Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 04 Issue: 07 | July-2026
ISSN (O) 3107-6696

AI Resume Screener and Student Resume Optimizer

L. Chavan
PG Department of Computer Applications
JSS College of Arts, Commerce and Science, Mysuru, India

Shwetha M R
Assistant Professor, PG Department of Computer Applications
JSS College of Arts, Commerce and Science, Mysuru, India

Abstract - Manual resume screening is slow, inconsistent, and prone to overlooking qualified candidates, while students often struggle to tailor resumes that satisfy Applicant Tracking Systems (ATS). This paper presents an AI Resume Screener and Student Resume Optimizer, a Flask-based web application that automates resume screening for HR users and provides AI-driven resume optimization for students. The system extracts resume content from PDF and DOCX files using a multi-fallback pipeline (pdfplumber, pypdf, pdfminer, and Tesseract OCR for scanned documents), identifies skills, educational qualifications, and contact details, and computes a weighted match score (skills 75%, education 20%, contact information 5%) against a given job description. HR users can screen multiple resumes simultaneously and export ranked results, while students receive personalized feedback and an ATS-friendly optimized resume generated using OpenAI GPT-4o-mini, with the original document formatting preserved. The system was implemented using Python, Flask, and SQLite, and evaluated on 15-20 unique resumes across 50-60 test runs, achieving a resume parsing success rate above 95% and a 100% authentication success rate. Results indicate that the proposed system reduces manual screening effort, improves candidate-job matching consistency, and enhances student employability through actionable, ATS-oriented feedback.

Index Terms - Applicant Tracking System, natural language processing, optical character recognition, resume optimization, resume screening.

I. INTRODUCTION

In today's competitive job market, organizations receive a large volume of applications for every opening, and Human Resource (HR) professionals spend considerable time manually reviewing resumes, comparing qualifications, and shortlisting candidates [1]. This manual process is time-consuming, labor-intensive, and susceptible to inconsistency and human error. At the same time, students and job seekers frequently struggle to present their skills and qualifications in a way that satisfies Applicant Tracking Systems (ATS) used by modern recruiters [3], reducing their likelihood of being shortlisted even when they possess relevant skills.

To address both challenges within a single platform, this work proposes an AI Resume Screener and Student Resume Optimizer. The system provides two role-based workflows: an HR workflow that screens and ranks multiple candidate resumes against a job description, and a student workflow that analyzes a single resume, identifies gaps relative to a target job description, and produces an AI-optimized, ATS-friendly version of the resume while preserving the original formatting.

The system extracts resume content from both PDF and DOCX formats using a multi-level fallback extraction strategy, falling back to Optical Character Recognition (OCR) via Tesseract when standard text extraction fails on scanned documents. Extracted skills and educational qualifications are compared against job requirements using a weighted scoring scheme, and AI-generated optimization suggestions are produced using OpenAI's GPT-4o-mini model. The remainder of this paper is organized as follows: Section II reviews related work, Section III describes the system requirements, Section IV presents the system architecture and design, Section V discusses implementation, Section VI presents testing and results, and Section VII concludes the paper with future enhancements.

II. RELATED WORK

Several studies have explored automated resume screening and recruitment systems using Natural Language Processing (NLP) and Machine Learning (ML). Sharma et al. proposed an NLP-based framework for keyword extraction and similarity analysis between resumes and job descriptions, improving shortlisting accuracy but offering no resume optimization functionality and limited format support [1]. Kumar and Verma presented a machine learning-based recruitment system using classification algorithms trained on historical recruitment data to rank candidates, though such approaches require large training datasets and produce decisions that are difficult to interpret [2].

Johnson et al. examined the role of Applicant Tracking Systems in modern recruitment, noting that ATS platforms efficiently filter large volumes of resumes but rely heavily on keyword matching, which can overlook qualified candidates whose resumes use different terminology [3]. More recent work has incorporated deep learning and transformer-based models: Jain

et al. reported approximately 94% candidate-matching accuracy using a BERT-based screening and ranking system, and Sharma and Gupta achieved 92% extraction accuracy with a deep-learning resume parser that integrated OCR for scanned documents.

