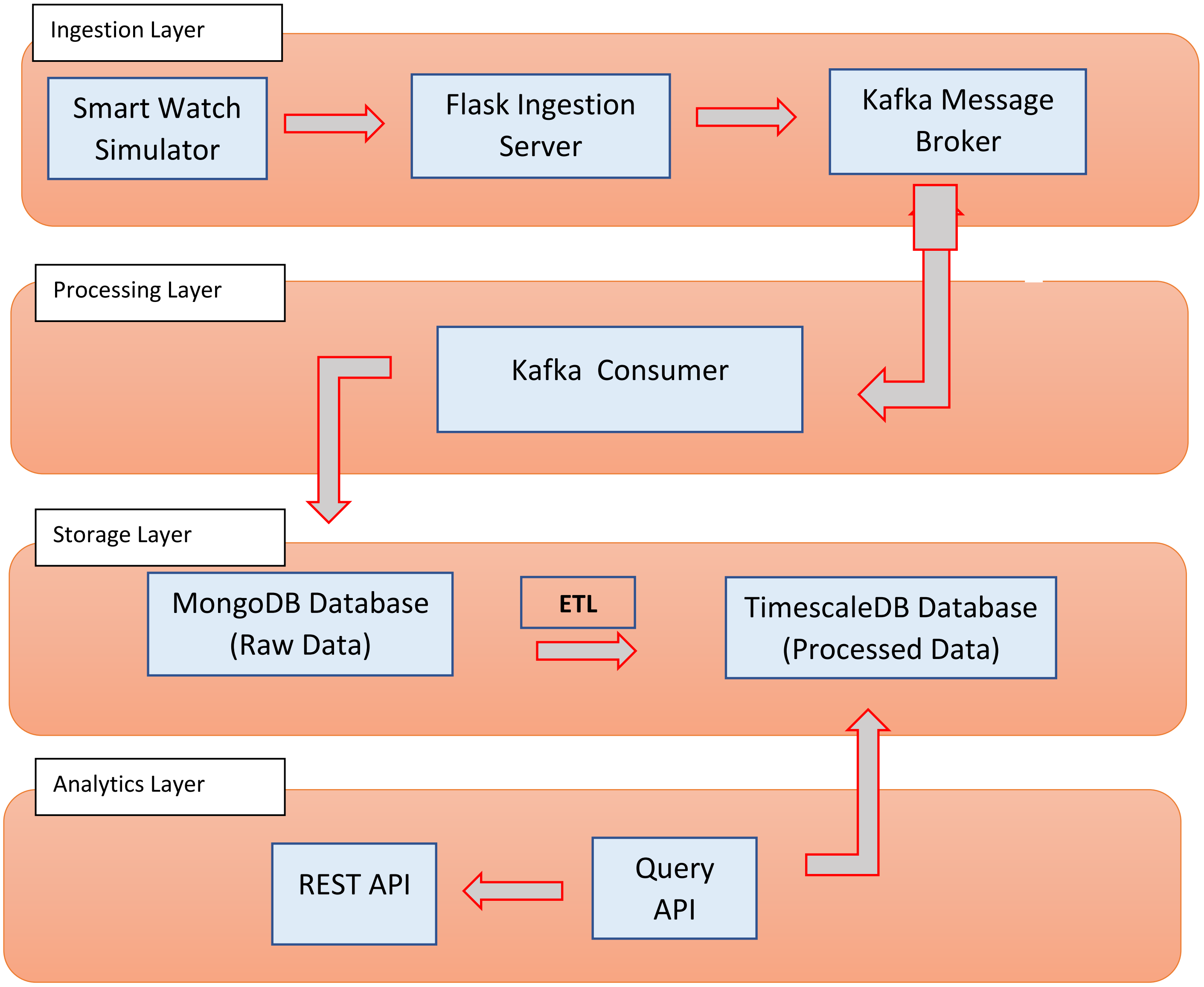
Smartwatch Data Pipeline Architecture : Justification



Data Sources & Ingestion

Used Flask as Ingestion server, because it provides a lightweight, easy-to-implement REST API for receiving data from devices. Also, there's a low overhead, quick to deploy, and sufficient for handling HTTP requests from IoT devices.

Message Broker

Used Kafka as message broker. It is ideal for handling high-volume, streaming data from numerous devices. Also, it

provides buffering between ingestion and processing,

ensures data durability with replication

allows for multiple consumers of the same data stream.

MongoDB for raw data

NoSQL document store is perfect for raw, semi-structured IoT data. Because its schema flexibility handling various data types horizontal scaling for handling large volumes and efficient storage of JSON-like data.

TimescaleDB for processed data

TimescaleDB is optimized for time-series data**,** which is exactly what the smartwatch data where data points associated with timestamps.

Also, its fast queries**,** aggregations, and time-based indexing**,** make it a perfect fit for efficiently querying.

Data Processing & Transformation

Minimal Transformations

The data receiving from client is mostly in a raw format and needs only basic cleaning and simple transformations according this scenario.

There for no need for complex processing such as joins from other sources, aggregation across different datasets, or real-time event-driven operations.

<u>So, I decided to go with direct transfer approach. (MongoDB to TimescaleDB)</u>

Direct transfer from MongoDB to TimescaleDB simplifies the pipeline by minimizing overhead. We can avoid unnecessary complexity from stream processors since the data doesn't require real-time processing or complex joins.

Data from MongoDB will be cleaned up and restructured before insertion into TimescaleDB using SQL scripts. This minimizes the need for a complex processing layer.

Query API

The Query API is built using Flask, which exposes endpoints for querying time-series data stored in TimescaleDB (PostgreSQL).

The APIs can retrieve data from the database based on time-based queries

e.g., user’s daily steps summary for the past month, avg heart rate among different age categories