Dhirubhai Ambani University
Technology

Formerly DA-IICT

PC649 – Summer Internship Report

Resume Ranker - AI Powered Recruitment Tool

Thanki Kishan Chetan
M.Sc. (IT) 202412117
SID 202412117

Dhirubhai Ambani Institute of Information and Communication Technology
Gandhinagar, Gujarat – 382001
August 2024

Resume Ranker – AI-Powered Recruitment

Internship Report

1. Introduction

Recruiters often struggle with efficiently evaluating large volumes of resumes for a single job opening. The Resume Ranker solves this problem by parsing resumes, extracting structured data (skills, experience, education, etc.), and ranking candidates against a job description using a multi-factor scoring system. This improves hiring efficiency, reduces manual efforts, and enhances the quality of candidate shortlisting.

2. Functional and Non-Functional Requirements

2.1 Functional Requirements

- Resume Parsing: Extracts structured data (skills, experience, education, contact info).
- Job Description Parsing: Extracts relevant skills and requirements.
- Ranking Algorithm: Ranks resumes based on:
    - Skills match (50%)
    - Text similarity (30%)
    - Experience match (20%)
- API & Frontend Interface:
    - Upload resumes and job descriptions.
    - Export candidate rankings and detailed scoring.

2.2 Non-Functional Requirements

- Performance: Resume parsing under 3 seconds.
- Accuracy: 94% skill matching accuracy.
- Scalability: Handles hundreds of resumes simultaneously.

- Maintainability: Modular and clean codebase for future improvements.
- Deployment: Easy setup with minimal configuration.

3. Methodology / Processes

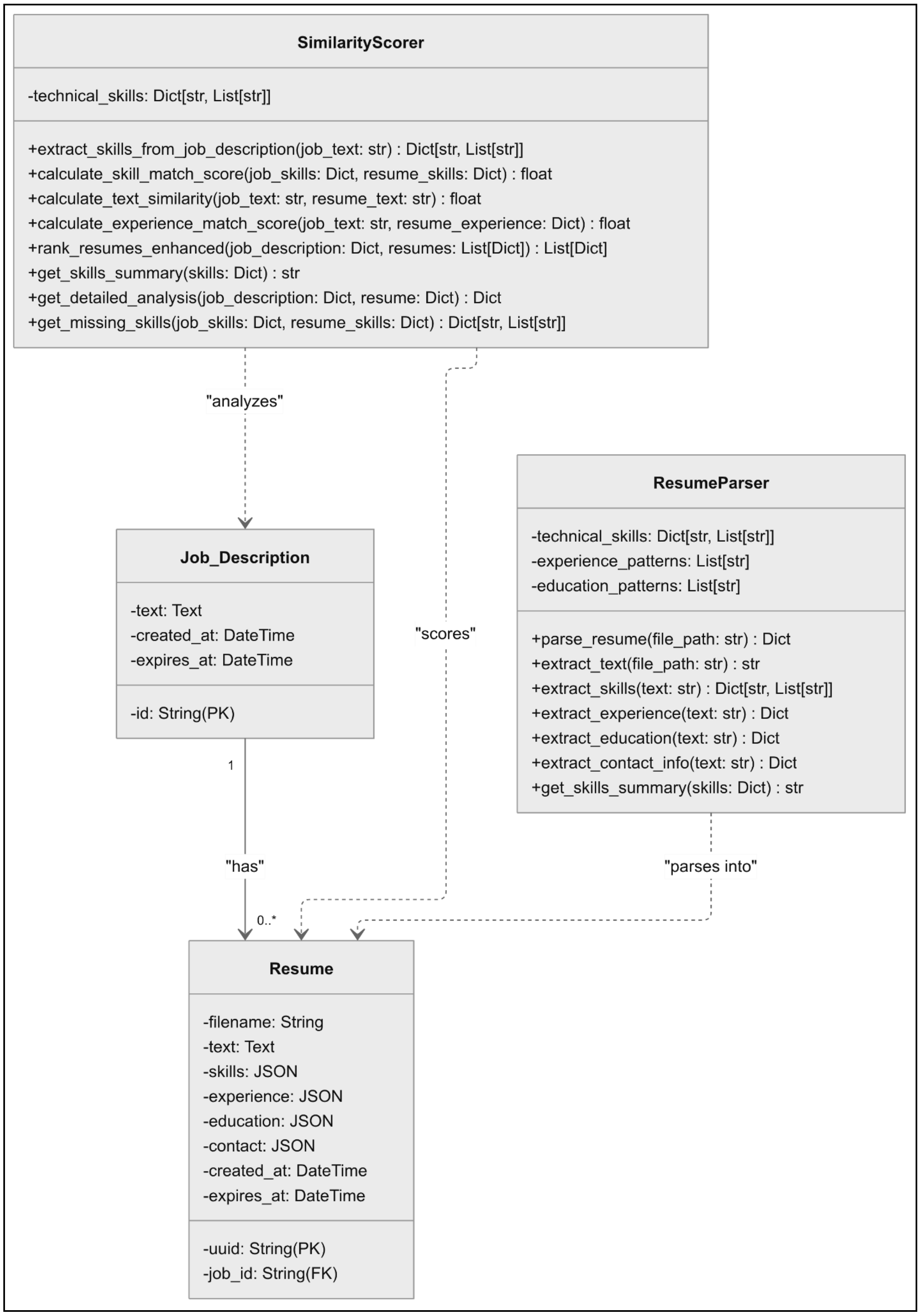
The project followed an iterative development process, inspired by Agile methodology, where features were developed and tested in focused sprints.

