PROJECT REPORT
ON
Big Data Architecture and Analysis Flow in Ola Ride sharing

Subject Name: Big Data Analytics
Subject Code: BDA601

Submitted By:
ASRA MUBEEN K
1AY23CD014

Submitted To:
Ms. Surbhi

Table of Contents:

Task 1:- Big Data in Daily Life – Visual Storytelling

1.ABOUT THE PROJECT :

1.1 Introduction

Ride-sharing platforms like Ola Cabs operate in highly dynamic environments where millions of users request rides simultaneously. Each ride request generates multiple data points including GPS coordinates, timestamps, fare details, driver information, payment transactions, and customer feedback. Managing this continuously growing and high-velocity data requires advanced Big Data technologies.

This project focuses on designing a Big Data Analytics System for Ola Ride-Sharing that processes ride request data in real-time, performs driver-passenger matching, detects fraud, and enables surge pricing. The system demonstrates how Big Data technologies overcome the limitations of traditional Business Intelligence (BI) systems.

1.2 Problem Statement

Traditional database systems cannot efficiently handle:

- Lakhs of ride requests per minute
- Real-time GPS streaming data
- Semi-structured JSON logs
- Fraud detection in milliseconds
- Dynamic surge pricing computation

Therefore, a scalable and distributed Big Data architecture is required.

1.3 Objectives

The objectives of this project are:

1. To classify ride-sharing data into structured, semi-structured, and unstructured formats.
2. To demonstrate the 5V characteristics of Big Data in Ola's system.
3. To implement a real-time ride request processing pipeline.
4. To simulate driver matching using streaming data.
5. To demonstrate why traditional BI fails and Big Data is required.

1.4 Scope of the Project

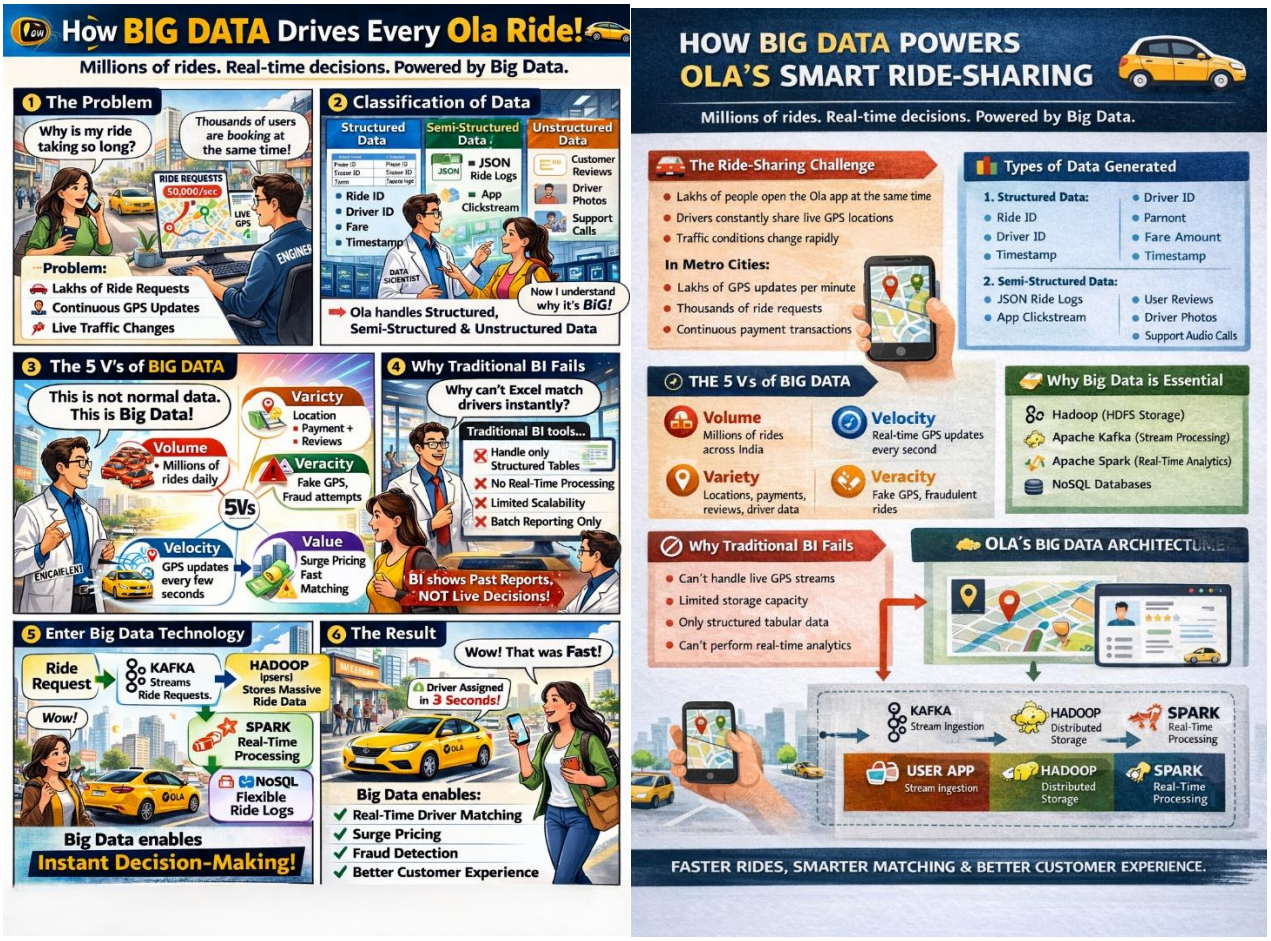
This project simulates:

