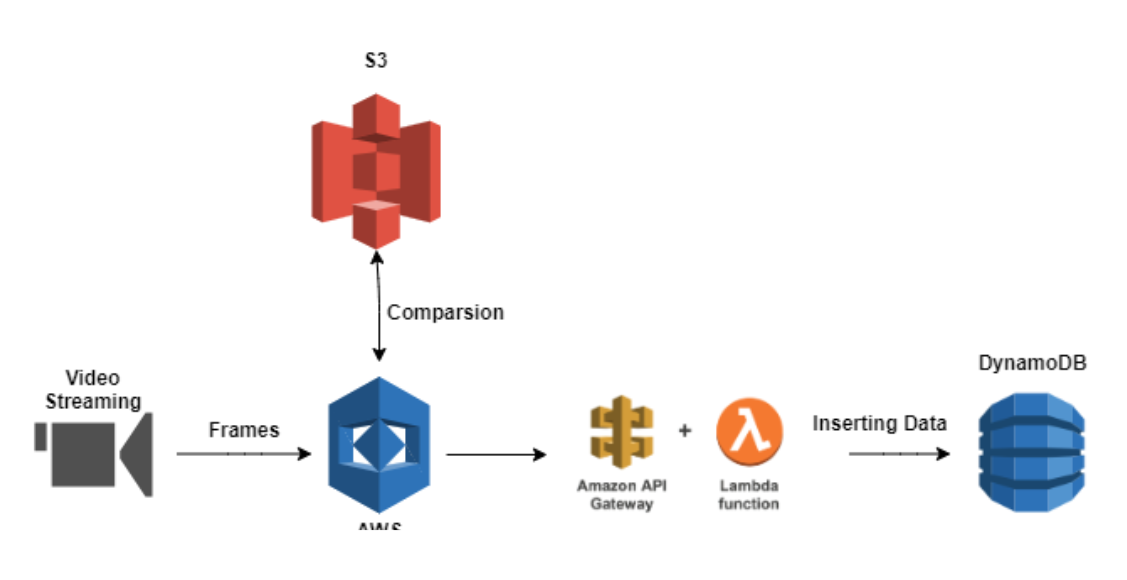
**AI-Powered Hourly Attendance Capturing System**

**Overview and Architecture of the Project**

This project focuses on creating an automated system that takes the attendance of students on hourly basis using preinstalled cameras in the classes. We make use of AWS services to marks the attendance, store the attendance in DB.

The architecture is given below:



From the above architecture, its components are:

1. **S3 Bucket** : It is the storage service of AWS, where image dataset will be stored.
2. **AWS Rekognition** : It provides the deep learning service, which would used to detect the faces in images stored in S3
3. **API Gateway** : It provides the REST API service, which would be used with Lambda function to connect to Dynamo DB, so that attendance is inserted to DB or attendance count can be read from DB. API Gateway creates a special token so that client can send attendace details as input to lambda function, which connects to DB to store the values of attendance details.
4. **DynamoDB** : It provide no-SQL service to store the name and period of attendace in a table.

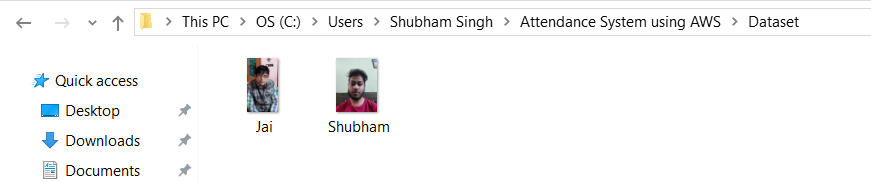
**Project Overflow or Steps**

1. Store the images of the students in S3 Bucket
2. Create the collection of students and add the faces of each student after detecting using AWS Rekognition into that collection
3. Capture the image of students from client web cam
4. The faces detected by openCV on client side will be matched against each face ID in collection using Face comparison algorithm
5. If the face is matched, then using API Gateway, attendance of the period and the name of student is stored in DynamoDB. This insertion is done by using lambda function by connecting to DynamoDB. This way attendance is marked and stored
6. Dashboard is created using Flask in python, that reads the attendance using different API Gateway from the DynamoDB for each student

**Data Collection**

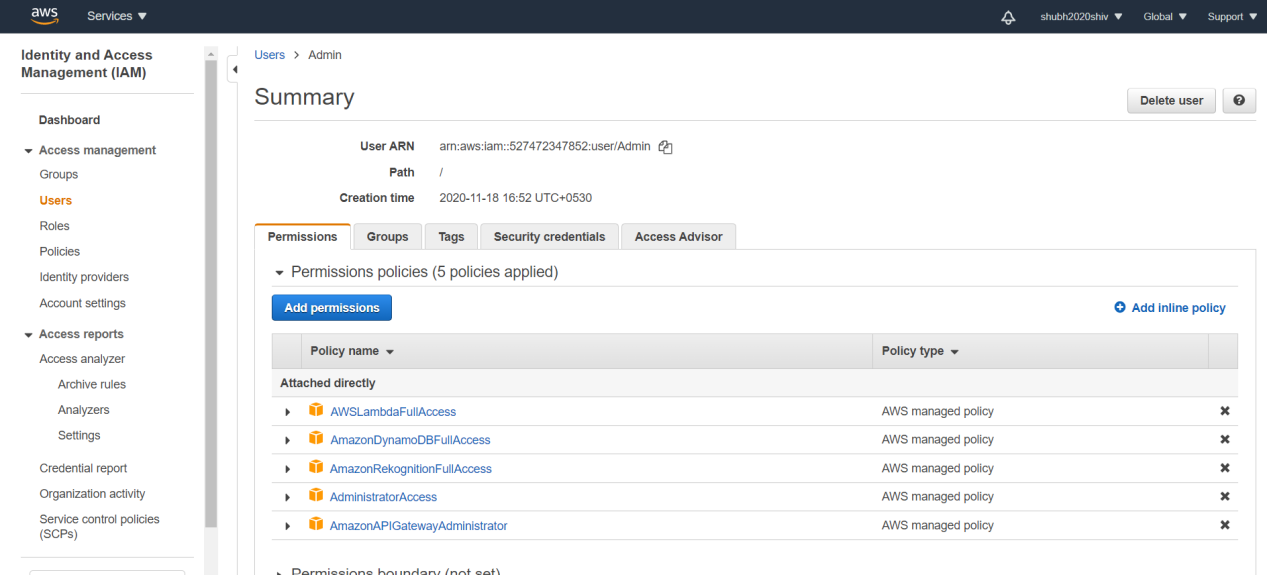
Suppose, there is a classroom having just two students : 1. Shubham and 2. Jai

I have collected each image of these students.



