Project Requirements Document: Automotive Parts Supply Chain Data Integrator

1. Introduction

1.1. Project Overview

This document outlines the requirements for building an **Automotive Parts Supply Chain Data Integrator**. The project aims to create a robust data platform that centralizes, processes, and analyzes data from various parts of a simulated automotive parts supply chain. The core components will include a data warehouse for unified data storage, an automated ETL pipeline using Airflow and Kafka for data orchestration, and machine learning models for predictive analytics.

1.2. Goal

The primary goal is to demonstrate proficiency in data engineering and machine learning principles by creating a functional, end-to-end data platform. The project will showcase skills in:

- Data Ingestion (streaming and batch)
- ETL/ELT pipeline design and orchestration
- Data Warehouse design and implementation
- Integration of Machine Learning models into data pipelines
- Data visualization and reporting

2. Functional Requirements

The system must perform the following functions:

2.1. Data Ingestion

- **FR-DI-01:** The system shall ingest real-time data streams from simulated manufacturing IoT sensors and logistics GPS trackers. This data will be crucial for real-time monitoring and predictive applications.
- **FR-DI-02:** The system shall ingest batch data from simulated operational systems, including supplier data (e.g., lead times, quality scores), inventory records (e.g., stock levels, movement), and historical sales (e.g., order history, returns).
- **FR-DI-03:** The system shall use **Apache Kafka** as the primary component for handling real-time data ingestion, ensuring high-throughput, low-latency, and fault-tolerant data streaming.

2.2. ETL and Orchestration

- **FR-ETL-01:** The system shall use **Apache Airflow** to orchestrate all data ingestion and transformation tasks. Airflow will define, schedule, and monitor the complex data workflows.
- **FR-ETL-02:** Airflow DAGs (Directed Acyclic Graphs) shall be designed to handle both

continuous data consumption from Kafka topics (for near real-time updates) and scheduled batch data loads (for periodic updates from operational systems).

- **FR-ETL-03:** The data pipelines must perform essential data quality operations, including cleaning (handling missing values, correcting inconsistencies), normalization (standardizing formats), and aggregation (summarizing data for analytical purposes) before loading into the data warehouse.
- **FR-ETL-04:** The pipelines must manage dependencies between tasks effectively, ensuring that a task only executes once its upstream dependencies have successfully completed (e.g., data transformation occurs only after raw data extraction).

2.3. Data Warehouse

- **FR-DW-01:** The project shall include the design and implementation of a dimensional data warehouse, primarily using a star or snowflake schema. This will enable efficient analytical querying.
- **FR-DW-02:** The data warehouse shall include:
  - **Fact Tables:** To store core business events and measurable metrics. Examples include fact_sales (quantity sold, revenue), fact_inventory_movement (stock-in/out), and fact_shipment (delivery times, status).
  - **Dimension Tables:** To store descriptive attributes related to the facts. Examples include dim_part (part details), dim_supplier (supplier information, performance), dim_date (time-based attributes), dim_warehouse (location, capacity), and dim_customer (customer segments).
- **FR-DW-03:** The data warehouse must be optimized for analytical queries, enabling fast retrieval and aggregation of data for reporting and machine learning model training.

2.4. Machine Learning Integration

- **FR-ML-01:** The project shall integrate at least one machine learning model into the data pipeline to derive actionable insights.
- **FR-ML-02:** A **demand forecasting model** shall be built and deployed to predict future sales for specific automotive parts. This model will be trained using historical sales data and other relevant features (e.g., seasonality, promotions) from the data warehouse.
- **FR-ML-03:** The output of the demand forecasting model (e.g., predicted quantities for upcoming periods) shall be loaded back into the data warehouse as a new table (e.g., fact_demand_forecast) or integrated into an existing fact table, making predictions available for business intelligence.
- **FR-ML-04:** (Optional, but highly recommended for a comprehensive project) A second ML model could be implemented. Potential examples include:
  - **Predictive Maintenance:** Using manufacturing IoT sensor data to predict equipment failures.
  - **Supplier Risk Scoring:** Assessing supplier reliability based on historical performance (on-time delivery, quality defects).
  - **Anomaly Detection:** Identifying unusual patterns in inventory levels or shipment delays.

