Platform Architect Take-Home Assignment

Commerce Transaction Platform (Monolith to Azure Migration)

Overview
You are designing a key aspect of a commerce platform that enables it to correctly process **orders**, **refunds**, **adjustments**, **fees**, and **partial fulfillments** from different source systems and data structures that are **not within your control**.

These external systems differ in file formats and in how business meaning is expressed. The platform must support reprocessing, corrections, auditability, reconciliation, and controlled evolution over time.

This assignment includes a short design component and a narrow hands-on coding exercise. You should approach the design questions from the perspective of overall platform architecture responsibility.

Hybrid environment and migration constraint:
The current on-premises system is a **monolith** that handles both **order fulfillment** and **billing**. The organization wants to move both functions to **Azure**, but will migrate **one capability at a time**. During migration, parts of the workflow will run in two locations and must remain correct, auditable, and operable.

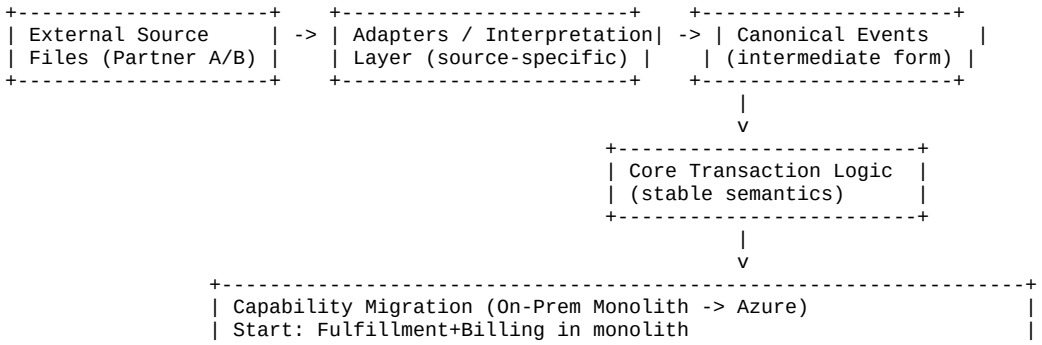
Time Expectation
Approximately 2 hours.

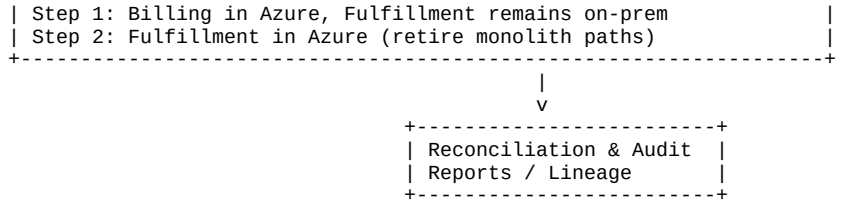
Submission
Please provide: Source code with instructions to run A short written design explanation (maximum 1 page) You may use any programming language.

Part A — Design

Written Explanation (maximum 1 page)
Describe: How your design separates source-specific interpretation from core transaction logic Whether and how you introduce an intermediate canonical representation between source files and transactions How auditability and reconciliation are achieved
Monolith-to-Azure migration questions (address briefly in your write-up):
The on-prem monolith owns both **fulfillment** and **billing**. If you migrate **billing** to Azure first while fulfillment remains on-prem, what changes do you make to enable the split and keep the system reliable and secure? What changes do you make to the application architecture and interfaces so that fulfillment and billing can evolve independently during the migration? Assume the connection between on-prem and Azure can occasionally fail or be degraded. Describe the runtime behavior you would choose for billing and fulfillment in those conditions, and how you reason about tradeoffs between consistency and availability. How do you execute progressive cutover and safe rollback for a single capability migration while limiting blast radius and avoiding regressions in existing flows?

Architecture Diagram (Reference)
You may use or adapt the following conceptual flow in your explanation:





Part B — Coding + Design Exercise

This section includes a small coding exercise and an accompanying design prompt. The goal is to demonstrate how you would approach normalization of different source structures into a consistent internal representation.

Scenario
Two external commerce systems provide files describing orders. Both systems describe similar business concepts, but their row structures differ.

System A (Line-per-item Orders)
File: **line_item_orders.csv**

```
source,file_id,row_id,order_id,customer_id,txn_id,item_sku,qty,unit_price,event_date
A,LI100,1,ORD2001,CUST7,T10001,SKU-AAA,1,40.00,2025-04-01
A,LI100,2,ORD2001,CUST7,T10002,SKU-BBB,2,15.00,2025-04-01
A,LI100,3,ORD2002,CUST8,T10003,SKU-CCC,1,25.00,2025-04-01
```

System A provides one row per item line in an order.

System B (Packed-SKU Orders)
File: **packed_orders.csv**

```
source,file_id,row_id,order_id,customer_id,order_ref,sku_1,price_1,sku_2,price_2,sku_3,price_3,order_date
B,PK200,10,ORD3001,CUST9,O-777,SKU-AAA,40.00,SKU-DDD,10.00,, ,2025-04-01
B,PK200,11,ORD3002,CUST10,O-778,SKU-EEE,12.00,SKU-FFF,18.00,SKU-GGG,5.00,2025-04-01
```

System B provides one row per order, but item data is embedded into multiple SKU/price columns on the same row.

Deliverables

Code: Implement a small program that reads both input files and emits a normalized representation of order items (your choice of output format: JSON, JSONL, CSV, etc.). **Design:** Propose a system design (high-level) for handling these kinds of format and semantic differences at scale as new sources are onboarded. **Clarifying questions:** Provide a list of the key questions you would ask stakeholders to refine the solution and reduce risk.

How to Run
Provide simple execution instructions, for example:

```
./run.sh line_item_orders.csv packed_orders.csv
```