Data Engineering Challenge

What is this?
This is a take-home test to evaluate your collaboration skills and your ability to solve both high- and low-level technical problems. We’re specifically interested in answering the following questions:

- How do you think about and decompose broad, ambiguous problems?
- How do you handle disagreements with other developers and their teams?
- Can you solve programming problems with good tests and coherent code?

What problems am I solving?
There are three problems in this challenge:

- The High Level Problem
- The Low Level Problem
- The Scenario Problem

They are designed to test your software design and architecture level, your programming skills, and your approach to collaborating with coworkers, respectively.

How do I submit my work?
Your submissions can take whatever structure you think best communicates your solutions. Part of what we’re interested in is how you communicate. You can send your submission to jfrausto@covalto.com.

What happens after I submit?
We review what you sent us and we schedule a time to talk with you about your solutions. You’ll have a chance to describe your approach and answer any questions that your solutions generated.

The High Level Problem

You are the tech lead of the Data Engineering squad at a bank. Your team has three stakeholders at the organization with three distinct data problems:

1. Customer Behavior Data Visualization
    a. **Problem.** Analysts would like to be able to build their own reports. Currently, they ask developers in your group to make reports for them on an ad-hoc basis. While the analysts understand the data they want to see, there is often confusion between the developers and the analysts due to polysemy. They are interested in all available data sources.
2. Risk Assessment for Home Buyers seeking a Mortgage
    a. **Problem.** The decision science team is developing models for pricing home loans given an applicant's financial history. Their models do not ingest raw data. Instead, they ingest characteristics composed of processed raw data.
3. Fraud Monitoring
    a. **Problem.** The fraud monitoring system observes customer bank transactions and credit card transactions in real time. They are interested in some of the higher-order features that the decision science team uses, and also some raw transaction data.

Available Data Sources

Annual Tax Returns *(3rd party XML API)*
Requires a Tax ID and digital authorization from the individual or company. No bulk endpoints, rate-limited by auth token.

Credit Card Transaction History *(3rd party JSON API and internal pg databases)* JSON API requires auth token, has bulk and streaming endpoints, not rate-limited. Internal pg databases have many data gaps and overall low data integrity. Data is spread across several databases which are relied upon by several production apps.

Bank Statements *(3rd party XML API and an S3 bucket of PDFs and images)* XML API requires auth token and digital signature, has bulk and streaming endpoints, and is not rate-limited. S3 bucket of PDFs and images are unstructured data, often photos of handwritten forms.

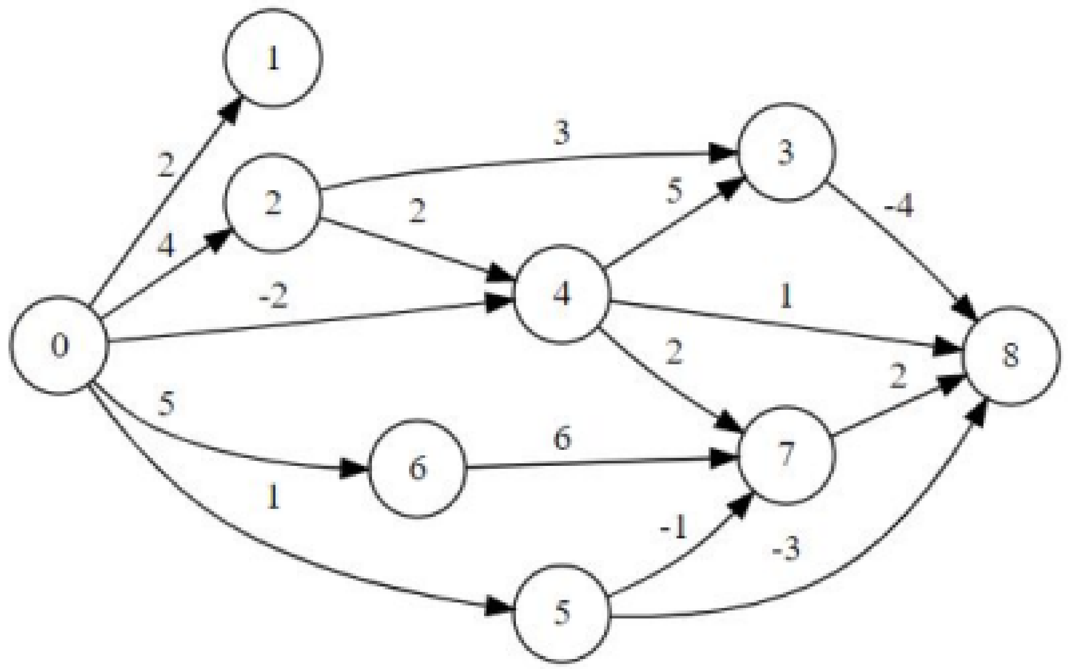
Challenge

Design an architecture that satisfies these constraints. Feel free to provide multiple levels of detail, but do not feel obligated to write code.

The Low Level Problem

Write tests and an implementation for the following problem.

Given a Directed Acyclic Graph:



write a tested program to accomplish the following:

1. Print the vertex *(**V**)* reachable by the greatest number of paths from the source vertex 0.
2. Sort and print those paths according to their cost (descending).
3. Introduce an additional vertex *(**V`**)* that satisfies the following conditions:
   a. ***V`*** is now the most reachable vertex (instead of ***V***).
   b. None of the vertices which share an edge with ***V*** share an edge with ***V`***.
4. If (3.b) is impossible, display an error message explaining why.
5. If (3) succeeds, print ***V`***'s insertion in the input format.

Input

{0, 1, 2},
{0, 2, 4},
{0, 4, -2},
{0, 5, 1},
{0, 6, 5},
{2, 3, 3},
{2, 4, 2},
{3, 8, -4},

{4, 3, 5},
{4, 8, 1},
{4, 7, 2},
{5, 7, -1},
{5, 8, -3},
{6, 7, 6},
{7, 8, 2}

Source vertex: 0

The Scenario Problem

Several services for which your team is responsible depend on another squad’s service in production, and mocks of that service in development and staging.

Your team uses the mocks they have provided in both development and staging. However, there are regular incidents in staging and production due to code changes made by that team which they have not propagated to their mocks.

Your developers are frustrated and losing confidence in their coworkers. The tech lead from the other squad has a reputation for over-confidence and being difficult to work with.

How do you handle this situation?

Please include the responses you anticipate from the other team, as well as your responses to those responses.