CSE 546 --— Project Report

Aditya Goyal (1225689049)

Anshul Lingarkar (1225118687)

Vibhor Agarwal (1225408366)

General requirements:

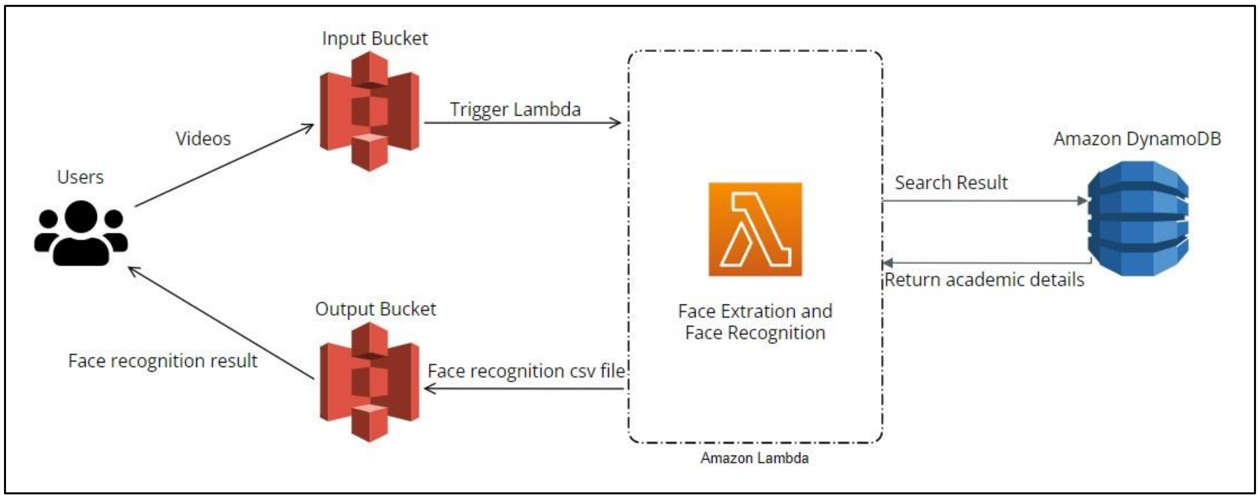
- Strictly follow this template in terms of both content and formatting.
- Submit it as part of your zip file on Canvas by the deadline.

1. Problem statement

Implement a smart classroom assistant for educators, using the Face recognition feature that will be run on AWS Lambda. This application will take videos from users, store them on an S3 bucket and process the video to extract frames, and run the face recognition module on the extracted frames using AWS Lambda, to return the details of the students like name, major, and year.

2. Design and implementation

2.1 Architecture



Architecture diagram for the implementation of smart classroom assistant for educators

We are using AWS Lambda to recognize an image from a video file uploaded by a user.

The user will upload a video file in S3 bucket. We are setting the Trigger point of Lambda to the input S3 bucket – "test-proj-cc-4". Once a file has been successfully uploaded to the input bucket, it triggers a lambda function.

The Lambda function has been created by using a containerized docker image. This docker image will then download the video uploaded by the user, process it to create frames, and then uses the first frame to perform image processing.

For identifying images, we are using the provided encoding file. This is used for comparing the image frame from the video. Once the image frame has been identified using the face recognition library, the image's name is sent to Dynamo DB. The Dynamo DB contains pre-loaded student data in the "student" table. We are matching the image processing result with Student's name, and returning the student's academic details like name, major, and year. These are stored as CSV file in the output bucket, and the output is returned to the users.

2.2 Autoscaling

Our application uses the features of PaaS and AWS Lambda to automatically Scale-out and in, based on the demand and load on the system.

The autoscaling part is managed by AWS, to support the demand in real time.

2.3 Member Tasks

Aditya Goyal

- Did the setup for AWS Lambda and Amazon Elastic Container Registry.
- Created the Docker image in Amazon ECR and created the AWS Lambda function using the Docker image from Amazon ECR.
- Worked on uploading the student_data.json file into the DynamoDB database.

