

Project Report: AI Resume Matcher & Candidate Ranking System



This is our project. Create a detailed project report for this project for the team of 5.

Team Size: 5 Members Date: Friday, July 18, 2025

I. Project Overview

The project aims to develop an AI-driven Resume Matcher and Candidate Ranking System using Streamlit. This tool enables HR teams and recruiters to upload multiple resumes and a job description, and automatically match and rank candidates based on resume-job relevance using natural language processing and machine learning[1]. The system streamlines recruitment, reducing manual effort and bias by providing a data-driven candidate shortlist.

II. Project Objectives

- Automate resume parsing and information extraction from PDF/DOCX files.
- Compare candidate profiles with job descriptions using AI-driven similarity scoring.
- Rank and filter candidates based on match percentage.
- Provide a user-friendly interface for uploading files and viewing results.
- Allow resume download for top-matched candidates.

3. Team Composition and Roles

Krish Tare

Management, Teamwork, Leadership, basic Python, Excel; oversees timeline, coordinates team [\[2\]](#)

Nayan Pandey

Python, scikit-learn, data analysis, model building, integration of AI/ML modules [\[3\]](#)

Abhishek Gupta

Streamlit, HTML/CSS, basic JS, UI/UX, ensures intuitive user flow [\[4\]](#)

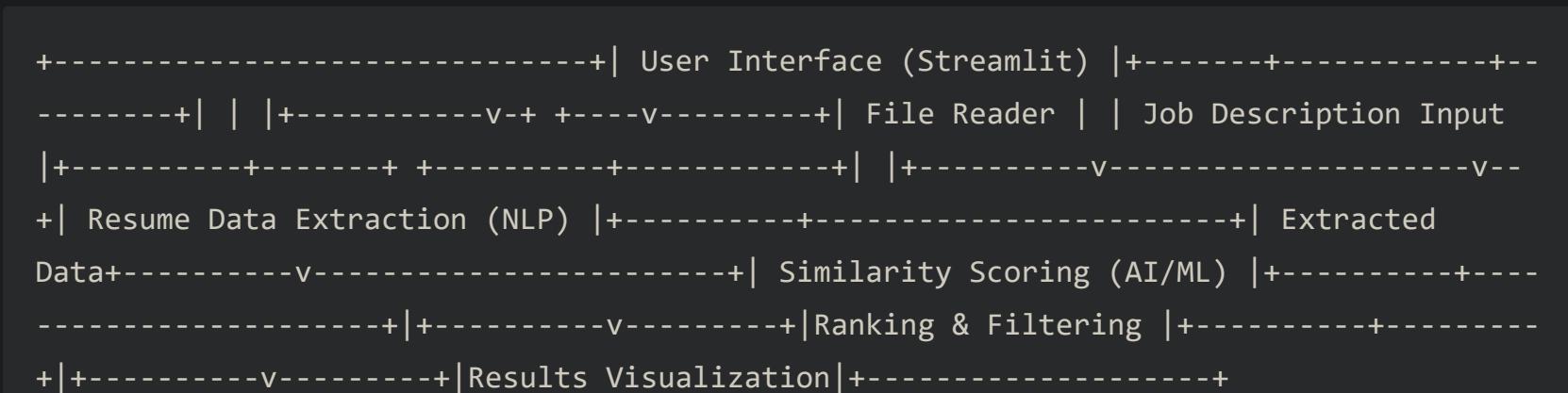
Khizar Saifee

Resume parsing QA, test case creation, assist with data extraction functions

