High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **HealthTrends - Distributed Health Data Analyzer**. The project aims to build a distributed pipeline for ingesting, storing, processing, and analyzing anonymized health data using big-data technologies, producing summary reports for healthcare trend analysis.

---

1. System Architecture Overview

Architecture Description:
The system consists of four main modules: data ingestion (Kafka), distributed storage (HDFS), data processing (PySpark), and workflow orchestration/report generation (Python scripts). Data flows sequentially through these components to produce analytical summaries.

Main System Components

Component	Role/Function
Kafka	Ingests and streams anonymized health records into the pipeline
HDFS	Stores raw and processed health data in a distributed, fault-tolerant manner
PySpark	Processes and analyzes data to extract trends and generate summaries
Python Scripts	Orchestrate workflow, trigger jobs, and output final CSV summary reports

---

2. Component Interactions

Step	Source Component	Target Component	Interaction Description
1	External Source	Kafka	Health data records are published to Kafka topics
2	Kafka	HDFS	Data is consumed from Kafka and stored in HDFS
3	HDFS	PySpark	PySpark jobs read data from HDFS for processing
4	PySpark	HDFS	Processed results and summaries are written back to HDFS
5	HDFS	Python Scripts	Scripts extract summaries and output CSV reports

---

3. Data Flow Overview

Data Stage	Description
Ingestion	Anonymized health records streamed into Kafka

Storage	Raw records persisted in HDFS
Processing	PySpark jobs analyze data, extract trends, and generate summaries
Output	Python scripts export summary reports as CSV files

---

4. Technology Stack

Technology	Purpose/Usage
Python	Workflow orchestration, scripting
Kafka	Distributed data ingestion/streaming
HDFS	Distributed data storage
PySpark	Distributed data processing/analytics

---

5. Scalability & Reliability

- **Scalability:**
  - Kafka and HDFS support horizontal scaling to handle increased data volume.
  - PySpark enables distributed processing across multiple nodes for large datasets.
- **Reliability:**
  - HDFS provides data replication and fault tolerance.
  - Kafka ensures reliable message delivery and buffering.
  - Modular design allows independent scaling and maintenance of components.
- **Security:**
  - Data is anonymized before ingestion.
  - Access controls and secure configurations are recommended for all components.

End of Document