**Configuring the AWS Account**

I created an Admin user in Identity and Access Management and provided permissions for each components mentioned above:

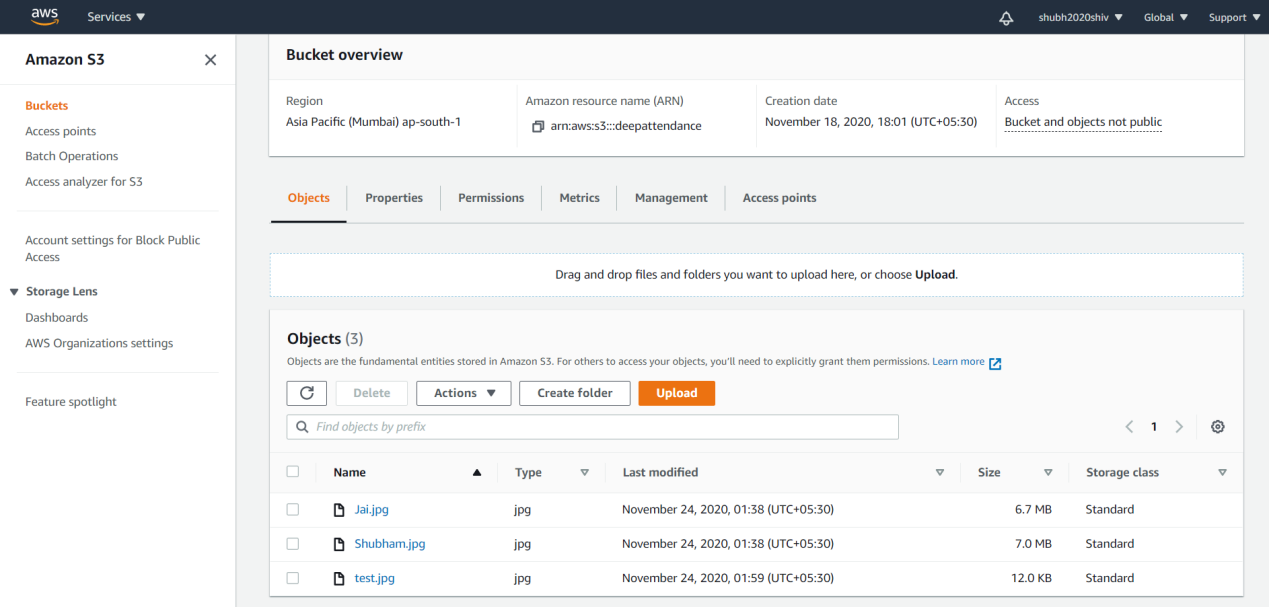


Steps to create IAM can be accessed from <https://serverless-stack.com/chapters/create-an-iam-user.html> for reference

**Uploading Images to Amazon S3 Bucket**

S3 bucket the storage service, where we will be storing the images.

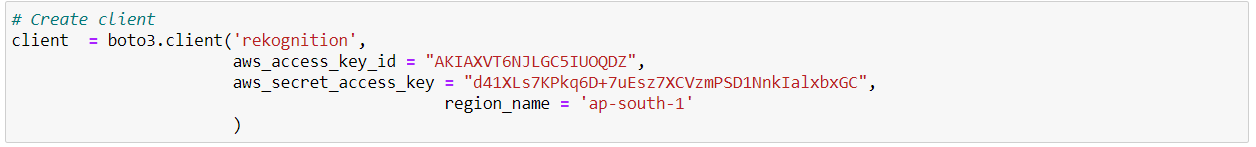
I created a bucket called "deepattendance" and uploaded the images of students: Jai and Shubham having face images in Jai.jpg and Shubham.jpg respectively



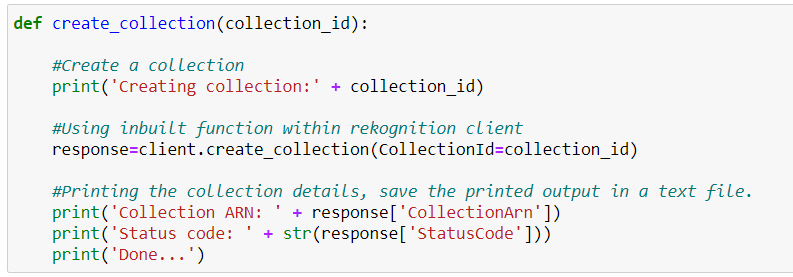
**Create the Collection**

Amazon Rekognition provides the functionality for detecting the faces in any image. This coordinates of the faces detected is kept on server-side as index in containers called “Collection”. The objective of creating collection is that faces in each frame of the streaming video will be matched with face index, to recognize the name of the person whose face is matched.

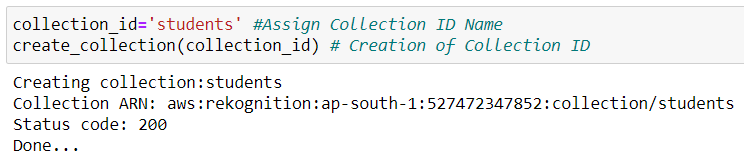
For connecting python with AWS, we use boto3 library, from which we can create a client to interact with AWS. Below script shows client being created in python.



Using this client, we will be defining a function, “create\_collection”, which also takes the collection ID to define the name of the collection.



Running this function will return the output as:



**Adding faces in created Collection using AWS Rekognition**

Given the collection, with name : “students” created, we will have to detect the faces we stored in Amazon S3 bucket and index the images.