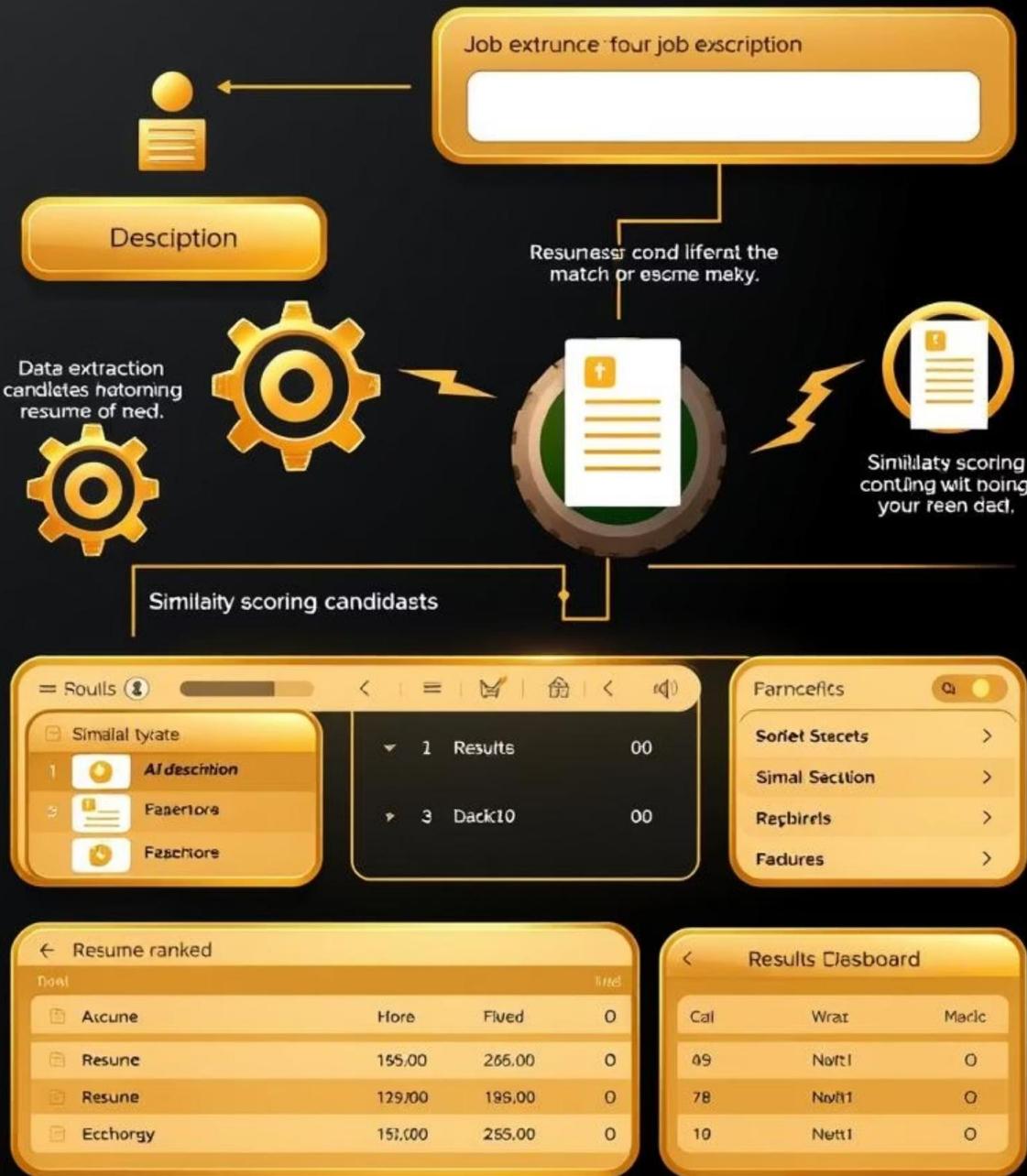
Arshad Shaikh

Writes documentation, manages requirements, supports deployment and maintenance

4. System Architecture



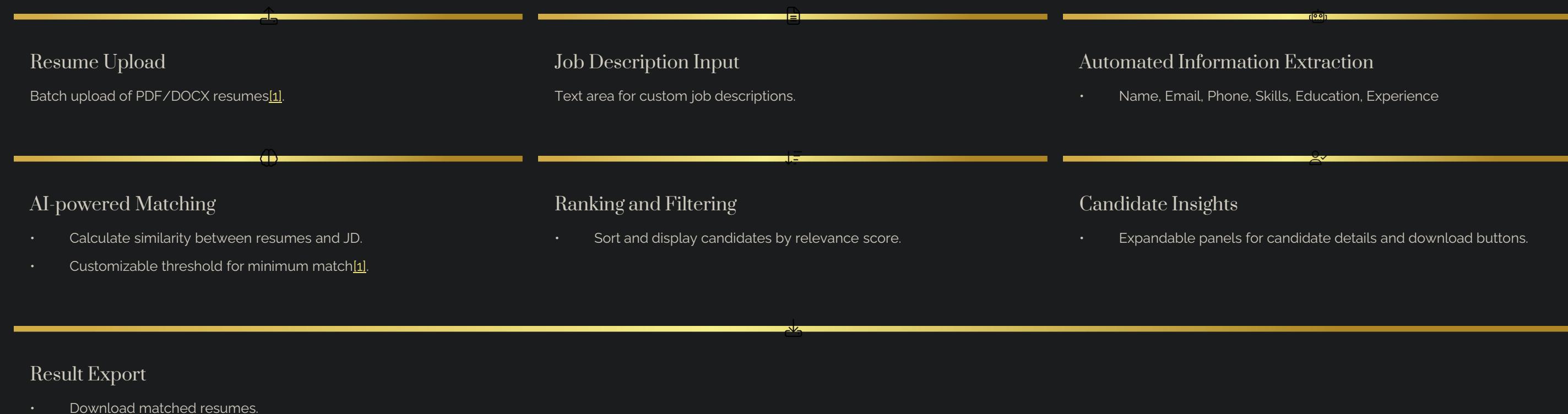
AI Resume Matcher Candidate Ranking



5. Technology Stack

- **Programming Language:** Python 3.x
- **Web Framework:** Streamlit
- **Key Libraries:**
 - PyPDF2, docx2txt (file reading)
 - pandas, numpy (data handling)
 - scikit-learn (vectorization, similarity scoring)
 - spaCy (NLP, entity extraction)
 - matplotlib (visualization, optional)[\[5\]](#)
- **Deployment:** Local, Streamlit Cloud, or internal server

6. Project Features



7. Implementation Details

1

Resume Parsing

Utilizes PyPDF2 and docx2txt to extract raw text from resumes[1][5]. spaCy analyzes the text to identify structured information.

2

JD Matching

scikit-learn vectorizes the text (possibly TF-IDF or CountVectorizer), then computes a similarity score (e.g., cosine similarity) between the resume data and job description[1][5].

3

Frontend Workflow

Built with Streamlit, providing drag-and-drop file upload, threshold slider, and results dashboard[1].

4

Configuration

All required dependencies are listed in the requirements.txt file[5].

8. Project Timeline (Sample/Indicative)

Requirement Analysis	1 week	Functional/technical spec, user flows
Initial Setup & Planning	1 week	Repo, environment, task breakdown
Resume Parsing Module	2 weeks	Working text extraction & NLP scripts
JD Matching Module	1 week	Similarity scoring prototype
