AI Engineer Assignment: Autonomous News Agent

Objective

Build a daily autonomous system that identifies trending global events, filters for impact, and generates high-quality, fact-grounded articles without human intervention.
The Core Challenge: This is a **System Architecture** and **Decision-Making** problem. We are evaluating how your system handles noise, prioritizes topics, and hallucinates less.

Problem Statement

Your system must run once per day (or on command) to execute this pipeline:

1. **Ingestion:** Fetch trending data from the web (via APIs, RSS, or scrapers).
2. **Selection:** Filter noise to identify the top 3–5 most significant global trends.
3. **Research:** Gather factual context for these trends from real-time sources.
4. **Generation:** Write publication-ready articles grounded in the retrieved facts.
5. **Output:** Return a structured JSON payload.

📦 Required Output Format

Your script must output a single JSON object adhering to this schema:

JSON

```
{
  "date": "YYYY-MM-DD",  // Strict ISO 8601 (UTC)
  "execution_time_seconds": 0.0,
  "articles": [
    {
      "title": "String",
      "category": "Technology | Finance | Politics | Other",
      "trend_score": 0.0, // Normalized (0-10) or absolute; explain logic in docs
      "summary": "String (TL;DR)",
      "article_body": "String (Markdown supported)",
```

```
    "sources": [
      "https://url-to-source-1.com",
      "https://url-to-source-2.com"
    ],
    "hallucination_check": "Pass/Fail/Unsure" // Optional: Self-verification metric
  }
 ]
}
```

Article Expectations

Each article must include:

- **Structure:** Headline -> Context/Background -> Key Developments -> Implications.
- **Grounding:** Content must be derived from the sources found during the research phase.
- **Citations:** The system should be able to reference where facts came from.
- **Length:** Comprehensive (approx. 600–800 words).
- **Tone:** Neutral, journalistic, objective.

Constraints & Guidelines

1. Data Ingestion vs. Scraping

- **Clarification:** While you *must* ingest data to find trends, we are **not** evaluating the complexity of your crawler.
- You may use official APIs (Twitter/X, Google Trends, NewsAPI), simplified scrapers, or RSS feeds.
- **Focus:** We care about how you **rank** and **verify** the data, not how hard it was to scrape.

2. Cost & Tools

- **Stack:** You decide everything (Python, Node, LangChain, LlamaIndex, OpenAI, Anthropic, Open Source Models, etc.).
- **Budget:** Do not spend personal funds. Use free tiers or trial credits.
- **Mocking:** If a production-grade solution requires a paid API (e.g., SerpApi, Twitter Enterprise), you may **mock the data response** for the demo, provided you explain the intended architecture in the documentation.

3. Quality Control

- **Hallucinations:** Pure LLM generation is not accepted. You must implement a strategy to ground the generation (e.g., RAG, Search-and-Verify, or Citation-Matching).
- **Deduplication:** The system must recognize if "Bitcoin" and "BTC Price" are the same trend and merge them.

What We Evaluate

1. Architecture & Engineering

- How do you handle the flow of data? (Sequential chains vs. Parallel agents).
- How robust is the error handling? (What happens if the Search API fails?).

2. "Intelligence" (Trend Ranking)

- Did the system pick random viral TikToks, or actual news?
- Explain your ranking logic (embeddings, keyword frequency, velocity).

3. Factuality

- Does the article cite real sources?
- Did the system hallucinate quotes?

4. Code Quality

- Is the repository clean, modular, and easy to read?
- Can we run it with a simple docker-compose up or python main.py?

Deliverables

1. **Repository:** A link to the working code (GitHub/GitLab).
2. **Run Instructions:** Precise steps to set up the environment and run the script.
3. **Sample Output:** A JSON file containing the results of a full run.
4. **Brief Documentation (Readme or PDF):**
    - **Approach:** Why did you choose this stack/architecture?
    - **Trade-offs:** What did you sacrifice for speed/cost?
    - **Trend Logic:** How do you calculate the trend_score?
5. **Demo Video (Loom/YouTube - max 5 mins):** Walk through the code and a live execution.

🏁 Bonus (Optional)

- **Verification Agent:** A second LLM pass that critiques the first draft against the sources.
- **Multi-Region:** Ability to toggle between trends in "US", "India", or "Global".
- **Dashboard:** A simple frontend (Streamlit/Next.js) to visualize the JSON output.