- **Requirement Gathering:** Identified key features such as resume parsing, job description analysis, and candidate ranking.
- **Design:** Structured data models and scoring logic were planned based on FastAPI architecture.
- **Development:** Built features incrementally—starting with resume parser, then JD parser, scoring engine, and API endpoints.
- **Testing:** Each module was manually tested after development; API responses and scoring accuracy were validated using sample data.
- **Deployment Setup:** Configured API routes, and deployment on Render.

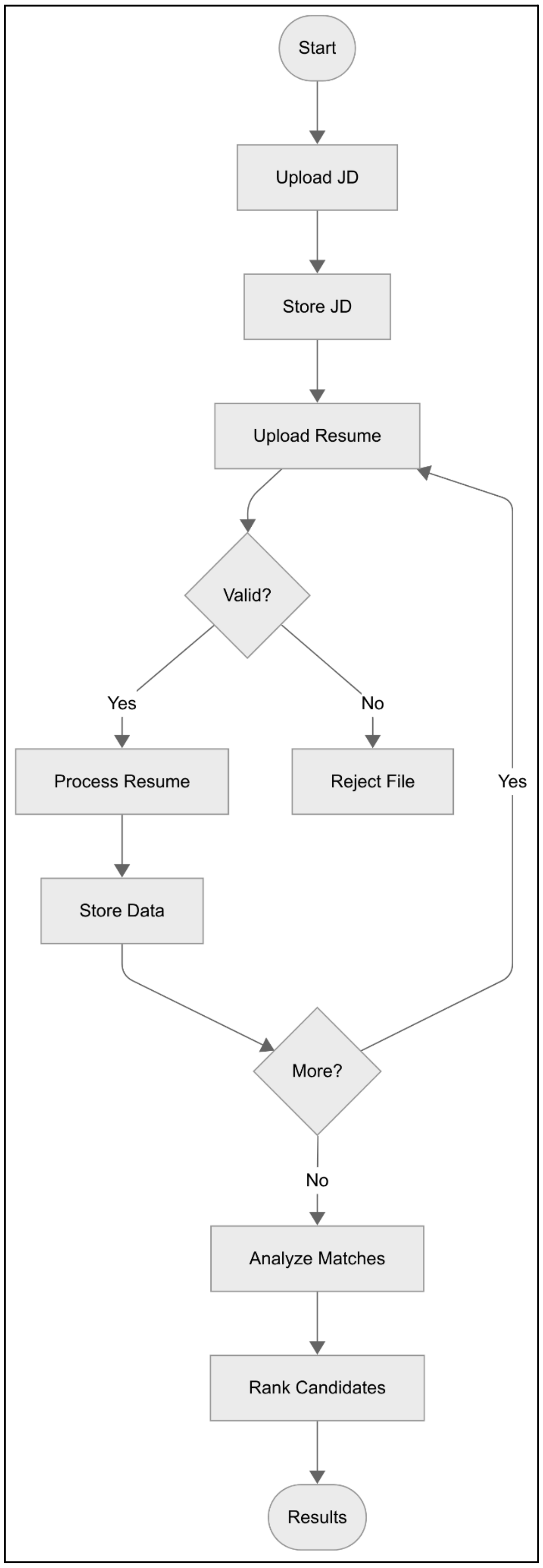
Process Flow: Resume Upload → NLP Parsing → Skill Mapping → Similarity Scoring → Candidate Ranking → Export