Across this body of work, a consistent gap emerges: existing systems are typically designed either for recruiters or for job seekers, but rarely both. Tools that focus on ATS compatibility and candidate-side feedback generally omit recruiter-facing features such as bulk screening, candidate ranking, and screening-history management, while recruiter-facing ATS platforms provide little personalized, actionable feedback for candidates seeking to improve their resumes. Additionally, robust handling of scanned or image-based resumes remains limited in many existing platforms. The proposed system addresses this gap by integrating resume screening, candidate ranking, OCR-based extraction, and AI-driven resume optimization within a single, role-based platform.

III. SYSTEM REQUIREMENTS

The functional and non-functional requirements of the proposed system were derived from the limitations identified in existing recruitment and resume optimization tools.

A. Functional Requirements

- User registration and role-based login (HR or Student).
- Upload of resumes in PDF and DOCX formats, with OCR fallback for scanned documents.
- Extraction of candidate name, contact information, skills, and educational qualifications.
- Job description analysis to extract required skills, qualifications, and keywords.
- Weighted match-score calculation and candidate ranking for HR users.
- AI-based resume optimization and ATS-friendly resume generation for students.
- Report generation, Excel export, and screening-history management.

B. Non-Functional Requirements

The system is required to process resumes efficiently with reasonable response time, maintain consistent and accurate database records, restrict access through authenticated, role-based sessions, and follow a modular architecture that supports future scalability, including additional resume formats and AI models.

C. Development Environment

The system was developed in Python 3.x using the Flask framework, with SQLite as the database engine. Table I summarizes the principal software components used.

TABLE I. SOFTWARE ENVIRONMENT

Component	Technology
Backend Framework	Flask (Python)
Database	SQLite
Frontend	HTML, CSS, JavaScript
Text Extraction	pdfplumber, pypdf, pdfminer
OCR Engine	Tesseract (pytesseract)
Document Parsing	python-docx
AI Optimization Model	OpenAI GPT-4o-mini

IV. SYSTEM ARCHITECTURE AND DESIGN

The system follows a three-tier architecture consisting of a presentation layer, an application layer, and a database layer, as illustrated in Fig. 1.

A. Architecture Overview

The presentation layer, built with HTML, CSS, JavaScript, and Jinja templates, handles user registration, login, resume upload, job-description input, and the display of analysis results. The application layer, implemented in Flask and Python, contains the core business logic: authentication, resume parsing, OCR processing, skill and education extraction, match-score calculation, and AI-based resume optimization. The database layer uses SQLite to persist user accounts, keyword repositories, screening runs, and student submissions.

B. Resume Processing Pipeline

To maximize extraction reliability across varied resume formats, the system applies a multi-level fallback strategy. For PDF files, text is first extracted using pdfplumber; if this fails or returns insufficient text, the system falls back to pypdf, then

to pdfminer, and finally to Tesseract OCR for scanned or image-based documents. OCR is automatically triggered when the extracted text length falls below an 80-character threshold. For DOCX files, the python-docx library is used as the primary method, with ZIP/XML parsing as a fallback for malformed documents.

C. Match Score Calculation

Candidate suitability is computed using a weighted scoring function that combines skill overlap, educational match, and the presence of verifiable contact information:

$$Score = (S \times 0.75) + (E \times 0.20) + (C \times 0.05) \quad (1)$$

where S, E, and C denote the normalized skills-match, education-match, and contact-information scores, respectively, each expressed as a percentage. The final score is normalized to the range 0-100 and rounded to two decimal places before being used to rank candidates.

D. Database Design

The database schema comprises five tables: users (credentials and role), skills_keywords and education_keywords (master keyword repositories used during matching), screening_runs (HR job descriptions, extracted requirements, and ranked results), and student_submissions (resume text, extracted skills/education, match score, and AI-generated feedback for each student session).

V. IMPLEMENTATION

The system was implemented as a Flask web application with two independent, role-based workflows, summarized below and in Table II.

A. HR Workflow

After authentication, an HR user submits a job description and uploads one or more candidate resumes. The job-description analysis module extracts required skills, qualifications, and keywords, which are then compared against each candidate's extracted profile. Match scores are computed using (1), candidates are ranked accordingly, and results can be exported as an Excel report or reviewed directly in the dashboard. The dashboard also retains a screening history so that HR users can revisit previous job postings and their corresponding ranked candidate lists.

B. Student Workflow

A student uploads a single resume and a target job description. The system extracts the resume content, identifies matched and missing skills and qualifications, and computes a match score. The relevant resume sections are then sent to the OpenAI GPT-4o-mini model, which returns optimized content emphasizing ATS-relevant keywords, stronger action-oriented phrasing, and improved structure. The optimized content is inserted back into the original DOCX while preserving fonts, bullet styles, tables, and spacing, and the result is made available for preview, editing, and download.

