

# **CSE 546 Project II Report**

## **Smart Classroom Assistant as a Service**

**Oct 30, 2022**

### **Team Members:**

- Rajkumar Kakumanu (1225696602)
- Vinay Edupuganti (1225210948)
- Nikhilesh Kumar Reddy Anam (1225424551)

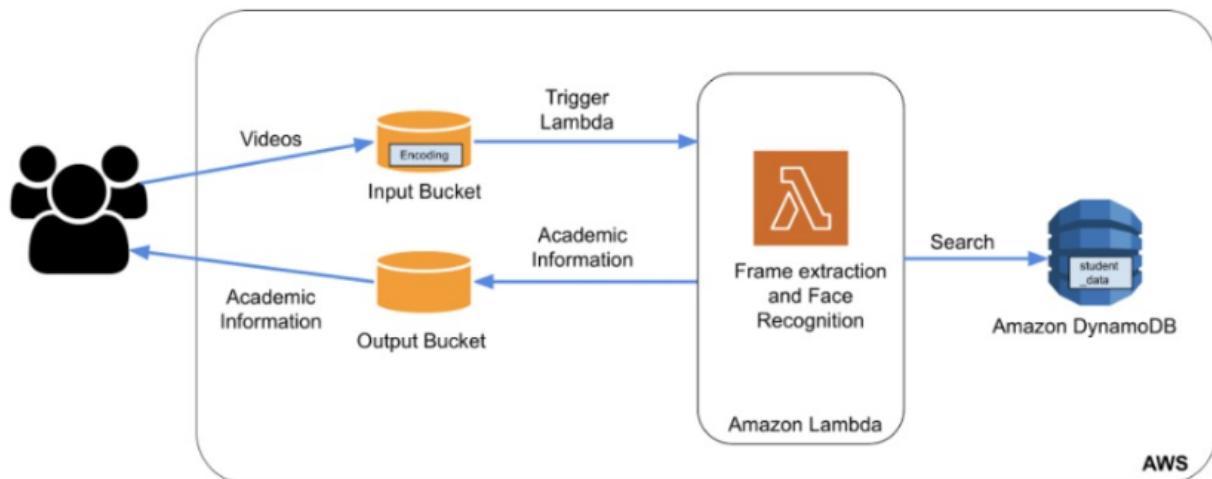
### **1. Problem statement**

The objective of this project is to create an elastic application that can use the Platform as a Service(PaaS) cloud resources to cost-effectively and autonomously scale out and in on demand. Specifically, AWS Lambda and other supporting services from AWS will be used to construct this application. Amazon Web Services provided the cloud resources used. This cloud-based application will provide teachers a smart classroom aide. This program collects videos from the user's classroom, runs face recognition on the footage, locates the recognized students in the database, and then gives the user access to each student's pertinent academic data. With a single server and limited resources, processing and returning results to the user may take longer. Using AWS (Platform as a Service)PaaS resources such as AWS Lambda the application should auto scale based on number of requests/hits, handle multiple concurrent requests, and return responses with low latency without missing out any requests. The desired characteristics of the application are **Low Latency, High Reliability, Auto Scaling**.

### **2. Design and implementation**

#### **2.1 Architecture**

As illustrated in the figure below, we have our application code in python (which uses the Python face\_recognition library to recognize faces from the frames) as an AWS Lambda function that is triggered on an upload of new video into the input S3 bucket. The Lambda function then downloads the video from the input S3 bucket into the '/tmp/' folder and performs face recognition on the first detected face of the uploaded video using python's face recognition library. The AWS Lambda automatically handles the algorithm that scales out the application based on the number of incoming requests. After the face recognition step, the function uses the name of the first recognized face to search the data preloaded to DynamoDB for this person's academic information. The Lambda function then saves a file containing the student's academic data in the output bucket of S3. The content is a CSV file with three fields: name, major, and year (for example, "president trump, physics, junior,"). The file name is set to represent the name of the video (for example, "test\_0"). After processing the input videos from S3 bucket, AWS Lambda automatically handles the scale in algorithm efficiently.



**Figure 1.1 Architecture**

## 2.2 Autoscaling

In Cloud computing, Autoscaling is a method that dynamically adjusts the number of computational resources based on the number of incoming requests.

*Automatic scale-up:* It is a method of adding more instances to the group. AWS Lambda automatically handles the scale out algorithm efficiently as described below in the implementation.