2.5. Reporting and Visualization

- **FR-RV-01:** The project shall include a basic reporting mechanism or dashboard to visualize key performance indicators (KPIs) and insights derived from the data warehouse and ML models.
- **FR-RV-02:** At a minimum, the visualization should display:
  - Actual sales vs. forecasted sales (from the demand forecasting model).
  - Key supplier performance metrics (e.g., on-time delivery rate, defect rate).
  - Current and historical inventory levels for critical parts.

3. Non-Functional Requirements

3.1. Performance

- **NFR-PERF-01:** The real-time data ingestion pipeline (Kafka) should be able to handle a high volume of events (e.g., thousands of events per second) with minimal latency (e.g., sub-second processing).
- **NFR-PERF-02:** The ETL batch jobs orchestrated by Airflow should complete within acceptable timeframes (e.g., daily jobs should finish within a few hours to ensure data freshness for next-day operations).
- **NFR-PERF-03:** Analytical queries on the data warehouse should return results efficiently, supporting interactive dashboards and reports.

3.2. Reliability and Scalability

- **NFR-REL-01:** The Airflow pipelines shall be robust, incorporating error handling, logging, and retry mechanisms to ensure data processing continues even with transient issues.
- **NFR-REL-02:** The system architecture (Kafka, Airflow, Data Warehouse) should be designed to be scalable, allowing for the future addition of new data sources, increased data volume, and more complex machine learning models without requiring a complete re-architecture.
- **NFR-REL-03:** Data integrity must be maintained throughout the ETL process and within the data warehouse.

3.3. Maintainability

- **NFR-MAINT-01:** All code (Airflow DAGs, Kafka producers/consumers, data generation scripts, ML models, and data warehouse schema definitions) shall be thoroughly documented, well-commented, and follow established coding best practices.
- **NFR-MAINT-02:** The project environment setup should be easily reproducible (e.g., using a requirements.txt file for Python dependencies and potentially Docker Compose for infrastructure components).

3.4. Security (Conceptual for this project)

- **NFR-SEC-01:** While a full security implementation is out of scope for a demo project, general best practices for data handling should be considered (e.g., avoiding hardcoding credentials, basic access control for simulated data).

4. Technical Stack

- **Data Streaming/Ingestion:** Apache Kafka
- **ETL/Workflow Orchestration:** Apache Airflow
- **Data Warehouse:** A SQL-based database (e.g., PostgreSQL is suitable for a local/containerized setup; cloud options like Google BigQuery, Snowflake, or AWS Redshift could be chosen for a more production-like environment).
- **Programming Language:** Python (for data generation, Kafka producers/consumers, Airflow DAGs, and Machine Learning models).
- **Machine Learning Libraries:** Scikit-learn, Pandas, NumPy, Statsmodels, or TensorFlow/PyTorch if deep learning is introduced.
- **Data Generation:** Python scripts utilizing libraries like Faker for realistic synthetic data.
- **Visualization/Reporting:** A business intelligence tool (e.g., Tableau Public, Power BI Desktop) or a simple web application framework (e.g., Streamlit, Flask with a basic front-end).
- **Containerization (Highly Recommended):** Docker and Docker Compose for setting up and managing Kafka, Airflow, and the database components easily.

5. Data Sources (Simulated)

For the purpose of this project, all data sources will be simulated to enable a complete end-to-end demonstration without reliance on proprietary systems.

- **Simulated Real-time Data Streams:**
  - **Manufacturing IoT Sensor Data:** Python scripts will generate continuous streams of data representing machine status, temperature, vibration, error codes, and production counts for various parts. These streams will be published to dedicated Kafka topics.
  - **Logistics GPS/Telematics Data:** Python scripts will simulate vehicle movements by generating continuous streams of truck_id, shipment_id, timestamp, latitude, longitude, speed, and fuel_level data, published to Kafka topics.
- **Simulated Batch Data Files (CSV/JSON):**
  - **Historical Sales and Demand Data:** Files containing sale_date, part_number, quantity_sold, customer_id, region, and promotion_applied to train demand forecasting models.
  - **Supplier and Part Information:** Files containing supplier_id, supplier_name, part_number, part_name, lead_time_expected, unit_cost.
  - **Inventory Records:** Files containing warehouse_id, part_number, stock_level, last_updated_date, min_stock_level, max_stock_level.
  - **Supplier Performance Data:** Files containing supplier_id, order_id, promised_delivery_date, actual_delivery_date, quality_inspection_result (e.g., defects found).

Document Sign-off:

- **Prepared By:** Rohan Kumar Sharma

- **Date:** 24/08/25