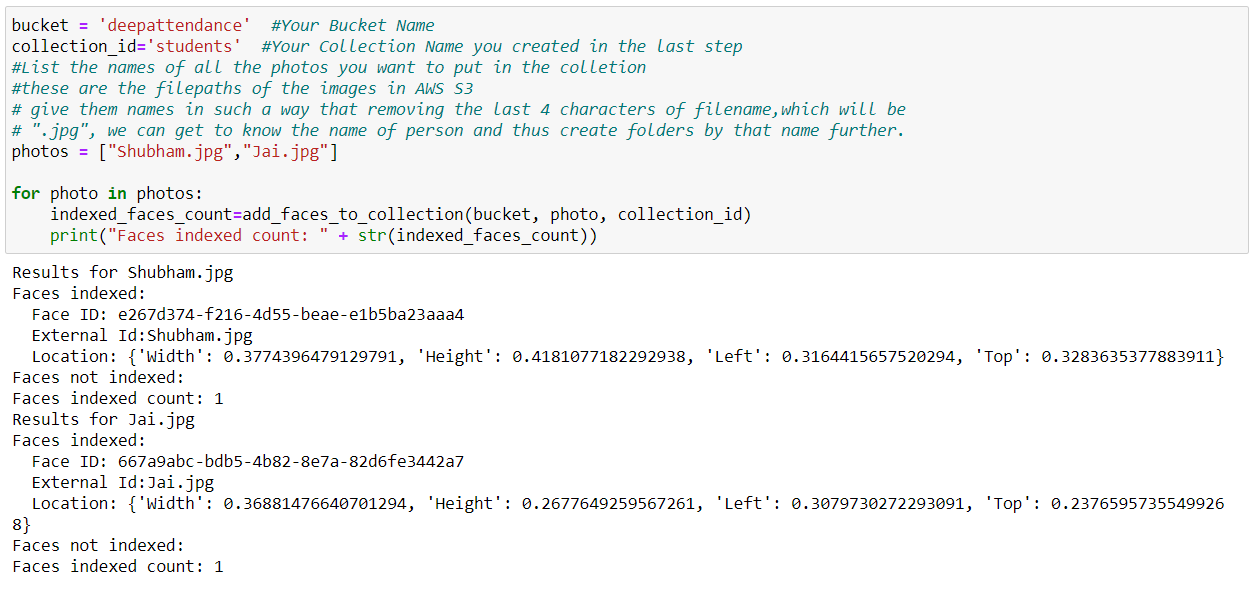
We have defined a function “add\_faces\_to\_collection”,



This function needs four inputs: client, bucket name, photo and collection\_id.

Client, for connecting to AWS Rekognition; Bucket name, to specify which bucket has the list of student images stored;Photo, to specify which image to index; Collection\_id, to contain the indexed faces of the images.

Running the “add\_faces\_to\_collection”function will produce below results:



In above output,we can see that each photo is given a unique ID and location of faces in those photos.

Now, after the faces detected and given an ID, we will store this information in dictionary format. Hence, this step is called listing faces in collection.

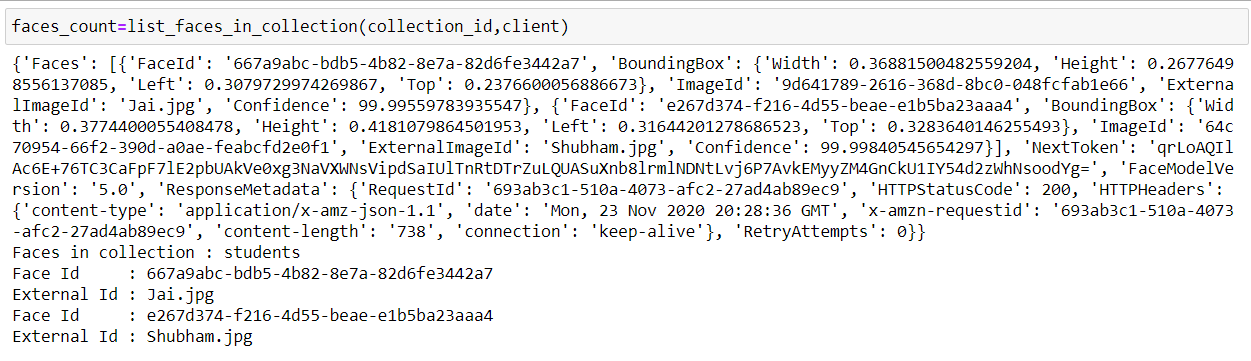
The listing faces from the collection describe the face properties such as the bounding box, face ID, image ID of the input image, and external image ID that you assigned

Therefore, we define the function “list\_faces\_in collection”as below:



The function will return the total number of faces in collection.