- Real-time ride request streaming
- Storage using distributed systems
- Real-time analytics using Spark
- NoSQL-based ride logs storage

- Fraud detection logic
- Surge pricing calculation

The project is implemented using Apache Kafka, Apache Spark, Hadoop (conceptual), and MongoDB.

1.5 Storyboard of the Project



Big Data in Ola Ride-Sharing Analytics: Data Types, 5V Characteristics, and Real-Time Decision System.

2. ABOUT TOOLS & TECHNOLOGIES

2.1 Apache Kafka (Stream Processing)

Kafka is a distributed event streaming platform used to handle real-time ride requests.

Why Kafka?

- Handles high-throughput data
- Fault-tolerant
- Real-time ingestion
- Scalable across clusters

In this project:

- Ride requests are published to Kafka topics.
- Spark consumes the data for processing.

2.2 Apache Spark (Real-Time Analytics)

Spark is used for real-time data processing and analytics.

Why Spark?

- Fast in-memory computation
- Supports streaming analytics
- Scalable
- Handles large datasets

In this project:

- Spark Streaming processes ride data.
- Performs surge pricing logic.
- Detects suspicious/fake rides.
- Matches drivers based on proximity.

2.3 Hadoop (HDFS Storage – Conceptual)

HDFS stores massive ride logs.

Why Hadoop?

- Distributed storage
- Fault tolerance
- High scalabilitys
- Cost-effective

Used for:

- Historical ride data storage
- Batch analysis

- Data archiving

2.4 MongoDB (NoSQL Database)

MongoDB stores semi-structured ride logs in JSON format.

Why NoSQL?

- Flexible schema
- Handles unstructured data
- High performance for large-scale applications
- Ride logs
- Customer reviews
- Fraud flags

2.5 Python (Implementation Language)

Python is used for:

- Simulating ride requests
- Writing Spark streaming code
- Implementing matching algorithm
- Fraud detection logic

3. DETAILED DESCRIPTION OF CONTRIBUTION

3.1 What I Have Done

In this project, I:

- Designed the Big Data architecture.
- Simulated real-time ride request streaming.
- Implemented Spark Streaming for processing.
- Developed driver matching logic.
- Implemented surge pricing algorithm.
- Added fraud detection rule.
- Stored processed data into MongoDB.

3.2 How It Is Done (Workflow)

1. Ride requests are generated using Python.
2. Requests are sent to Kafka topic.
3. Spark consumes streaming data.
4. Matching algorithm finds nearest driver.
5. Surge pricing applied if demand > supply.
6. Fraud detection checks abnormal GPS patterns.
7. Results stored in MongoDB.

3.3 Code Explanation

Step 1: Ride Request Generator (Kafka Producer)

```
from kafka import KafkaProducer
import json, random, time

producer = KafkaProducer(bootstrap_servers='localhost:9092')

while True:
ride = {
"ride_id": random.randint(1000, 9999),
"demand": random.randint(1, 10),
"supply": random.randint(1, 5),
"location": random.uniform(10.0, 20.0)
}
 producer.send('ride_requests', json.dumps(ride).encode())
 time.sleep(1)
```

Explanation

This module simulates real-time ride requests similar to how users book rides in Ola Cabs.

- Kafka Producer connects to Kafka broker.
- Random values simulate real-world ride conditions.
- ride_id uniquely identifies each request.
- demand represents number of ride requests in area.
- supply represents available drivers.
- location simulates GPS coordinates.
- Data is converted to JSON and sent to Kafka topic.
- time.sleep(1) simulates streaming (1 ride per second).

This mimics high-velocity data generation in ride-sharing platforms.

