1. **Define high level design principles, if any, that should govern the system**

High-Level Design Principles:

Scalability: The system should be designed to handle large volumes of data and be able to scale horizontally or vertically as the platform grows.

Flexibility: The architecture should be flexible to accommodate evolving data requirements and allow for easy integration of new data sources or components.

Reliability: The infrastructure should ensure data integrity, availability, and reliability by implementing robust backup, disaster recovery, and fault tolerance mechanisms.

Security: A strong emphasis should be placed on data privacy, access control, encryption, and compliance with relevant regulations and standards.

Performance: The system should be optimized for efficient data processing, storage, and retrieval to provide real-time or near-real-time insights and analytics.

Architecture Design:

A proposed architecture for the data infrastructure of The Room's intelligence platform could include the following components:

Data Ingestion Layer: This layer handles the collection of data from various sources, including external APIs, user interactions, and internal systems. It should support batch and real-time streaming data ingestion.

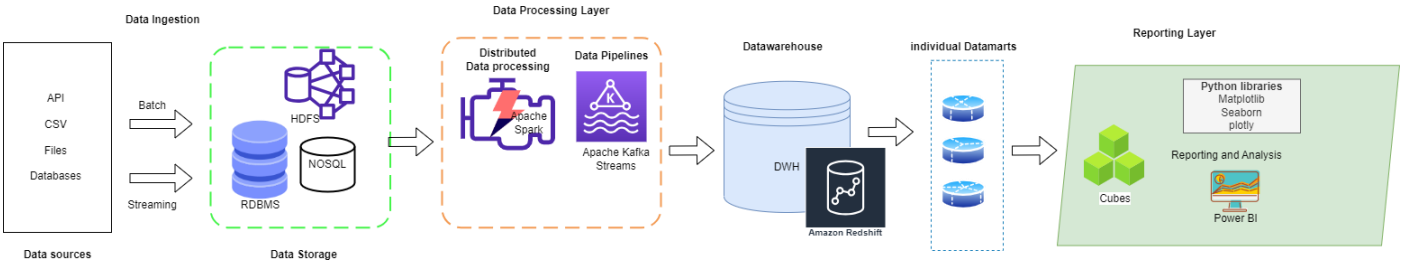
Data Storage Layer: This layer stores the ingested data in a scalable and efficient manner. It can incorporate a combination of technologies such as a distributed file system (e.g., Hadoop Distributed File System or HDFS), a NoSQL database (e.g., MongoDB, Cassandra), and a relational database management system (RDBMS) for structured data.

Data Processing Layer: This layer encompasses the processing and transformation of raw data into usable formats. It includes components like extract, transform, load (ETL) processes, data pipelines, and batch/stream processing frameworks (e.g., Apache Spark) for data cleansing, enrichment, and aggregation.

Data Warehouse/Database Layer: This layer serves as a centralized repository for structured and curated data. It can employ a data warehouse (e.g., Amazon Redshift, Google BigQuery) or a data lake (e.g., Apache Hadoop) for efficient storage, querying, and analysis of large volumes of structured and semi-structured data.

Analytics and Reporting Layer: This layer facilitates data exploration, visualization, and reporting. It may include business intelligence tools (e.g., Tableau, Power BI) or custom-built dashboards to provide insights and actionable intelligence to stakeholders.

2. **<u>Design the architecture for such a system (you can make any assumptions you wish to, but please state these assumptions in your submission)</u>**



Assumptions :

In our proposed architecture, we have made certain assumptions to ensure scalability, flexibility, and efficiency. Here are the key assumptions:

Scalability: The architecture is designed to handle large volumes of data and can scale horizontally or vertically as the data requirements and user base increase. This ensures that the system can accommodate future growth and increasing demands.

Real-time and Batch Data: The data ingestion layer supports both real-time streaming data and batch data processing. We assume that appropriate technologies like Apache Kafka, Apache Nifi, AWS Kinesis, Apache Airflow, or AWS Glue are used to handle these data types effectively.

Distributed File System: The data storage layer assumes the utilization of a distributed file system, such as Hadoop Distributed File System (HDFS). This choice enables efficient storage and management of large volumes of data across multiple nodes in a distributed environment.

NoSQL and Relational Databases: The data storage layer assumes the use of NoSQL databases like Apache Cassandra or MongoDB for storing semi-structured or unstructured data. It also assumes the use of relational databases like MySQL or PostgreSQL for structured data storage. The specific choice of database depends on the characteristics and requirements of the data being stored.

Data Processing with Apache Spark: The data processing layer assumes the use of Apache Spark, a powerful distributed data processing framework. Apache Spark enables data transformation, aggregation, and analysis at scale, ensuring efficient processing of large datasets.

Data Warehouse: The data warehouse/database layer assumes the utilization of cloud-based data warehousing solutions such as Amazon Redshift, Google BigQuery, or Snowflake. These solutions offer efficient storage, querying, and analysis capabilities for structured data, supporting the needs of a data-driven product like The Room.

