Assignment 3:

Submission Deadline

June 30, 2025 (23:59).

Goal

Implement an event-driven serverless application to perform profanity check and sentiment analysis of customer's reviews. You will be working with the same group as in Assignment 1 and 2.

> Please note, the scope of the assignment is on the design and the implementation of a serverless application rather than of specific application KPIs (e.g., processing time, accuracy).

Enviroment

This time, you'll be using your laptops (or whatever you prefer) properly configured with LocalStack (please refer to the "Enviroment Setup", "Tutorial Instructions" and "Tips & Tricks" documents).

> Please note, LocalStack enviroment is ephimeral, which means that when you shut it down and then restart it, you **MUST** recreate all the resources (S3 buckets, DynamoDB tables, Lambda functions, etc...).

Introduction

You're a team of data scientists and sw. engineers who are required to design and develop a serverless application using AWS Lambda functions and related AWS PaaS services (i.e., S3 and DynamoDB).

The application aims to analyze customer's reviews and perform the following main tasks:

- Preprocess the review text and summary (i.e., tokenization, stop word removal, lemmatization).
- Checking for the presence of bad words, that is profanity check.
- Analyzing the sentiment of a review.
- Counting the number of unpolite reviews (i.e., containing bad words) inserted by a customer.
- Mark a customer as banned when they inserted more than 3 unpolite reviews.

Requirements

1. The functionalities have to be implemented in **at least three Lambda functions**, such as, `preprocessing`, `profanity-check`, and `sentiment-analysis`.
2. The function chain has to start when a **new, single review** is inserted in a S3 bucket.
3. Other function invocations must be triggered by S3 buckets and/or DynamoDB events (i.e., object created, item inserted, ...).
4. For each review, `summary`, `reviewText`, and `overall` fields have to be taken into consideration for profanity and/or sentiment analysis.
5. Bucket names, table names, and other parameters must be retrieved by the parameter store (SSM) (i.e., check the tutorial lambdas' code).

6. Automated integration tests that verify your function chain main functionalities must be implemented (preprocessing, profanity check, sentiment analysis, counting unpolite reviews and banning a user). Check the `tests/` folder of the tutorial for a starting point.

Dataset

You can test your application with the `reviews_devset.json`, plus you can add your own reviews for testing corner cases, if necessary.

> If you insert additional reviews for testing corner cases (e.g., unpolite reviews), please insert them in the final submission archive.

Results & Report

The results you need to provide are:

- The number of positive, neutral, and negative reviews present in the `reviews_devset.json`;
- The number of reviews that didn't pass the profanity check;
- Users resulting in a ban, if any.

> These results should take into account only the `reviews_devset.json` provided, not for your additional testing reviews.

Produce a `report.pdf`, that contains detailed at least five sections:

1. Introduction
2. Problem Overview
3. Methodology and Approach
4. Results
5. Conclusions

The report **MUST** contain an architectural diagram showing the function chain, how they interact with other PaaS service and what are the trigger events.

Submission

Submission Files

Please submit a single file named `<GroupID>_DIC2025_Assignment_3.zip` that contains:

- `report.pdf`: 8 page report (max), 11pt font size, one column format
- `instructions.pdf`: a document describing how to execute your applications (i.e., similar to the **Tutorial_Instructions** document)
- `src/`: subdirectory containing **very well documented source code**, additional testing reviews (if any), your **integration test** files.

Submission Procedure

Submit your solution via TUWEL before **30.06.2025 (23:59)**.