Project Brief: AI Job Search

Modern job search is broken:

- Job descriptions are **messy, inconsistent, and unstructured**
- Requirements, tech stacks, seniority, and benefits are often implied—not cleanly labeled
- Users think in **intent**, not filters (“something senior but not management”, “remote-friendly startups”, “fast-growing companies”)

Your goal is to build a prototype AI-powered job discovery engine that helps users find relevant job postings using natural language queries and iterative refinement. The system should *feel* smart when someone searches for jobs.

The Challenge

You are given a dataset of USA job postings from HiringCafe. This dataset contains 100,000 most recent job postings from the United States, scraped directly from company career pages.These are scraped directly from company career pages.

What to build

Two capabilities. You design the function signatures, data structures, and API however you think is best.

1. Search

The user enters a natural language query, and the system returns the most relevant jobs.

2. Refine

The user has a conversation to narrow down results. They might say "data science jobs", then "at companies/non-profits that care about social good", then "make it remote". The system should use the conversation context to improve results.

The history could include previous queries, previous results, parsed intents, or whatever you think is useful.

Example flow:

```
User: "data science jobs"
```

```
→ Data scientist, ML engineer, analytics roles, etc.

User: "at companies or non-profits that care about social good"
→ Filters to mission-driven companies and non-profits

User: "make it remote"
→ Filters to remote data science roles at mission-driven orgs
```

Jobs dataset

We're providing jobs.jsonl - a collection of 100,000 jobs (unstructured text like title and job description + structured features we extracted using an LLM + text embeddings for the job description and company).

Jobs.jsonl dataset:
https://drive.google.com/file/d/1RRVWYAvfb4hUus1hUDY1nPQUJGqpiBiq/view?usp=sharing

Each job is row in the file. Some have more detail than others. Some mention qualifications explicitly, others don't. Some list salary clearly, others don't. Etc. A critical part of working at HiringCafe is navigating this messy real-world job data and learning how to make sense of it under ambiguity.

Your task in this challenge is to figure out how to work with this scale + complexity of data.

Data structure (Structured + Unstructured)

Field	Description
id	Unique job identifier (format: {source}_{board_token}_{job_id})
apply_url	Direct link to apply on HiringCafe
job_information (unstructured)	Raw job data (title, description HTML)
v5_processed_job_data (structured)	GPT-extracted structured fields (legacy)
v5_processed_company_data (structured)	Company metadata (subsidiaries, funding, employees)
v7_processed_job_data (structured)	Normalized structured data + 3 embeddings
_geoloc	Geo coordinates for location-based search

3 Embeddings are in v7_processed_job_data) All embeddings are generated using OpenAI *text-embedding-3-small* (1536 dimensions).

Embedding	Purpose	Text Source
embedding_explicit_vector	Match on what the job explicitly states	Job title, listed skills, required majors, certifications
embedding_inferred_vector	Match on related/implied qualifications	Related titles, inferred skills, likely relevant majors
embedding_company_vector	Match on company characteristics	Company name, industry, org type, activities

Deliverables

1. Working code
    a. Implements both functions described in “What to build”
    b. Runnable demo that shows 5+ queries with results
    c. Include the refinement flow in your demo

2. Readme - an explanation of your approach
    a. How did you process/represent the jobs data?
    b. How does your search work?
    c. How do you determine relevance/ranking?
    d. What trade-offs did you make?
    e. What queries work well? What's tricky?
    f. What would you improve with more time?

3. Tokens report. How many tokens did you use to develop this? How many tokens does the system consume per query? Please remember that we ask you to **self-monitor OpenAI usage and limit spend to $10 or less**

4. Demo video (optional)
    a. Record a short screen-share showing how to run your demo script, one initial search, and one multi-turn refinement flow.
    b. Please do not edit the video.
    c. We are not evaluating presentation quality, speaking ability, or production value.
    d. This is purely to help us understand your system faster.

Evaluation

We care about:

- *Do the results feel right* - Run your queries, do the rankings make sense
- *Thoughtful approach* - Did you think through the problem or just throw code at it?
- *Handling ambiguity* - Real queries are messy. How do you deal with that?

- *Handling scale* - Our job dataset is massive. Can you wrangle it?

We don't care about:

- Perfect code style
- Test coverage
- Fancy infrastructure
- Fancy UI/UX

Tips

- Start simple, then iterate - Get something working, then improve
- Test with real queries - The best way to know if it works
- Be opinionated - We want to see YOUR judgment
- Use AI tools freely - Cursor, Copilot, whatever. The thinking is yours.

Submission

Please email a link to a private Git repository (invite Github usernames hamedn and alimir1) or a zipped folder containing

1. **Your Code.** Code should be runnable via a single command (e.g, “node demo.js” or “python demo.py”)
2. **Readme File**
3. **Tokens report**
4. **Demo video (optional)**

Email to hamed@hiring.cafe and ali@hiring.cafe

Estimated time to completion

Depending on your speed of execution, this take-home assignment typically takes between 8-24 hours to complete. Please do not spend more than 24 hours total (approximately 3 working days) on it. If you choose to spend less time, that’s completely fine. Many strong candidates submit partial solutions. Please include how long you spent, and we will evaluate your work in that context. We respect that candidates have personal lives and other commitments.

To be clear, this is **not free consulting work**. This task reflects work we have already completed internally. We intentionally chose a more substantial assignment to give strong but non-obvious candidates (“hidden gems”) a real opportunity to demonstrate their thinking, depth, and problem-solving approach beyond surface-level signals.

Use of AI Tools

We strongly encourage using AI programming tools (e.g., ChatGPT, Cursor, Manus) during development to help you design, write, and debug your solution. However, your final solution must not be a wrapper for a 3rd party AI search solution (e.g., Algolia AI search).