Anshul Lingarkar

- Implemented the code to download the video uploaded by the user in the S3 bucket.
- Implemented the code to extract the first frame from the video and store it in /tmp/ folder on AWS Lambda.

Vibhor Agarwal

- Implemented the face_recognition_handler function to work on the frames extracted from the video uploaded in the S3 input bucket.
- Implemented the code to Fetch the student data from the Dynamo DB and create a CSV file for the result data and upload it to the S3 Output bucket.

3. Testing and evaluation

Initially, we tested the result of our application by uploading a single video file in the S3 bucket and we placed some custom logs to monitor the flow of the application. We have used the in-house CloudWatch application provided by AWS to watch the logs. We were successfully able to upload the video file which internally triggered AWS Lambda that performed image recognition on the first extracted frame from the video file. Once the image recognition result was received, we used it to perform a search in Dynamo DB for the match, and the output from DB was stored as an excel file in the S3 output bucket.

After getting the desired outputs, we used the provided workload generator, to upload the test video files to the Input S3 bucket and trigger the Lambda function once the video was available in the S3 bucket.

Number of Files Uploaded	Time taken to record the output (in Seconds)
1	10
8	90
10	100
100	412

Note: Major time was taken in uploading the file as the file sizes were different

4. Code

Functions present in the code:

1. open_encoding(filename)
   To open and store the encoding file provided to compare the result of face recognition

2. face_recognition_handler(event, context)
   The main function for running the face recognition module on AWS Lambda.
   It calls download_video_from_s3() function to get the video uploaded by the user from the input bucket.
   Then this function calls get_image_frame() function to extract the first frame of the video and runs the face recognition module of the frame stored in the /tmp/ folder on AWS Lambda.
   After receiving the Image processing result on the video frame, this function calls create_csv_file() function to store the result in CSV file.

3. download_video_from_s3(bucket,file_name,file_key)
   This function downloads the video file uploaded by the user in the S3 input bucket.

4. process_image(image_path)
   Runs the face_recognition module on the image frame from the video uploaded by the users.
   Compares the faces with the provided encodings.dat file and returns the result of the comparison.

5. get_image_frame(video_path,image_path)
   This function is used to extract the frame from the video available in the S3 bucket, and return the first frame of the video to run the face recognition module.

6. create_csv_file(name,major,year,video_name)
   Function to create the CSV file and add the student details – name, major, and year received after running the face recognition function.
   This function then uploads the CSV file to the output bucket – “demo-output-csv-1”

Steps to install and run the code:

1. Create a Docker Container with the encodings file and handler.py file. The main code of our application will be present in the handler.py file.
2. Build the Docker image using the following command:
   a. docker build -t test-cc-proj .
3. Use the command from AWS Console to login to the AWS Account and push the Docker image to the AWS Elastic Container Registry.
   a. aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 341967751550.dkr.ecr.us-east-1.amazonaws.com
   b. docker tag test-cc-proj:latest 341967751550.dkr.ecr.us-east-1.amazonaws.com/test-cc-proj:latest
   c. docker push 341967751550.dkr.ecr.us-east-1.amazonaws.com/test-cc-proj:latest
4. Create an input bucket with the name "test-proj-cc-4" and an output bucket with the name "demo-output-csv-1" in S3.
5. Create a DynamoDB table with the name – "student" and load the "student_data.json" file in this table.
6. Create AWS Lambda function using the docker image from ECR and set the trigger point to S3 input bucket – "test-proj-cc-4".
7. Increase the memory and timeout for AWS Lambda to 1024 MB and 1 minute respectively.
8. Now you can invoke the workload generator which will upload the video files to the S3 input bucket. Once the video is available in the S3 input bucket, the AWS Lambda function will be triggered.
9. AWS Lambda function will process the image, run the face-recognition module and store the student details in CSV file to the S3 output bucket.