WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

JD Analyz Matcher- An AI-Based Resume Analyzer and Job Matching System

Kokila G*, Mokisha A, Jane Jeba Kirubha J, Jayashree N, Kowsika S,

Student, Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology
*Tutor, Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology

ABSTRACT

Choosing suitable job opportunities and understanding how a resume aligns with job requirements can be challenging for students, fresh graduates, and job seekers. JD Analyz Matcher is a web-based resume analysis and job-matching application developed to simplify this process. The system accepts resumes in PDF, DOCX, and TXT formats and extracts readable text for analysis. It identifies technical skills using a predefined skill collection and compares resume content with available job descriptions using Term Frequency–Inverse Document Frequency (TF-IDF) vectorization and cosine similarity. Based on the calculated similarity scores, job descriptions are ranked according to textual relevance. The application also presents matching and missing skills and generates a downloadable PDF report containing the analysis results. Implemented using Python and Flask with HTML, CSS, and JavaScript, the system combines document processing, keyword-based skill identification, text-similarity analysis, and report generation in a single web application. The system is intended to reduce manual comparison effort and help candidates review potential skill gaps and job relevance. Its scores indicate textual similarity and are not a guarantee of employment.

Index Terms—Resume Analysis, Job Matching, TF-IDF, Cosine Similarity, Skill Identification, Skill-Gap Analysis, Flask, Text Processing.

I. INTRODUCTION

The use of digital recruitment platforms has increased the number of job opportunities available to candidates. However, identifying roles that align with a candidate's skills and qualifications still requires reviewing job descriptions and comparing them with resume content. This manual process can be time-consuming, particularly when several opportunities are being considered.

Keyword-based searches and manual comparisons may overlook relevant information when a resume and a job description use different wording. Candidates may also find it difficult to identify which required skills are already represented in their resumes and which are not explicitly mentioned.

JD Analyz Matcher addresses this problem through a web-based workflow that extracts resume text, identifies predefined skills, compares resume content with available job descriptions, and presents ranked results. The application uses TF-IDF and cosine similarity for textual comparison and provides matching and missing skill information. The system is designed as a decision-support tool for resume review and job discovery, not as a replacement for human assessment.

II. EXISTING SYSTEMS

Existing job portals provide searchable vacancies and filters, but candidates generally need to interpret job requirements and compare them with their own resumes. Manual resume screening can require substantial time and may produce inconsistent comparisons. Basic keyword searches may also fail to identify related terms when the wording differs.

Resume analysis tools can extract candidate information or check keywords, while job recommendation systems may rank opportunities using different matching methods. However, the usefulness of these systems depends on their data, extraction quality, and matching approach. JD Analyz Matcher focuses on a lightweight workflow that combines document text extraction, predefined skill identification, TF-IDF similarity, job ranking, and report generation.

Table 1. Comparison of common approaches and the proposed system.

Approach	Common Limitation	JD Analyz Matcher
Manual resume comparison	Time-consuming and repetitive	Automates text comparison and ranking
Keyword-based search	May miss terms not present in the search query	Uses TF-IDF representation and predefined skill checks
Standalone resume review	May not compare against several job descriptions	Compares a resume with available job descriptions
Unstructured feedback	Skill gaps may be difficult to identify	Displays matching and missing predefined skills

III. SYSTEM DESIGN AND OBJECTIVES

A. System Architecture

The proposed system is organized as a web application with a user interface, Flask backend, resume text-extraction stage, and analysis modules. The uploaded resume is processed according to its file type. The extracted text is examined for predefined skills and represented using TF-IDF. Cosine similarity compares the resume representation with available job descriptions. The resulting scores are used to rank the descriptions, while skill comparison identifies matching and missing requirements.

