

Take Home Assignment

To be completed latest within one calendar week

Overview

In this assignment, you'll design and implement an **agentic system** that constructs a **graph database of academic papers**.

Your agents should be capable of:

- Reading research papers
- Extracting key concepts and entities
- Connecting those concepts across papers through **semantic relationships**, not just citations

The goal is to demonstrate how intelligent agents can extract structure and meaning from unstructured academic text and represent it as an interconnected knowledge graph.

Your submission should include a **proof-of-concept backend system** that extracts nodes and relationships from a selected corpus of papers. You do **not** need to implement user-facing features (e.g., search, visualization), but you should discuss how you would extend your system to support them in the future.

What We're Looking For

This challenge is designed for candidates applying to the **AI Engineer** role—one of the most technically demanding and strategically important positions in our company.

We're not only evaluating your coding ability, but also your **system design thinking**, **agentic reasoning**, and **approach to real-world scalability**. A strong submission will show both solid engineering and thoughtful consideration for long-term usability and reliability.

Key Design Considerations

You don't need to implement every item below, but you should address each area in your documentation and make deliberate design choices.

1. Representing Data in the Graph

- What node and edge types will your graph include (e.g., Paper, Concept, Author, Method, Dataset, Metric)?
- Should agents be able to create new node or edge classes dynamically?
- How will you represent semantic relationships (e.g., *introduces*, *improves_on*, *evaluates*, *extends*)?

2. Extracting Entities

- How will you prompt or instruct agents to identify entities and relationships?
- How will your system validate or correct extraction errors?
- Will you rely on prompting alone, or would fine-tuning or few-shot learning improve accuracy?

3. User Experience and Use Cases

- What real-world use cases could this graph enable (e.g., semantic search, literature mapping, novelty discovery)?
- How would users interact with or navigate the graph?
- How might you surface explainable insights (e.g., "Paper B improves on Paper A by introducing concept X")?

4. Scalability and Maintenance

- How could this system scale to process the entire corpus of AI research?
- How would you keep it up to date as new papers are published?
- How would you ensure performance, consistency, and fault tolerance at scale?

Corpus: Papers to Include

To narrow scope, your system should focus on papers related to **Gaussian Splatting** — a well-defined and fast-growing subfield in computer graphics and 3D reconstruction.

Start with the seminal paper:

[3D Gaussian Splatting for Real-Time Radiance Field Rendering](#)

This paper has over 6,000 citations. Select a **subset of 50–100 related papers** (more if you wish). You may choose papers by relevance, citation strength, or semantic similarity—just explain your selection strategy in the documentation.

Deliverables

1. Backend Codebase

A working backend that:

- Agentically analyzes papers and extracts graph nodes and edges
- Stores extracted data in a Postgres database (schema defined below)
- Demonstrates a proof-of-concept pipeline end-to-end

Languages:

- TypeScript preferred (especially for the agent layer)
- Python acceptable for experimentation, parsing, or fine-tuning

You do **not** need to build a frontend or search interface, though you're welcome to include one if time permits.

2. SQL Schema Definition

Define your graph structure in **Postgres**, including:

- Table definitions for nodes, edges, and metadata
- Example queries (e.g., "Which papers improve on the original 3DGS method?")

While Postgres isn't a native graph database, it's widely used for production graph-like systems (e.g., Facebook's TAO).

Using **Supabase** to manage your Postgres environment is optional but recommended.

3. Documentation

Your documentation is as important as your code. It should clearly communicate how you approached the problem and how you would evolve your solution.

Include:

- **System architecture overview:** data flow diagrams, key components, and agent logic
- **Design rationale:** how your approach addresses the four key consideration areas
- **Limitations and trade-offs:** what's out of scope and why
- **Future roadmap**, including:
 - Plans for scaling ingestion and processing
 - Concepts for a user interface or visualization layer
 - How the system could support advanced research discovery features (e.g., semantic queries, trend analysis)

Not Required

You are **not** expected to deliver:

- A production-ready frontend
- A full-scale dataset ingestion pipeline
- High-fidelity LLM training or fine-tuning

Focus your time on building a **clean, well-architected proof of concept** that demonstrates strong reasoning, modularity, and potential for real-world impact.

Summary

At a high level, we're looking for:

- **A thoughtful architecture** for an agentic system that builds a research knowledge graph
- **Working backend code** showing proof of concept
- **Clear documentation** that demonstrates how you think about scaling, usability, and reliability

This is intentionally an ambitious challenge—our goal is to see how you think, structure, and communicate complex systems.