CSE 546 — Project Report

Abhishek Masetty, Shashank Reddy Baradi, Thejas Naik

General requirements:

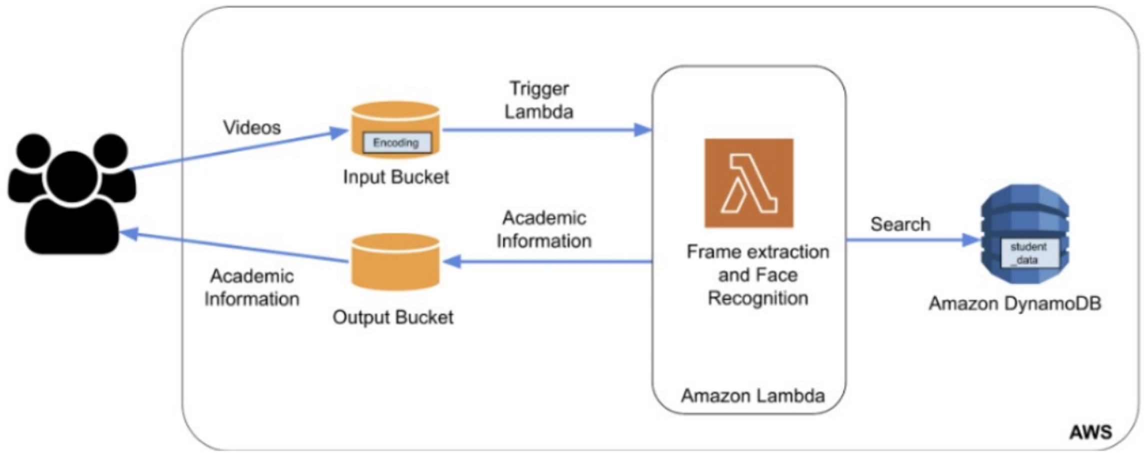
- Strictly follow this template in terms of both contents and formatting.
- Submit it as part of your zip file on Canvas by the deadline.

1. Problem statement

To offer a face recognition service by doing face extraction on the user-provided video clips using cloud resources like AWS lambda, AWS S3, AWS DyanmoDB. It is significant because it enables users to conduct direct searches based on faces seen in videos and receive pertinent results. For instance, you might prefer a particular lecturer from a selection of lecture videos, documentaries about certain people.

2. Design and implementation

2.1 Architecture



1. Input Bucket - This bucket receives the user input and it stores the videos in the bucket.
2. Output Bucket - This bucket gets the input from the lambda function after the running the face recognition algorithm and output of the algorithm is stored in this bucket.
3. AWS Lambda - This is a serverless application which uses a docker image and builts the docker container during the run time. This function has a trigger set and when ever a user uploads a video this function executes and runs a face recognition algorithm and send the output to the S3 bucket.

4. AWS DynamoDB - This a NoSQL DB which stores the data of students, the lambda function will query this db and will returns the relevant information back to the lambda function.

2.2 Autoscaling

AWS will automatically scale itself to handle the traffic because the application employs a lambda function to perform the facial recognition. Nothing more was used in this situation.
Our own multithreaded workload generator was used to test this, and the results showed a considerable improvement in speed with a processing time of less than 2 minutes for 100 photos.

2.3 Member Tasks

1. Abhishek Masetty:
   a. Developed the logic for getting the inputs from the bucket and sending it to the lambda function.
   b. Created the logic along with Shashank to recognize the first face from the video and send the output to the s3 bucket.
   c. Created the logic for reading the student data from dynamodb.
2. Shashank Reddy Baradi:
   a. Created the logic along with Abhishek to recognize the face with ffmpeg and coded the logic
   b. Developed the logic for testing the application using the workload and to test the application's scaling abilities
   c. Created the logic to upload the output csv data to output bucket.
3. Thejas Naik:
   a. Created the initial code base along with the repository.
   b. Created the resources on AWS such as a private repository in ECR and uploaded the dockerfile image to it. As well as the input and output buckets used for data transmission.
   c. Created the logic for iterating through the frames and stopping once a frame with a matching face is found.

3. Testing and evaluation

Through the given workload generator, we carried out load testing by repeatedly submitting 100 requests; each time, the output was completed in less time than the allowed five minutes. On top of that we came up with a multiload generator and tested the application but sending the video clips to the input bucket and the application returned the output in 2 mins.

4. Code

Dockerfile: This file has the implementation of the docker image which contains all the required libraries and various packages in order to make this application running.

Handler.py: This file has the actual logic to extract a face from a video clip and performs the actual face recognition algorithm. This file gets the input from the input bucket, after retrieving the inputs from the bucket it utilized the ffmpeg library by following line:

```
os.system("ffmpeg -i " + str(video_file_path) + " -r 1 " + str(path) + "image-%3d.jpeg")
```

This above line will create various frames and we go through the frames and finds the frame which has the first image. Then the face recognition is used to get the face encoding of the above returned frame and then it verifies with the encoding file provided in the project. Once the face is found it stores the output in the s3 bucket.

5. Running the application

In order to run the application just run the workload generator provided in the project, In order to see the output of the application just go and check contents of the output s3 bucket.