Template - Requirements Specifications Document

# Introduction - *This introduction is very important as it sets expectations that we will come back to throughout the SRS.*

## Purpose - The purpose of this Requirements Specifications Document (SRS) is to outline the functional and non-functional requirements for implementing a Big Data analytics solution for a Health Care insurance company. This solution aims to analyze competitor data received from various sources, such as web scraping and third-party providers, to enhance revenue, understand customer behavior, and provide customized insurance offers. The ultimate goal is to increase the company’s market share by better serving customers and maximizing revenue through targeted policies and royalty calculations.

## 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.*

## Product Scope –

The Big Data analytics solution aims to provide the following benefits, objectives, and goals:

* **Benefits:**
  + Enhanced understanding of customer behavior and preferences.
  + Increased revenue through customized insurance offers.
  + Improved customer satisfaction by offering tailored policies and calculating royalties effectively.

**Goals:**

* Increase market share by attracting new customers with competitive offers.
* Retain existing customers by offering royalties and personalized policies.
* Support strategic decision-making with data-driven 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.*

# 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.*

## User Needs –

* To access, process, and analyze large datasets from competitor companies and other sources to generate insights.
* They will use the product to clean, transform, and analyze data, using advanced analytics techniques to uncover trends and patterns in customer behavior and market conditions.
* The goal of the project is to create data pipelines for the Health Care insurance company which will make the company make appropriate business strategies to enhance their revenue by analysing customers behaviours and send offers and royalties to customers respectively.

## Assumptions and Dependencies –

# 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 –

* Which disease has a maximum number of claims.
* Find those Subscribers having age less than 30 and they subscribe any subgroup
* Find out which group has maximum subgroups.
* Find out hospital which serve most number of patients
* Find out which subgroups subscribe most number of times
* Find out total number of claims which were rejected
* From where most claims are coming (city)
* Which groups of policies subscriber subscribe mostly Government or private
* Average monthly premium subscriber pay to insurance company.
* Find out Which group is most profitable
* List all the patients below age of 18 who admit for cancer
* List patients who have cashless insurance and have total charges greater than or equal for Rs. 50,000.
* List female patients over the age of 40 that have undergone knee surgery in the past year

## 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

* + - 1. AWS S3
      2. AWS Redshift
      3. Databricks
      4. AWS EMR Studio
      5. Pyspark
      6. Jira
      7. GitHub

### Communications

* **DATASET CREATION**
  + Upload the given sample data on AWS s3 in a folder named input-data.
* **DATA CLEANING**
  + Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset.
  + Cleaning Activity
    - First check if there are null values in dataset
    - Count the total Null values for each column
    - And then replace the null values for specific columns by NA
    - Check the If three are duplicates records
    - If there are duplicates then drop duplicates
  + Clean data for at least for following datasets
    - Patients
    - Subscriber
    - Claims
    - Group\_subgroup
  + Upload cleaned data corresponding to each data set into a redshift table.
  + Please create a schema design doc for target tables.
* **RESULT CREATION ON REDSHIFT**
  + Create a separate redshift table for each use case output in a redshift schema
    - Schema = Project-Output

## System Features –

The Big Data analytics solution for the Health Care insurance company will include the following system features:

1. **Data Ingestion and Integration**:
   * **Feature Description**: The system will have the capability to ingest data from various sources, including web scraping, third-party data providers, and internal systems. It will also integrate this data into a unified data lake for further processing.
   * **Functional Requirements**:
     + Ability to connect to multiple data sources (APIs, databases, web scraping tools).
     + Support for batch and real-time data ingestion.
     + Data cleansing and transformation capabilities to ensure data quality.
     + Integration with existing CRM and policy management systems.
2. **Data Storage and Management**:
   * **Feature Description**: The system will store large volumes of structured and unstructured data in a scalable and secure environment.
   * **Functional Requirements**:
     + Use of cloud-based storage solutions (e.g., AWS S3, Azure Blob Storage) for scalability.
     + Support for various data formats (CSV, JSON, Parquet, etc.).
     + Data cataloging and metadata management to facilitate data discovery and governance.
3. **Data Processing and Analytics**:
   * **Feature Description**: The system will process the ingested data using Big Data technologies and provide analytical capabilities to generate insights.
   * **Functional Requirements**:
     + Implementation of distributed processing frameworks (e.g., Apache Spark) for handling large datasets.
     + Support for advanced analytics techniques, including predictive modeling and machine learning.
     + Ability to generate reports, dashboards, and visualizations for different stakeholders.
     + Tools for ad-hoc querying and exploratory data analysis.
4. **Custom Offer and Royalty Calculation**:
   * **Feature Description**: The system will analyze customer data to customize insurance offers and calculate royalties for existing policyholders.
   * **Functional Requirements**:
     + Algorithms to assess customer behavior and predict future needs.
     + Dynamic rule engine to generate personalized insurance offers.
     + Calculation engine to determine royalties based on customer history and policy details.
     + Integration with marketing tools to automate offer delivery.
5. **Monitoring and Alerts**:
   * **Feature Description**: The system will include monitoring and alerting features to ensure smooth operation and quick response to issues.
   * **Functional Requirements**:
     + Real-time monitoring of data pipelines and processing tasks.
     + Automated alerts for data quality issues, system performance degradation, and security incidents.
     + Dashboard for system health and operational metrics.

## Nonfunctional Requirements -

### Performance requirements

 **High Throughput**: The system must be capable of ingesting and processing large volumes of data in near real-time, ensuring that data is available for analysis without significant delays.

 **Low Latency**: The system should provide near real-time analytics and insights, with minimal delay between data ingestion and result availability.

### Safety requirements

 **Data Privacy Compliance**: The system must comply with relevant data privacy laws and regulations, such as HIPAA, ensuring that all customer data is handled with the highest level of confidentiality.

 **Failover and Disaster Recovery**: The system must include failover mechanisms and a disaster recovery plan to ensure data is not lost and the system remains operational during unexpected failures.

### Security requirements

 **Data Encryption**: All data at rest and in transit must be encrypted using industry-standard encryption methods to protect sensitive information from unauthorized access.

 **Access Control**: The system must implement role-based access control (RBAC) to ensure that only authorized users can access certain data and functionalities.

 **Audit Logging**: All user actions and system events must be logged for audit purposes, with logs stored securely and monitored regularly.

### Usability requirements

 **User Interface**: The system should have an intuitive user interface that allows non-technical users, such as marketing and sales teams, to easily access insights and generate reports.

 **Documentation and Training**: Comprehensive documentation and training materials must be provided to help users understand how to interact with the system and utilize its features effectively.

### Scalability requirements

* + - 1.  **Horizontal Scalability**: The system must support horizontal scaling to accommodate increasing volumes of data and growing numbers of users without performance degradation.
      2.  **Modular Architecture**: The system should be designed with a modular architecture, allowing individual components (e.g., data ingestion, processing, storage) to be scaled independently based on demand.

## 