**Low-Level Design**

**Project 2**

**Upload EOD Transactions files into System**

**Infrastructure Components:**

* AWS S3: Bucket to store incoming transaction files.
* AWS Lambda: Serverless compute service to process transactions.
* DynamoDB: NoSQL database for storing Account Master and Source System Master data.
* AWS Glue: Optional service for data transformations.

**Project Flow Block Diagram:**

**A picture containing text, screenshot, diagram, line

Description automatically generated**

**Detailed Workflow:**

* Incoming Files: Two types of files are received: Parquet and Avro.
* File Upload: The files are uploaded to the designated S3 bucket.

**Lambda Function:**

* Trigger Configuration: Configure the S3 bucket as a trigger for the Lambda function.
* Event Notification: Configure event notifications for new file arrivals.
* File Processing: Lambda function gets triggered when a new file is uploaded.
* Read the file from the S3 bucket.
* Determine the file type (Parquet or Avro) based on the file extension.
* Parse and extract the relevant data from the file.

**Parquet File Processing:**

* Data Extraction: Extract the necessary fields (trnrefid, code, tdate, trn\_amount, vat, excise\_duty).
* Transaction Generation:
* Generate a voucher code from the trnrefid.
* Create three separate transaction entries for each transaction:
* Credit "Product Sale" with the trn\_amount.
* Credit "Value Added Tax" with the vat.
* Credit "Excise Duty" with the excise\_duty.
* Prepare the transaction data to be stored in the ledger\_txn table.
* Validation:
  + Check for the validity of the transaction date (should not be a future date).
* Failed Transaction Handling:
* Create a separate file containing the failed records for reloading purposes.

**Avro File Processing:**

* Data Extraction: Extract the necessary fields (Tran\_ref\_id, Transaction\_dt, amt, gst, custom\_duty).
* Transaction Generation:
* Generate a voucher code from the Tran\_ref\_id.
* Create three separate transaction entries for each transaction:
* Credit "Product Sale" with the amt.
* Credit an appropriate account (not specified in the requirements) with the gst.
* Credit "Custom Duty" with the custom\_duty.
* Prepare the transaction data to be stored in the ledger\_txn table.
* Validation:
* Check for the validity of the transaction date (should not be a future date).
* Failed Transaction Handling:
* Create a separate file containing the failed records for reloading purposes.

**Database Operations:**

* Account Master Table (acct\_master):
* Store account-related information (acc\_no, acc\_name, acc\_desc, acc\_type).
* Use DynamoDB for its scalability and performance.
* Define appropriate indexes for efficient querying.
* Source System Master Table (src\_sys\_mst):
* Store information about various data source systems (system\_id, system\_name).
* Use DynamoDB for its scalability and performance.
* Define appropriate indexes for efficient querying.
* Transaction Table (ledger\_txn):
* Store the generated ledger transactions.
* Use DynamoDB for its scalability and performance.
* Define indexes to support efficient querying and reporting.

The LLD provided outlines the detailed design considerations and steps involved in the system implementation. However, the actual implementation may require additional modifications and optimizations based on specific requirements, resource constraints, and best practices.