Technical Assessment Document

# Task: Customer Counter Change Detection Export and Upload Job (job\_CustomerCounterExportUpload)

## 1. Introduction

## The SK\_II Beauty Project is a microservices-based application used by beauty counsellors in SK\_II stores to analyse customers' skin and recommend personalized beauty products. It is already deployed in the Japan market and is now expanding to Hong Kong and Taiwan.

## As part of this project, a new requirement has been introduced to track customer visits across multiple counters (stores). A default counter is initially assigned based on where the customer makes their first purchase. However, if the customer frequently visits a different counter throughout the month, the default counter needs to be updated to reflect the most visited store.

## To achieve this, a job will execute at the end of each month, analysing visit data for all customers and updating default counters if needed. The job will export data for all customers whose default counters have changed and upload the file to Cloud Storage, as per the client's requirement. The project leverages Spring Boot 2.5.13 for backend logic and Azure services for database, storage, key vault, and service bus integration.

## 2. Scope of the Task

The scope of this task is to implement a job that exports customer details when their default counter is changed based on monthly visit analysis. This task complements an existing job that already updates the default counters by identifying the most visited counter for each customer. Below are the key objectives:

* **Monitor Counter Changes:**

Track changes made to customers’ default counters by the existing job, which updates the counter based on the customer’s monthly visit patterns.

* **Export Data for Changed Counters:**

Generate a CSV or Excel file containing business-relevant information for customers whose default counter has changed. The file will include key fields such as Customer ID, Old Counter, New Counter, Change Date, and other relevant business data.

* **Upload the Exported File to Cloud:**

The generated file will be uploaded to Cloud Storage, ensuring secure and reliable storage for the client's reporting and analytics needs.

* **Ensure Timely Execution:**

The job will run immediately after the monthly counter updates are completed to capture and export the most recent changes to customer counters.

## 3. Functional Requirements

## The following functional requirements outline the key features and behaviours of the job responsible for exporting customer counter changes and uploading the data to AWS Blob Storage:

## Detect Counter Changes:

## The job will identify customers whose default counter has been updated by the existing monthly counter update job.

## Retrieve Business Data:

## Fetch relevant business data from the database for all customers whose default counter has changed.

## Include information such as Customer ID, Previous Counter, New Counter, Change Date, and Additional Business Data as needed.

## Generate CSV/Excel File:

## Create a CSV or Excel file with the retrieved customer data.

## The file will follow a predefined structure with columns like Customer ID, Old Counter, New Counter, Change Date, etc.

## Upload to Cloud Storage:

## Upload the generated file to the designated AWS Blob Storage location provided by the client.

## Ensure the upload is secure and reliable, using proper authentication and error handling mechanisms.

## Error Handling and Logging

## Log the status of the job (success/failure) using a logging framework.

## In case of an error during file generation or upload, retry the operation and log the error details.

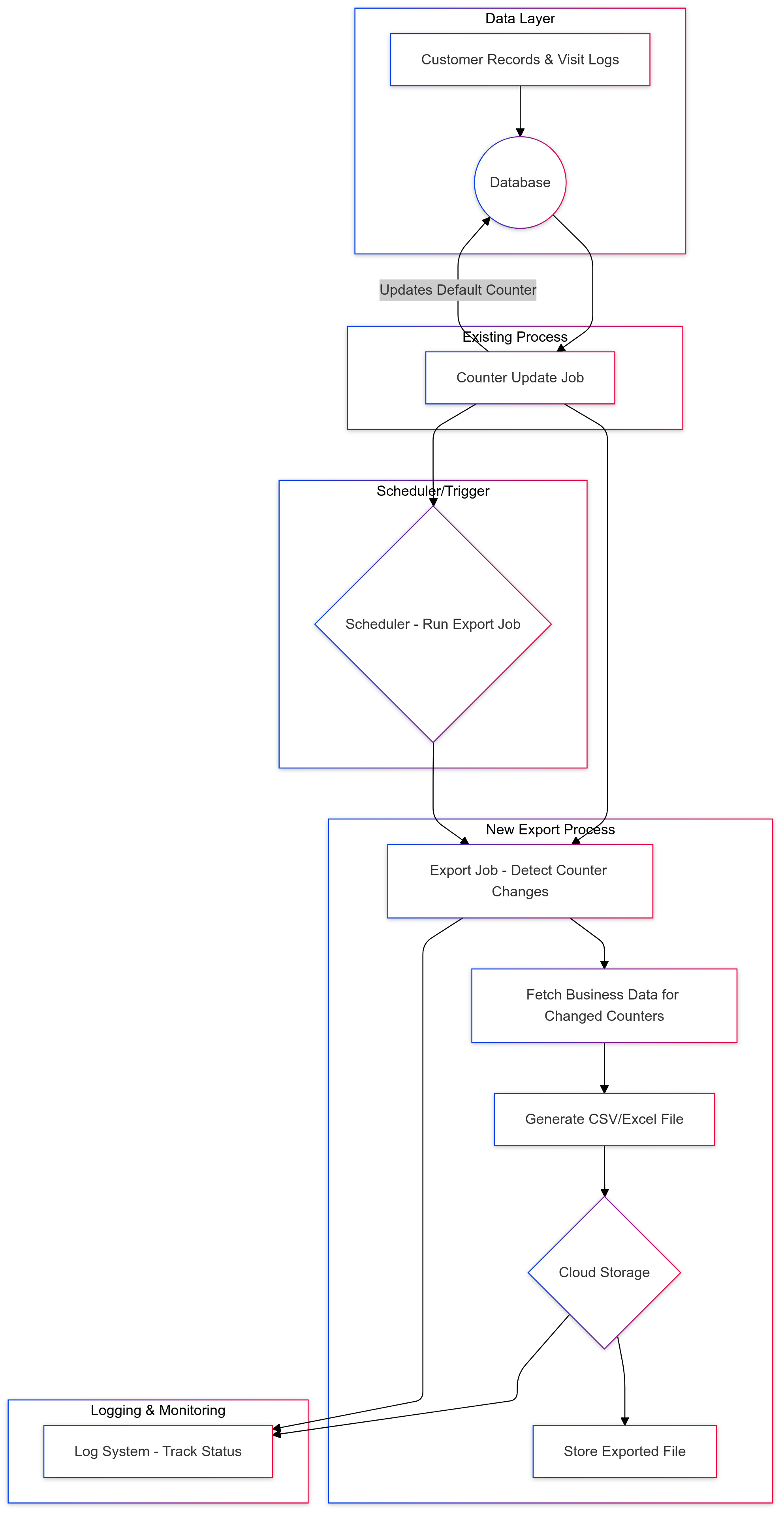
## Job Execution and Scheduling:

