HIGH LEVEL DOCUMENTATION -3

PROJECT-3

**Project Title:** **Capture transaction data from SQS queue**

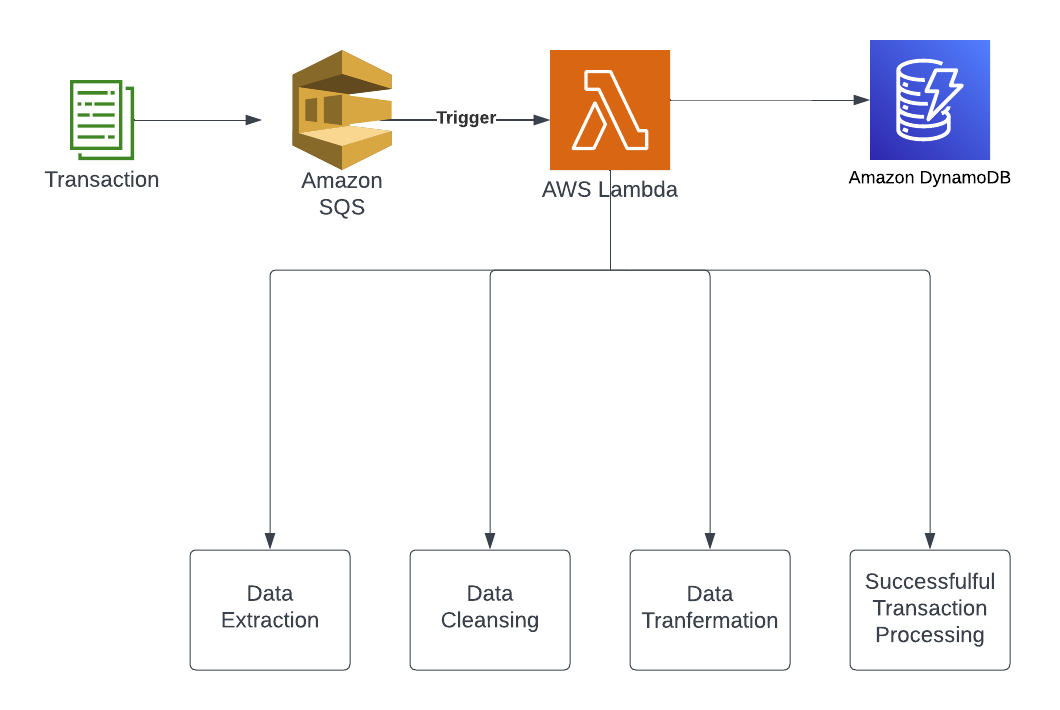
**Infrastructure Components:**

* AWS SQS: Message queue service for receiving live transaction data from a third-party system.
* AWS Lambda: Serverless compute service for processing and handling the received messages.
* DynamoDB: NoSQL database for storing Account Master and Source System Master data.

**Detailed Workflow:**

* Third-Party System Integration:
* The third-party system will post live transaction data to the SQS queue in JSON format.
* The SQS queue acts as a buffer, decoupling the transaction processing from the third-party system.
* Message Processing:
* AWS Lambda functions are triggered by messages from the SQS queue.
* The Lambda function retrieves the message from the queue.
* The JSON message is parsed to extract the relevant transaction details.
* Validations are performed on the transaction data.
* Transaction Creation:
* The Lambda function interacts with DynamoDB to fetch the necessary data from the Account Master and Source System Master tables.
* Based on the transaction details, three separate transactions are created for each incoming transaction.
* The transactions are recorded in the Ledger Transaction table in DynamoDB.
* Transaction Credit:
* The transactions are credited to the appropriate accounts based on the transaction type and amount.
* The "Product Sale" account, "Value Added Tax" account, and "Excise Duty" account are credited accordingly.
* Error Handling:
* If a transaction fails for any reason during processing or validation, it is flagged as an error.
* Failed transactions are recorded in a separate file for reload purposes.
* System Maintenance and Reporting:
* Regular maintenance tasks are performed to ensure the system's availability, performance, and data integrity.
* Reporting mechanisms can be implemented to generate transaction reports, monitor system performance, and track any errors or failures

**Dataset Block Diagram:**



**Data Models:**

* Account Master Table:
* Stores information about various accounts, including account number, name, description, and type.
* Ledger Transaction Table:
* Stores details of each transaction, such as transaction ID, voucher code, transaction type, date, account number, amount, source system ID, and source system transaction ID.
* Source System Master Table:
* Contains information about different source systems, including system ID and system name.

**Acceptance Criteria:**

* For each incoming transaction, three separate transactions are created in the Ledger Transaction table.
* The transaction amounts are credited to the appropriate accounts, namely "Product Sale," "Value Added Tax," and "Excise Duty."
* In case of any transaction failure, a separate file of failed records is generated for reloading purposes.

# **REQUIREMENTS :**

**Retrieval from SQS**: Retrieve messages in JSON format from the SQS queue.

**Lambda Processing**: Route the messages to AWS Lambda for further processing.

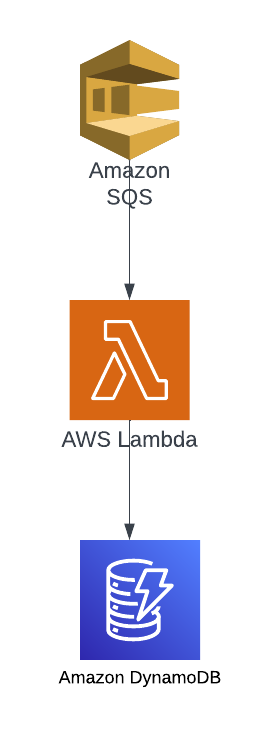
**Lambda Function Usage:** Utilize AWS Lambda functions to perform serverless data processing.

**Event Trigger:** Trigger Lambda functions based on events received from the SQS queue.

**Data Validation and Transformation:** Perform necessary validation and transformation operations on the data within the Lambda functions**.**

**Data Storage in DynamoDB**: Store the processed data in a DynamoDB table for persistence and future retrieval**.**

**High-level diagram**



## **ARCHITECTURAL PLAN :**

## The architecture follows a distributed and scalable approach to handle the transaction data. It consists of the following key components:

**Data Retrieval Component**

* Identify and integrate with transaction data sources as message, such as message from AWS SQS in json format..
* Implement a data loading process to extract data from the sources and transform it into a suitable format for processing.

**Data Processing Component**

# Lambda is a serverless compute service that allows running code without provisioning or managing servers. It is used to process the incoming messages from the SQS queue, performing necessary transformations and validations on the JSON data.

**Store Processed Data Component:**

DynamoDB is a fully managed NoSQL database service provided by AWS. It offers high availability, scalability, and low latency for storing and retrieving data. In this project, DynamoDB is used to store the processed transaction data.

**OUTPUT SCOPE :**

The project does not involve the development of a user interface or visualization components.

The focus is on retrieving data from the SQS queue, processing it using AWS Lambda, and storing the processed data means message in a DynamoDB table.