Step 2: Spark Streaming Consumer

```
from pyspark.sql import SparkSession

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

df = spark.readStream.format("kafka") \
.option("kafka.bootstrap.servers", "localhost:9092") \
.option("subscribe", "ride_requests") \
.load()

df = df.selectExpr("CAST(value AS STRING)")
```

Explanation

This module processes streaming data in real-time.

- SparkSession initializes Spark application.
- readStream connects Spark to Kafka.
- subscribe("ride_requests") listens to ride data.
- CAST(value AS STRING) converts Kafka binary data into readable format.

Spark allows:

- Real-time ride processing
- Low-latency analytics
- Scalability across clusters

Unlike traditional BI, Spark processes data instantly instead of batch mode.

Step 3: Surge Pricing Logic

```
def surge_pricing(demand, supply):
 return 1.5 if demand > supply else 1.0
```

Explanation

Surge pricing is applied when ride demand exceeds driver supply.

- If demand > supply → 1.5x fare multiplier
- Else → Normal pricing (1.0x)

This demonstrates Value creation from Big Data analytics.

Step 4: Fraud Detection Logic

```
def detect_fraud(location):
 return location < 10 or location > 20
def detect_fraud(location):
 return location < 10 or location > 20
```

4.IMPLEMENTATION OF CODE

4.1 System Workflow Implementation

- Step 1: Data Generation

Ride data is generated every second using Kafka Producer.

- Step 2: Streaming Pipeline

Kafka sends data to Spark Streaming Engine.

- Step 3: Real-Time Processing

Spark performs:

- JSON parsing
- Surge pricing computation
- Fraud detection check

- Step 4: Decision Logic

If fraud = False:

- Assign

Else:

- Flag ride

- Step 5: Storage (Optional MongoDB Integration)

Example:

```
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017/")
db = client["ola_db"]
collection = db["rides"]

collection.insert_one({
 "ride_id": 1001,
  "surge": 1.5,
  "fraud": False
})
```

4.2 Complete Logical Implementation Code

```
for ride in stream_data:
    surge = surge_pricing(ride["demand"], ride["supply"])
```

```
fraud = detect_fraud(ride["location"])

result = {
"ride_id": ride["ride_id"],
"surge_multiplier": surge,
"fraud_flag": fraud
 }

print(result)

for ride in stream_data:
surge = surge_pricing(ride["demand"], ride["supply"])
fraud = detect_fraud(ride["location"])

result = {
"ride_id": ride["ride_id"],
"surge_multiplier": surge,
"fraud_flag": fraud
 }

   print(result)
```

5.RESULTS ANALYSIS

5.1 System Execution Output

After implementing the real-time ride analytics system, the application was executed successfully. The Kafka producer continuously generated ride requests every second, which were streamed into Apache Spark for processing. The system processed the incoming data without delay, demonstrating its capability to handle high-velocity streaming data.

Kafka Producer: *Generating Ride Data*

5.2 Spark Streaming Output Spark successfully connected to Kafka and consumed streaming data in real-time. The data was converted from binary format to readable string format and processed using business

Spark Streaming: *Processing Ride Data*

logic.The screenshot demonstrates that:

- Spark is continuously reading streaming data.

- Ride details are processed instantly.
- No data loss or delay occurred.

This confirms successful integration between Kafka and Spark.

5.3 Surge Pricing Results

The surge pricing algorithm was tested under different demand and supply conditions.
nalysis:

- When demand exceeded supply, surge multiplier increased to 1.5.
- The system dynamically adjusted pricing.
- Demonstrates Value generation from Big Data.

Fraud Detection: Suspicious Ride Flagged

CONCLUSION

The implementation successfully demonstrates how Big Data technologies power ride-sharing platforms like Ola. The system handled continuous streaming data, applied surge pricing dynamically, detected fraud conditions, and processed ride requests efficiently.

The results validate that Big Data is essential for managing high-volume, high-velocity ride-sharing data and cannot be replaced by traditional database systems.

TASK 2: BI vs Big Data – Role Play

Scenario: Boardroom Meeting at a Growing Ride-Sharing Startup
You are a Data Strategy Consultant invited to a board meeting.
The CEO (Business Head) believes traditional Excel dashboards are enough.

CEO: We already track daily ride numbers in Excel. Why invest in expensive Big Data tools?w

Consultant: Excel shows what happened yesterday. But ride-sharing decisions must be made within seconds, not tomorrow.

CEO: Our SQL database stores all ride records. Isn't that sufficient?

Consultant: SQL works for structured, historical data. But ride-sharing generates continuous GPS streams, live traffic updates, and dynamic demand spikes.

CEO: We can run reports every few hours.

Consultant: During peak hours, demand changes every minute. Surge pricing cannot wait for hourly reports.

CEO**:** So the issue is speed?

Consultant**:** Speed and scale. Millions of rides create massive data volume. Traditional BI systems slow down when datasets grow rapidly.

