Atlan Platform Internship Challenge 2025 - Observability

Overview

As an intern at Atlan, you will contribute to internal Observability improvements that simplify and speed-up debugging for all Atlan engineers, allowing everyone to detect issues earlier. This task evaluates your problem-solving skills.

You will be given a scenario and set of tasks to complete. Please submit your response in PDF, ensuring clarity and conciseness in your explanations.

Scenario

Imagine you're in a team of engineers that is being slowed down by a painful debugging experience, where finding the cause & corrective actions about why a new software release is broken/malfunctioning is slow, repetitive, manual, inconsistent, and heavily dependent on the finite time of a few experienced engineers. Logs if any, were added a long time ago and are not too relevant.

Example

A REST API endpoint is having high response times/error rates. Note that this API endpoint may have many other actions (DB queries/other API calls/etc) inside it.

Assuming a frontend/backend web framework of your choice:

1. Which metrics would you consider important, and why?
2. How would you decide which logs to keep/remove/add?
3. Provide a sample dashboard (public image link/image).

Here, we're equally interested in looking at your thinking behind the problem, as well as the final dashbaord itself.

1. What improvements would you make to **address** the above problems?

2. Without having to ask the team repeatedly, how would you track how much your improvements have helped the team?

If needed, you may use Atlan's existing architecture as a starting point.

Follow-up:

1. What could be done to reduce repetitive work to expand this dashboard to more API endpoints?
2. Which problems might affect the team after the above problems have been fixed?
3. What would you additionally do to **prevent** the above issues from re-appearing in the future?

Deliverables

High-level system diagram

This outlines the major components involved in the system, and the interactions between them.

Explanatory document

A brief 1-2 page document explaining:

1. Major, high-level design decisions and tradeoffs.
2. Proof of the solution, solving the stated problem.
3. Known gaps that the solution does not cover, and why those gaps are safe to ignore.

Video submission

Record a 5-7 minute video where you:

1. Walk through your solution, demonstrating key features.
2. Explain the high-level design with technical decisions made.
3. Challenges faced whilst solving this problem, and how they were resolved.

Submission guidelines

If applicable:

1. **Code:** Upload your project to a GitHub repository and share the link. To prevent plagiarism, we recommend using a meaningless or obfuscated project name.
2. **App:** If applicable, deploy your solution using a platform like Vercel, Netlify, or Heroku and share the link.
3. **Documentation, ER Diagram, and Video:** Share links to your architecture diagram, ER diagram, explanation document (PDF), and video walkthrough.

Evaluation criteria

1. **Clarity of communication:** Ability to articulate ideas clearly and effectively.
2. **Depth of current knowledge:** How well do you know what you know?

Expectations

This task evaluates your ability to:

1. Think about the same problem from multiple angles.
2. Identify which part of your current knowledge to apply to a given problem.
3. Absorb new information quickly, and communicate succinctly.