## Formative 3 – Mini Project (Architecture Diagram) Scenario: Data Ingestion & Quality

## Management for Inter-Party Relationships (IPRs) in Consumer Relationships

## ****Section 1:Introduction****

This formative documents data ingestion and quality management approach for Inter-Party Relationships (IPRs) in a large retail bank's Consumer Relationships domain. IPRs define connections such as parent/guardian/ward and govern how communications and access rights are applied. Recent incidents have highlighted that misapplied IPRs can cause misdirected correspondence, unauthorised changes, and regulatory exposure. The objective is to design an end-to-end ingestion architecture that accommodates diverse sources, embeds automated quality controls, and produces audit-ready outputs. This solution aims to reduce risks by not only ensuring compliance but also by enabling more efficient customer service and reducing manual data clean-up efforts by over 50%.

### ****Section 2: Revised System Sketch****

The architecture diagram (Figure 1) shows how the data flows through the system.  
The design is easier to follow than earlier drafts because it separates each major step:

* **Ingestion methods** – three options:
  + Batch ingestion for legacy systems that send daily files.
  + Streaming ingestion for real-time updates, such as new IPR records created during the day.
  + Change Data Capture for core databases, so we only collect rows that have changed instead of full tables.
* **Secure staging** – a protected area inside a cloud data lake.  
  This is where all raw data lands first.  
  It is read-only so nothing can be overwritten by mistake.
* **Processing layer** – a step that checks the data against agreed rules, enforces the correct schema (column names and types), and standardises formats.  
  For example, it can make sure dates are all in the same format and that relationship codes match the official list.
* **Orchestration layer** – a tool that schedules jobs, starts them when triggers occur (like a new file arriving), and automatically retries if something fails.
* **Governance layer** – controls such as Role-Based Access Control, audit logs, and a data catalogue that records data lineage (where the data came from and how it changed).

All sources feed into a single, governed data lake, which becomes the trusted place for downstream analytics or reports.

### ****Section 3: Methodology Write-up****

This section details the design choices and methodology employed to create a robust and scalable data ingestion pipeline.

#### Data Collection

Different systems provide data in different ways, so the pipeline supports multiple ingestion patterns.

* **Batch ingestion** handles older systems that output daily flat files.  
  These files are collected overnight so a full set of records is processed at once.
* **Real-time ingestion** uses REST APIs to pick up important or high-risk changes quickly, such as a new IPR being created.  
  This ensures customer service teams always see the latest information.
* **Change Data Capture (CDC)** watches the bank’s key relational databases and pulls only the rows that changed since the last run.  
  This reduces network traffic and speeds up processing.

#### Automation

Automation keeps the pipeline reliable and removes the need for manual uploads.  
The orchestration tool starts jobs as soon as new data arrives or on a set schedule.  
It runs hourly checks to confirm all expected files have been received, retries failed jobs automatically, and sends alerts if something goes wrong.  
This design cuts down on human error and means the system can scale without extra staff.

#### Data Quality and Pre-processing

Every record first lands in the **secure staging area** for initial checks, such as verifying the file format and schema.  
Next, the **processing layer** applies detailed business rules to clean and standardise the data.  
Examples include:

* Checking that parent–child relationships have sensible age gaps.
* Ensuring that household links are valid and not duplicated.

The cleaned and standardised data is then made ready for analytics or machine learning models that can flag unusual IPR patterns.

### ****Section 4: Design Choices & Architecture****

The architectural design was guided by a number of key considerations to ensure the solution is fit for purpose and provides long-term value.

#### Scalability

The system runs on a **cloud-based data lake** with a distributed processing tool such as Apache Spark.  
This allows scaled storage and computing power separately, so it can handle growing volumes without over-provisioning resources .

#### Reliability

The pipeline is built for fault tolerance. If a job fails, the orchestration layer retries it automatically.  
The system **queue** captures records that still fail after retries so they can be reviewed. The system uses **idempotent writes** so reprocessing data does not create duplicates.

#### Security

Security is built into every step:

* Data is encrypted both **in transit** and **at rest**.
* Role-Based Access Control ensures only authorised people can view or change sensitive information.
* API keys and passwords are stored using a secure secrets manager.

#### Compliance

To meet regulatory needs, the system records **data lineage** in a catalogue, showing where each record came from and how it was changed. Immutable audit logs capture every action, which provides evidence for reviews such as GDPR audits. This gives confidence in the accuracy and traceability of IPR data.

### ****Section 5: Reflection & Continuous Improvement****

This design is a clear improvement over earlier drafts because it highlights governance and makes the flow easy to explain to both technical and non-technical stakeholders.

