# **Take-Home Assessment**

## **Overview**

You're tasked with designing and implementing a data pipeline for a fictional e-commerce company, "ShopStream," that needs to analyze user behavior across multiple touchpoints to improve conversion rates and detect potential fraud.

## **Business Context**

ShopStream receives data from multiple sources:

* Real-time clickstream events from web and mobile apps
* Order transactions from the payment system
* Product catalog updates from the inventory system
* Customer support interactions

The company needs to:

1. Create unified customer profiles
2. Calculate real-time metrics for fraud detection
3. Generate daily business intelligence reports
4. Enable data scientists to query historical data efficiently

## **The Task**

### **Part 1: System Design (60% weight)**

Design a data pipeline architecture that can handle the requirements above. Your design should include:

1. Architecture Diagram showing:
   * Data sources and ingestion methods
   * Processing layers (streaming/batch)
   * Storage solutions for different use cases
   * How different teams will access the data
2. Technology Choices with justification for:
   * Message queue/streaming platform
   * Processing frameworks
   * Storage systems (data lake, warehouse, etc.)
   * Orchestration tools
   * Data quality/monitoring solutions
3. Design Decisions addressing:
   * How you'll handle late-arriving data
   * Schema evolution strategy
   * Data retention and archival policies
   * Failure recovery mechanisms
   * Scaling strategy for 10x growth

### **Part 2: Implementation (40% weight)**

Implement a simplified version focusing on ONE of these scenarios (your choice):

Option A: Streaming Pipeline

* Process the provided sample clickstream data
* Implement sessionization (group events into user sessions with 30-minute inactivity timeout)
* Calculate real-time metrics: events per minute, unique users per minute
* Detect potential anomalies (e.g., >100 events from single user in 1 minute)

Option B: Batch Pipeline

* Process daily transaction and clickstream data
* Build user behavior aggregates (purchase patterns, browsing categories)
* Implement slowly changing dimension (SCD Type 2) for customer profiles
* Create data quality checks and quarantine bad records

## **Data Files Provided**

You'll find the following sample data files in the data folder:

* clickstream\_events.json: User interaction events from web/mobile
* transactions.csv: Completed and failed transactions
* product\_catalog.json: Current product information
* customer\_support\_interactions.json: Support ticket data

## **Requirements & Constraints**

### **Functional Requirements:**

* Handle 1M events/minute during peak hours
* Sub-second latency for fraud detection
* 99.9% uptime for critical pipelines
* Support for GDPR compliance (data deletion requests)
* All timestamps are in UTC

### **Non-functional Requirements:**

* Cost-conscious design (explain trade-offs)
* Monitoring and alerting strategy
* Data lineage tracking
* Development environment setup

## **Deliverables**

1. System Design Document (PDF/Markdown):
   * Architecture diagram
   * Technology justifications
   * Design decisions with trade-offs
   * Capacity planning calculations
2. Code Implementation:
   * Working code for chosen scenario
   * README with setup instructions
   * Basic tests demonstrating functionality
   * Configuration for local development
3. Data Model:
   * Schema definitions for your chosen storage systems
   * Example queries for common use cases
4. Production Readiness (1-2 pages):
   * Deployment strategy
   * Monitoring/alerting plan
   * Disaster recovery approach
   * Performance optimization ideas

## **Time Expectation**

This assessment should take 4-6 hours. We value quality over quantity - a well-thought-out partial solution is better than a rushed complete one.

## **Submission Instructions**

1. Create a private GitHub repository
2. Commit your work with meaningful commit messages
3. Include all documentation in the repo
4. Share access with [interviewer email]

## **FAQ**

Q: Should I implement everything in the design? A: No, implement only the chosen scenario. The design should be comprehensive, but the code should focus on demonstrating your implementation skills for one component.

Q: Can I use cloud-specific services? A: Yes, but explain why and provide cloud-agnostic alternatives.

Q: What if I'm unfamiliar with some technologies? A: Document your learning process and reasoning. We value problem-solving over memorization.

Q: Which programming languages can I use? A: Python is preferred, but Scala, Java, or Go are acceptable. Use what you're most comfortable with.

Q: Should I consider costs in my design? A: Yes! Include rough cost estimates and discuss trade-offs between cost and performance.