

One-Pager: Autonomous Resume Screening Agent

– Use Case & Impact

Project Title:

Autonomous Resume-Screening Agent for HR Teams






Use Case Overview

Hiring managers spend hours manually scanning resumes for each open role. To streamline this process, we developed an autonomous AI agent that reads multiple resumes from a folder, compares them against a given job description, and outputs the top 3 most relevant candidates, along with GPT-based reasoning — saving time, increasing precision, and mimicking a human recruiter's thinking.







Agent Workflow (What the Agent Does)

- ✓ Accepts Job Description: Reads text input from `job_description.txt`
- ✓ Scans Resumes Folder: Reads `.pdf` resumes from the `resumes/` folder
- ✓ Parses Resume Text: Extracts content using `PyPDF2`
- ✓ Analyzes with GPT: Sends JD + resumes to OpenAI GPT (3.5/4) for intelligent matching
- ✓ Ranks & Justifies: GPT ranks top 3 matches and provides explanations
- ✓ Saves Output: Result is saved to `ranked_output.txt` and shown in terminal

Agentic Features Implemented

-  Looping – Automatically loops through all resumes in the folder
-  Reasoning – Uses GPT to evaluate resumes against the JD with human-like logic
-  Tool Usage – Integrates OpenAI API + PDF Reader
-  Memory (optional) – Can be extended to remember past rankings or JDs
-  Outcome-Focused – Delivers ranked output with justifications without user supervision

Tools & Technologies Used

-  Platform: VS Code (Locally) or Google Colab (Optional)
-  Language: Python 3.11
-  PDF Parsing: `PyPDF2`
-  LLM Reasoning: OpenAI GPT-3.5 / GPT-4
-  Output: `ranked_output.txt` (summary + ranking)
-  Packaging: `requirements.txt` for dependencies





Sample Output (Generated by Agent)

Top 3 Candidates:

- `resume2.pdf` – Strong backend skills, Django + REST API match the JD very well.
- `resume4.pdf` – Full-stack developer with Python + Flask, good team experience.
- `resume1.pdf` – Good Python knowledge, lacks direct framework experience but promising.

Reasoning: `resume2` is a near-perfect match to JD keywords and role requirements.

Impact

-  Time Efficiency: Screens 50+ resumes in under 30 seconds
-  Matching Accuracy: GPT enables human-like selection, improving candidate quality
-  Autonomous Workflow: No manual review required; agent loops, analyzes, and reports
-  Submission-Ready: All outputs saved and ready to share with hiring teams