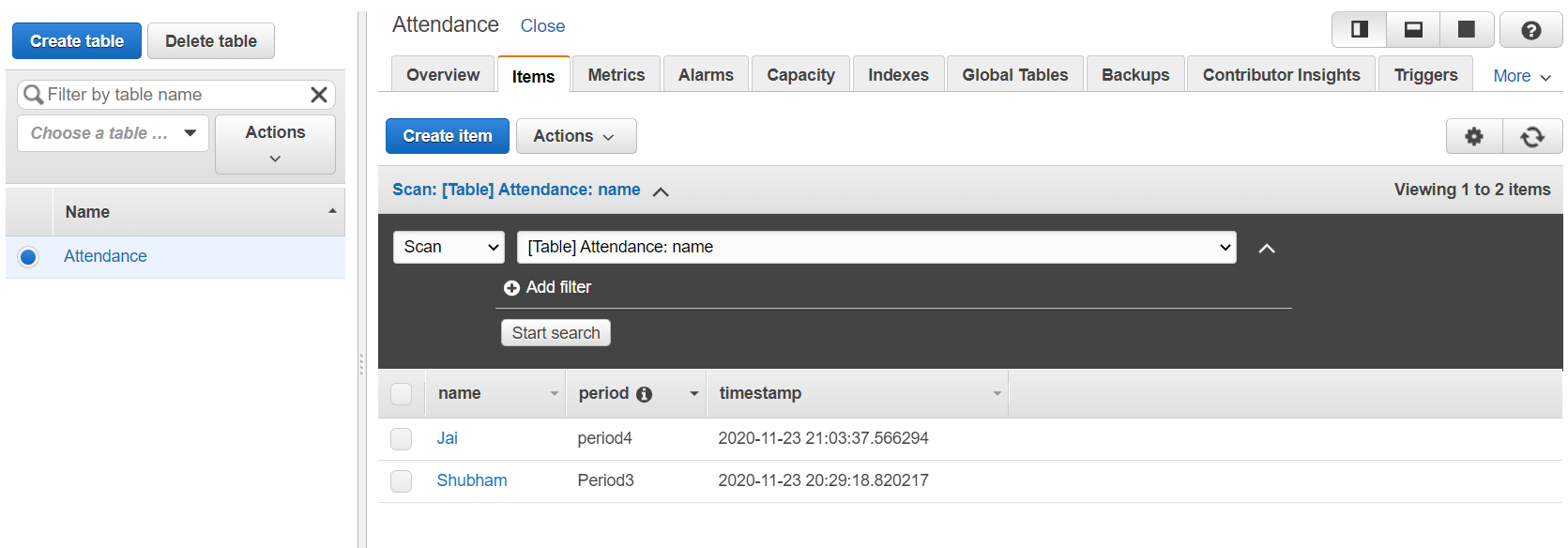
Running above function will provide below output:



**Creating Table in DynamoDB**

DynamoDB is no-SQL service, where we will be storing the Names and Period for which Student is present.

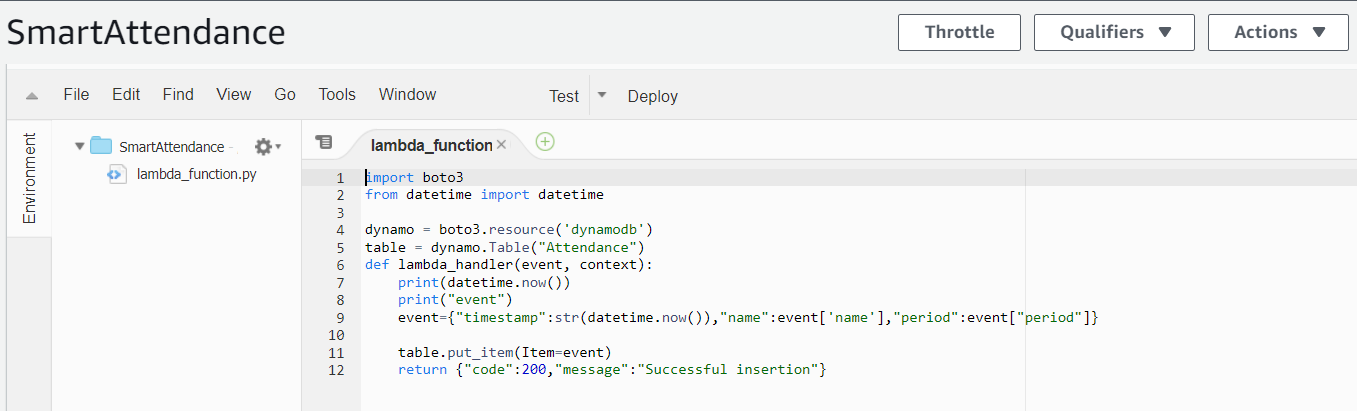
For this purpose, I had created a table called “Attendance” in DynamoDB as follows:



**Creating Lambda Function**

We will define a function, which will run on AWS Lambda, hence called Lambda function, that will keep on receiving the requests whenever the face Matches and make an entry in DynamoDB against the person whose face is matched.

I have defined a Lambda function called “SmartAttendance”as given below to make the entry into DynamoDB.

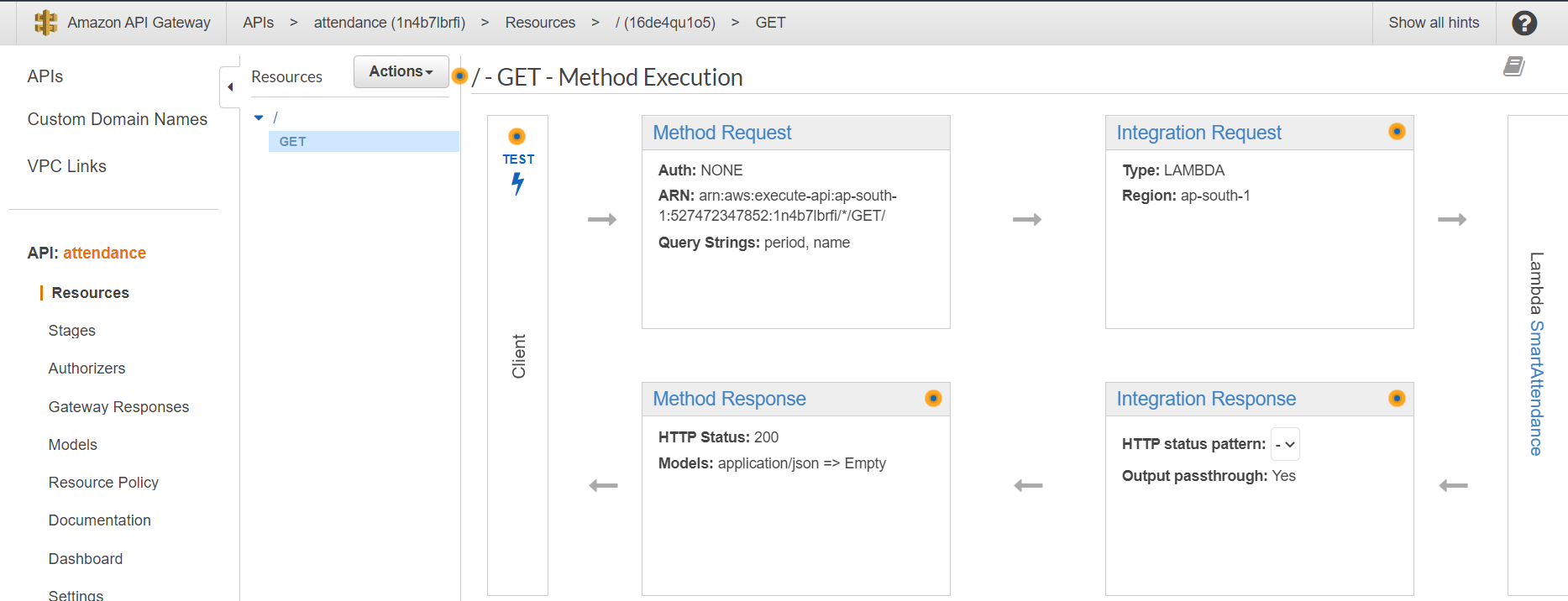
Here, we can see, DynamoDB name “Attendance’, is called to create new records using Name and Period of Attendance.

