
  + Corrects skew and adjusts saturation.
  + Shifts images for alignment.
* **Implementation:** Uses OpenCV techniques for better OCR readability.

##### **3.2.3 OCR Processing Module**

* **Folder:** TesseractProcessor
* **Purpose:** Extracts text from processed images using Tesseract OCR.

##### **3.2.4 Text Comparison & Similarity Analysis Module**

* **Folder:** ModelComparision
* **Files:**
  + CosineSimilarityCalculator.cs
  + EmbeddingGeneratorService.cs
  + ProcessingTimeTracker.cs
* **Purpose:**
  + Converts text into vector embeddings.
  + Computes similarity using cosine similarity.
  + Tracks processing time for performance evaluation.

##### **3.2.5 Output Storage Module**

* **Folder:** Output
* **Subfolders:**
  + Comparision: Stores similarity results.
  + ExtractedText: Saves extracted text from images.
  + Logs: Maintains logs of processing steps.
  + ProcessedImage: Saves preprocessed images.

##### **3.2.6 Utility Functions**

* **Folder:** Utils
* **Purpose:** Provides helper functions for file handling, logging, and utility tasks.

##### **3.2.7 Core Execution & Integration**

* **Files:**
  + Program.cs: Main entry point orchestrating all modules.
  + OCRProject.csproj: Manages dependencies and configurations.

### **4. Workflow**

1. **Image Input Handling:** Image is placed in the Input folder.
2. **Preprocessing:**
   * Grayscale conversion, thresholding, deskewing, and other enhancements are applied.
   * The processed image is stored in Output/ProcessedImage.
3. **OCR Text Extraction:**
   * Processed image is passed to TesseractProcessor.
   * Extracted text is stored in Output/ExtractedText.
4. **Text Embedding & Similarity Calculation:**
   * EmbeddingGeneratorService converts text to vector embeddings.
   * CosineSimilarityCalculator compares extracted text with reference text.
5. **Logging & Performance Tracking:**
   * ProcessingTimeTracker records execution times.
   * Logs are stored in Output/Logs.

### **5. Conclusion**

This modular architecture ensures:

* **Efficiency** in OCR processing.
* **Scalability** for handling large datasets.
* **Accuracy** in text extraction and comparison.
* **Maintainability** with structured code and clear module separation.