Frontend Development	1 week	Streamlit interface
Integration & Backend	2 weeks	End-to-end system
Testing & QA	1 week	Bug fixes, functional/performance QA
Documentation & Deployment	1 week	User manual, deployment on Streamlit
Buffer/Misc	1 week	

9. Testing Plan

- Upload mixed-format resumes (PDF/DOCX).
- Test with different job descriptions.
- Validate extracted information for accuracy.
- Check ranking consistency and download functionality.
- Edge cases: corrupted files, missing fields.

10. Challenges & Risk Mitigation

File format inconsistencies

Extensive input validation and error handling in file parsing.

Resume template diversity

Use robust NLP models for flexible entity extraction.

Performance

Employ async processing & optimize matching for larger datasets.

Data privacy

Do not store resume data after processing; advise users on secure handling.

11. References to Team Experience

The team brings a range of competencies:

- Data science, Python programming, AI/ML, web/app development, and presentation skills [2][3][4].
- Prior experience in dashboarding, prediction models, and AI assistant tools ensures readiness for both back-end and UX/UI tasks.

12. Future Enhancement Ideas

- Support for additional file types (e.g., LinkedIn exports).
- Advanced analytics: heatmaps, candidate benchmarking.
- Automated recommendation feedback to applicants.

This structured approach and division of responsibilities will empower the team to deliver a robust, scalable solution that meets modern recruitment needs.