TABLE II. MODULE-WISE FUNCTIONALITY

Module	Function
Authentication	Registration, login, role-based access
Resume Upload	PDF/DOCX validation and storage
Parsing/OCR	Multi-fallback text extraction
Skill/Education	Keyword-based information extraction
Scoring	Weighted match-score calculation
Optimization	GPT-4o-mini resume enhancement
Reporting	Excel export, resume download

VI. RESULTS AND DISCUSSION

The system was evaluated using approximately 15-20 unique resumes across 50-60 processing runs, covering login, upload, extraction, scoring, and AI-optimization scenarios. Table III summarizes the key performance outcomes.

TABLE III. PERFORMANCE SUMMARY

Metric	Result
Authentication success rate	100%
Resume parsing success rate	95%+

PDF / DOCX support	Yes / Yes
OCR fallback support	Yes
Skill/education extraction	High accuracy
AI optimization success	High

Sample screening results, summarized in Table IV, show that candidates with higher skills-match percentages consistently received proportionally higher final scores, confirming that the weighting scheme (75% skills, 20% education, 5% contact information) produces an intuitive and consistent ranking.

TABLE IV. SAMPLE SCREENING RESULTS

Candidate	Skills (%)	Score
A	90	92.5
B	80	85.0
C	70	77.5
D	60	65.0
E	50	57.5

The multi-fallback extraction pipeline proved effective at handling heterogeneous resume formats: documents that failed under pdfplumber were frequently recovered by pypdf or pdfminer, and the OCR fallback successfully extracted text from scanned PDFs that contained no machine-readable text layer. On the student side, the AI optimization module consistently preserved the original document's fonts, tables, and bullet structure while improving keyword alignment with the target job description, and qualitative feedback (e.g., missing skills, suggested certifications, and recommendations to quantify project outcomes) was generated for every test resume.

A limitation observed during testing is that OCR accuracy decreases for low-resolution scanned documents, and the keyword-based matching approach, while effective and transparent, does not capture deeper semantic relationships between differently worded but equivalent skills (e.g., "ML" versus "Machine Learning"). These limitations motivate the future enhancements discussed in Section VII.

VII. CONCLUSION AND FUTURE WORK

This paper presented an AI Resume Screener and Student Resume Optimizer that unifies recruiter-facing resume screening and candidate-facing resume optimization within a single Flask-based platform. By combining a multi-fallback text-extraction pipeline, OCR support for scanned documents, a transparent weighted scoring model, and GPT-4o-mini-based resume enhancement, the system reduces manual screening effort for HR users while giving students actionable, ATS-oriented feedback. Testing across 15-20 resumes and 50-60 runs demonstrated a resume-parsing success rate above 95% and reliable, format-preserving resume optimization.

Future work will focus on incorporating semantic and embedding-based matching to capture contextual relationships between skills beyond exact keyword overlap, extending support to multilingual resumes, integrating with external job portals such as LinkedIn, and deploying the system on cloud infrastructure to support larger-scale recruitment scenarios with real-time analytics dashboards.

ACKNOWLEDGMENT

The authors thank the PG Department of Computer Applications, JSS College of Arts, Commerce and Science, Mysuru, for the support and resources provided during the development of this project.

REFERENCES

[1] P. Sharma, R. Gupta, and S. Verma, “Automated resume screening using natural language processing,” Int. J. Comput. Appl., 2022.

[2] A. Kumar and P. Verma, “Intelligent recruitment system using machine learning,” 2021.

[3] M. Johnson, K. Smith, and R. Brown, “Applicant tracking systems and resume ranking techniques,” 2020.

[4] R. Jain et al., “AI-powered resume screening and candidate ranking system,” LinkedIn Resume Dataset Study, 2024.

[5] S. Sharma and A. Gupta, “Intelligent resume parser using deep learning,” Real-world Resume Dataset Study, 2024.

[6] Y. Wang et al., “Automated recruitment using NLP techniques,” Kaggle Resume Dataset Study, 2023.

[7] A. Kumar et al., “Resume classification and job matching system,” Resume Screening Dataset Study, 2023.

[8] S. Chen, X. Wang, and Y. Zhao, "OCR-based resume information extraction," Scanned Resume Dataset Study, 2021.