🤖 ResumeScreening Agent – Rank Resumes by Job Description

🔍 Resume Screening Tool 🧰 Project & AI Stack

🧠 ResumeScreening Agent – Project Overview

This project demonstrates an **AI-powered Resume Screening Agent** that:

- Takes a **Job Description (JD)** as input
- Accepts multiple **candidate resumes** (PDF/DOCX/TXT)
- Ranks resumes based on **semantic similarity** + **keyword coverage**
- Shows per-candidate highlights for explanations

🔗 Target AI Stack (Conceptual Design)

Even though this notebook uses local Python + TF-IDF, the project is **designed** to plug into a modern AI stack:

🤖 AI Models

- **OpenAI GPT** – main reasoning engine (fit scoring, explanations)
- **Claude** – great for long resumes and multi-page JDs
- **Gemini** – ideal if heavily integrated with Google Workspace

🧰 Frameworks

- **LangChain** – tools, prompts, chains (JD parsing, scoring tools)
- **CrewAI** – multi-agent setup (Screening Agent + Interview Agent, etc.)
- **LlamaIndex** – building RAG over historical candidate data & job archives

🗂️ Vector Databases

- **Pinecone** – managed, scalable semantic search for resumes & JDs
- **ChromaDB / Weaviate** – flexible, self-hosted or managed options
- **FAISS** – fast local vector search for prototypes / small teams

🖥️ UI Layer

- **Streamlit** – internal recruiter dashboard (upload JDs/resumes, view rankings)
- **Gradio** – quick demo UI (what you're using now)
- **HTML/JS** – public-facing preview / portfolio website

🗄️ Databases

- **Firebase / Supabase** – auth, logs, recruiter accounts, audit trails
- **Notion DB** – store candidate pipelines and interview notes as tables
- **Google Sheets** – simple ATS for smaller teams / college projects

🌐 APIs & Integrations

- **Google Calendar** – automatically schedule interviews for shortlisted candidates
- **Calendly** – send scheduling links to candidates
- **Notion / Sheets** – sync ranked candidate lists
- **Zapier** – trigger Slack/Email notifications from shortlist events
- **Shopify** – (optional) sync hires or roles for e-commerce teams

---

🚀 How to Present This as a Project

You can describe your system as:

1. **Frontend** – Streamlit/Gradio app where HR pastes JD and uploads resumes.
2. **Backend** – Python + LangChain/CrewAI orchestrating:
    - JD parsing (LLM)
    - Resume embedding (vector DB)
    - Scoring & ranking (LLM + similarity)
3. **Data Layer** – Pinecone/ChromaDB/FAISS for embeddings, plus Notion DB / Sheets for pipeline.
4. **Automation Layer** – Zapier + Google Calendar + Calendly for interview scheduling.

This notebook gives you a **working core (ranking logic + UI)** that you can later extend with actual API keys and real LLM calls.