4. Design

4.1 Class Diagram


SimilarityScorer
-technical_skills: Dict[str, List[str]]
+extract_skills_from_job_description(job_text: str) : Dict[str, List[str]]
+calculate_skill_match_score(job_skills: Dict, resume_skills: Dict) : float
+calculate_text_similarity(job_text: str, resume_text: str) : float
+calculate_experience_match_score(job_text: str, resume_experience: Dict) : float
+rank_resumes_enhanced(job_description: Dict, resumes: List[Dict]) : List[Dict]
+get_skills_summary(skills: Dict) : str
+get_detailed_analysis(job_description: Dict, resume: Dict) : Dict
+get_missing_skills(job_skills: Dict, resume_skills: Dict) : Dict[str, List[str]]
"analyzes"
"scores"
Job_Description
-text: Text
-created_at: DateTime
-expires_at: DateTime
-id: String(PK)
ResumeParser
-technical_skills: Dict[str, List[str]]
-experience_patterns: List[str]
-education_patterns: List[str]
+parse_resume(file_path: str) : Dict
+extract_text(file_path: str) : str
+extract_skills(text: str) : Dict[str, List[str]]
+extract_experience(text: str) : Dict
+extract_education(text: str) : Dict
+extract_contact_info(text: str) : Dict
+get_skills_summary(skills: Dict) : str
1
"has"
0..*
"parses into"
Resume
-filename: String
-text: Text
-skills: JSON
-experience: JSON
-education: JSON
-contact: JSON
-created_at: DateTime
-expires_at: DateTime
-uuid: String(PK)
-job_id: String(FK)

4.2 Sequence Diagram



5. Coding and Frameworks

Tech Stack

- Backend: Python, FastAPI (REST API), SQLAlchemy (ORM), SpaCy (NLP)
- Frontend: Streamlit (recruiter-facing UI)

Key Features

- Resume parser with 7 skill categories
- Weighted multi-factor similarity scoring
- API Endpoint: GET /ranker/resume-analysis/{job_id}/{resume_uuid}

6. Testing

Functional Testing Outcomes:

- **Resume Parsing**
  Ensured that resumes in various formats (PDF, DOCX) were correctly parsed. Extracted fields like name, contact info, skills, experience, and education were validated for accuracy and completeness.

- **Skill Extraction**
  Tested the mapping of extracted skills from resumes and job descriptions against a predefined skill set. Verified correct categorization into 7 distinct skill categories.