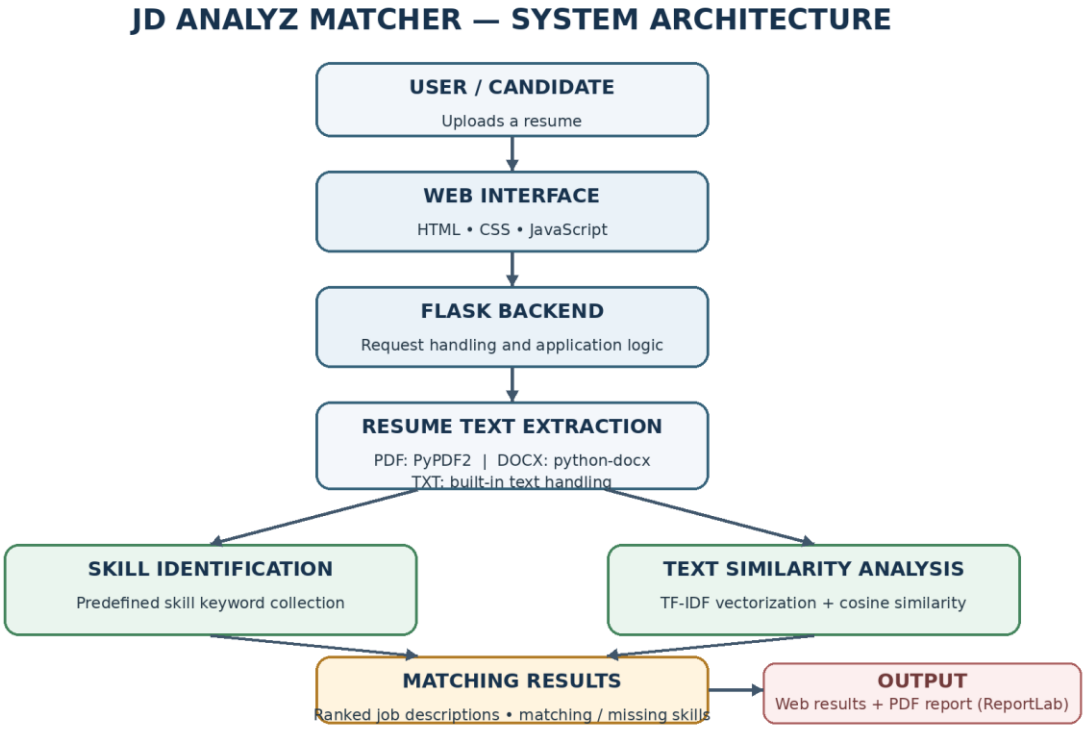


Figure 1. Proposed system architecture of JD Analyz Matcher.

B. System Workflow

The user opens the web application through a browser.

The user uploads a resume in PDF, DOCX, or TXT format.

The Flask backend receives the file and selects the corresponding text-extraction method.

The extracted resume text is checked against a predefined skill collection.

TF-IDF vectorization converts resume and job-description text into numerical representations.

Cosine similarity calculates the textual similarity between the resume and available job descriptions.

The application ranks job descriptions and displays matching and missing skills.

The results are presented through the web interface and can be exported as a PDF report.

C. Objectives

Develop a web-based application for resume analysis and job-description matching.

Extract readable text from PDF, DOCX, and TXT resume files.

Identify predefined technical skills mentioned in the resume.

Compare resume content with available job descriptions using TF-IDF and cosine similarity.

Rank job descriptions and present matching and missing skills.

Generate downloadable PDF reports to support resume review.

IV. SYSTEM DESIGN

A. High-Level Design (HLD)

At a high level, the system contains a browser-based frontend and a Flask backend. The frontend accepts a resume upload and displays the analysis output. The backend validates and processes the uploaded file, extracts text, performs skill identification and similarity calculations, and prepares the results. A PDF-generation component creates a report from the analysis output.

B. Low-Level Design (LLD)

The upload handler receives the selected document and routes it to the relevant extraction method. PDF text is read using PyPDF2, DOCX text using python-docx, and TXT content through standard Python file handling. The extracted text is then used by the skill-identification and job-matching logic.

The skill-identification logic checks the extracted text against a predefined collection of skill keywords. The text-matching component applies TF-IDF vectorization and cosine similarity to compare the resume with each available job description. The ranking stage sorts the descriptions by calculated similarity. The output stage presents the scores and skill comparisons and uses ReportLab to generate a PDF report.

Figure 2. Resume processing, matching, and report-generation workflow.

Stage	Processing	Output
Input	Resume upload (PDF / DOCX / TXT)	Uploaded resume
Extraction	PyPDF2 / python-docx / text handling	Readable resume text
Skill analysis	Predefined skill keyword matching	Detected skills
Text matching	TF-IDF and cosine similarity	Similarity scores
Result generation	Ranking and skill comparison	Ranked jobs and skill gaps
Reporting	ReportLab PDF generation	Downloadable report

V. METHODOLOGY

A. Input Data

The primary input is a user-uploaded resume in PDF, DOCX, or TXT format. The system also uses the job descriptions available within the application for comparison. The usefulness of the output depends on the readability of the resume text and the relevance and completeness of the job descriptions.

