Requirements Specifications Document

1. Introduction – This project aims to help a healthcare insurance company improve revenue and customer understanding by analyzing competitor and customer data using Big Data tools. By building data pipelines that clean and process this information, the company can uncover insights like top diseases, customer behavior, and policy trends. These insights will guide better offers, targeted policies, and reward strategies. This document outlines the expectations, use cases, and technologies that will drive the solution.

Purpose – The main goal of the project is to help the healthcare insurance company gain a deeper understanding of customer behavior, policy trends, and operational effectiveness. We are going to analyze factors such as common diseases, age demographics, claim rejections, hospital usage, and regional patterns, the company so that we can identify high-risk areas, optimize policy offerings, and improve customer targeting. With these techniques, we can customizing plans for younger subscribers, focusing on profitable groups, or offering better coverage for high-value cases.

Intended Audience and Use - The SRS will be accessed by key stakeholders involved in the project, including developers, testers, data engineers, business analysts, and project managers. Developers will use the SRS to understand the functional and technical requirements they need to implement. Testers will refer to it for defining test cases and validating outputs against business goals. Project managers and analysts will use the SRS to track progress, ensure alignment with objectives, and manage scope.

Product Scope - The primary goal of this product is to enable the healthcare insurance company to make smarter, data-driven decisions that boost revenue and improve customer satisfaction. By leveraging Big Data technologies, the system will uncover key insights into customer behavior, policy performance, and competitor trends. This will help the company customize offers, reward loyal policyholders, and streamline operations.

Definitions and Acronyms –

* SRS – Software Requirements Specification
* ETL – Extract, Transform, Load
* AWS – Amazon Web Services
* S3 – Simple Storage Service
* EMR – Elastic MapReduce
* SQL – Structured Query Language
* DB – Database
* PK – Primary Key
* FK – Foreign Key
* GUI – Graphical User Interface
* API – Application Programming Interface
* CSV – Comma-Separated Values
* ID – Identifier
* BI – Business Intelligence

## Overall Description - The product is designed to help a healthcare insurance company gain insights from large-scale customer and competitor data in order to improve revenue, policy targeting, and customer satisfaction. We have developed a newdata analytics solution from the ground up, utilizing modern cloud and big data technologies. This system will integrate multiple datasets that may include claims, subscribers, and hospital records which is then processed into a central Redshift warehouse through an automated PySpark pipeline. It will not replace an existing product but will enhance the company’s decision-making framework with deeper analytics and reporting capabilities.

## User Needs - This system will serve multiple users across the organization, each with distinct needs. Data analysts need access to clean, structured data to perform deep dives into trends like disease prevalence, claim rejection rates, and policy profitability. Business managers and strategists require high-level summaries and insights to make informed decisions on pricing, customer targeting, and loyalty programs. Data engineers and developers need a reliable and scalable pipeline that automates ingestion, cleaning, and output generation. All users expect the system to be accurate, fast, and easy to maintain, with clear documentation and integration into existing tools like Databricks, AWS Redshift, and GitHub.

## Assumptions and Dependencies –

## The solution assumes access to accurate raw data from third-party and internal sources.

* The infrastructure is built on AWS services (S3, Redshift, EMR), and the solution assumes these services are fully operational and accessible.
* It is assumed that Databricks (Community Edition) will be used for development and initial testing, with final deployment on EMR or enterprise Databricks.
* Dependencies include libraries and integrations with PySpark, JDBC for Redshift, and GitHub for version control and CI/CD.
* It is also assumed that users have basic familiarity with cloud platforms and querying tools (like SQL or notebooks).

1. System Features and Requirements

The system must meet a variety of functional and nonfunctional requirements. These requirements define how the system will behave, how it will interact with users and other systems, and how well it will perform under different conditions.

Functional Requirements

* The system must ingest raw data (CSV/JSON) from AWS S3.
* Data must be cleaned using PySpark (null handling, deduplication).
* Cleaned datasets (patients, claims, etc.) must be loaded into Redshift tables.
* SQL queries must be executed to generate outputs for each use case.
* The system must store query results in Redshift under project\_output schema.
* The system must allow integration with visualization tools (e.g., Databricks or BI dashboards).
* The pipeline must support automated runs (daily/weekly) using a job scheduler.

External Interface Requirements

* **User Interface:**Users will interact via Databricks notebooks or SQL clients (e.g., Redshift Query Editor).
* **Hardware Interface:**AWS-hosted infrastructure; users access via browser and cloud services.
* **Software Interface:**Integration with AWS S3, Redshift, Databricks, and GitHub.
* **Communication Interface:** Secure data transfer between S3, EMR/Databricks, and Redshift using AWS IAM roles and secure protocols.

System Features

* Data cleansing feature using PySpark with modular scripts.
* Use case result generation as parameterized SQL jobs in Redshift.
* Logging and monitoring for job execution success/failure.
* Version control of pipeline scripts through GitHub.
* Result snapshot support for visualization and reporting.

Nonfunctional Requirements

* **Performance Requirements:** System must handle and process datasets with up to 10 million records with a 30-minute SLA per batch.
* **Safety Requirements:** Only anonymized or masked data will be used for testing/demo purposes.
* **Security Requirements:** Role-based access control via IAM; sensitive data is encrypted at rest and in transit.
* **Usability Requirements:** Notebooks and queries must be modular and reusable; documentation must be clear.
* **Scalability Requirements:**Must scale to accommodate additional data sources and new use cases without redesigning the pipeline.