Requirements Document: GPT-Powered Semantic Resume Matcher (NexResume)

Project Overview

NexResume is an AI-powered matching system that analyzes a job description and a collection of resumes to produce a semantic fit report for each candidate. It leverages large language models to understand and compare resume content to job requirements using natural language understanding.

This system is designed specifically for **Information Technology (IT) job families in India**.

Core Functional Requirements

1. Job Description Input

- Accept a job description file in YAML format.
- The file must include the following fields:
- `Job_Title`
- `Job_Summary`
- `Required_Skills`
- `Optional_Skills`
- `Minimum_Experience_Years`
- `Educational_Qualification`
- `Optional_Certifications`
- `Location`
- `City_Tier`: One of `Tier-1`, `Tier-2`, or `Tier-3`
    - **Tier-1** cities include: Chennai, Bangalore, Delhi, Hyderabad, Kolkata, Mumbai
    - **Tier-2** cities include (examples for Tamil Nadu): Coimbatore, Trichy, Madurai; similar-sized cities from other states
    - **Tier-3** includes all other cities not listed above
- `Maximum_Job_Gap_Months`: Specifies the maximum allowable duration (in months) between two consecutive jobs

2. Resume Input Folder

- The system must load and iterate through all resumes placed in a designated folder.
- Resume files may be in PDF or plain text format.
- Each resume will be treated as a unique candidate for semantic comparison.

3. Prompt Construction (Language Model Interface)

- The system must construct an effective prompt for the LLM that compares the job description with a given resume.
- The prompt must request the LLM to return a JSON object with the following fields:
- matched_required_skills

- missing_required_skills
- matched_optional_skills
- education_match
- experience_match
- keywords_matched
- soft_skills_match
- resume_summary
- match_score (numeric value between 0 and 1)
- city_tier_match: Whether the candidate's current city matches the tier specified in the job description
- longest_tenure_months: The **longest duration** (in months) the candidate was employed in a single job based on their work history
- final_score: An integer between 0 and 100 summarizing the overall resume match quality against the job description based on all criteria above

4. Language Model Integration

- The system must interact with a GPT-based API endpoint (such as OpenAI's `chat/completions`).
- For each resume, the system must:
- Send the constructed prompt
- Receive and parse the structured JSON response

5. Report Generation

- The system must generate an individual structured JSON report per resume.
- Reports must be saved to a `reports/` folder.
- Each report should include the candidate name and a summary of their semantic match with the job description.

6. Batch Processing

- The entire folder of resumes must be processed in a single run.
- The system must not crash or halt when encountering invalid or incomplete resumes.

Non-Functional Requirements

- System should be implemented in Python 3.8+
- Should handle edge cases like empty fields, missing education, inconsistent formatting, or long job gaps.
- Should be able to run locally with minimal configuration.
- Output JSON should be consistent and parseable regardless of LLM response variability.

Deliverables

- Python script or module with:
- Input loading from a YAML job description
- Resume folder ingestion

- Prompt generation logic
- GPT API integration
- JSON output for each resume
- Sample folder structure with:
- `job_description.yml`
- `resumes/` folder with sample `.pdf` or `.txt` files
- `reports/` output folder with generated JSON files

Evaluation Criteria (for Data Scientist)

- Ability to correctly extract and summarize key resume details
- Effectiveness and clarity of prompt design
- Handling of city-tier and job-gap logic:
- Candidates from **Tier-3 cities** should receive **higher weightage** to promote geographic diversity
- Candidates with **longer tenures in one job** should receive **higher scores** as a proxy for stability and depth of experience
- Accurate and meaningful `final_score` calculation in LLM output
- Quality of JSON output and alignment with job description fields
- Robustness across different resume formats and content styles

---

Sample Job Description

```
Job_Title: Python Backend Developer

Job_Summary: >
  We are hiring a Python developer to join our product engineering team. The
role involves developing

scalable APIs, working with cloud-based microservices, and contributing to the
architectural design of our backend.

Required_Skills:
  - Python
  - REST APIs
  - SQL
  - Version Control (Git)
  - Debugging and Code Review

Optional_Skills:
  - Docker
  - AWS Lambda
  - Flask or FastAPI
  - CI/CD Pipelines
```

```
Minimum_Experience_Years: 2

Educational_Qualification:
  - Bachelor’s Degree in Computer Science or related field

Optional_Certifications:
  - AWS Certified Developer Associate
  - Certified Python Programmer

Location: Madurai

City_Tier: Tier-2

Maximum_Job_Gap_Months: 6

Additional_Notes: >
  Preference will be given to candidates with startup experience and continuous
work history.
```

End of Requirements Document