**Creating API Gateway**

API Gateway service is needed to assist in sending the Name and Period attribute from the client, a system where camera is running. After matching the faces in images captured from live camera or webcam, the names of the matched face of the student and attendance of the corresponding period will be sent via this API and will be given as input to the Lambda function that we created earlier. After that, this lambda function will insert the Name of the student and period of attendance in DynamoDB.

Hence, we will be created this step by step procedure using API Gateway as below:

I had created an API Gateway named as “attendance”:

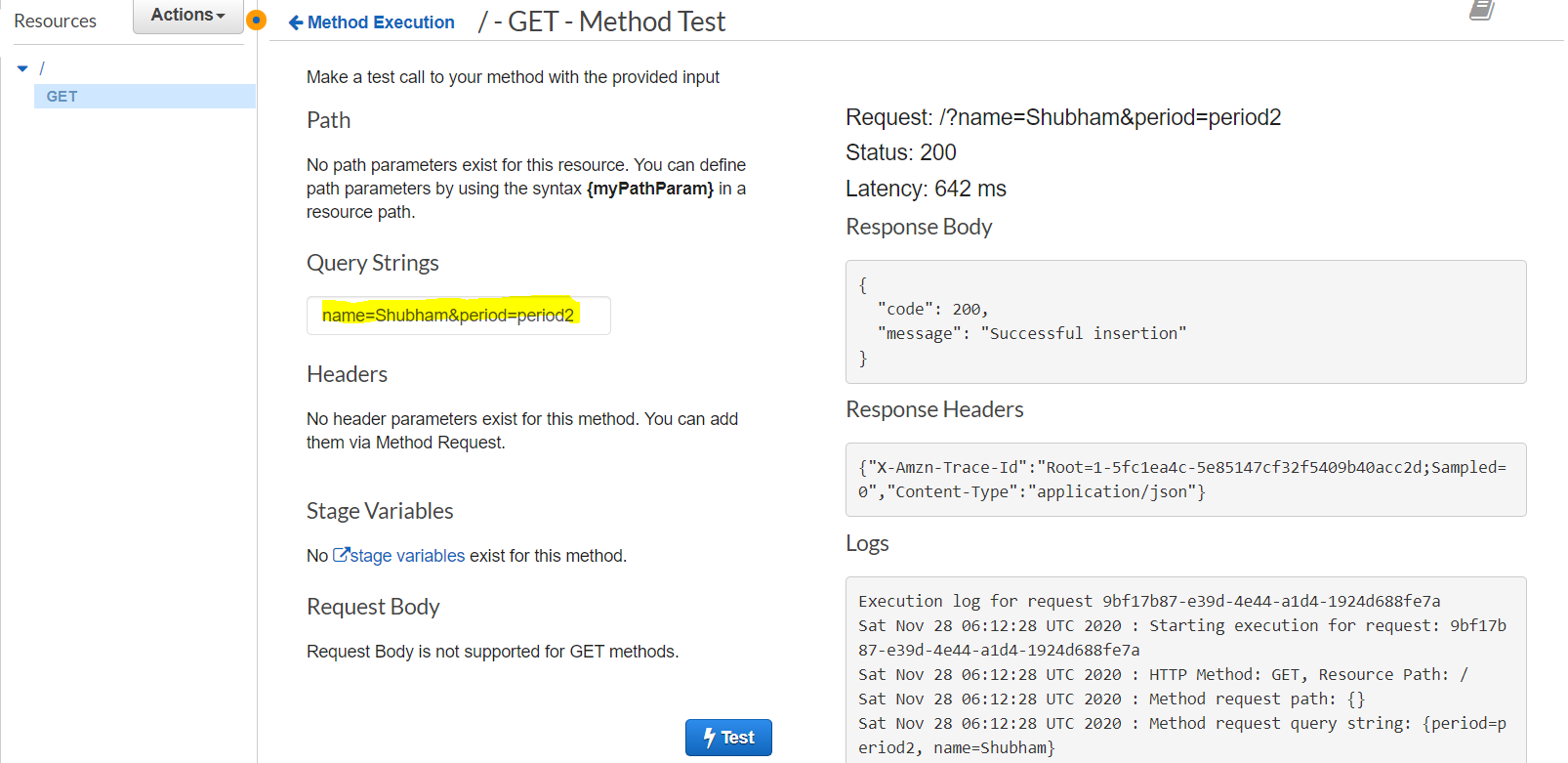


Here we can see that, client will send the Name of the student and period of attendance, which is expected to be received as parameters in “Method Request”. These parameters will be send to “Integration Request”, which is Lambda type request, which would send those parameters to Lambda function “SmartAttendance”that we created earlier. After the successful insertion, it will be sending the response according, which would specify if the insertion is success or not.

We can verify if the API “attendance” works manually by clicking on Test button on left side of the above process. This test basically, emulates the activity on client side by manually giving the parameters expected by this API.

That’s the reason, test button is on Client side as we will be giving the parameter manually to test manually of the insertion occurs successfully in DynamoDB using Lambda function “SmartAttendance”.

The parameters are also given in format expected by this API.



