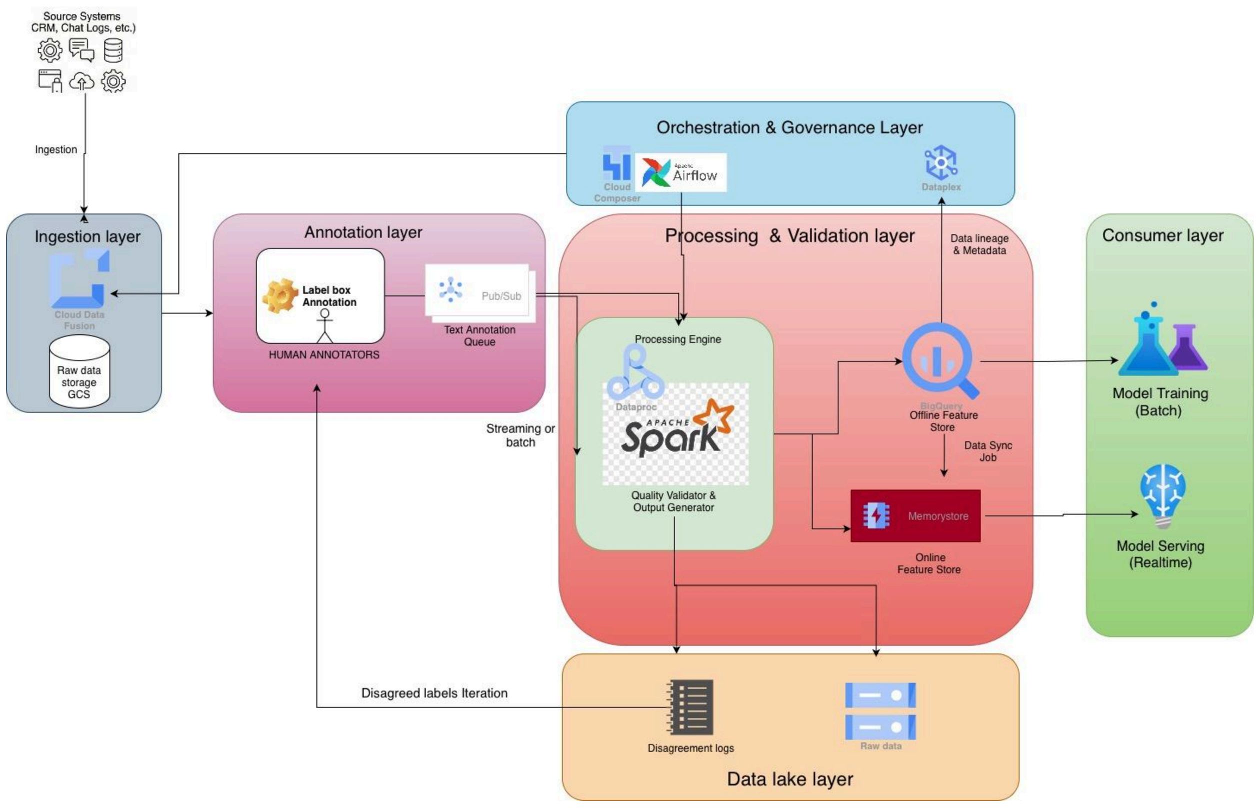
1.) Architectural Diagram



2.) Components Description

Ingestion Service: Cloud Data Fusion

High-level, managed ETL/ELT Ingestion service with built-in connectors for databases, files and SaaS apps, reducing the need to write complex connector code.

Annotation Queue: Pub/Sub

Serverless, fully managed, high-throughput message queue native to the GCP ecosystem. Ensures decoupling and reliable delivery of text samples for annotation and labels back to the system. This service is used to continuously stream the data as annotations are being labeled to Apache spark for processing from annotation platform such as label box, label studio etc.