Data Visualization: The analytics and reporting layer assumes the use of visualization tools like Tableau or Power BI for data exploration, reporting, and generating actionable insights. Additionally, it allows for the flexibility of custom-built solutions using Python libraries such as Matplotlib, Seaborn, or Plotly for more specific visualization requirements.

By making these assumptions within the architecture, we ensure that the system can handle large-scale data processing, storage, and analysis efficiently. It provides the necessary flexibility, scalability, and visualization capabilities to support the ambitious goals of The Room.

3. **Lay out the technology choices (with justifications) for your proposed architecture**

Technology Choices:

Data Ingestion	Apache Kafka, Apache Nifi, or AWS Kinesis for real-time streaming data, Apache Airflow or AWS Glue for batch data.
Data Storage	Hadoop Distributed File System (HDFS), Apache Cassandra, MongoDB, or a combination based on the data requirements.
Data Processing	Apache Spark for distributed data processing, Apache Beam for data pipelines, and Apache Kafka Streams for stream processing.
Data Warehouse/Database	Amazon Redshift, Google BigQuery, or Snowflake for scalable data warehousing and analysis.
Analytics and Reporting	Tableau, Power BI, or custom-built solutions using Python libraries like Matplotlib, Seaborn, or Plotly for data visualization.

Justifications of chosen technologies :

Data Ingestion Layer:

Real-time streaming data: Apache Kafka / Apache Nifi / AWS Kinesis

Justification: These technologies are widely used for handling real-time streaming data. Apache Kafka is a distributed streaming platform known for its scalability, fault-tolerance, and low-latency capabilities. Apache Nifi provides a visual interface for data flow management and supports various data sources and destinations. AWS Kinesis is a fully managed service that simplifies the handling of real-time data ingestion on the AWS cloud platform.

Batch data: Apache Airflow / AWS Glue

Justification: Apache Airflow is a popular open-source platform for orchestrating and scheduling batch data workflows. It provides a rich set of features for managing dependencies, retries, and monitoring. AWS Glue is a fully managed extract, transform, load (ETL) service that can handle large-scale data ingestion and transformation in batch mode, particularly well-suited for AWS environments.

Data Storage Layer:

Distributed file system: Hadoop Distributed File System (HDFS)

Justification: HDFS is a widely used distributed file system known for its scalability and fault-tolerance. It is suitable for storing and managing large volumes of data across a cluster of machines, making it a popular choice for big data storage and processing.

NoSQL database: Apache Cassandra / MongoDB

Justification: Apache Cassandra and MongoDB are popular NoSQL databases known for their ability to handle large amounts of semi-structured or unstructured data. They offer high scalability, fault-tolerance, and flexible data modeling capabilities, making them suitable for storing diverse data types and accommodating rapid growth.

Relational database: ***Depending on requirements (e.g., MySQL, PostgreSQL)***

Justification: The choice of a relational database depends on specific requirements such as data structure, transactional consistency, and relational querying capabilities. MySQL and PostgreSQL are well-established, open-source relational databases known for their reliability, performance, and extensive feature sets.

Data Processing Layer:

Distributed data processing: Apache Spark

Justification: Apache Spark is a powerful distributed computing framework that provides fast and scalable data processing capabilities. It supports in-memory data processing, fault tolerance, and a wide range of data transformation operations. Spark's flexibility and efficiency make it a suitable choice for processing large volumes of data in parallel.

Data pipelines: ***Apache Beam / Apache Kafka Streams***

Justification: Apache Beam is a unified programming model for building batch and streaming data processing pipelines. It provides a high-level API that supports multiple execution engines, allowing flexibility in choosing between different processing backends. Apache Kafka Streams is a stream processing library that enables developers to build real-time applications using Apache Kafka as the underlying messaging system. Both technologies offer powerful capabilities for building and managing data pipelines.

Data Warehouse/Database Layer:

Data warehousing: ***Amazon Redshift / Google BigQuery / Snowflake***

Justification: These cloud-based data warehousing solutions offer scalable and cost-effective options for storing and analyzing large volumes of structured data. They provide optimized query performance, data compression, and parallel processing capabilities, making them well-suited for data warehousing use cases.

Analytics and Reporting Layer:

Data visualization: ***Tableau / Power BI***

Justification: Tableau and Power BI are industry-leading data visualization tools that provide a wide range of interactive visualizations, dashboards, and reporting capabilities. They offer user-friendly interfaces, extensive integration options, and support for connecting to various data sources.

Custom-built solutions: ***Python libraries (Matplotlib, Seaborn, Plotly)***

Justification: Python libraries such as Matplotlib, Seaborn, and Plotly provide flexible options for creating custom data visualizations. They offer a rich set of visualization functions, allowing for greater customization and specialized visual representation as per specific requirements.