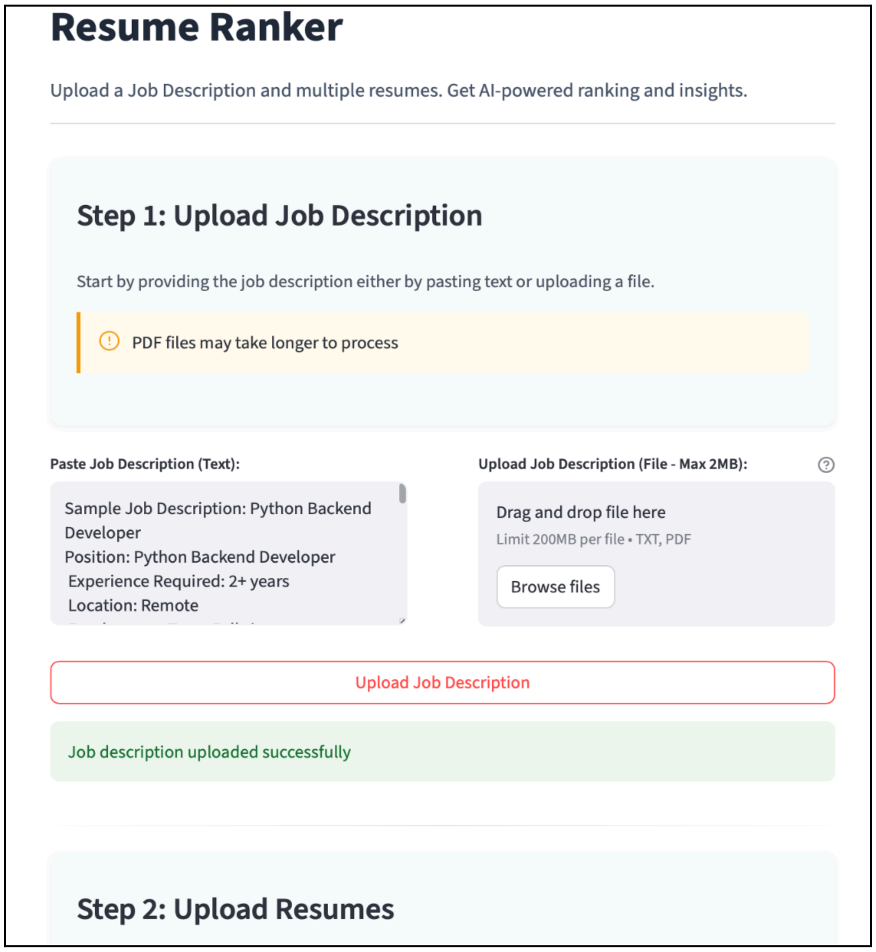
- **Similarity Scoring**
  Verified that the scoring logic accurately weighted skills (50%), text similarity (30%), and experience (20%) across multiple test resumes. Results were cross-checked against expected rankings.

- **API Responses**
  All API endpoints were tested using tools like Postman. Confirmed correct request handling, data validation, and structured JSON responses for different job and resume inputs.

7. Snapshots

- **GitHub Repository:** Resume-Ranker GitHub
- **Live Demo:** https://resume-ranker-n5l1.onrender.com/
- **Screenshots:**

Job Description Input:



Resume Ranker

Upload a Job Description and multiple resumes. Get AI-powered ranking and insights.

Step 1: Upload Job Description

Start by providing the job description either by pasting text or uploading a file.

PDF files may take longer to process

Paste Job Description (Text):

Sample Job Description: Python Backend Developer
Position: Python Backend Developer
Experience Required: 2+ years
Location: Remote

Upload Job Description (File - Max 2MB):