## The job will be triggered automatically after the monthly counter update job is completed.

## Ensure the job runs only once per month and captures the latest changes accurately.

## 4. System Design Overview

### 4.1 Architecture Diagram



### 4.2 Flow of Execution

## The following outlines the step-by-step flow of execution for the customer counter change export job:

## Counter Update Job Execution

## At the end of each month, the existing Counter Update Job runs.

## This job analyzes the customer visit data from the database and updates the default counter to the most visited counter for each customer, if necessary.

## Triggering the Export Job

## Once the Counter Update Job completes, the Export Job is triggered automatically by a scheduler or event-based trigger.

## Detect Counter Changes

## The Export Job identifies customers whose default counter was changed by the Counter Update Job.

## It queries the database to retrieve the updated counter details along with relevant business data.

## Fetching Business Data

## For each customer with a counter change, the job fetches additional business information (e.g., Customer ID, Previous Counter, New Counter, Change Date, and other required fields).

## Generating the Export File

## The retrieved data is compiled into a CSV or Excel file.

## The file follows a predefined structure, including all necessary fields to support reporting or analytics.

## Uploading the File to Cloud Storage

## The generated CSV/Excel file is uploaded to Cloud Storage.

## The job ensures the file is uploaded securely, using appropriate authentication mechanisms.

## 5. Database

**1. Tables**

VIP\_SUMMARY, need to add the new column

## 6. Technical Approach

### 6.1 Job Implementation Using YAML Configuration and Java Classes

### This job leverages YAML configuration files to define essential properties such as FTP/SFTP details, paths, and cron expressions. It also uses Spring Boot configuration classes to manage the job triggers and logic classes for data export and upload. Below are the key aspects of the implementation:

### YAML Configuration

### The YAML file contains the job settings such as:

### SFTP connection details (host, port, username, password)

### Local and remote paths for data files

### Cron expression for scheduling

1. **Configuration Class for Job Setup**

* The **ScheduleConfiguration** class reads the YAML properties and creates the JobDetail and Trigger beans required to schedule and trigger the job.

1. **Job Logic Implementation**

* The **CustomerCounterExportUpload** class contains the logic for fetching customer data, generating the CSV file, and uploading it to cloud Storage.
* Key Logic of the Job Class:
* Fetching data from the database: Retrieves customers whose default counters were updated.
* Generating the CSV file: Creates a structured export file with relevant customer data.
* Uploading to AWS: Uploads the generated file to AWS Blob Storage.

### ****7. Testing Plan****

The testing plan ensures that the **export job logic** works as expected through **unit tests** for isolated components and **integration tests** for validating the complete flow export and upload.

## 12. Assumptions

This section outlines the assumptions and constraints for the successful execution of the export job.

**Assumptions**

1. **Data Accuracy and Accessibility:**

* It is assumed that the customer data and visit logs in the database are accurate and accessible at the time the job is executed.
* All counter updates are completed by the existing counter update job before the export job begins.

1. **Cloud Storage Availability:**

* Cloud Storage will be available and responsive at the time of upload.
* Necessary access permissions and credentials are correctly configured to allow the job to upload files securely.

1. **Scheduled Job Execution:**

* The export job will run immediately after the counter update job is completed at the end of each month.
* Any change to the scheduled execution time or frequency will be communicated and configured in advance.

1. **No Concurrent Modifications:**

* It is assumed that no concurrent updates to customer counters will occur while the export job is running, ensuring data consistency.

## 13. Timeline and Milestones

* Phase 1: Gather requirements and design (2 days).
* Phase 2: Implement logic (3 days).
* Phase 3: Test and deploy (2 days).
* Phase 4: Monitor and optimize (1 day).