We see that after giving the parameters of Name and Period is proper format under “Query Strings”, we get the response status as success.

**Inserting the attendance in DynamoDB**

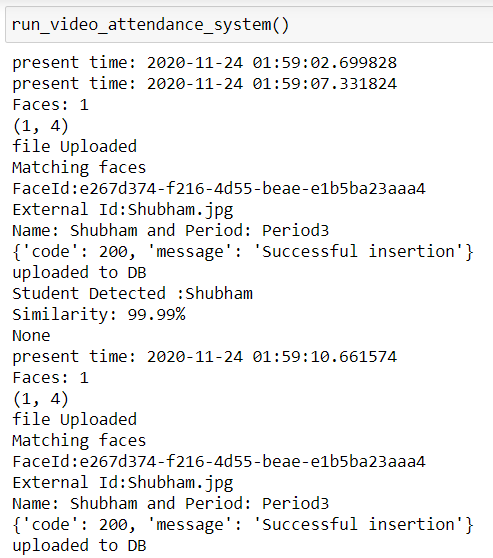
Now, our API, DynamoDB table and lambda function is created, we are ready to write function from Client side in python.



The frames will be captured as file name ‘test.jpg’and will be uploaded to S3 bucket and faces will be recognized from using AWS Rekognition in ‘test.jpg’. The AWS Rekognition will also assist in searching the faces already in collection with faces found in ‘test.jpg’. With every face that is matched from ‘test.jpg’, the Name of matched student face and Period will be inserted in DynamoDB by hitting on API URL, that we created earlier. The Period of attendance is determined from the system time running on client side.Also, the name of the student is determined from indexed images of student faces already there in collection.

**Running the Video Attendance System**

After running the whole setup in python, we get the output as below:



As it be seen from the output of whole Video Attendance system, it

First uploads the image from webcam present on client side into S3

Bucket as ‘test.jpg’ file and it tries to match the faces already indexed in collection and returning the Face ID: e267d374-f216-4d55-beae-e1b5ba23aaa4, which is the student name : Shubham. It also returns the similarity measure in percentage with which the face matched after extracting the student faces from ‘test.jpg’. It then tries to hit the API URL : https://1n4b7lbrfi.execute-api.ap-south-1.amazonaws.com/attendance\_input?name="+name+"&period="+period, to send the Name and period of attendance as parameter to Lambda function “SmartAttendance”, which will insert the names of student and period in DynamoDB and will return the response if the insertion is successful.

**Creating the Dashboard to show attendance using Flask**

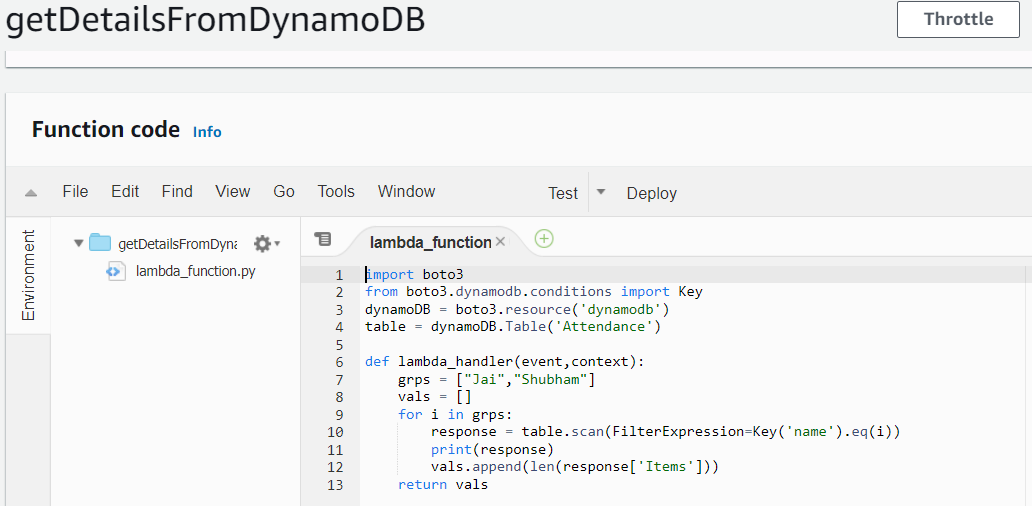
We have created python application in Flask ‘app.py’, which would

populate the total attendance of the student on browser.