Drag and drop file here
Limit 200MB per file • TXT, PDF

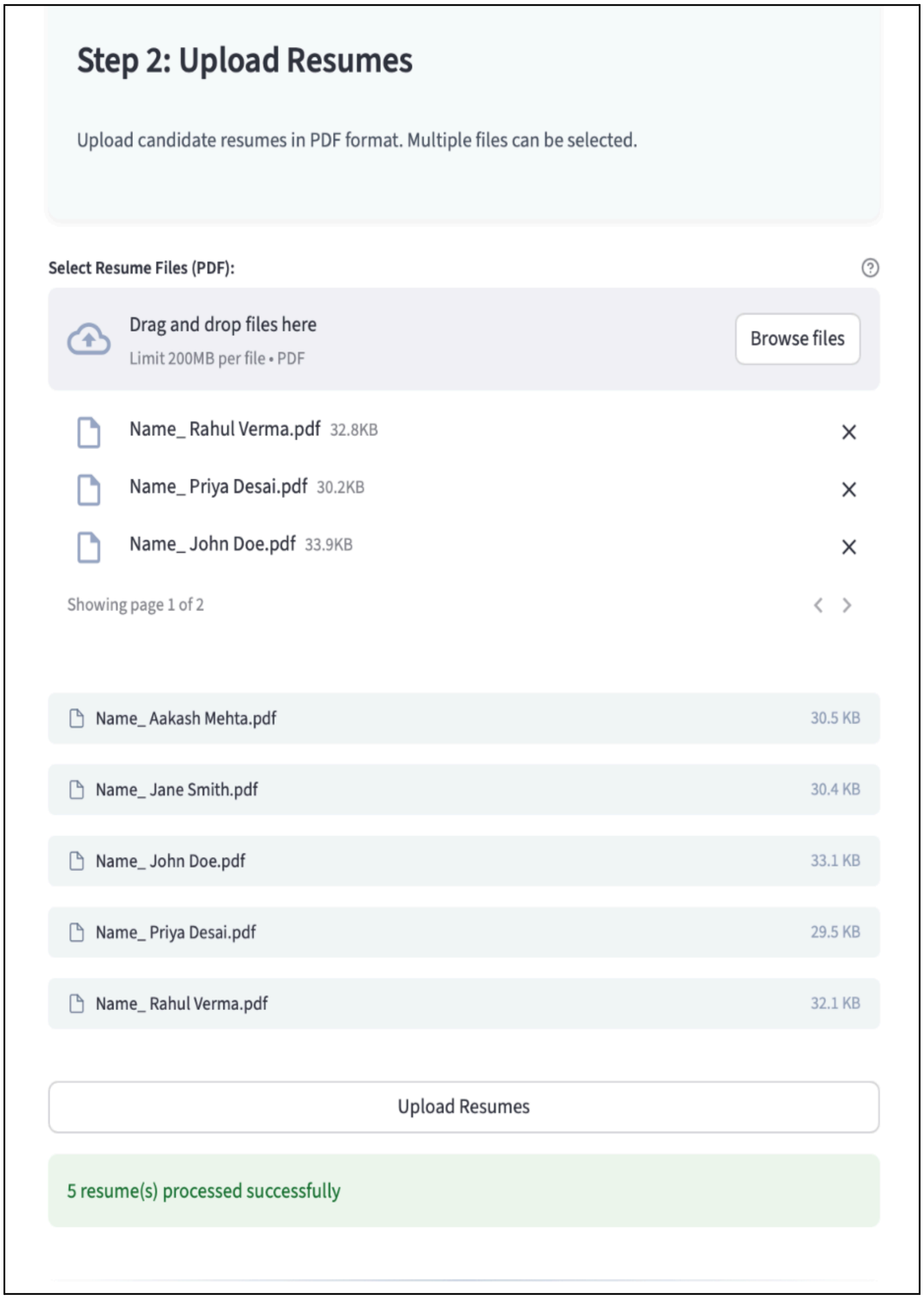
Browse files

Upload Job Description

Job description uploaded successfully

Step 2: Upload Resumes

Resume Upload Interface:



Step 2: Upload Resumes

Upload candidate resumes in PDF format. Multiple files can be selected.

Select Resume Files (PDF):