B. Resume Text Extraction

The application extracts readable text from the uploaded document. PyPDF2 is used for PDF files and python-docx is used for DOCX files, while TXT files are read as plain text. Scanned image-only PDFs may not provide extractable text without an additional OCR component, which is outside the current implementation.

C. Skill Identification

The extracted resume text is searched for terms contained in a predefined skill collection. Detected terms are recorded as skills explicitly represented in the document. The method is straightforward and efficient, but it may not recognize synonyms or skills described using wording that is absent from the collection.

D. TF-IDF Vectorization

Term Frequency–Inverse Document Frequency represents text documents as weighted numerical vectors. The term-frequency component reflects how often a term appears in a document, while inverse document frequency reduces the influence of terms that occur broadly across the document set. These vectors provide an input representation for similarity calculation.

E. Cosine Similarity and Job Ranking

Cosine similarity measures the cosine of the angle between two numerical vectors. In this application, it is used to compare the TF-IDF representation of the resume with each available job description. The job descriptions are then ordered by their calculated similarity scores. These scores indicate textual alignment; they are not probabilities of hiring or direct measures of candidate ability.

F. Skill Comparison

The application compares skills identified in the resume with the relevant skill requirements associated with job descriptions. Skills found in both are presented as matching skills, while required terms not detected in the resume are presented as missing skills. A missing term indicates that it was not detected in the resume text and does not prove that the candidate lacks the capability.

G. PDF Report Generation

After analysis, ReportLab is used to generate a PDF report containing the results. The report provides a portable record of the analysis for the user to save and review.

VI. SOFTWARE IMPLEMENTATION

JD Analyz Matcher is implemented as a Python Flask web application. The frontend is developed using HTML, CSS, and JavaScript. Python libraries support document extraction, text comparison, and PDF generation.

A. Development Environment

Programming language: Python for backend logic and analysis.

Backend framework: Flask for request handling, resume uploads, and result delivery.

Frontend: HTML, CSS, and JavaScript for the browser-based user interface.

PDF processing: PyPDF2 for extracting readable text from PDF documents.

DOCX processing: python-docx for reading DOCX document content.

Text analysis: scikit-learn for TF-IDF vectorization and cosine similarity.

Report generation: ReportLab for downloadable PDF reports.

Development tools: A compatible code editor and pip for package installation.

B. Project File Organization

File / Directory	Purpose
app1.py	Flask backend and resume-analysis logic
templates1/index1.html	HTML structure of the web interface
static/style1.css	Frontend styling and layout
static/script1.js	Client-side interactions
requirements1.txt	Python package dependencies

C. Technology Summary

Layer	Technology	Purpose
Frontend	HTML, CSS, JavaScript	User interface and interaction
Backend	Python, Flask	Application logic and request handling
Document extraction	PyPDF2, python-docx	PDF and DOCX text extraction

Layer	Technology	Purpose
Text analysis	scikit-learn	TF-IDF and cosine similarity
Reporting	ReportLab	PDF report generation

VII. SYSTEM MODULES

A. Resume Upload Module

Accepts user resumes in PDF, DOCX, and TXT formats and forwards the selected document for processing.

B. Resume Parsing Module

Extracts readable text from the uploaded file using the relevant document-processing method.

C. Skill Identification Module

Checks the extracted text against a predefined skill collection and records detected skills.

D. Job Matching Module

Compares resume and job-description text using TF-IDF vectorization and cosine similarity.

E. Ranking and Skill Comparison Module

Sorts job descriptions by similarity score and identifies matching and missing skills.

F. Results and Report Module

Displays analysis results in the web interface and creates a downloadable PDF report using ReportLab.

G. Privacy, Security, and Responsible Use

Resumes may contain personal information such as contact details, education, and employment history. Uploaded documents should be handled only for the stated analysis purpose. File-type validation, careful file handling, secure deployment configuration, and appropriate data-retention practices are important. The matching score should be treated as decision-support information and should not be used as the sole basis for recruitment or career decisions.

