# CSE 546 — Project 2 Report

*Ramachandra Sai Nayani(jnayani@asu.edu)*

*Disha Agarwal(dagarw24@asu.edu)*

*Sanket Duhoon(sduhoon@asu.edu)*

## 1. Problem statement

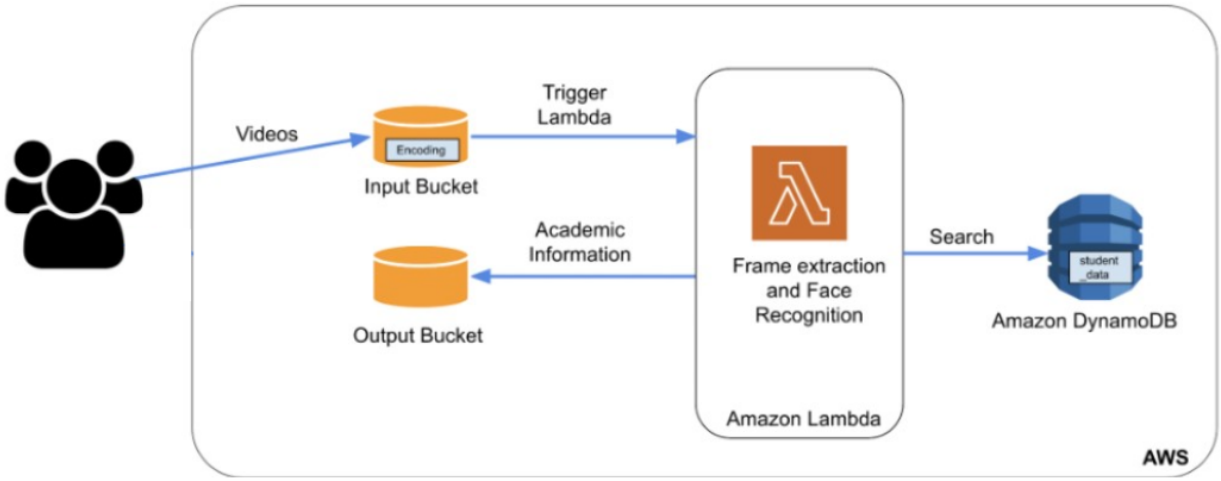
Nowadays, with widespread use of ML, and requirement of ML api's to scale cloud technologies become an essential skill. In this project a face recognition app that analyses a video stream and retrieves information from a database based on face recognition is created. The application needs to be elastic in nature and efficiently and automatically adjust its scale in response to demand while maintaining cost-effectiveness. AWS Lambda is used to scale in and scale out based on the traffic of the application. The information from the database DynamoDB based on the recognized output is returned to the user as the response.

This challenge holds significant importance for several key reasons. Firstly, it optimizes resource utilization, mitigating the waste of computing power and financial resources, as we are using lambda which is a serverless computing resource for short loads. Secondly, it aligns with cost-effective cloud computing practices, crucial for competitiveness and profitability in a pay-as-you-go environment. Moreover, it enhances the user experience by maintaining service reliability under fluctuating loads, ultimately increasing customer satisfaction.

## 2. Design and implementation

### 2.1 Architecture

The below architecture diagram shows all the major modules present in the app and the data flow in the system.



The Main Modules present in the app are:

- **Lambda Function:** This is serverless compute which gets called when a video object is uploaded to the s3 input bucket. It First segments the video into images using ffmpeg and uses the first image for face recognition using face recognition library and provided encoding file. Based on the results of the recognition it queries dynamodb and returns the name, major and year of the student in the video.
- **Lambda Trigger and Elastic Container Registry:** A Trigger is created on the input s3 bucket for upload. This trigger is called whenever we put an object in the s3 bucket and calls the lambda function. A container is built and uploaded into the Elastic container register. The required container is choosed to run the lambda function. All required access for other services like s3 and dynamoDB are provided to the lambda function.
- **DynamoDB:** Applications requiring quick and adaptable data storage can benefit from AWS's fully managed NoSQL database service, DynamoDB, it offers great scalability and low latency performance. The student information is stored in Amazon DynamoDB table, created using AWS CLI and console and it is queried everytime the face recognition function returns a valid student name from the video given as input.
- **Data Tier/S3:** This module is responsible for storing data permanently, the S3 storage service is used for this to store data as object in key,value form. For the videos in input-store bucket the video filename is used as key and value is the video file, this bucket is also used to trigger the lambda function, for results in output-store bucket the key is the video filenameand value is name,major,year in csv format.

**2.3 Member Tasks**

| **Team Member** | **Contribution** |
|---|---|
| Ramachandra Sai Nayani | ● Created the input buckets and lambda trigger with required access. Uploaded the docker image to the container registry.<br>● Wrote code to get and store the information for s3 buckets. |
| Disha Agarwal | ● Implemented the face recognition code and video splitting code in lambda function .<br>● The code takes the video filename from the event in lambda and stores it locally first, and then performs the processing on it. |

| | |
|---|---|
| Sanket Duhoon | ● Created the DynamoDB table though AWS console and used AWS CLI to upload the data into the table from file provided<br>● Implemented the function to query dynamoDB table to fetch the required information of identified student from the face recognition function |

## 3. Testing and evaluation

The application was tested in phases or units and also end to end testing was performed to ensure that application worked as expected, processing 100 requests in 7minutes as per the specified requirements.

Phase Testing/ Unit Testing:

Trigger and dummy lambda function: We created a dummy lambda function to test the trigger from s3 along with testing of access from s3 and dynamodb from the lambda function. We have uploaded some dummy code in the container to the registry to check whether the container is working properly or not.

Face Recognition: The Face recognition and video splitting code, as well as retrieving values from dynamodb and uploading results to s3 were tested on WSL locally at first and then integrated with the trigger and event lambda code.

End-to-End testing:

Using the provided workload generator, the lambda function was tested rigorously by running simultaneosuly, debugging and rerunning multiple times. The cloudwatch logs were seen in the monitor tab on lambda and were used to debug and correct code issues . Further, it was tested whether the system could process the 100 requests in 5 minutes or not.

## 4. Code

Lambda function consists of a single python file called handler. Following functions.

1. open_encoding: Given this reads the encoding file and stores known face encodings and names.
2. get_images: Given video file path it breaks the video into images , returning the path and name of the first image file generated.
3. face_recognition_1: Given image file and encoding data, it compares the encoding of the image to the stored encoding and returns the name of the person whose image it is based on this comparison.

4. face_regnition_handler: This is the main function which executes when lambda gets triggered, the event and context is passed here. We get the video filename from the event and download it from s3, store it temporarily, pass it to get images function, and then the received path to faceregnition function. Finally, it calls the dynamodb_util function to retrieve the major and year from the database.
5. dynamodb_util: Returns major, year of the person whose name is passed to the function by querying the details stored in a dynamoDB table

**Steps to execute the application:**

1. We run the workload.py script to test our test cases. This script inputs the s3 input bucket from there it triggers the lambda function and from the lambda function the outputs are stored in the output bucket.
2. We check the input buckets whether the videos are getting updated in the bucket or not. After a while we check the output bucket for results. Logs are created for debugging.