CSE 546 — Project Report

Cyriac Biju Narayamparambil

Aksa Elizabeth Sunny

Saran Prasad

1. Problem statement

Build a PaaS solution in AWS for performing facial recognition from video frames to recognize students and return relevant academic information for the same. Use PaaS services like Lambda functions, S3 buckets for persistence and DynamoDB for storing academic information for students.

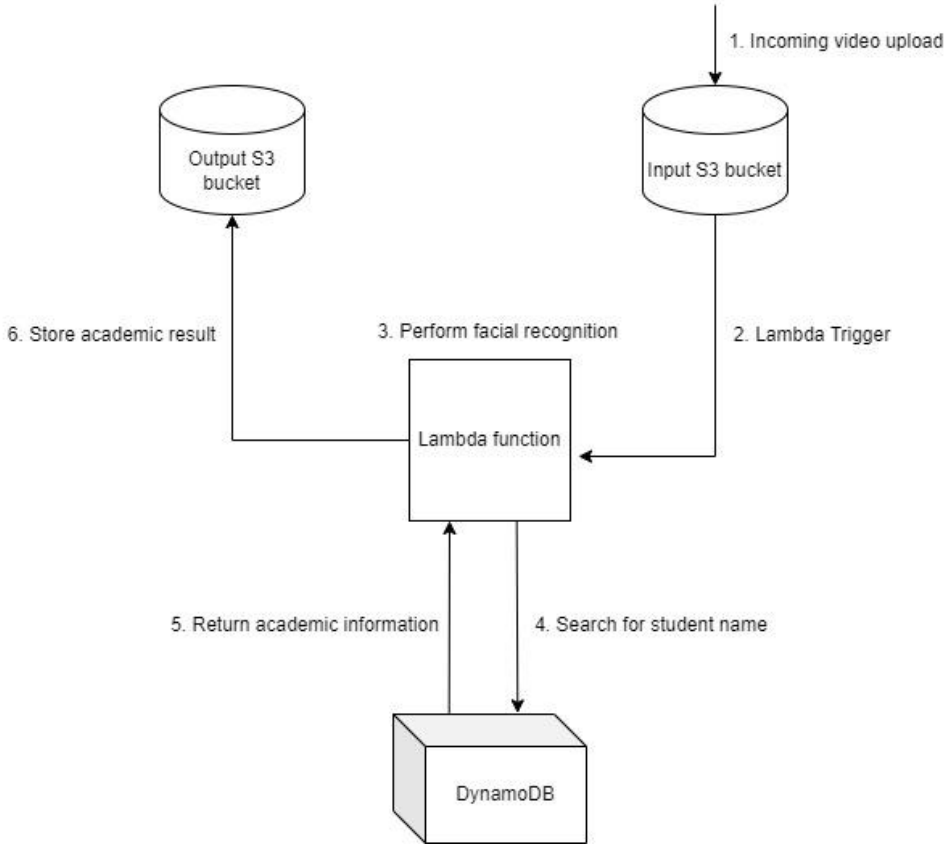
2. Design and implementation

The technical details are the following,

1. **Input S3 bucket and Lambda triggers**
    a. Input s3 bucket is created to store input video files.
    b. An event notification is created for all object create events to trigger the facial-recognition lambda function.
    c. When a file is uploaded to the S3 bucket, an event notification is sent to the AWS Lambda service. This event contains information about the S3 object that was created. AWS Lambda then invokes the function that is associated with the S3 bucket event.
2. **Facial recognition**
    a. The Lambda function is set up as a container which is created according to specifications in the Dockerfile. This includes downloading dependencies, copying files and triggering the handler (endpoint).
    b. The handler (handler.py) recognizes the bucket and uploaded file (key) using the event parameter passed.
    c. Using boto3, the handler downloads the .mp4 file from the input S3 bucket.
    d. After the download, the first frame is extracted using ffmpeg and stored as a .jpeg file.
    e. Face_recognition module is used to create an encoding for the first recognized face from the .jpeg file and compared against known encodings given.
    f. After the best match, the corresponding student's name is obtained.

