# Software Requirements Specification (SRS)

Smart Manufacturing – Fuel Efficiency & Turbine Health Analytics

Platform: Azure Cloud (ADLS, ADF, Databricks, FastAPI, Python Visualization)

## 1. Introduction

### 1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the Smart Manufacturing project that aims to monitor turbine health and optimize fuel efficiency using cloud-based analytics. This system will enable ingestion of telemetry data, real-time data transformation, KPI computation, anomaly detection, and visualization of operational trends. The SRS will serve as a reference for developers, testers, and future maintainers.

### 1.2 Scope

The system will ingest data from CSV logs into Azure Data Lake, clean and transform data using Python scripts and Databricks workflows, compute KPIs such as torque differential and efficiency ratios, expose REST APIs for real-time retrieval of health summaries, detect anomalies, and visualize data trends using Matplotlib and Seaborn.

## 2. Overall Description

### 2.1 Product Perspective

The system will function as a cloud-native analytics platform. Data will be ingested from edge devices (CSV logs) into ADLS, processed into curated layers, and served to analytics consumers through APIs and visualization scripts.

### 2.2 Product Functions

• Data Ingestion: Automated loading of CSV sensor logs into cloud storage  
• Feature Engineering: Calculation of temperature-pressure ratios, torque differences  
• Analytics: Efficiency benchmarking, anomaly flagging  
• APIs: Endpoints to retrieve KPIs and log alerts  
• Visualization: Correlation heatmaps, time-series plots, anomaly trend charts

### 2.3 User Classes

• Operators – View KPIs and detect anomalies  
• Maintenance Engineers – Analyze decay coefficients for predictive maintenance  
• Developers – Extend pipelines, add new data sources  
• Analysts – Perform EDA, tune anomaly detection thresholds

### 2.4 Operating Environment

Cloud: Azure ADLS, ADF, Databricks  
APIs: FastAPI, hosted locally or on Azure  
Visualization: Python scripts executed in VS Code using Matplotlib/Seaborn  
Database/Storage: CSV layers (raw, curated), SQL tables for structured data

## 3. Functional Requirements

|  |  |
| --- | --- |
| S.no | Requirement |
| 1 | System shall ingest telemetry data from CSV files into Azure Data Lake (raw layer). |
| 2 | System shall clean and transform data into curated datasets using Python/Databricks. |
| 3 | System shall compute derived metrics such as fuel efficiency, torque differential, and decay coefficients. |
| 4 | System shall provide REST APIs to retrieve turbine health summaries, sensor metrics, and KPI data. |
| 5 | System shall log and expose anomaly detection alerts based on threshold breaches. |
| 6 | System shall visualize sensor data, anomalies, and trends using Matplotlib and Seaborn. |

## 4. Non-Functional Requirements

|  |  |
| --- | --- |
| Category | Requirement |
| Performance | The system shall process 1 million sensor readings per day with <5 minutes pipeline latency. |
| Scalability | The system shall support scaling horizontally to ingest higher volumes of telemetry data. |
| Reliability | Data pipelines shall recover from transient failures and ensure at least-once delivery of data. |
| Security | APIs shall be protected with token-based authentication; sensitive data shall be encrypted in storage. |
| Maintainability | Codebase shall be modular, documented, and version-controlled using Git. |
| Usability | APIs shall be self-documented via Swagger UI, making them easy to consume by clients. |

## 5. Acceptance Criteria

• Data ingestion jobs successfully load 100% of provided CSV logs  
• KPI computations match validation dataset results within tolerance  
• API endpoints respond within <1 second for single record queries  
• Visualization outputs must include at least 3 plots: trend line, correlation heatmap, anomaly chart