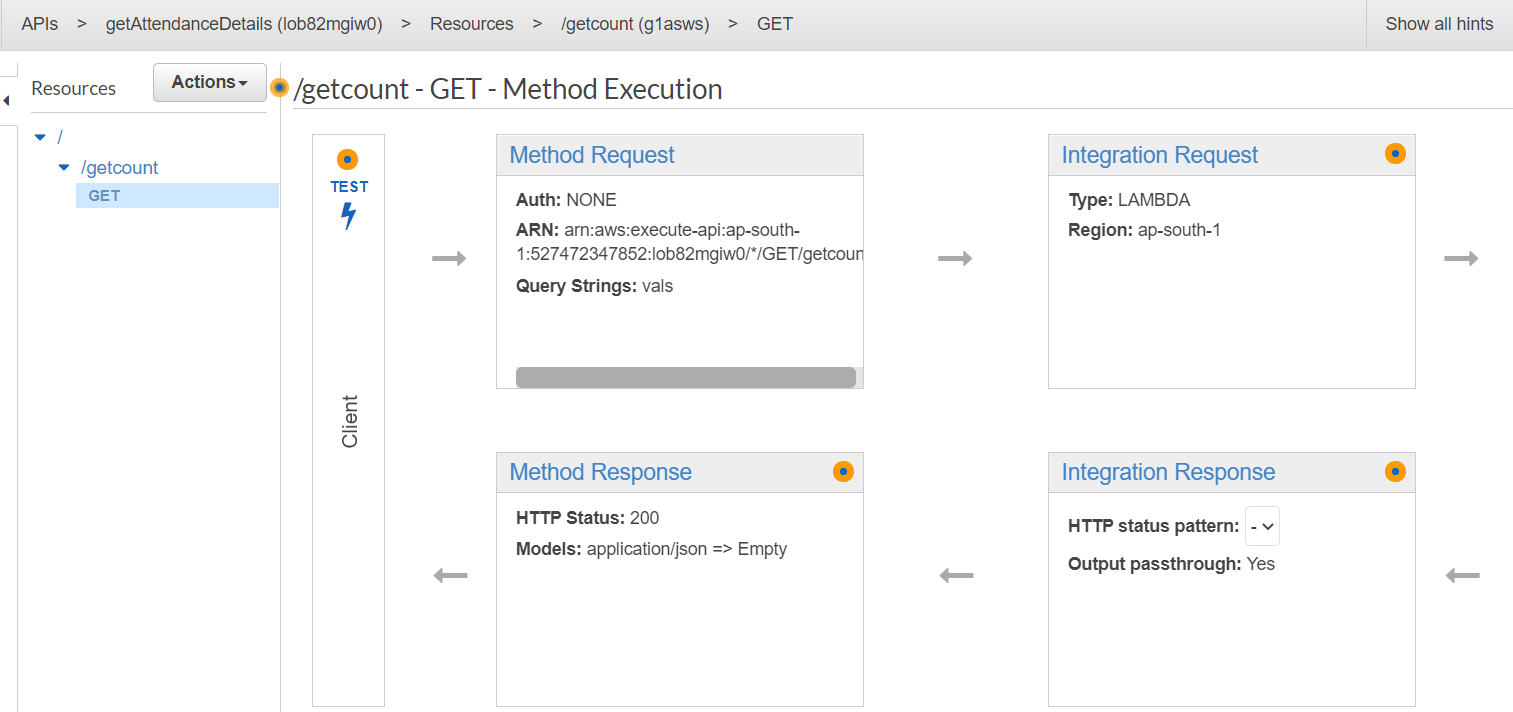


Again, in order to get the total attendance count of a student from DynamoDB, we need to create a lambda function to fetch the record from the DynamoDB table and an API to send the count of attendance on client machine, which could be seen as a dashboard on browser.

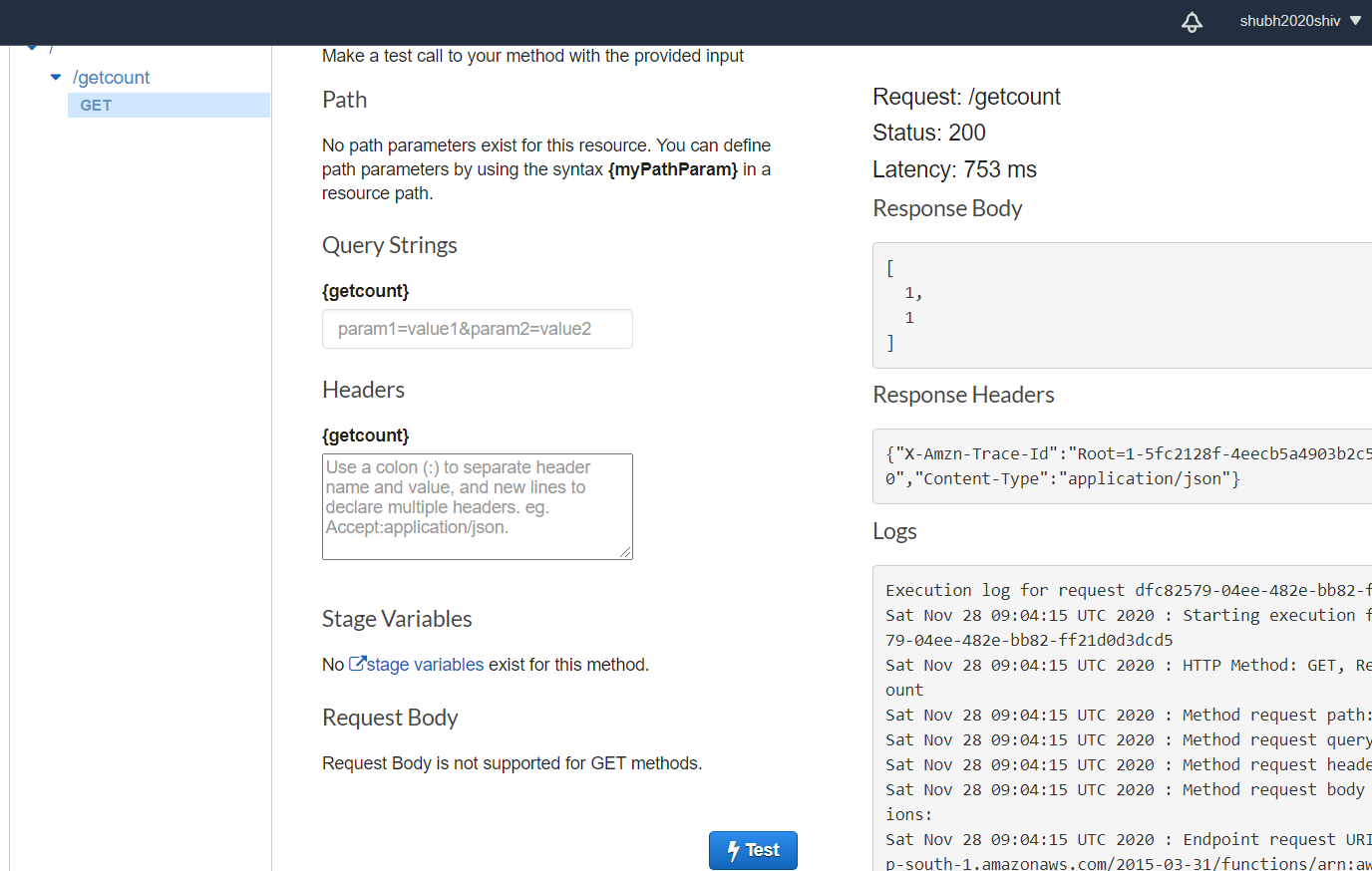
We created another Lambda function called ‘getDetailsFromDynamoDB’, to send the Name of the students from DynamoDB table.From the below function, it is returning the count of each name in DynamoDB table. This value is returned via API. That’s why a separate API must be created to send this response back to client browser to be seen as dashboard



We have created API called ‘getAttendanceDetails’, which would send the response back to client as count of attendance.