**Implementation:** When Lambda function is called for the first time, AWS creates an instance of it and executes its handler method to handle the event. When the function responds, it remains active while it awaits further events to handle. When the function is called again while the first event is still being handled, Lambda creates a new instance and handles both events at once. Lambda routes new events to available instances as they arrive and generates additional instances as necessary.

Lambda function's cumulative concurrency in a Region can initially reach a level of between 500 and 3000, which varies by Region, for an initial surge of traffic. Lambda function's concurrency can increase by 500 extra instances each minute after the initial burst. This keeps going until there are sufficient instances to handle every request or a concurrency limit is reached.

*Automatic scale-in:* It is a method of removing instances from the group. AWS lambda automatically handles the scale in algorithm efficiently once the number of incoming requests decreases.

**Implementation:** Lambda disables idle instances when the number of requests declines to free up scalability capacity for other tasks resulting in automatic scale in.

## 2.3 Member Tasks

Initially we discussed and analyzed the problem statement collectively and went through the AWS documentation for PaaS resources and came out with a design. Later we divided the implementation into three parts (Image recognition, Data retrieval and storing , Configuring AWS and Testing) and handled them.

**Vinya Edupuganti:** Developed the Image recognition functionality in python which included the following tasks.

- Downloading the input video from the S3 input bucket..
- Generating the frames for the video using ‘ffmpeg’ library.
- Recognizing the first face in the video from the frames with the help of python’s ‘face\_recognition’ library.

**Nikhilesh kumar reddy Anam:** Developed the data retrieval and storing functionality which included the following tasks.

- Retrieving the student details of the recognized person from the AWS DynamoDB.
- Saving the details in a csv file.
- Uploading the csv file into S3 output bucket.

**Rajkumar Kakumanu:** Tested thoroughly the whole application and configured the following in the AWS portal.

- Created a User group in the IAM dashboard and added all the team members as users in the group and provide **AdministratorAccess** to all the users.
- Generated Access key credentials for all the users to access the AWS services through the application.
- Created S3 Input,Output buckets and a table containing the student details in the DynamoDB .
- Deployed the docker image to Elastic Container Registry and created the lambda function using the uploaded docker image.

### 3. Testing and Evaluation

- Tested with the multithreaded workload generator given by the TA with 100 videos which took < seven minutes to complete.