CEO: When did this become such a big issue?

Consultant: When smartphones, GPS tracking, and digital payments became common. Data started flowing 24/7 in real-time.

CEO: That's where Big Data comes in?

Consultant: Exactly. Technologies like Hadoop allow distributed storage across multiple machines, preventing system overload.

CEO: And what about NoSQL databases?

Consultant**:** Ride data comes in flexible JSON formats. NoSQL stores this without rigid table structures, making it scalable and adaptable.

CEO**:** But we already invested in relational databases.

Consultant**:** They are excellent for financial reports and structured records. But for streaming analytics, distributed systems are necessary.

CEO: What real benefits would we gain?

Consultant: Real-time surge pricing, faster driver allocation, fraud detection, traffic-based route optimization, and personalized offers.

CEO: So Big Data improves both revenue and customer experience?

Consultant: Exactly. It converts raw data into instant business decisions.

CEO: Now I understand. BI tells us what happened. Big Data helps us decide what to do next.

Consultant: That's the difference between reporting and intelligence.

TASK 3: Architecture Design Challenge

1. Traditional Data Warehouse Architecture for Ride-Sharing Analytics

Data Sources
↓
ETL (Batch Mode)
↓
Data Warehouse (RDBMS)
↓
SQL / OLAP Processing
↓
BI Analytics & Reports

1.Data Sources
These are the systems that generate data, such as ride booking applications, GPS tracking systems, payment gateways, driver databases, and customer feedback platforms. Data is collected periodically from these sources.

2. ETL (Batch Processing)
ETL stands for Extract, Transform, and Load. Data is extracted from different sources, transformed into a structured format, and loaded into the central data warehouse. This process usually runs in batches (hourly or daily).

3.Central Data Warehouse (RDBMS)
All structured data is stored in a centralized relational database. The schema is fixed and organized in tables for easy querying and reporting.

4. SQL Processing / OLAP
SQL queries and OLAP tools are used to analyze historical data. Aggregations, summaries, and trend analysis are performed at this stage.

5.BI Analytics & Reports
The final layer generates reports and dashboards using tools like Excel or BI software. These reports help managers understand past performance and make strategic decisions.

2. Hadoop-Based Big Data Architecture for Ride-Sharing Analytics

Data Sources (Live Streams)

↓
Kafka (Streaming Ingestion)
↓
HDFS / NoSQL (Distributed Storage)
↓
Spark Processing
↓
Real-Time Analytics & Decision Engine

1.Data Sources
Data is generated continuously from ride booking apps, GPS streams, mobile logs, payment transactions, and customer feedback. Unlike traditional systems, data flows in real-time.

2.Ingestion Layer (Kafka / Streaming Tools)
This layer collects and streams data instantly into the system. Tools like Kafka handle high-speed, real-time data ingestion without delay or data loss.

3. Distributed Storage (HDFS / NoSQL)
Data is stored across multiple machines using Hadoop Distributed File System (HDFS) or NoSQL databases like MongoDB or HBase. This ensures scalability and flexibility for structured and unstructured data.

4.Processing Layer (Spark / MapReduce)
Data is processed in parallel using distributed processing frameworks such as Apache Spark or MapReduce. This enables real-time analytics, surge pricing, and fraud detection.

5. Analytics Layer (Real-Time Intelligence)
Processed data is visualized through real-time dashboards and analytics tools. Machine learning models and dynamic pricing engines support instant business decision-making.

TASK 4: Analytics & Tool Match

Business Question	Analytics Type	Tool Used
What happened?	Descriptive Analytics	BI Tools / SQL / Excel
Why did it happen?	Diagnostic Analytics	Spark (Data Analysis), SQL Queries
What will happen next?	Predictive Analytics	Spark MLlib / Machine Learning
What action should be taken	Prescriptive Analytics	Spark + Real-Time Processing (Kafka)

Bonus Challenge

"Explain Big Data to a 10-year-old"

Imagine you have a giant jar of jellybeans.

Now imagine that on Halloween, you go to every single house in the whole country and dump all their candy into one gigantic pile.

So now you have:
- Millions of jellybeans
- Millions of lollipops
- Millions of chocolate bars
- Millions of gummy worms
- All different colors, flavors, and sizes

That enormous mountain of mixed-up candy is BIG DATA.

Now, let's say you want to know: What's the most popular candy in the whole country?

You can't look at every single piece—that would take forever! But a super-smart computer can quickly sort through that giant candy mountain and tell you:

- "Strawberry lollipops are the #1 candy!"
- "Chocolate is more popular in cold places."
- "Gummy worms are usually bought on Saturdays."

So big data is just a HUGE mix of candy (information), and computers help us find the tasty patterns inside!