Health Monitoring System

1. Introduction

The **Health Monitoring System** is designed to analyze and track patient health data using **Big Data technologies** such as **Apache Spark, Hadoop, Kafka, and NoSQL databases**. This system processes health records of **10,000 patients**, identifies trends in health parameters, and provides real-time insights to doctors while also collecting patient feedback for further analysis.

2. Objectives

- **Generate** 10,000 patient profiles with health parameters.
- **Process** patient data using **Hadoop (MapReduce) and Apache Spark**.
- **Perform** statistical analysis on patient health records.
- **Stream** processed data to doctors via **Kafka**.
- **Store** patient feedback in a **NoSQL database**.
- **Analyse** feedback using **Machine Learning** for sentiment classification.
- **Implement database sharding** for scalability.

3. System Architecture

The system follows a **Lambda architecture**, which consists of three layers:

1. **Batch Layer (Hadoop & MapReduce):** Processes historical patient data stored in HDFS.
2. **Speed Layer (Apache Spark & Kafka):** Streams new patient data for real-time analysis.
3. **Serving Layer (NoSQL Database & Dashboard):** Stores processed data and provides visualization.

4. Implementation Details

4.1 Data Generation

- **Tools Used:** Python (Faker Library)
- **Generated Data:**
    - Patient ID
    - Name ○
    - Age
    - Blood Pressure (BP)

    - Sugar Level
    - Cholesterol
    - Hemoglobin
- **Storage:** HDFS (Hadoop Distributed File System)

4.2 Data Processing

- **Tools Used:** Apache Spark, Hadoop MapReduce
- **Processing Steps:**
    1. Load patient data from HDFS.
    2. Compute average values for BP, Sugar, Cholesterol, and Hemoglobin.
    3. Categorize patients into risk groups.
    4. Store processed data in HDFS.

4.3 Real-Time Streaming

- **Tools Used:** Apache Kafka
- **Process:**
    1. Processed data is published to a Kafka topic.
    2. Doctors subscribed to Kafka receive real-time updates.
    3. Patients provide feedback through a web interface.

4.4 NoSQL Database for Feedback Storage

- **Database Used:** MongoDB
- **Data Structure:** {"patient_id": "UUID", "feedback": "String"}

4.5 Feedback Sentiment Analysis

- **Tools Used:** NLP (TextBlob, VADER)
- **Process:**
    1. Analyze patient feedback to determine sentiment (Positive/Negative).
    2. Generate trends on patient satisfaction.

4.6 Database Sharding

- **Technique Used:** MongoDB Sharding

- **Purpose:** Distributes patient records across multiple database nodes for scalability.

5. Dashboard & Visualization

- **Tools Used:** Flask, D3.js, Grafana
- **Features:**
    - Display **patient health statistics** (graphs & charts) ○ Show **real-time streaming data**
    - Display **patient feedback trends**

6. Results & Discussion

- Successfully processed **10,000 patient records** using Hadoop & Spark.
- Provided real-time **health insights to doctors** using Kafka.
- Collected and analyzed **patient feedback using NLP**.
- Implemented **scalable storage solutions with NoSQL and database sharding**.

7. Future Enhancements

- **Integration with Wearable Devices:** Collect real-time patient vitals.
- **Predictive Analytics:** Use **Machine Learning** to predict health risks.
- **Cloud Deployment:** Deploy the system on AWS/Azure for scalability.

8. Conclusion

This **Health Monitoring System** demonstrates how **Big Data technologies** can effectively process and analyze **large-scale health records**. By leveraging **Apache Spark, Hadoop, Kafka, and NoSQL databases**, the system provides real-time health insights, patient feedback analysis, and scalable data storage solutions.

Technologies Used:

- **Big Data:** Hadoop, Apache Spark
- **Real-time Streaming:** Apache Kafka
- **Database:** MongoDB (NoSQL), HDFS
- **Machine Learning:** NLP for Sentiment Analysis
- **Visualization:** Flask, Grafana, D3.js