3. **Search using DynamoDB**
    a. Table is created in DynamoDB (Student-Data) and populated with the student data provided in student_data.json.
    b. Student name is used as the partition key for the table.
    c. Using boto3, the handler searches and retrieves the student name matched using facial recognition in DynamoDB if it exists.
    d. Only the name, major and year are retrieved from the database for the matching record.
    e. The retrieved data is stored as a csv file in the output bucket in S3 using boto3 with the name of the input file as the key.

2.1 Architecture



Steps in process pipeline:

1. A video (.mp4) is uploaded into the input S3 bucket by a client.
2. S3 bucket stores the video file and triggers an event for a configured Lambda function.

3. The Lambda function receives the event and performs facial recognition. This is done using ffmpeg to extract the first frame of the mp4 and perform facial recognition to identify the first face in the corresponding frame.
4. The identified student's (by comparing face encodings of known students against the identified face from first frame) name is searched in DynamoDB.
5. If there is a match, DynamoDB returns the corresponding student's academic information.
6. The academic information is stored in an output S3 bucket for persistence.

2.2 Autoscaling

Autoscaling is achieved using lambda functions. We first build a dockerized image that is pushed to a private Elastic Container Repository. The lambda function is given that prebuilt image as the entity to be executed. We also set a trigger for the lambda function such that the function starts up when an object is created in the input bucket. Therefore, whenever a new object is created in the input bucket, a new lambda function instance is created and executed that does the face recognition task and sends output to the output bucket. Thus, the number of functions is proportional to the number of input objects given.

2.3 Member Tasks

- **Saran Prasad**
    - Completed tasks mentioned in **Facial recognition** section in **Design and implementation.**
    - Set up input and output S3 buckets.
    - Developed code for the handler function (handler.py) executed within Lambda functions for handling incoming .mp4 file and its conversion to a .jpeg corresponding to the first frame.
    - Developed code for facial recognition and matching to a known list of students.
- **Cyriac Biju Narayamparambil**
    - Set up a test lambda function to test the working of lambda functions when using dockerized containers.
    - Set up a private Elastic Container Repository for the same. Changed the lambda function configuration to get it working.
    - **Constructed** an **image** from the code developed by Sharan and Aksa and pushed it to the ECR repository. Created the **lambda function** that deploys the pushed image.
    - Tested the working of the code by running workload.py

- **Aksa Elizabeth Sunny**
    - Completed tasks mentioned in **Search using DynamoDB** section in **Design and implementation**.
    - Created and populated table in DynamoDB with student data.
    - Verified data fetching using key value from DynamoDB (tried Global Secondary Index as well).
    - Developed code for the handler function (handler.py) executed within Lambda functions for fetching only the required attributes from DynamoDB based on input key as well as storing the data as file in output S3 bucket.

3. Testing and evaluation

The code was tested both manually and using the automated workload.py tool.

3.1. Manual testing

Some video files were randomly chosen from the test_cases folder and uploaded into the input S3 bucket. The Lambda functions were triggered as expected and output was verified manually in the output S3 bucket.

3.2. Automated testing using workload.py

cmd: *python3 workload.py*

This runs the entire set of test cases.

All the run test cases had corresponding outputs in S3 bucket and were verified to be correct.

4. Code

From the base (forked) project, only the following were changed

1. Dockerfile - The encodings were not copied into the container. The following COPY command was added,

   COPY encoding ${FUNCTION_DIR}

2. handler.py **-** This is the entry point specified in Dockerfile. The python program gets invoked when a trigger is observed. From the triggered event, we obtain the event and context objects which we use to identify the bucket and key that triggered the event. Then, the following functions are done by the handler,
    a. Download video from input bucket that caused the trigger
    b. Run ffmpeg on the video to obtain the first frame and store it as .jpeg
    c. Identify a face and produce its encoding using faciaL_recognition module.

    d. Compare this encoding against known encodings to identify the student name.
    e. Search this student's academic information in DynamoDB using the identified name.
    f. Retrieve and store the academic information in the output S3 bucket.
3. workload.py - The input and output bucket names were changed.