VIII. RESULTS AND DISCUSSION

A. Evaluation Setup

The application can be evaluated using sample resumes in the supported formats and a set of available job descriptions. Testing should verify whether text is extracted, predefined skills are identified, similarity scores are calculated, job descriptions are ranked, and PDF reports are generated. The test data and observed outcomes should be recorded to support reproducible evaluation.

B. Evaluation Measures

Text extraction success: whether readable content is retrieved from each supported format.

Skill identification precision and recall: correctness and completeness of detected predefined skills against manually reviewed labels.

Ranking relevance: whether job descriptions considered relevant by reviewers appear near the top of the ranked list.

Processing time: time required to analyze a resume and produce results.

Report generation: whether the PDF report is created and can be opened.

Usability feedback: user assessment of the clarity and ease of using the interface.

C. Expected Outcomes

The intended outcome is a functional web application that accepts common resume formats, extracts readable text, identifies predefined skills, compares the resume with available job descriptions, and displays ranked matching results with matching and missing skills. The application is also intended to generate a downloadable PDF report. Quantitative accuracy or performance claims should be made only after testing and measurement.

Table 2. Functional evaluation plan.

Evaluation Area	Suggested Check	Expected Evidence
Resume upload	Submit PDF, DOCX, and TXT files	File accepted and processed
Text extraction	Review extracted content	Readable text obtained where supported
Skill identification	Compare output with manual review	Detected predefined skill terms
Job matching	Inspect calculated similarity scores	Scores produced for available descriptions
Ranking	Review sorted descriptions	Descriptions ordered by score
PDF generation	Open generated report	Readable downloadable PDF

D. Discussion and Limitations

The system provides a lightweight method for comparing resume text with job descriptions. Its results depend on document extraction quality, the predefined skill collection, and the content of the available job descriptions. TF-IDF and cosine similarity measure textual alignment and do not fully interpret context, experience quality, or practical proficiency. Keyword matching may also miss synonyms and alternative expressions.

A missing skill in the output means that the term was not detected in the extracted resume text; it does not establish that the candidate does not possess that skill. Similarly, a high similarity score does not guarantee an interview or job offer. The system should therefore be used as an aid for resume review and job exploration.

IX. CONCLUSION

JD Analyz Matcher is a web-based resume analysis and job-matching application that combines document text extraction, predefined skill identification, TF-IDF vectorization, cosine similarity, job ranking, skill comparison, and PDF report generation. Developed using Python and Flask with standard frontend technologies, the system provides a unified workflow for comparing resume content with available job descriptions.

The application aims to reduce repetitive manual comparison and help candidates review textual alignment and potential skill gaps. Its current implementation is based on conventional text-analysis methods and therefore has limitations in semantic understanding and keyword coverage. Future work may include expanded skill vocabularies, OCR for scanned resumes, semantic text representations, improved evaluation, and stronger privacy controls. The system is intended to support user judgment rather than replace it.

ACKNOWLEDGMENT

The authors would like to thank Kokila G and the Department of Artificial Intelligence and Data Science

, Sri Shakthi Institute of Engineering and Technology, for their guidance, encouragement, and support during the development of JD Analyz Matcher. The authors also acknowledge the developers and maintainers of the open-source libraries and tools used in this project.

REFERENCES

[1] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[2] Scikit-learn Developers, “TfidfVectorizer,” Scikit-learn Documentation. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[3] Scikit-learn Developers, “Cosine Similarity,” Scikit-learn Documentation. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

[4] Pallets Projects, “Flask Documentation: File Uploads.” Available: https://flask.palletsprojects.com/en/stable/patterns/fileuploads/

[5] PyPDF2 Contributors, “PyPDF2 Documentation.” Available: https://pypdf2.readthedocs.io/en/3.x/

[6] Python Software Foundation, “Python Documentation.” Available: https://docs.python.org/3/

[7] python-docx Contributors, “python-docx Documentation.” Available: https://python-docx.readthedocs.io/en/latest/

[8] ReportLab, “ReportLab PDF Toolkit Documentation.” Available: https://docs.reportlab.com/

[9] Mozilla Developer Network, “HTML Documentation.” Available: https://developer.mozilla.org/en-US/docs/Web/HTML

[10] Mozilla Developer Network, “CSS Documentation.” Available: https://developer.mozilla.org/en-US/docs/Web/CSS

[11] Mozilla Developer Network, “JavaScript Guide.” Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide