

AI Recruitment Platform Documentation

Project Overview

The AI Recruitment Platform is an end-to-end job matching system that leverages Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) to match candidate resumes with job descriptions (JDs). The platform processes PDF resumes to extract structured data, processes JDs to store in MongoDB Atlas, and recommends the top 5 matching jobs based on skills and experience. The system is built with Python, uses the Google Gemini API for data extraction, `sentence-transformers` for embeddings, and Streamlit for a modern, user-friendly interface.

Project Features

Resume Processing:

- Upload PDF resumes via the Streamlit interface.
- Extracts structured data (name, skills, experience, education) using the Google Gemini 2.5 Flash API.
- Outputs data in JSON format for display and further processing.

Job Description Processing:

- Generates 200 realistic JDs.
- Processes JDs to extract structured data (title, responsibilities, skills, experience, location, company, education) using the Gemini API.
- Stores processed JDs in MongoDB Atlas (`recruitment_platform.job_descriptions`).

Job Matching:

- Uses sentence-transformers (`all-MiniLM-L6-v2`) to generate embeddings for resume and JD data.
- RAG with cosine similarity to recommend the top 5 matching JDs based on skills and experience.

User Interface (UI)

The Streamlit UI is designed for simplicity and ease of use.

Layout:

- Two-Block Design:
 - Left Block: Resume upload section with a file uploader for PDF resumes, displaying extracted JSON data.
 - Right Block: Job search section with a "Find Jobs" button, a table of top 5 recommended JDs, a CSV download button, and a dropdown to view full JD details.
- Responsive Columns: Uses `st.columns([1, 1], gap="medium")` for balanced spacing.

Styling:

- Background: Gradient from light blue (`#E6F0FA`) to white (`#FFFFFF`).
- Blocks: White containers with rounded corners and subtle shadows (`box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1)`).
- Text: Dark blue (`#003087`) for headings, markdown, and text.
- Buttons: Light blue (`#E6F0FA`) with dark blue borders and text, darkening on hover (`#D1E0F5`).
- Table: Clean, bordered table for job recommendations with a light blue header.

User Flow:

1. Upload a PDF resume in the left block.
2. View extracted resume data (JSON) below the uploader.
3. Click "Find Jobs" in the right block to see the top 5 job recommendations.
4. Download recommendations as a CSV or select a job to view its full JSON details.

System Flow and Components

System Flow:

The platform operates in three main stages:

1. JD Generation:

- Script: `generate_jds.py`.

- Process: Uses the Google Gemini 2.5 Flash API to generate 200 realistic JDs, saved as `job_descriptions.json`.

2. JD Processing:

- Script: `process_jds.py`.

- Process: Reads `job_descriptions.json`, extracts structured data using the Google Gemini 2.5 Flash API, and stores it in MongoDB Atlas.

3. Resume Processing and Job Matching:

- Script: `app.py`.

- Process:

- Uploads a PDF resume via Streamlit.

- Extracts text using `PyPDF2` and processes it with the Google Gemini 2.5 Flash API to obtain structured JSON data (name, skills, experience, education).

- Generates embeddings for resume data and JDs using sentence-transformers.

- Computes cosine similarity to recommend the top 5 matching JDs.

- Displays results in a table, with options to download as CSV or view full JD details.

Components:

Frontend:

- Streamlit: Provides the web interface for resume upload, job recommendations, and JD viewing.

- CSS: Custom styles for background, blocks, and buttons.

Backend:

- Google Gemini API: Extracts structured data from resumes and JDs (`gemini-2.5-flash` model).

- MongoDB Atlas: Stores processed JDs in the collection.

- PyPDF2: Extracts text from PDF resumes.

- sentence-transformers: Generates embeddings (`all-MiniLM-L6-v2`) for RAG-based job matching.

- scikit-learn: Computes cosine similarity for ranking JDs.

Scripts:

- `generate_jds.py`: Generates 200 JDs.
- `process_jds.py`: Processes JDs and store them in MongoDB.
- `test_mongo.py`: Tests MongoDB connectivity.
- `app.py`: Main application for resume processing and job matching.

Architecture:

User: Interacts with Streamlit UI.

-Streamlit UI:

- Resume Upload: Sends PDF to `app.py`.
- Job Search: Triggers job recommendations and displays results.

- Resume Processing:

- Input: PDF resume.
- Process: `PyPDF2` extracts text → Google Gemini 2.5 Flash API extracts JSON (name, skills, experience, education) → `sentence-transformers` generates resume embedding.

- JD Processing:

- Input: `job_descriptions.json` (from `generate_jds.py`).
- Process: Google Gemini 2.5 Flash API extracts structured JD data → MongoDB Atlas stores JDs → `sentence-transformers` generates JD embeddings.

- Job Matching:

- Process: Cosine similarity between resume and JD embeddings → Top 5 JDs returned.
- *MongoDB Atlas*: Stores processed JDs.
- *Output*: Table of recommendations, CSV download, full JD JSON display.

Deployment

The application is deployed on Streamlit Cloud.

Deployed URL:

- <https://ai-recruitment-platform-az8moar9nrxnahrteju6dqy.streamlit.app/>

The screenshot displays the 'AI Recruitment Platform' interface. It features a top navigation bar with the title 'AI Recruitment Platform'. Below this, the interface is divided into two main sections. The left section, titled 'Upload Your Resume', includes a file upload area with a 'Drag and drop file here' instruction, a 'Browse files' button, and a preview of a resume titled 'machine-learning-resume-example.pdf'. Below the upload area, the 'Extracted Resume Data' is shown as a JSON object:

```
{  "name": "Atticus Hawthorn"  "skills": [    0: "Pandas"    1: "Featuretools"  ]}
```

. The right section, titled 'Find Matching Jobs', contains a 'Find Jobs' button and a 'Recommended Jobs' table. The table lists five job recommendations with columns for Title, Company, Location, and Match Score. At the bottom of the right section, there is a 'Download Results as CSV' button and a 'Manage app' link.

AI Recruitment Platform

Upload Your Resume

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

machine-learning-resume-example.pdf

Extracted Resume Data

```
{  "name": "Atticus Hawthorn"  "skills": [    0: "Pandas"    1: "Featuretools"  ]}
```

Find Matching Jobs

Find Jobs

Recommended Jobs

	Title	Company	Location	Match Score
0	Senior ML Engineer	Analytix Inc.	New York, NY	52.82%
1	AI Research Scientist	NextGen Solutions	Toronto, Canada	51.70%
2	Senior ML Engineer	NextGen Solutions	Bangalore, India	50.66%
3	Senior Data Scientist	Analytix Inc.	Boston, MA	48.46%
4	Deep Learning Specialist	CloudPeak	London, UK	48.36%

Download Results as CSV

Manage app