CSE 546 — Project Report

Samarth Patel (1220111230)

Shubham Bansal (1224102750)

Sulabh Soneji (1219617139)

1. Problem statement

This project focuses on developing an elastic cloud application which will be cost-effectively using PaaS cloud. The PaaS cloud will be using two services of Amazon AWS namely Lambda, S3 for storarge and DyanmoDB for working with data. The application will take video from the user's classroom camera and it will perform face recognition on the first frame of the fetched video. The Lambda service which provides serverless function-based computing, will look up into the first recognized student in the DynamoDB database. And finally, the lambda function will store the students' academic information into another S3 bucket. The lambda function will be responsible to trigger the face recognition algorithm when there's a video uploaded to the input S3 bucket. For this project we will be using a customized docker container image file which is preinstalled with the necessary libraries. This project aims to understand AWS services like DynamoDB, S3 Storage, Serverless Lambda Computing and developing other skills like the face recognition module, working with Docker and putting it together in a cost-effective way.

2. Design and implementation

2.1 Architecture

AWS Services:

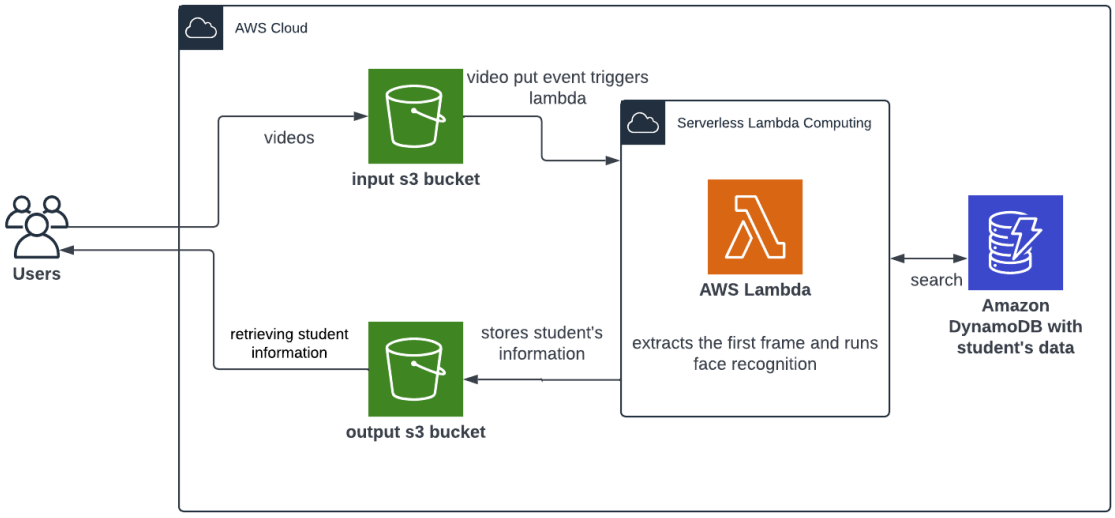
- **AWS S3 bucket:**
    - S3 buckets are used for storing the input videos and results of the face recognition corresponding to the first frame of the video as csv in the output bucket.The users will upload the video of the classroom in the input bucket, and the output will be stored in another S3 bucket. The output format would be a csv file, which will contain three fields name, major, and year as per the requirements.

- **AWS DynamoDB database:**
    - DynamoDB is used to store the information of the students. We have created a table with four columns which is best suited for our use case: student id, student

name, major, and year. This DynamoDB table will be queried when the lambda function extracts the name of the recognized person in the video and then the lambda function uses the corresponding name of the person to extract the student's information from Amazon DynamoDB table named 'students'.

- **AWS Lambda Function:**
  - Lambda function will be responsible for extracting the first frame from the video, running the face recognition algorithm on that frame of the video, and querying the database for the information of the student. The lambda function will be triggered with the put event on the input S3 bucket. The code for the lambda function resides in the handler.py file and the whole module is being provided to lambda as Docker container image.

Architecture Diagram:



Architecture Description:

As seen from the diagram, the architecture consists of two S3 buckets, a lambda function for serverless computing, and a DynamoDB table for holding students' data. The end user performs a put operation including a video to the input S3 bucket. When the video is available on the input bucket, the lambda function will be triggered and performs various activities.

The activities done by the lambda function will be as below which happens in a sequential manner,

1. Get the video from the input bucket,
2. Stores it in a temporary folder (/tmp)
3. Extracts the first frame from the video using ffmpeg library
4. Runs facial recognition algorithm on the extracted frame
5. Recognizes the face and encodes it
6. Loads the ‘encoding’ file and uses it to get the name of the corresponding recognized face in step 5
7. Searches for the recognized student name in the dynamoDB table
8. Puts the information about the student in the csv file, and uploads it to output S3 bucket

The Docker file provided is used to create a customized container image, and the container image is pushed to the dockerhub. The pushed image is registered on Amazon ECR (Elastic Container Registry) so that it can be used by lambda function. We pull the docker image as as base image for the lambda function and it automatically picks all the required code and libraries for the lambda.

