Taboola

SW Engineer, Internal Tools: Home Assignment

Objective

Evaluate your ability to build a small but real AI agent that monitors public chatter about **Taboola** and **Realize** and reports **field sentiment** clearly and reproducibly.

The Challenge: Build a Mini Social Listening Agent

Create an agent that:

1. **Reddit**: fetches recent posts/comments mentioning **“Taboola”** and/or **“Realize”**, analyzes them, and outputs sentiment at the **field level** (see schema below).

2. **Bonus source (choose one)**: add **one additional public source** (e.g., LinkedIn public posts, Facebook public pages, X/Twitter, YouTube comments, Hacker News, Product Hunt, news RSS). If live APIs are hard to access, you may ingest a CSV/JSON export you produce manually.

3. **Visual Insight UI:** Build a simple interface (e.g., lightweight React app, or HTML dashboard) to visualize your results. It should display sentiment by field, trends over time, and allow basic filtering by platform, entity, and topic.

   You’re not expected to build a full product, **a working script/CLI or notebook is enough**, but it should run end-to-end on sample or live data.

Also produce **aggregates**:

- Sentiment distribution per **entity** and **field** (e.g., performance: 62% positive / 18% neutral / 20% negative).

- Top **3 themes** per entity with representative quotes/links.

- **Trend over time** (by day/week) if you fetch timestamped data.

Requirements

- **Language:** Python (preferred) or TypeScript/JavaScript.

- **Use at least one external API** (LLM for analysis; data API or simple fetch for content).
  - Examples: OpenAI/Claude for LLM; Reddit API/Pushshift or manual export; a second source API or CSV.
  - **If you need our help with API keys, please let us know and we will help.**
- **Structured data:** ingest + output JSON/CSV and write aggregates programmatically.
- **Include at least one simple unit or integration test** to validate core logic (e.g., sentiment parser, JSON schema validation, or API call wrapper).
- **Design doc** (≤1 page) covering:
  - Problem & success criteria
  - Architecture (inputs → processing → outputs)
  - Model & prompt choices; how you ensure consistency (e.g., output schema, few-shot prompts)
  - Limitations / next steps
- **Reproducibility:** use env vars for API keys; document how to run (README).
- **Ethics & compliance:** respect TOS; only use publicly available content. If a platform is restricted, use a **manual export** workflow.

Evaluation Criteria

Category	What We're Looking For
System Design & Reasoning	Clear pipeline, sensible data model, defensible choices.
Execution & Code Quality	Modular, readable, runs as described.
AI Integration Skill	Reliable schema-constrained outputs; prompt design; handling LLM edge cases.
Business Value	Field-level insights that a GTM or product team could use.

Documentation	Clear run steps, design doc, and limits/next steps.
Velocity Mindset	Practical, end-to-end prototype within scope (3–4 hours).
Insights & UX	Does the UI make it easy to spot patterns and drill down to examples?

Submission

1. **Repo or zip** with code + `README.md` (how to run, env vars).
2. **Design doc** (Google Doc or Markdown, ≤1 page).
3. **Outputs** sample (`items.jsonl`, `aggregates.json`, `report.md` or screenshots).
4. **(Optional)** 2–3 minute Loom or GIF demo.

Estimated time: 6–8 hours total

Notes & Tips

- If platform auth blocks you, **mock the ingest**: paste public URLs into a CSV and run the same pipeline. The point is the **agent + analysis**, not API gymnastics.
- Be explicit about brand/entity mapping (e.g., “Realize” vs generic “realize”).
- Keep prompts short; use a **JSON schema** + `response_format` (if your LLM supports it) to reduce parsing errors.
- Include a small **“edge cases”** section (sarcasm, mixed sentiment per field, duplicates, reposts).

If you want, I can also draft a starter `README.md`, the JSON schema, and a minimal Python CLI scaffold you can drop into a repo.