Name_ Rahul Verma.pdf 32.8KB

Name_ Priya Desai.pdf 30.2KB

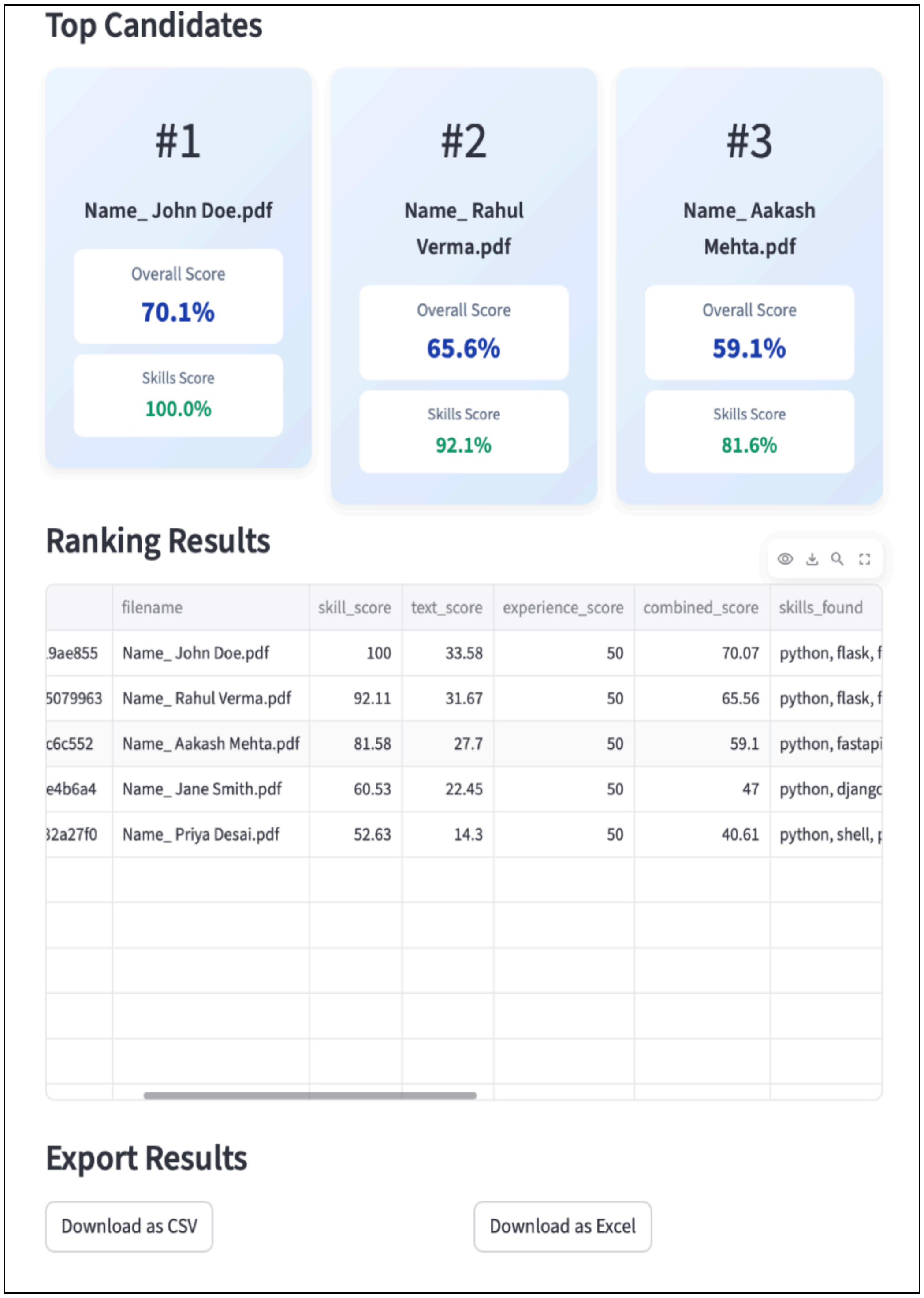
Name_ John Doe.pdf 33.9KB

Showing page 1 of 2

Name_ Aakash Mehta.pdf	30.5 KB
Name_ Jane Smith.pdf	30.4 KB
Name_ John Doe.pdf	33.1 KB
Name_ Priya Desai.pdf	29.5 KB
Name_ Rahul Verma.pdf	32.1 KB

5 resume(s) processed successfully

Ranked Output with Scoring Breakdown:



Top Candidates

#1	#2	#3
Name_ John Doe.pdf	Name_ Rahul Verma.pdf	Name_ Aakash Mehta.pdf
Overall Score 70.1%	Overall Score 65.6%	Overall Score 59.1%
Skills Score 100.0%	Skills Score 92.1%	Skills Score 81.6%

Ranking Results

	filename	skill_score	text_score	experience_score	combined_score	skills_found
.9ae855	Name_ John Doe.pdf	100	33.58	50	70.07	python, flask, f
5079963	Name_ Rahul Verma.pdf	92.11	31.67	50	65.56	python, flask, f
c6c552	Name_ Aakash Mehta.pdf	81.58	27.7	50	59.1	python, fastapi
e4b6a4	Name_ Jane Smith.pdf	60.53	22.45	50	47	python, djangc
32a27f0	Name_ Priya Desai.pdf	52.63	14.3	50	40.61	python, shell, p

Export Results

8. Summary

The Resume Ranker addresses a critical need in the recruitment process by automating and enhancing resume screening using AI-powered techniques. It simplifies candidate evaluation by providing:

- Structured and accurate resume parsing
- Intelligent skill and experience matching
- Multi-factor scoring based on recruiter-defined priorities
- Faster and more objective shortlisting for hiring teams

9. Lessons Learnt

- Importance of writing clean, modular, and extensible code for real-world scalability
- Practical application of NLP tools like SpaCy for extracting structured data from unstructured resumes
- Benefits of designing an API-first architecture for integration and reusability
- Deeper understanding of recruiter pain points and how AI can be leveraged to solve them effectively