# Requirements Specifications Document

## 1. Introduction

## a. Purpose

The purpose of this document is to define the requirements for a data pipeline system tailored for a healthcare insurance provider. The objective is to build a scalable data solution that can ingest, clean, and analyze extensive datasets from customers and competitors. The insights generated will enhance customer understanding, enable targeted insurance offerings, promote customer retention, and increase company revenue.

## b. Intended Audience and Use

This document is intended for internal stakeholders, including:

- Developers/Data Engineers: to design and implement the pipeline.

- Testers: to verify the integrity and correctness of the data.

- Project Managers: to track progress and manage deliverables.

- Analysts/Business Stakeholders: to interpret insights and apply them to strategic decisions.

## c. Product Scope

The proposed solution will provide the ability to:

- Analyze patterns in customer behavior and medical claims.

- Generate actionable insights for strategic decision-making.

- Use modern cloud-based infrastructure, including AWS (S3, Redshift), Databricks, and PySpark.

- Deliver scalable, reliable, and repeatable processes accessible to both technical and business users.

## d. Definitions and Acronyms

- S3: Amazon Simple Storage Service for storing raw data.

- Redshift: AWS cloud-based data warehouse for structured data analysis.

- PySpark: Python API for Apache Spark used for big data processing.

- Databricks: Unified analytics platform for running Spark workloads.

- EMR: Amazon Elastic MapReduce, a service to run distributed big data frameworks.

- Use Case: A scenario or problem statement the system aims to solve using data.

## 2. Overall Description

The system will be a greenfield implementation that processes and transforms raw data into meaningful insights for the healthcare insurance industry. It will support the complete data lifecycle, from ingestion to analysis, without integrating with legacy systems. This independent architecture will empower cross-functional teams to leverage data more efficiently.

## a. User Needs

- Data Engineers: Build, test, and maintain the pipeline.

- Data Analysts: Query and analyze the processed data.

- Software Developers: Enhance and optimize pipeline components.

- Quality Assurance Teams: Validate the reliability and accuracy of data.

- Business Teams: Use data insights to shape policies and marketing strategies.

## b. Assumptions and Dependencies

- Input data will be provided in standard formats like CSV.

- AWS services such as S3, Redshift, and EMR are available.

- Databricks Community Edition will be used during development phases.

- Python, Spark, and related tools are accessible and supported.

- Data received from external sources is reasonably clean and structured.

- The solution will operate in a cloud-agnostic environment, not tied to a specific OS.

## 3. System Features and Requirements

## a. Functional Requirements

- Upload CSV or similar raw data to S3.

- Preprocess data with PySpark to handle missing, inconsistent, or duplicate entries.

- Load the curated data into Redshift.

- Use SQL and Spark to answer predefined business questions.

- Store results in Redshift for easy reporting.

- Maintain codebase and documents in GitHub with proper version control.

- Use Jira for task planning, progress tracking, and sprint management.

## b. External Interface Requirements

- User Interface: No traditional UI; interactions through notebooks, SQL clients, and code editors.

- Hardware Interface: Cloud infrastructure only; no dependency on local hardware.

- Software Interface: Integrates with tools such as Databricks, AWS CLI, Redshift, and GitHub.

- Communications Interface: Collaboration and task coordination via GitHub and Jira.

## c. System Features

- Efficient handling of large-scale healthcare data.

- Structured storage for seamless analysis.

- Built-in logic to answer strategic queries like most frequent claims or top cities for submissions.

- Reusable, consistent data pipelines for long-term operation.

- Modular and extensible design for easy enhancements.

## d. Nonfunctional Requirements

- Performance: Capable of processing large volumes swiftly; support near real-time querying.

- Safety: If real personal data is used, privacy measures will be implemented.

- Security: Role-based access to AWS and code repositories.

- Usability: Clean, readable, and modular code; descriptive naming for outputs.

- Scalability: System should accommodate data growth and support future use cases without full redesign.