Orchestration: Apache Airflow (via Cloud Composer)

Industry standard for managing complex DAGs. Composer provides a managed Airflow environment on GCP, simplifying scaling and maintenance. Ideal for managing job dependencies (e.g., ensuring feature engineering starts only after ingestion is complete). Also provides the Monitoring of jobs, Scheduling and Backfill strategy.

Processing Engine: Apache Spark (via Dataproc)

Provides distributed, scalable compute power essential for feature engineering, running quality validation checks across massive datasets, and performing backfills. Dataproc is the managed Spark service on GCP.

Offline Feature Store: Bigquery

BigQuery is a fully serverless managed Google Data warehouse, It has native time travel, which is crucial for point-in-time correctness. BigQuery is excellent for large-scale analysis with high performance and scalability.

OnlineFeature Store: MemoryStore

Extremely fast, in-memory key-value store providing the **sub-millisecond latency** required for real-time model serving lookups. Memorystore is the managed Redis/Memcached service on GCP.

Data lake: Google Cloud Storage

This serves as the foundation of your Data Lake architecture. It is the central, highly durable, and cost-effective storage layer for all raw data. GCS is used here to store the outputs of the **Quality Validator**, specifically the files containing the records that failed quality checks or agreement checks. This data is kept separate from the clean training set for auditing and for the Disagreed Labels Iteration loop.

3.) Technology Justification

Cloud Data Fusion

Chosen for its managed, low-code interface and extensive pre-built connectors, minimizing custom code development for common data sources like CRMs and databases. This reduces maintenance overhead and accelerates the initial data onboarding process.

Google Cloud Pub/Sub

Selected as a serverless, high-throughput message queue that reliably decouples the fast ingestion rate from the slower human annotation workflow. Its managed nature native to GCP simplifies scaling and operational management.

Apache Airflow (via Composer)

Used as the central control plane for defining, scheduling, and monitoring complex workflow dependencies across all layers (Ingestion →Processing → Materialization). Composer provides a scalable, managed Airflow environment.

Apache Spark (via Dataproc)

Provides the essential distributed computing power necessary for large-scale feature engineering, running quality checks, and efficiently executing massive historical backfill jobs across terabytes of data.

BigQuery

Delta Lake is chosen for its Time Travel and ACID properties, guaranteeing point-in-time correctness required for training data integrity. BigQuery serves as a highly scalable analytics warehouse for large-scale historical queries.

Memorystore

Selected for its sub-millisecond latency and high throughput to serve the latest feature values needed for real-time model predictions. It acts as a fast, in-memory cache synchronized from the Offline Store.

Google Cloud Storage (GCS)

The storage layer, chosen for its extreme durability, security and cost-efficiency to store all raw, immutable data files. Its native integration with Dataproc and BigQuery makes it the scalable backbone of the entire data lake.

4.) Data Governance Implementation

Data Lineage Tracking (Source → Feature)

- Implementation: Every record is tagged with its Raw Data GCS Path and the Airflow DAG Run ID upon ingestion and processing. This ensures that any final feature vector can be traced back to the exact input data file and the specific execution run that created it.

Code Versioning (Transformation Logic)

- Implementation: The feature engineering code within the Quality Validator is strictly managed in a Git repository. Airflow passes the Git Version Tag (e.g., `v1.2.5`) as a parameter to the Spark job, and this tag is stored as the `feature_code_version` column in the Offline Feature Store.

Label Versioning (Point-in-Time)

- Implementation: Labels from the Annotation Layer are stored with a Label Creation Timestamp. The Offline Feature Store uses this time when generating training datasets to ensure Point-in-Time Correctness, preventing data leakage by guaranteeing that feature values are always historical relative to the label.

Central Metadata Registry

- Implementation: A Data Catalog or relational database indexes all gathered metadata (source paths, code versions, execution logs) to provide a single, searchable repository for auditing and reproducibility. This allows data scientists to easily query and understand the history of any feature set.