We can test the working of this API as below:

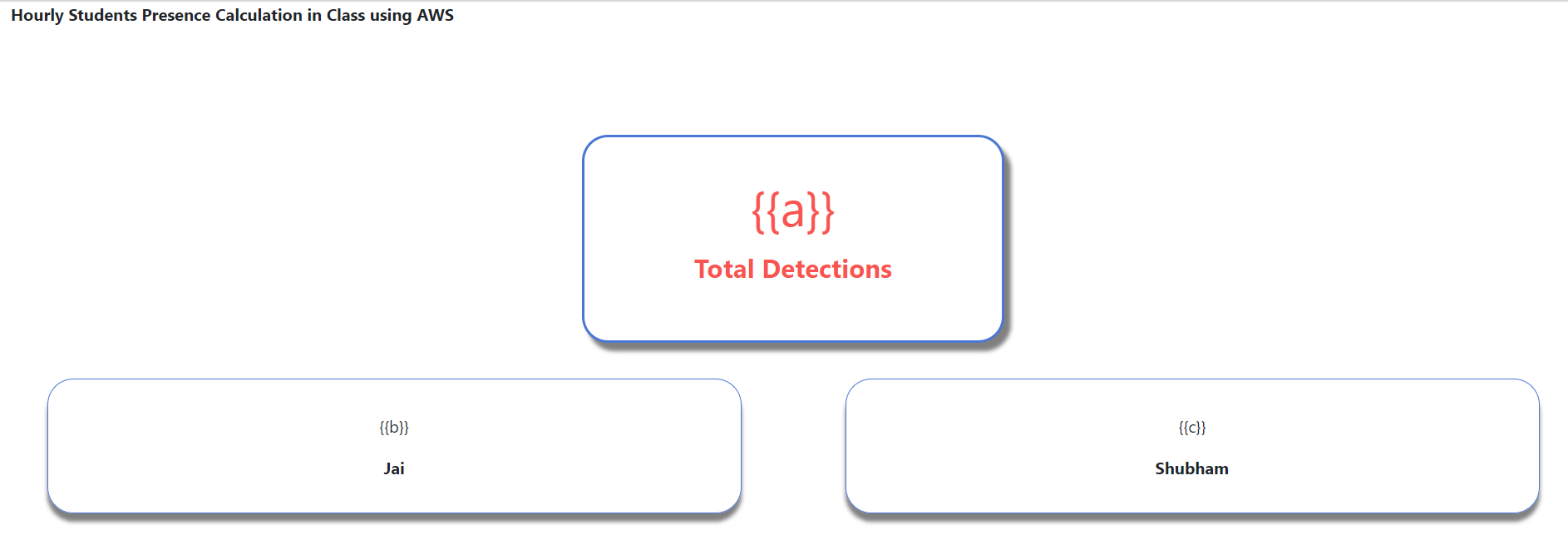


The API is returning the list containing the count of students attendance for each student. We can use this list to identify the count of attendance of each student as we specified in Lambda function: ‘getDetailsFromDynamoDB’.

**Writing the template of Dashboard on browser in HTML**



The template in browser can be as below:

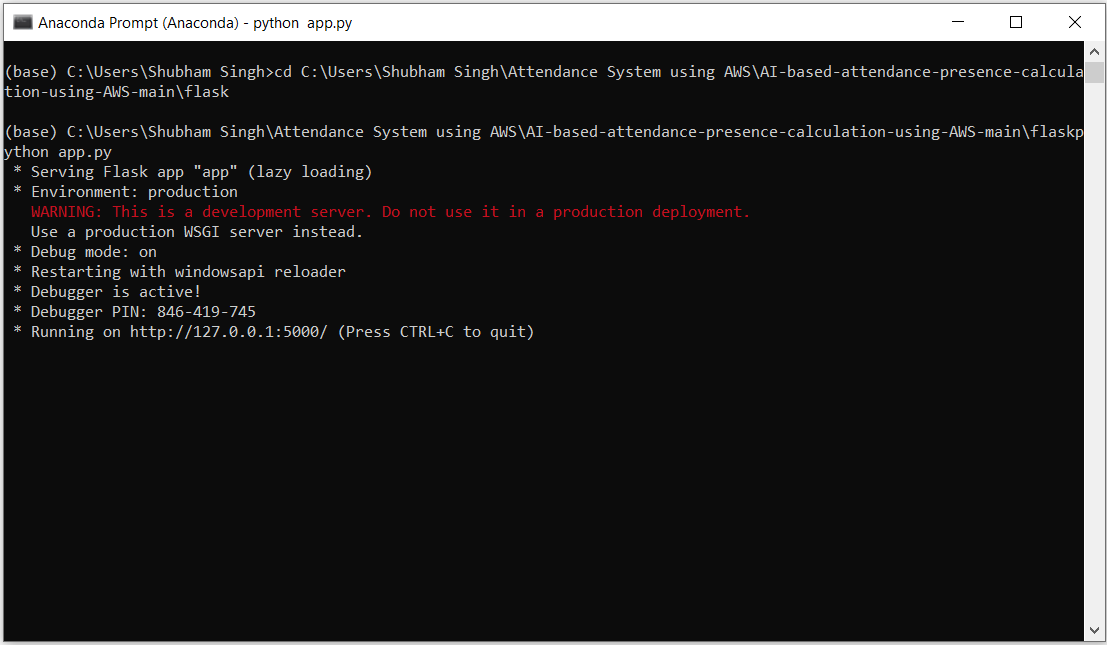


The value in bracket, for example: {{a}}, is variable which will receive its values from the API after getting count of each student’s attendance count.

**Running the Flask application**

We will be running the flask application written in ‘app.py’ using Anaconda command prompt.

It will show the output as below:



**Final Output**

The final output will look as below on browser Dashboard.

It shows total attendance and attendance against each student names.