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
Uploading to input bucket.. name: test_57.mp4
Uploading to input bucket.. name: test_58.mp4
Uploading to input bucket.. name: test_59.mp4
Uploading to input bucket.. name: test_60.mp4
Uploading to input bucket.. name: test_61.mp4
Uploading to input bucket.. name: test_62.mp4
Uploading to input bucket.. name: test_63.mp4
Uploading to input bucket.. name: test_64.mp4
Uploading to input bucket.. name: test_65.mp4
Uploading to input bucket.. name: test_66.mp4
Uploading to input bucket.. name: test_67.mp4
Uploading to input bucket.. name: test_68.mp4
Uploading to input bucket.. name: test_69.mp4
Uploading to input bucket.. name: test_70.mp4
Uploading to input bucket.. name: test_71.mp4
Uploading to input bucket.. name: test_72.mp4
Uploading to input bucket.. name: test_73.mp4
Uploading to input bucket.. name: test_74.mp4
Uploading to input bucket.. name: test_75.mp4
Uploading to input bucket.. name: test_76.mp4
Uploading to input bucket.. name: test_77.mp4
Uploading to input bucket.. name: test_78.mp4
Uploading to input bucket.. name: test_79.mp4
Uploading to input bucket.. name: test_80.mp4
Uploading to input bucket.. name: test_81.mp4
Uploading to input bucket.. name: test_82.mp4
Uploading to input bucket.. name: test_83.mp4
Uploading to input bucket.. name: test_84.mp4
Uploading to input bucket.. name: test_85.mp4
Uploading to input bucket.. name: test_86.mp4
Uploading to input bucket.. name: test_87.mp4
Uploading to input bucket.. name: test_88.mp4
Uploading to input bucket.. name: test_89.mp4
Uploading to input bucket.. name: test_90.mp4
Uploading to input bucket.. name: test_91.mp4
Uploading to input bucket.. name: test_92.mp4
Uploading to input bucket.. name: test_93.mp4
  
```

- Verified the S3 Input and Output Buckets.

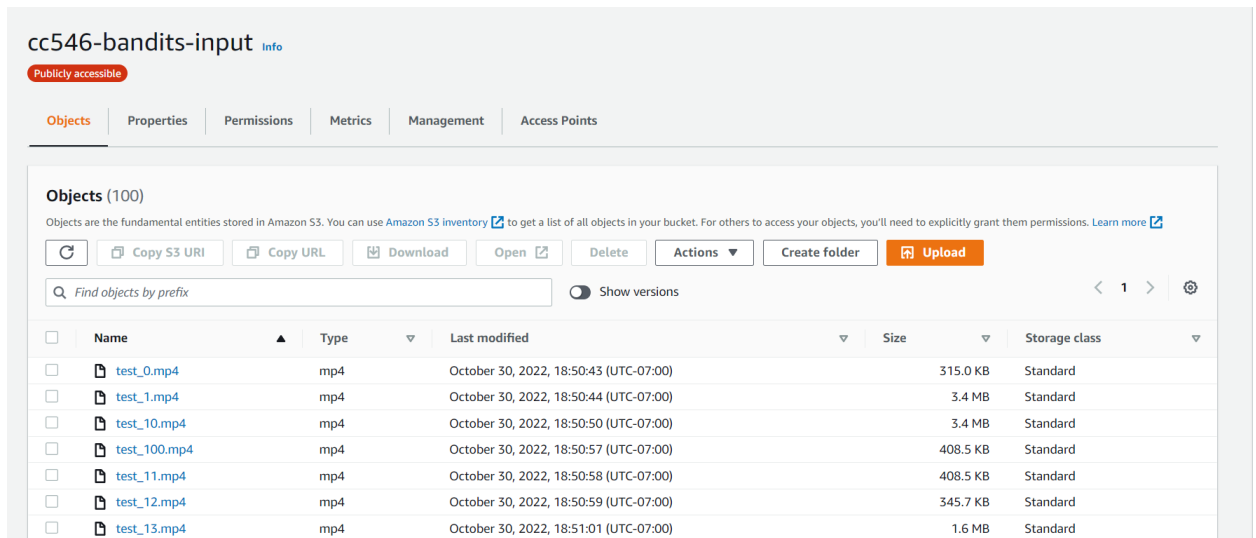


Figure 1.6 S3 Input Bucket

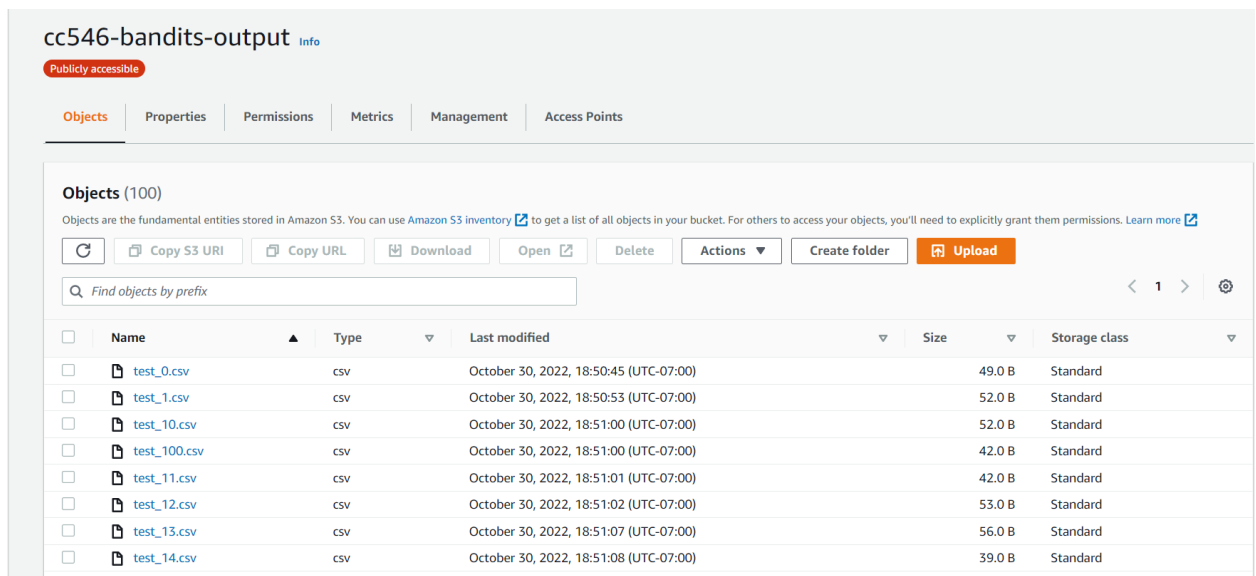
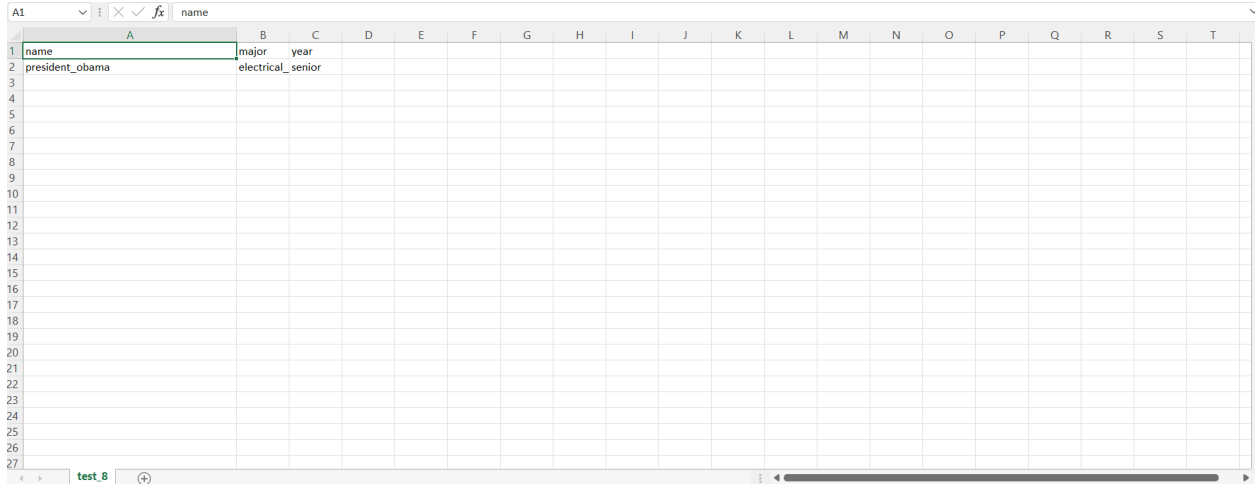


Figure 1.7 S3 Output Bucket

- Verified the logs in AWS CloudWatch.
- Verified the generated CSV files in output S3 bucket for the accuracy of the retrieved information.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	name	major	year																	
2	president_obama	electrical	senior																	
3																				
4																				
5																				
6																				
7																				
