SP-114 Data Lake Implementation Design Documentation

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 document is written to provide a detailed description of the system that will be implemented in the creation of the Data Lake Project. While the requirements documentation discusses the necessary parts to creating a functioning data lake and gives a brief overview, this document will go into further detail on the relevant design components.

Firstly, this document will discuss the considerations that must be considered when our group is designing the data lake, including any dependencies, constraints, and the methodology we will use to develop the program. Next, we will discuss the system architecture and the strategies employed to create the system. The closing section will be an in-depth exploration of the components essential to the construction of the data lake. Among these sections, detailed diagrams will be included to illustrate the relationship between components and the process flow.

A data lake is a repository of data that can be structured or unstructured, stored for use by end users in data analytics. This project will create a mini data lake using open-source software to create storage and querying structures that allow a user to access the data and sort it according to their needs.

2.0 Design Considerations

2.1 Assumptions and Dependencies

Our data lake is going to be built on a Windows installation, so the assumption for end users is that they possess a computer that is running Windows or can run Windows programs. Each of the tools used to create the components of the data lake will also be using Windows installations.

Many of the tools that will be used to create the data lake are from the Apache Software Foundation, which is a group providing open-source computing software for communities. The assumption in using these tools is that they will work with each other to connect the separate components of the data lake.

On the user side, it is assumed that they have knowledge of SQL queries and working with databases, as pulling data from the data lake requires knowledge of database systems and how to manipulate and pull data from them.

2.2 General Constraints

Most of the programming written in the creation of the data lake will be written in Python, which means that the tools we use to create the components of the data lake must have support for the Python programming language. If there is no official or community supported version of the software, we will find an alternative tool.

When developing the project, it will be developed with the limitation that it must run on the hardware of all team members, so that every project member can help contribute to project development.

Additionally, since the data lake requires a stream of input data, our project will be much smaller in scope since we will have to create the raw data ourselves.

2.3 Development Methods

The primary development methodology we will employ for this project is the Agile methodology. From the requirements we determined in the specification document, we will find the most key features that need to be implemented.

Once found, our team will begin a sprint to develop and implement a few chosen features of the final product. Following each sprint, the team will meet and discuss what needs to be implemented next and what might need for revision. Afterwards, another iteration of planning and sprinting can begin.

3.0 Architectural Strategies

This section details the various strategies that will be utilized in creating our data lake. As mentioned in previous sections, the programming aspect of the project will be managed using the Python programming language. The reasoning behind the use of this language is familiarity and ease of use for team members.

Another component to our data lake is the user's input. The end user of the database will not be able to directly enter raw data stored in the system, as the query engine is only going to be used for system output. If any data requires introduction into the data storage, it should be added through the data acquisition component of the architecture.

3.1 Apache Kafka

Apache Kafka is an open-source data collection tool in the Apache suite of tools. This tool allows users to create applications that can read and write “events,” which are records of data from the client service. These events can be anything from a timestamp of a webpage access to a payment message. From here, the data can be stored in folders for later access.

We decided to use Kafka for the data input collection due to its ability to manage large, continuous streams of data, and its ability to record data of diverse types. This aligns with the main goals of our data lake, which is a repository of substantial amounts of unstructured data that can be later structured through schema from other tools we will be utilizing.

While official Kafka documentation does not list any Python support, there is a supported Python library for accessing Apache Kafka, which aligns with our stated programming language constraint.

3.2 Apache Parquet

Apache Parquet is an open-source data file format that is used for retrieving data from a database. A data lake requires some form of formatting for data retrieval, since the data format is also how structure can be added to the raw data through schema. There are several options for file formats, but we chose Parquet for its speed.
This file format is column-oriented; meaning data is stored by column instead of by row. As such, the Parquet queries only read the user's specified columns, speeding up the time it takes to retrieve the data. The goal of our project is to create a repository holding large amounts of data, so having a file format that can scale to higher volumes of data while maintaining speed in querying is imperative.

3.3 MinIO

MinIO or the Alstor object server, is a large-scale data storage system that can be deployed through local computers or the cloud to hold data. This was chosen for use in creating our project since it is primarily used in analytics and has support for Python application connections.

3.4 DuckDB

DuckDB is a database management system that is ideal for large databases. This system was chosen since it offers support for Apache Parquet files, and Apache Parquet has common libraries for use in DuckDB. The supported connection between these two parts of our data lake will make implementation much easier.

3.5 Future Support

Since this project is a collaborative effort among students, much of its final design will remain unchanged following the conclusion of the project. If any team member wishes to continue iterating on the project, the final package will remain available for members to fork and create updates to continue support on the data lake.

