**Genna Hilbing**

**Section 1: Consulting Soft Skills**

A key function of this position is producing or contributing to functional specifications and other written or presentation quality technical documentation. Please provide responses that address the following hypothetical “client concerns”. These responses should illustrate your personal writing style and your ability to provide clients with cogent and technically relevant information.

**What is a Data Lake? Explain its benefits, how it differs from a data warehouse, and how it might benefit a client.**

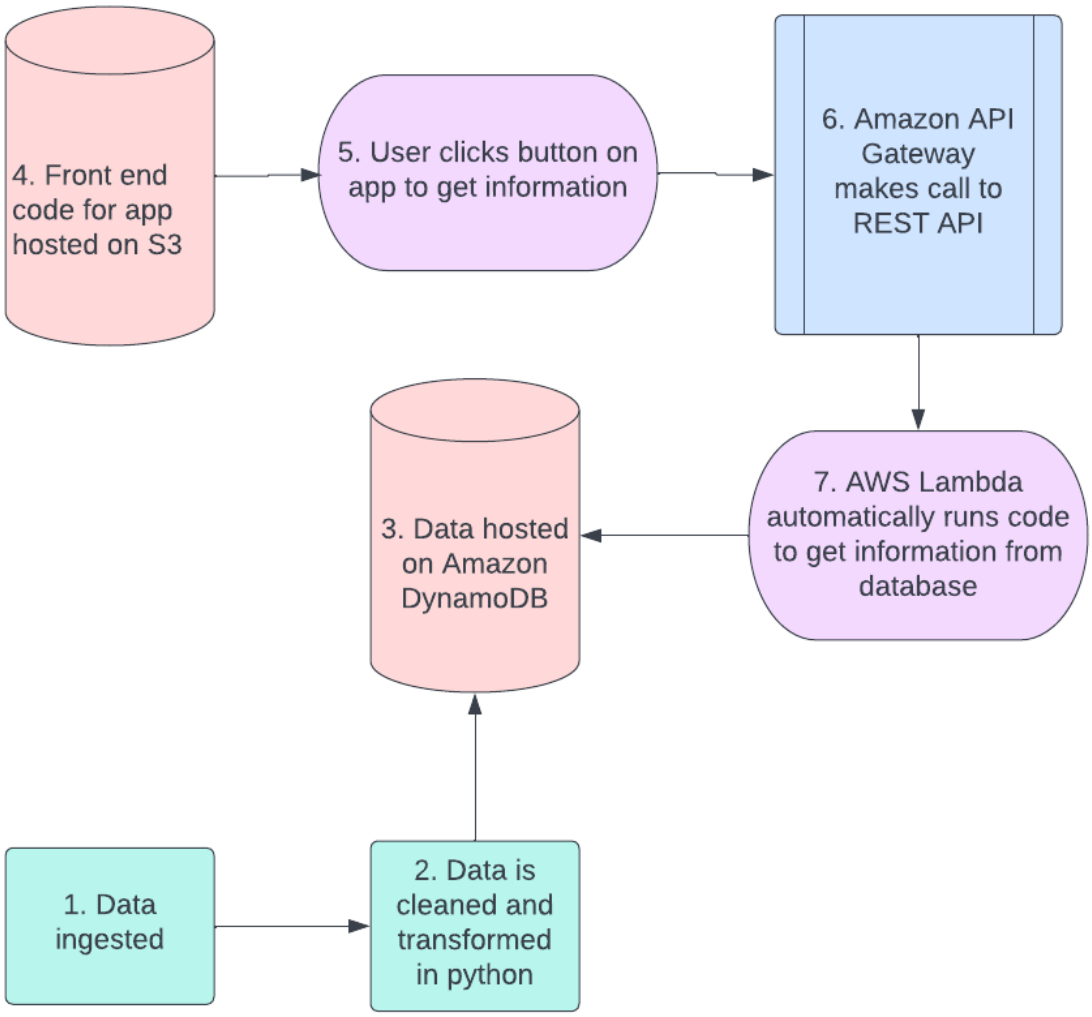
A data lake is a centralized repository, which holds both structured and unstructured data. Data is held “as-is”, in its raw form, from sources like websites, corporate applications, apps and more in both relational and non-relational databases. Data lakes and Data warehouses differ in their underlying architecture, data sources, and use cases. Unlike data lakes, the data in a data warehouse is held in a relational database. This data comes from business applications (e.g. transactional systems) and is highly curated for use in business intelligence and visualizations. In contrast, the raw data in a data lake can be used by data scientists for purposes such as big data, analytics, and machine learning. A data warehouse would benefit a client who is looking to query data in a “neat and tidy” format for business intelligence purposes, while a data lake would benefit a client looking to perform larger-scale actions such as big data analysis or creating machine learning models (https://aws.amazon.com/data-warehouse/).

**Explain serverless architecture.  What are its pros and cons?**

Serverless architecture refers to operations and tools (e.g. databases, api’s, etc.) that are hosted on a platform that takes care of all “server” operations. While the platform still runs on a server, the configuration of this server is taken care of by the platform host (e.g. aws, gcp, etc.). This lack of control over the “server” has its pros and cons; while it can be helpful to have another entity handle all server configuration, this lack of control can lead to vendor lock-in. If ease of access and use is most important to a client, then serverless architecture could be a viable solution. Serverless architecture also allows clients to interact with databases and services without needing technical knowledge of the underlying processes. However, for clients that want more control over their processes (e.g. they would like to configure servers for multiple types of processes from multiple vendors), then traditional server architecture should be used. (https://aws.amazon.com/serverless/)

**Please provide a diagram for an ETL pipeline (e.g., Section 2) using serverless AWS services. Describe each component and its function within the pipeline.**

ETL pipeline for AWS lambda with DynamoDB:




1. Data is ingested from a source, such as: internal data, external, data, an API, etc.
2. Data is cleaned and transformed in python using tools such as Pandas and PySpark.
3. The transformed data is hosted on DynamoDB.
4. The front end code for an app is hosted on S3.
5. A user clicks a button to request information from the website.
6. The Amazon API Gateway makes a call to the REST API.
7. The REST API call triggers AWS Lambda to run code to get information from the database.

**Describe modern MLOps and how organizations should be approaching management from a tool and system perspective.**

Modern MLOps should focus on three major factors: efficiency, scalability, and risk reduction. Since MLOps require multiple steps (e.g. data ingestion, data cleaning, model training, etc.) with multiple people working on each step (e.g. data engineers, data scientists, data analysts), it's important for pipelines to run smoothly and allow for this constant collaboration. With this in mind, processes on a cloud, like aws, will need to be configured with the correct security group rules to allow individuals to access these operations. Containerized environments, like Docker, can also be used to facilitate environment creation for better communication and set up of processes. Processes should also be created with scalability in mind: when more data/processes are needed, how can this be added to the operation? For example, while the Pandas python package works great for exploratory data analysis, packages like Apache Spark work better when mass amounts of data need to be ingested and altered. Architecture also requires scalability, and decisions should be made such as whether servers will run on distributed systems and how much and how often systems should be backed up. Finally, MLOps need to incorporate risk reduction. Given the incessant flow of data preparation, model training, testing, and model deployment, models and processes need to be constantly checked for errors such as model drift and failed processes.

(https://databricks.com/glossary/mlops)