## SOURCE IDENTIFICATION

1. **Social Media:** The data generated from social media will be textual. On that note, the format will be in **.txt**. But by assumption, if the social media in question is **X**, formerly known as **Twitter**, the data will be in JSON format because we will be using the Twitter API to extract such.

2. **Call center logs:** Generally, logs are in .log format.

3. **SMS:** SMS are generally in .txt format.

4. **Website Forms:** The data collected from Website forms will be in CSV or XLSX format.

## INGESTION STRATEGY

1. **Social Media:** For social media, it is safe to use API Ingestion with Streaming. Reason being that Platforms like **FB**, **Instagram,** or **X**(Formerly known as **Twitter**) continuously generate data. So the API pulls such data in real-time or near real-time.

2. **Call Center Logs:** Logs generate data periodically(i.e, daily, weekly, monthly. This all depends on the structure logic governing the underlying system. Therefore, the Ingestion strategy for this data source will be **File Upload**(Batch Processing)

3. **SMS:** Just like social media, SMS will adopt the API Ingestion with Streaming. Reason being that SMS are also continuous data. An API is readily available to consume such data in real-time or near real-time.

4. **Website Forms:** Website forms are a little tricky here. Forms are submitted in real-time, which makes it streaming, but if volume is low, it can be batched.. An API is used to ingest such data. On that note, I would say API Ingestion with Streaming/Batch

## PROCESSING AND TRANSFORMATION

1. **Social Media:** One of the cleaning steps to take is the removal of metadata, removal of emojis, or unwanted texts. For standardization, we will be interested in **username or sender_name**, **message** or **complaint**, and **timestamp(for cases where we may have YYY-MM-DD)**.
2. **Call center logs:** We will adopt the same standardization step we took for Social Media. We get the **username or sender_name**, **complaint** or **message**, and **timestamp**.
3. SMS: Same method as the aforementioned. **Sender** or **username**, **message** or **complaint**, and **timestamp**.
4. **Website Forms:** These are naturally structured data. So we standardize it by collating the **sender** or **username**, **message** or **complaints**, and **timestamp**.

**Classifying complaints into categories:** We will have to include a logic to search through the messages or complaints sent out for special **keywords** that will include catch phrases for **PoorNetwork**, **Billing errors,** or **bad customer service.** With these keywords identified in our message, we can then group them into separate entities.

## STORAGE OPTIONS

In our data pipeline, we will need both the **Data Lake** and the **Warehouse**. A data lake will be regarded as our one source of truth, whereby it holds all the different data from different sources in their respective formats. Then, once the transformation is done, we can move the standardized data into a data warehouse for analytics and reporting. The cleaned data will be in parquet format. We will convert all data from the various sources to parquet. We will be dealing with millions of user complaints. Parquet is best because it will be smaller in size.

## SERVING

**Querying:** The data querying strategy depends on the storage platform used. Since our transformed data is in the Warehouse, it is safe to assume we will be using SQL queries for it.
**Downstream Users' Usage:** Every complaint will be channeled to their respective support teams for handling. The teams may include Network engineers(for network complaints), the Billing Team( for billing complaints), the BI Team, ML Team, etc..

## ORCHESTRATION AND MONITORING

**Pipeline schedule:** A hybrid approach can be used since we are dealing with both batch processing( log data, which can run daily, hourly, etc.) and stream processing(for sms, web forms, and social media data).
**Failure detection:** Monitor metrics to account for API responses, job completion or failure alerts, and storage errors, and input measures to check for data quality during transit in the pipeline. Proper logging can also help detect and alert engineers to errors.

## DATAOPS

The pipeline can run on.

1. On premises,
2. On cloud, or
3. Hybrid.

Before making the pipeline production, alot of factors and conditions are to be put in place.  These conditions involve Security, scalability, and Reliability, etc.