SP-114 Data Lake Implementation

Requirements Specification

CS 4850, Section 02, Spring 2026

Professor Perry, 2/8/2026

Roles	Name	Role	Contact
Team Leader	Caleb Cox	Document project progress; Coordinate communications and development between team members	470-819-7917
Team Members:	Bryce Wishart	Documentation, development, programming, and testing.	404-683-1147
Advisor/Instructor	Sharon Perry	Facilitate project progress; advise on project planning and management.	770-329-3895

Caleb Cox
Team Leader

Bryce Wishart
Group Member

Table of Contents

1.0 Introduction

This requirements document is divided into several sections based on the criteria for constructing this project. The introduction section contains an overview of the project, which details the basic premise of the project in this document. Next is the project scope, which is a more detailed description of the software specified in this project. The definitions section will be reserved for describing any shorthand acronyms used within the document, while the references will contain the other resources used to create this document and project.

Following the introduction, the constraints detail what limitations are present in the creation of this project, while system features will detail the behavior of the system and any core features that are integral to the function of this project. The concluding section, analysis, will detail use cases for the project, as well as the flow of data.

1.1 Overview

The project detailed within this document is the creation of a mini data lake for cleaning raw data to add the cleaned data sets to reports for business analytics. From here on, the document shall refer to this project as the Data Lake Project.

1.2 Project Scope

The Data Lake Project requires several main constructions to ensure a complete product. Firstly, the data lake must have a system to intake raw data from various formats. This raw or bronze zone should encompass where all input data is first run through. Primarily, our project will be using Apache Kafka to create an input data stream.

Secondly, this data lake needs some form of storage for the massive amounts of data that are streaming in. This is where the object store is used. With an object store, the raw data can be checked, cleaned, and organized into a standard format or several formats for ease of access. This is the silver layer, and it is here that our project will add schemas to the data to provide structure.

The final main construction of the Data Lake Project is the query engine. This part allows the user to directly pull data from the object store using the file formatting system applied. Here, data can be collated into a proper report for analysis purposes. This gold zone is where the data is put into its most organized state.

1.3 Definitions

SQL- A language used for working with data stored in database systems. It stands for Structured Query Language

1.4 References

Apache Parquet:

https://parquet.apache.org/docs/

Apache Hive:

https://cwiki.apache.org/confluence/display/HIVE

Apache Kafka:

https://kafka.apache.org/41/getting-started/introduction/

MinIO:

https://docs.min.io/enterprise/aistor-object-store/

2.0 Constraints

The main constraint of the Data Lake project is the tools that we can use. This project is made using open-source tools like MinIO or the Apache suite of software for data analytics. If these tools ever lose support in some fashion, this project will need to be moved to other data manipulation tools that are still in service.

For now, we will be using Apache Kafka for data streaming, Apache Parquet for a file formatting system for structuring the input data, MinIO as an object store for holding the data in our data lake, and Apache Hive for the database that the end user can make queries on.

Additionally, the programming side of this project will be limited to Python as a programming language, since it is the language that all team members have familiarity with. What this means for the tools we are using is that they must have some support for python. If there is no official version that can be used on Python, then alternatives will need to be found.

3.0 System Features

3.1 Data Acquisition

3.1.1 Description

This component of the Data Lake system handles intaking inputs from a source and creating an entry for the object store. This function encompasses the raw data and the bronze layer of medallion architecture.

3.1.2 Functional Requirements

- Connect to input stream.
- Read input data (text files, application logs, website clicks, etc.)
- Create key for input to use as access when in database.
- Place data into proper folders for type of input data.
- If input fails to validate, system moves on to another input.
- Connect to object store to send input data.

3.2 Data Storage

3.2.1 Description

Data storage in the data lake is an object store tool that can hold massive amounts of data with varying types of formats.

3.2.2 Functional Requirements

- Intake data from the data acquisition component
- If no folder exists for the data type being added, create new folder to store data.
- Sort the incoming data into the appropriate storage folders.

3.3 Data Formatting

3.3.1 Description

Data formatting in this data lake system includes both the file format used to access the data from the object storage, as well as the file format used to structure the raw data. With this feature, data becomes more structured and enters the silver zone of medallion architecture.

3.3.2 Functional Requirements

- Connect to the object store.
- Create schema according to user specification.
- Sort data by the created schema.
- Access the data according to the query given by a user.
- Display all valid data that conforms to the specified query.
- Any errors in displaying data should be documented by the system in a secondary log.

3.4 Data Querying

3.4.1 Description

This final system feature allows for the querying of data stored in the object store using a secondary database that a user can access. Data can be pulled through SQL queries to find all data that fits the user criteria. On this final layer, all the data is curated for easy analysis and direct retrieval.

3.4.2 Functional Requirements

- Take in user SQL input for data.
- Send user input through the data format portion of the system.
- Receive the output from the object store.
- Display the output for the user to read.
- If data cannot be retrieved, send an error message to the user.

4.0 Interface Requirements

4.1 User Interface Requirements

The user should be able to access the data lake through a query engine that connects to the object store. The engine should have an input section that allows the user to type in SQL queries for the system to read. If there is any error in the user's input, the system should display an error message asking the user to rewrite the input. Any errors in data retrieval should also be documented and displayed to the user.

4.2 Software Interface Requirements

In the Data Lake Project, several software components are used for the creation of the object store, query engines, and file formatting. Each of these needs to be connected to each other in a way that allows the data flowing in to be accessed by each software and cleaned to create more filtered data. The data acquisition component should be connected to the object store, so that the raw data can be stored.

From the object store in MinIO, the file formatting using Apache Parquet should be able to access the data for retrieval and cleaning. Parquet should send this data back to the query engine on Apache Hive to display it to the user.

5.0 Analysis

5.1 Use Cases

There are many diverse types of users or businesses that may want access to a data lake for gathering and filtering data. Some examples include:

- A web developer wants to analyze the traffic on their website and how often users remain active.
- A business wants to monitor sales figures for specific products or during specific times of the year.

5.2 Data Flow

In this data lake, data will first enter the system from the data stream created using Apache Kafka. At this time, the data can be in any format, from excel files to text documents or even images or website clicks. From the data stream, these pieces of data are organized into folders for easier control over data flow.

After checking and validating the data, it is moved into the object store. At this point, the data can be further organized using a standard format to create schemas that structure the data. This structured form of the data is what the end user can access when they make SQL calls to the data store.