Software Requirements Specification

(SRS)

Project Title: Supply Chain Data Engineering & Analytics on Azure Cloud

Prepared For: Capstone Project

Prepared By: Team 3

Date: 24/09/2025

1. Introduction

1.1 Purpose

The purpose of this document is to define the software requirements for a cloud-native supply chain analytics platform. The system will consolidate and process data from shipments, inventory, vendors, delivery logs, and claims to provide visibility, improve decision-making, and reduce operational inefficiencies.

1.2 Intended Audience

- Business Stakeholders: To track shipments, monitor inventory, and oversee claim resolution.
- Data Engineers: To build ingestion pipelines and ensure data quality.
- Developers: To implement FastAPI microservices for system interaction.
- Cloud Engineers: To deploy and manage Azure-based infrastructure.
- End Users: Operations and supply chain managers accessing dashboards and reports.

1.3 Intended Use

The system will be used for:

- Tracking shipment progress and delivery timelines.
- Monitoring inventory health and reordering needs.
- Summarizing and analyzing claims raised against deliveries.
- Providing dashboards and APIs for real-time supply chain insights.

1.4 Product Scope

The product provides:

- Data ingestion pipelines for multi-source data.

- Data cleaning & transformation using SQL and Python (Pandas).

- APIs (via FastAPI) for shipment, inventory, and claims visibility.

- Cloud integration using Azure Data Lake (ADLS), Azure Data Factory (ADF).

- Visualization via Power BI or Python for KPIs like delivery trends, claim summaries, and stock health.

1.5 Definitions, Acronyms, Abbreviations

- ADLS: Azure Data Lake Storage

- ADF: Azure Data Factory

- KPI: Key Performance Indicator

- API: Application Programming Interface

2. Overall Description

2.1 Product Perspective

The system will act as a centralized analytics platform integrated with Azure cloud services. It will provide both batch data ingestion pipelines and real-time REST APIs.

2.2 Product Features

- Shipment Tracker: Monitor shipment progress, delays, and costs.

- Claims Monitor: Track claims lifecycle and analyze trends.

- Vendor Inventory Viewer: Manage stock levels, vendor contracts, and restocking schedules.

2.3 User Classes and Characteristics

- Admin: Full access to configure data pipelines and APIs.

- Analyst: Access dashboards and aggregated data.

- Customer Support: Review claims and delivery logs.

2.4 Operating Environment

- Backend: FastAPI, SQLAlchemy, Python (Pandas).

- Database: SQL Database (Azure SQL / PostgreSQL).

- Cloud: Azure (ADLS, ADF, possibly Synapse simulation).

- Frontend (optional): Power BI / Python dashboards.

2.5 Constraints

- Data must be processed within Azure ecosystem.

- APIs should follow REST standards.

- Large-scale data ingestion must be handled efficiently.

2.6 Assumptions and Dependencies

- Source data available in CSV or SQL tables.

- Azure subscription and access to ADLS & ADF.

- Users have access to visualization tools.

3. System Features

3.1 Shipment Tracker

Description: Track and analyze shipments.

Inputs: shipment_id, origin, destination, dates, product_id, quantity, cost.

Outputs: Delivery timelines, freight cost summaries.

3.2 Claims Monitor

Description: Manage claims raised against deliveries.

Inputs: claim_id, delivery_id, claim_reason, status, amount.

Outputs: Claim resolution times, percentage of claims per carrier.

3.3 Vendor Inventory Viewer

Description: Monitor inventory stock levels and vendor contracts.

Inputs: warehouse_id, product_id, stock_level, vendor details.

Outputs: Reorder alerts, vendor performance metrics.

4. External Interface Requirements

4.1 User Interfaces

Web-based dashboards (Power BI or Python visualization).

Swagger/Redoc documentation for APIs.

4.2 APIs

- GET /claims-summary → Claim percentages by carrier.

- GET /inventory-health → Stock levels & reorder status.

- POST /log-shipment → Add new shipment.

- File upload API → Import new delivery logs.

4.3 Hardware Interfaces

Standard Azure cloud infrastructure (VMs, Data Lake storage).

4.4 Software Interfaces

SQLAlchemy, Pandas, Azure SDKs, FastAPI.

5. Non-Functional Requirements

- Performance: APIs should respond within 2 seconds for typical queries.

- Scalability: System should handle increasing data volumes with Azure scaling.

- Security: Access control via Azure IAM; data encryption at rest and in transit.

- Reliability: ≥ 99.5% uptime in cloud deployment.

- Usability: Dashboards should be intuitive for business users

5.1 Shipment Tracker

The functional requirements describe the expected behavior of the system in terms of inputs, processing, and outputs. These requirements define what the system must do to satisfy business objectives.

5.2 Claims Monitor

- FR1: The system shall allow logging of new shipment details including origin, destination, product_id, quantity, and ship_date.

- FR2: The system shall calculate and display delivery delays by comparing ship_date and delivery_date.

- FR3: The system shall generate reports on freight costs aggregated by city, vendor, or carrier.

- FR4: The system shall provide shipment history accessible via REST APIs.

5.3 Vendor Inventory Viewer

- FR5: The system shall allow users to file new claims linked to delivery_id with claim reason, amount, and claim_date.

- FR6: The system shall update claim status (Pending, Approved, Rejected, Resolved).

- FR7: The system shall generate claim resolution reports including resolution time and claim aging.

- FR8: The system shall calculate claim percentages grouped by carrier.

5.4 Data Processing & Integration

- FR9: The system shall monitor warehouse stock levels against reorder thresholds.

- FR10: The system shall trigger reorder alerts when stock_level < reorder_threshold.

- FR11: The system shall display vendor contract details including start and end dates.

- FR12: The system shall provide reports of vendor performance based on vendor_rating.

- FR13: The system shall ingest shipment, vendor, inventory, claims, and delivery logs data from CSV files into Azure Data Lake.

- FR14: The system shall process raw data through bronze and silver layers in Azure Data Factory.

- FR15: The system shall clean and transform data using Python (Pandas) and SQL queries.

- FR16: The system shall expose APIs to fetch metrics like claim summary, inventory health.