Key lessons learned:

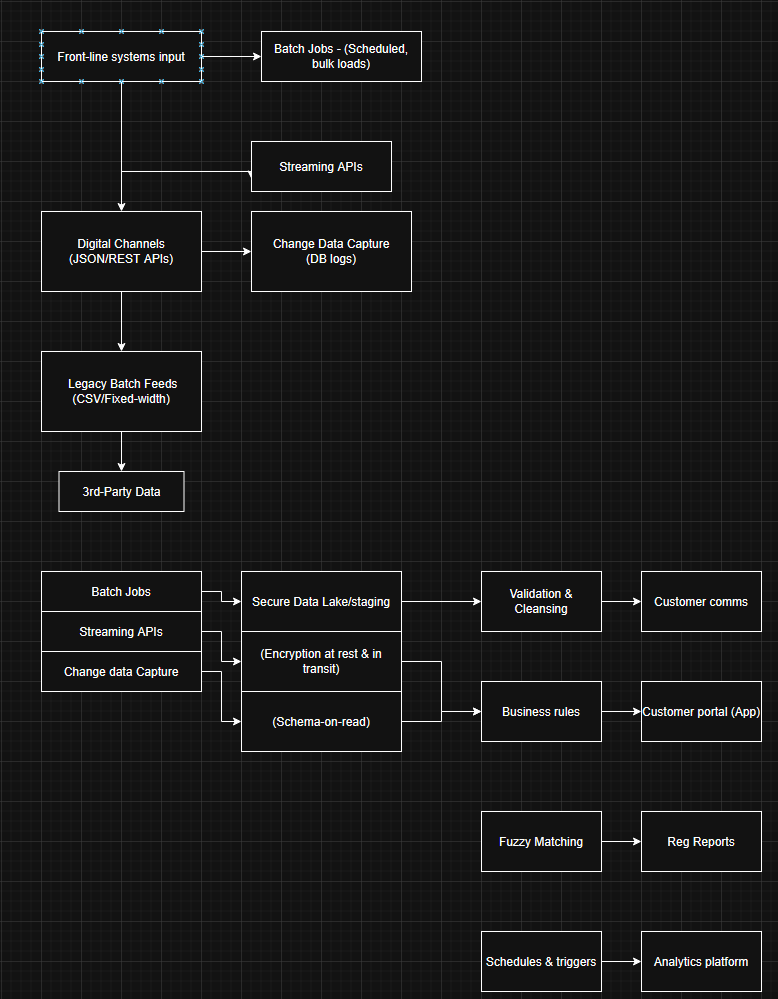
* **Quality checks must start early.** Catching issues at ingestion prevents larger problems downstream.
* **Flexible ingestion methods are valuable.** Supporting batch, streaming, and CDC means the bank can work with a variety of source systems.
* **Automation reduces risk.** Scheduled and event-driven triggers remove the need for manual uploads and reduce human error.

However, there are still some limitations.  
The pipeline can quarantine bad records but **cannot fix the original source data**.  
Future improvements could include:

* Automatic reconciliation of quarantined records back to the source system.
* Steward dashboards so data owners can see and correct issues quickly.
* Machine learning models to detect anomalies and reduce false positives.

**Conclusion:** The proposed ingestion architecture delivers timely, governed data for IPRs, reducing operational risk and enabling confident decision-making. By combining fit-for-purpose ingestion patterns, automated quality controls, and strong governance, the design provides a scalable foundation for continuous improvement.

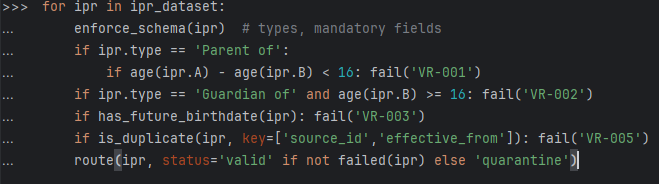
*Fig. 1 – Architecture Diagram (sources → ingestion → staging → processing/orchestration → targets*

**

*Fig. 2 – Example Validation Rules (excerpt)*

|  |  |  |  |
| --- | --- | --- | --- |
| Rule ID | Description | Type | Action on Fail |
| VR-001 | Parent must be ≥16 years older than child | Business | Quarantine record; raise task |
| VR-002 | Guardianship expires at adulthood threshold | Business | Auto-expire; notify steward |
| VR-003 | No future birth dates | Schema/Validity | Reject; log error |
| VR-004 | Mandatory fields present and typed | Schema | Reject; log error |
| VR-005 | Duplicate IPR (source\_id + effective\_from) | DQ/Dedup | Mark superseded; keep latest |

*Fig. 3 – Pseudocode for IPR validation pipeline (excerpt)*

**