Detailed Report on Attribute Extraction and Matching

# Overview

This Flask-based application is designed to analyze resumes and job descriptions, extracting key attributes and calculating their similarity. The system integrates various NLP and machine learning techniques to provide a comprehensive similarity score.

# System Components

## 1. Flask Application

* Purpose: Serves as the central interface for uploading resume and job description files and returns a calculated similarity score.
* Endpoints:
  + calculate\_similarity: Accepts POST requests with a resume and a job description file, processes them, and returns a similarity score.

## 2. File Processing

* Function: allowed\_file
  + Checks if the uploaded files have allowed extensions (PDFs).
* Function: extract\_file\_data
  + Extracts structured data from the uploaded resume and job description files.

3. Similarity Calculation

* Function: calculate\_final\_similarity\_score
  + Calculates the final similarity score between the extracted data from the resume and job description.
* Utilizes various similarity measures:
  + skills\_similarity\_score: Compares skills lists.
  + location\_similarity\_score: Assesses geographical relevance.
  + median\_designation\_similarity: Evaluates alignment in designations.
  + median\_degree\_similarity: Compares educational qualifications.
  + industry\_similarity\_score: Analyzes industry-related content.
* The final score is a weighted sum of these individual scores.

# Workflow

* File Upload: Clients upload a resume and a job description as PDF files via the /calculate\_similarity endpoint.
* File Validation: The application verifies that the uploaded files are in the correct format (PDF).
* Data Extraction: The extract\_file\_data function processes the files to extract relevant information.
* Similarity Scoring: The extracted data is passed to the calculate\_final\_similarity\_score function, which calculates individual similarity metrics and computes a final aggregated score.
* Result: The application returns the final similarity score as a JSON response.

# Error Handling

* File Upload: The system checks for the presence and format of the uploaded files, returning an error if they are missing or not in the allowed format.
* Data Extraction: Any errors during data extraction are logged, and the process continues with available data.
* Similarity Calculation: Exceptions in calculating similarity scores are caught, logged, and a default value (usually zero) is used for the failing part of the score.

# Deployment

* The application is designed to be run as a standalone Flask application.
* Can be containerized using Docker for easier deployment and scalability.
* Should be deployed on a secure server, especially when handling sensitive personal data.