8																				
9																				
10																				
11																				
12																				
13																				
14																				
15																				
16																				
17																				
18																				
19																				
20																				
21																				
22																				
23																				
24																				
25																				
26																				
27																				

#### 4. Code

The handler.py file contains the lambda function (face\_recognition\_handler). When the function is called in response to an S3 upload event, we download the uploaded video and save it to the '/tmp/' folder. We generate frames from the video using the ffmpeg library and save them in the '/tmp/' folder before deleting the uploaded video. Then, using the 'face\_recognition' library, we can recognize the face in the frames. We pass this recognized face name to the 'get\_items\_from\_dynamo' function, which returns student details from AWS DynamoDB, and then call the 'write to csv and upload to s3' function, which stores the details in a csv file and uploads the csv into the output.

##### Function details:

**1. 'open\_encoding':** It takes an input encoded file as an argument and returns the encoded data in the file.

**2. 'get\_data\_from\_dynamo':** It takes three arguments (DynamoDB resource, Table name, Recognized name). This function connects to the DynamoDB and gets the relevant information of the recognized face in the table.

**3. 'write\_to\_csv\_and\_upload\_to\_s3':** It takes four arguments (csv file name, student details, field names, output bucket name). It stores the student details in a csv file and uploads the csv file into the output s3 bucket.

##### Installation :

Step 1: Deploy the submitted dockerfile in the Elastic Container Registry using the following push commands.

- docker build -t dockerfile .
- docker tag dockerfile:latest 614453164629.dkr.ecr.us-east-1.amazonaws.com/dockerfile:latest
- docker push 614453164629.dkr.ecr.us-east-1.amazonaws.com/dockerfile:latest

Step 2: Create a lambda function using the deployed docker image.

Step 3: Create a S3 trigger to the lambda function.

Step 4: Create a student details table (student\_data) in the Amazon DynamoDB.

Step 4: Upload a video in the S3 input bucket and check the output S3 output bucket.