Requirements Specifications Document

# Introduction -

## Purpose -*Define the purpose of these requirements here.*

The goal of this project is to create a full data pipeline that allows a healthcare insurance company to process and analyze lots of data from customers and competitors. By doing this, the company can understand its customers better, tailor better insurance offers, build loyalty, and ultimately boost its revenue.

## Intended Audience and Use - *Define who in your organization will have access to the SRS and how they should use it. This may include developers, testers, and project managers.*

This document will be accessed by developers, testers, project managers, and analysts within the organization. Developers and data engineers will use it as a guide to build and maintain the data pipeline. Testers will use it to ensure data quality and validate the outputs. Project managers and analysts will refer to it to track project progress and understand how the data insights support business decisions.

## Product Scope - *What are the benefits, objectives, and goals we intend to have for this product? This should relate to overall business goals, especially if teams outside of development will have access to the SRS.*

This project aims to help the company gain a deeper understanding of customer behavior and preferences, analyze key trends like diseases and insurance claims, and use these insights to create targeted offers and reward loyal customers. By doing so, the company can improve overall business performance and increase revenue. The solution will be built using scalable, cloud-based technologies such as AWS (S3 and Redshift), Databricks, and PySpark, ensuring that both technical teams and business stakeholders can access and benefit from the data insights.

## Definitions and Acronyms -*Clearly define all key terms, acronyms, and abbreviations used in the SRS. This will help eliminate any ambiguity and ensure that all parties can easily understand the document.*

* **S3** – Amazon’s cloud storage service where we keep all the raw data files.
* **Redshift** – A cloud database from AWS used to store cleaned and organized data ready for analysis.
* **PySpark** – A tool that lets us process and analyze big datasets using Python and Apache Spark.
* **Databricks** – A platform that helps us run Spark jobs and explore data in an easy, interactive way.
* **EMR** – Amazon’s service for running big data processing clusters, like Spark or Hadoop, in the cloud.
* **Use Case** – A specific business problem or question we want to answer using the data.

# Overall Description - *Your next step is to give a description of what you’re going to build. Why is this product needed? Who is it for? Is it a new product? Is it an add-on to a product you’ve already created? Is this going to integrate with another product? Understanding and getting your team aligned on the answers to these questions on the front end makes creating the product much easier and more efficient for everyone involved.*

We're building a complete data pipeline from the ground up. It takes in raw data, cleans and processes it, and turns it into useful insights that can answer real business questions. This is a brand-new system—not something we’re adding to an existing product. It’s meant to help analysts, developers, and business teams better understand customer behavior and trends so they can make smarter decisions. The system will run independently and doesn’t need to connect to any existing tools or platforms.

## User Needs - *Describe who will use the product and how. Understanding the various users of the product and their needs is a critical part of the SRS writing process.*

* **Data engineers** will build and manage the data pipeline.
* **Analysts** will use the cleaned data to find trends and create reports.
* **Developers** will maintain and improve the system.
* **Testers** will make sure the data is processed correctly.
* **Business teams** will use the insights to make better decisions, target customers, and increase revenue.

Each group depends on the system to turn raw data into useful insights that support their roles and business goals.

## Assumptions and Dependencies - *What are we assuming will be true? Understating and laying out these assumptions ahead of time will help with headaches later. Are we assuming current technology? Are we basing this on a Windows framework? We need to take stock of these technical assumptions to better understand where our product might fail or not operate perfectly.*

We’re working with a few key assumptions to make sure the project runs smoothly:

* The data we need is already available in files like CSVs.
* AWS services like S3, Redshift, and EMR are set up and ready to use.
* Databricks Community Edition is available for testing during development.
* Tools like Python, PySpark, and Spark are installed and supported.
* The external data we receive is mostly reliable and in a format we can work with.
* The system will run in a cloud environment (not tied to Windows or any specific OS).

If any of these things aren’t true, it could slow down the work or cause issues with how the pipeline runs.

# System Features and Requirements -*In order for your development team to meet the requirements properly, we must include as much detail as possible. This can feel overwhelming but becomes easier as you break down your requirements into categories.*

## Functional Requirements - *Functional requirements are essential to your product because, as the name implies, they provide some sort of functionality. Asking yourself questions such as “does this add to my tool’s functionality?” or “what function does this provide?” can help with this process. You may also have requirements that outline how your software will interact* *with other tools*

* Upload the raw data files (like CSVs) into AWS S3 for storage.
* Clean and process the data using PySpark — this includes fixing missing values, removing duplicates, and making sure everything is in the right format.
* Load the cleaned data into AWS Redshift, where it can be easily queried.
* Write and run code to answer 13 specific business questions using the data, and save those results in Redshift.
* Keep track of tasks and progress using Jira, organized into a 2-week sprint plan.
* Use GitHub to store all code, documents, and updates so everything is version-controlled and easy to share.

All of these steps help turn raw data into useful insights while making sure the work stays organized, repeatable, and easy to maintain

## External Interface Requirements - *You may also have requirements that outline how your software will interact with other tools There are several types of interfaces you may have requirements for, including:*

### User

### Hardware

### Software

### Communications

**i. User**  
There isn’t a typical user interface like a website or app. Most of the work happens through notebooks, Redshift queries, or writing and running code.

**ii. Hardware**  
Everything runs in the cloud using AWS services—no local hardware needed.

**iii. Software**  
We’ll be using tools like PySpark, Redshift, AWS CLI, Databricks, GitHub, and Jira to build and manage the system.

**iv. Communications**  
GitHub is where the team shares and controls the code, while Jira helps us plan and keep track of tasks.

## System Features - *System features are a type of functional requirements. These are features that are required in order for a system to function.*

The system can clean and process large healthcare datasets efficiently.

* + - It stores the data in Redshift so it’s easy to query and analyze.
    - It generates insights to answer key business questions, like which diseases have the most claims or which cities submit the most claims.
    - It supports consistent and repeatable data workflows to ensure reliable results.
    - It’s designed to be scalable and built for long-term use.

## Nonfunctional Requirements - *Nonfunctional requirements, which help ensure that a product will work the way users and other stakeholders expect it to, can be just as important as functional ones. These may include:*

### Performance requirements

### Safety requirements

### Security requirements

### Usability requirements

### Scalability requirements

**i. Performance**  
The system needs to efficiently process large amounts of data, and queries should run quickly enough to support timely analysis.

**ii. Safety**  
While we’re working with sample data here, if real personal information were involved, it would require careful handling to protect privacy.

**iii. Security**  
Access to AWS resources and the GitHub repository must be well-controlled to keep data and code secure.

**iv. Usability**  
The code should be well-organized, easy to understand, and reusable. Output tables should have clear, descriptive names.

**v. Scalability**  
The data pipeline should be built to handle increasing volumes of data and allow adding new business use cases without needing a complete overhaul.