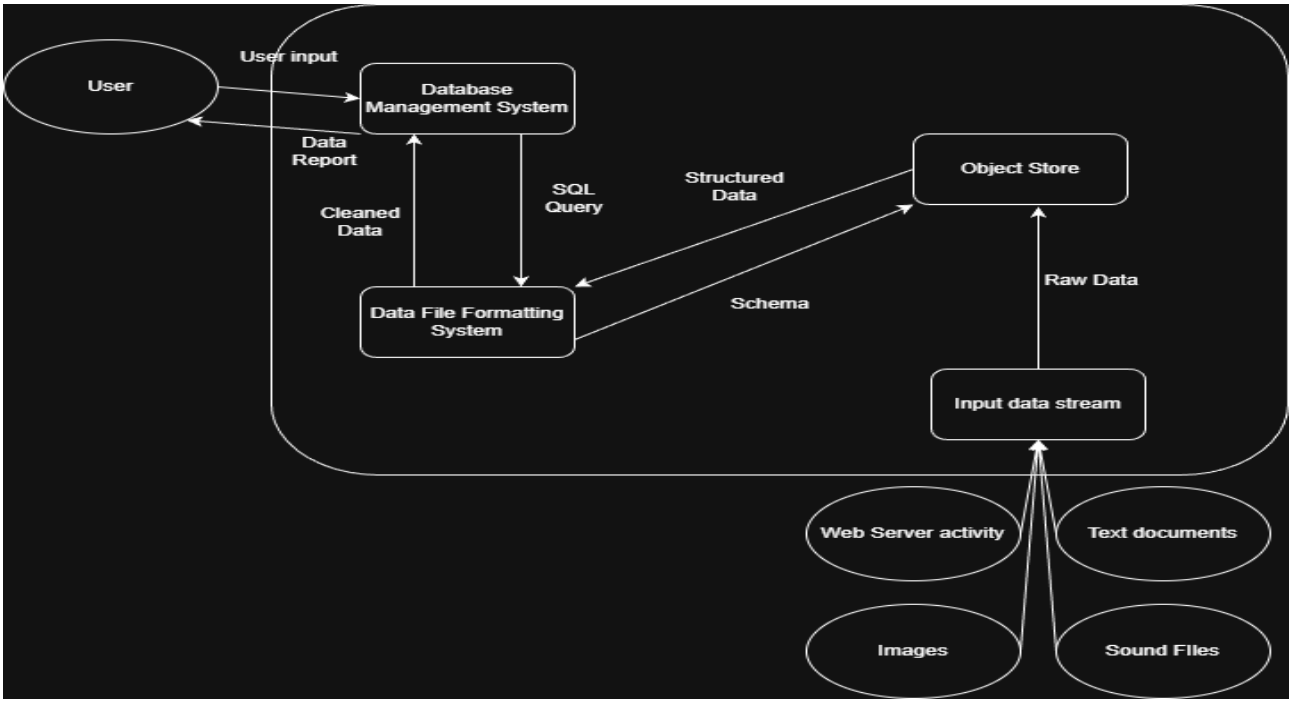
4.0 System Architecture

A Data Lake requires several different systems working in tandem to create a functional data storage that can be accessed and analyzed. Our team partitioned the creation of the data lake into four major sections: Data Acquisition, Data Storage, Data Formatting, and Data Querying.

The data acquisition subsystem encompasses all functions and libraries that are used to create a program that pulls in raw data as a stream and transfers it to the data storage subsystem. Data storage encompasses all programs that deal with storing raw data, and some minor structures with folders for storing diverse types of data.

From the storage segment, the data formatting section should pull data from the data storage section and apply the neccessary schema to the data to gather all valid data that can be used for analysis. Finally, the data querying section will take the structured data from data formatting and place it into clean reports that are easy for the end user to read and gather information from.

Below is a figure of the connection between users and the functions of the system.



5.0 Detailed System Design

5.1 Classification

- Input data stream- subsystem.
- Object store- subsystem.
- Data file formatting- subsystem
- Database management system- subsystem
- Raw data- file
- Schema- function
- Apache Kafka- package
- Apache Parquet-package
- MinIO- package
- DuckDB- package

5.2 Definition

- Input data stream- The functions and packages used for gathering raw data from an outside source. The package of Apache Kafka is a part of this subsystem, and the files obtained from outside sources.

- Object store- The group of components that encompass the storage of raw data. Packages like MinIO as well as raw data files are included in this subsystem.

- Data file formatting-The packages, functions, and classes used to apply structure to the raw data stored within the object store. Apache Parquet and the functions used for Schema are included in this subsystem.

- Database management system- The group of components used to send queries to the object store and receive structured data reports. Packages like DuckDB and Apache Hive are included in this subsystem.

- Raw data- All files brought into the system by the input data stream.

- Schema- The structures applied to raw data that filters it into more refined data sets that can be arranged into a single report.

- Apache Kafka- Package that gives methods for creating a data stream into the system.

- Apache Parquet- Package that gives methods to convert data and manage schema.

- MinIO- Object storage system that can be connected to a supported input stream.

- DuckDB- Relational database managment system that can send SQL queries to an object store.

5.3 Constraints

- Input data stream- The data that streams into this component should be unstructured, meaning it can be of any type. Additionally, raw data must not be stored in this subsystem; it must be sent to the object store after validation.

- Object Store- While multiple users can interact with the object store when running queries, if the query results in a structure change for the data, lockouts must be enforced so that the system maintains up-to-date state information of all stored data.

- Data file formatting- Data brought into this part of the system must be in the same file format, so that schema application and data modification can be applied to create a thoroughly cleaned set of data for analysis.

- Database management system- User inputs should be in SQL query format, while system outputs should be a report file either as a text document or pdf.

6.0 Bibliography

AIStor Object Store Documentation. (n.d.). *Operations*. AIStor Object Store Documentation. https://docs.min.io/enterprise/aistor-object-store/operations/

Getting started. Apache Kafka. (n.d.). https://kafka.apache.org/41/getting-started/

Morley, L. (2026, January 27). *Building a modern data lake using Open-source tools*. OpenMetal IaaS. https://openmetal.io/resources/blog/building-a-modern-data-lake-using-open-source-tools/

Overview. Parquet. (2025, November 12). https://parquet.apache.org/docs/overview/

User, G. (n.d.). *Documentation*. DuckDB. https://duckdb.org/docs/stable/

What is a data lake? - introduction to Data Lakes and analytics - AWS. (n.d.). https://aws.amazon.com/what-is/data-lake/