**Lovely Professional University**

**Name: Tejasva Agrahari**

**Reg no.: 12221307**  
  
**B.tech Computer Science And Engineering**

**Big Data Health Monitoring System - Project Report**

**1. Introduction**

**1.1 Project Overview**

The **Health Monitoring System** is a big data-driven solution designed to process and analyze patient health parameters efficiently. By leveraging **Apache Spark, Hadoop, Kafka, and NoSQL databases**, this system can manage large-scale patient data, perform real-time analytics, and offer predictive insights.

**1.2 Objectives**

* Generate and process health records for **10,000 patients**.
* Utilize **Apache Spark & Hadoop** for large-scale data processing.
* Implement **real-time data streaming** via **Apache Kafka**.
* Store processed records in a **NoSQL database (MongoDB/Cassandra)**.
* Provide **data visualization** for better insights.
* Explore **Machine Learning models** for predictive analytics (optional).

**2. Technologies Used**

| **Technology** | **Purpose** |
| --- | --- |
| **Apache Spark** | Distributed data processing |
| **Hadoop (HDFS)** | Storing large patient datasets |
| **Apache Kafka** | Real-time data streaming |
| **NoSQL (MongoDB/Cassandra)** | Fast retrieval and storage |
| **Tableau/Power BI** | Data visualization |
| **Machine Learning (Scikit-Learn, TensorFlow)** | Predictive analytics (optional) |

**3. System Architecture**

**3.1 Workflow**

1. **Data Generation:**
   * Simulating data for **10,000 patients** with health metrics.
   * Each record includes **BP, Sugar, Cholesterol, and Hemoglobin levels**.
2. **Data Storage:**
   * Raw data stored in **HDFS** for distributed access.
   * Processed insights stored in **NoSQL** with **(Patient ID, Health Metrics)**.
3. **Data Processing with Spark:**
   * **MapReduce** operations to compute insights.
   * Identifying high-risk patients based on defined thresholds.
4. **Real-Time Streaming with Kafka:**
   * **Doctors receive live updates** via Kafka topics.
   * **Patients receive alerts** on abnormal health readings.
5. **Visualization & Analytics:**
   * Dashboards created in **Tableau or Power BI**.
   * Displaying real-time and historical health trends.
6. **Predictive Analysis (Advanced Feature):**
   * Using **ML models** to predict potential health risks.
   * Personalized recommendations based on trends.

**4. Implementation Details**

**4.1 Data Processing Using Apache Spark**

* Apache Spark reads patient data from **HDFS**.
* Performs **MapReduce** for:
  + Calculating **average BP, Sugar, Cholesterol levels**.
  + Identifying high-risk patients.
* Stores insights in a **NoSQL database** for fast retrieval.

**4.2 Real-Time Processing using Kafka**

* Kafka **producers** push processed health stats.
* **Consumers (Doctors & Patients)** receive real-time updates.

**4.3 Storage in NoSQL Database**

* Patient records stored in **MongoDB / Cassandra** for scalability.

**4.4 Data Visualization**

* **Tableau / Power BI dashboards** show health trends.
* Insights on **BP trends, sugar variations, risk analysis**.

**4.5 Code Implementation**

The following Python code demonstrates the **data processing workflow using Apache Spark, Kafka, and NoSQL databases**:

from pyspark.sql import SparkSession

from pyspark.sql.functions import col, avg

# Initialize Spark Session

spark = SparkSession.builder.appName("HealthAnalysis").getOrCreate()

# Load Data (Assuming CSV format stored in HDFS)

df = spark.read.csv("hdfs://path\_to\_patient\_data.csv", header=True, inferSchema=True)

# Compute Average Health Metrics

health\_avg = df.groupBy("PatientID").agg(

avg("BloodPressure").alias("Avg\_BP"),

avg("SugarLevel").alias("Avg\_Sugar"),

avg("Cholesterol").alias("Avg\_Cholesterol")

)

# Save Processed Data to NoSQL (MongoDB Example)

health\_avg.write.format("mongo").mode("append").option("uri", "mongodb://localhost:27017/health\_db.patients").save()

print("Data processing completed successfully!")

**5. Results & Discussion**

**5.1 Key Findings**

* Processed health data for **10,000 patients** efficiently.
* Enabled **real-time alerts** using Kafka.
* NoSQL database ensures **fast access to patient records**.
* Predictive analytics identified **high-risk patients**.

**5.2 Challenges & Solutions**

| **Challenges** | **Solutions** |
| --- | --- |
| Handling large-scale data | Optimized Spark jobs |
| Real-time streaming delays | Kafka topic partitioning |
| NoSQL performance issues | Used **indexing & sharding** |

**6. Conclusion & Future Scope**

**6.1 Conclusion**

This project demonstrates how **big data technologies** enhance healthcare analytics. The system enables:

* Efficient patient health monitoring.
* Real-time alerts and predictive analytics.
* Scalable data storage and processing.

**6.2 Future Enhancements**

* **Deep Learning** for **more accurate health predictions**.
* AI-powered **chatbot** for automated health insights.
* Deploying as a **cloud-based service** for scalability.

**Appendix**

* **Source Code:** Attached in the project repository.
* **Dataset:** Simulated for 10,000 patients.
* **Tools Used:** Apache Spark, Hadoop, Kafka, MongoDB, Tableau/Power BI.