2.2 Autoscaling

For effective and efficient execution we have considered many different options.

Maximum number of parallel lambda functions to be run at the same time in free tier is 50. Considering the computation required to deliver output for our task, the default memory was insufficient, we allowed a lambda function to utilize 2 gbs of memory and it will automatically shut down when the task is completed. This procedure turned out to be less time consuming and more efficient and it serves as scaling in for the architecture.

In a sample run, the facial recognition module was taking 1.5 to 2 mins to complete the task. But the default timeout duration was insufficient for processing the task, so we changed the timeout of each lambda function to 5 mins from 3 seconds which will be helpful for letting the function execute for a sufficient time.

2.3 Member Tasks

Samarth Patel

Design Contributions:-

1. Brainstormed and collaborated with the team about the possible design of the project. Decided to follow the architecture which was elaborated in the project problem statement.
2. Brainstormed the possibilities for scaling in the app-tier using different configuration options for the lambda function.

Code Contributions:-

1. Developed the function to read the encoding file and store the data into a pickle file in the handler.py file.
2. Helped Shubham Bansal to write the face_recognition_handler function which handles the put event trigger on the S3 input bucket.
3. Wrote down the insert_into_table function in main.py which is responsible for reading data from the students_data.json file and writing it to the dynamoDB table.

Testing Contributions:-

1. Helped with the team to create the github workflow and test the docker image file for the lambda function.
2. Conducted final round of the testing, made sure that the architecture was working fine as a whole.

Project Report Contributions:-

1. Designed and draw the architectural diagram for the project.
2. Explained the working and structure of the cloud architecture.
3. Collaborated with Shubham and drafted the final version of the report.

Sulabh Soneji

Design Contributions:-

1. Brainstormed the idea about the design from different aspects and how the application can be implemented.
2. Helped the team to implement the design as it is the important thing, and how to use lambda function inorder to do the auto-scaling.

Code Contributions:-

1. Collaborated with Samarth and wrote down the create_table method to create the dynamoDB table using the boto3 library.
2. Wrote down the check_table method for the main.py file which returns the boolean value according to the existence of the table.
3. Helped other team members to write down the code for creating the csv file and reading the first frame of the fetched video.

Testing Contributions:-

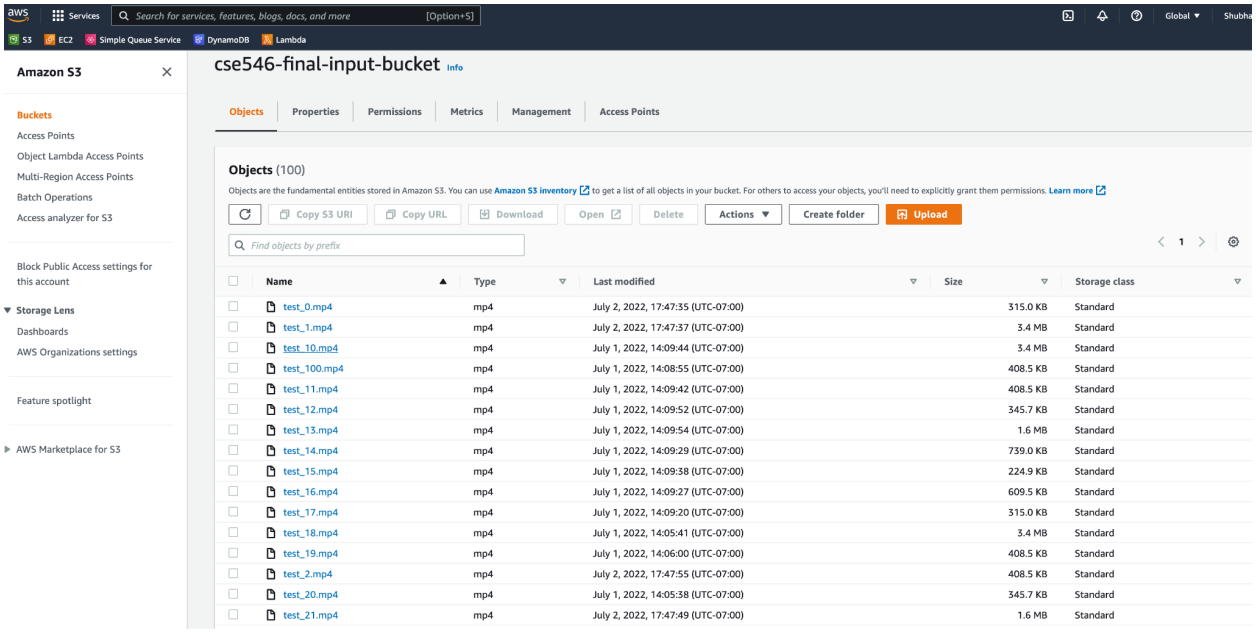
1. Helped Samarth with configuring the customized image using docker hub and deploying it to the Amazon ECR.
2. Carried out the testing for the dynamoDB table, made sure that the python lambda code was correctly fetching the student data given to us.

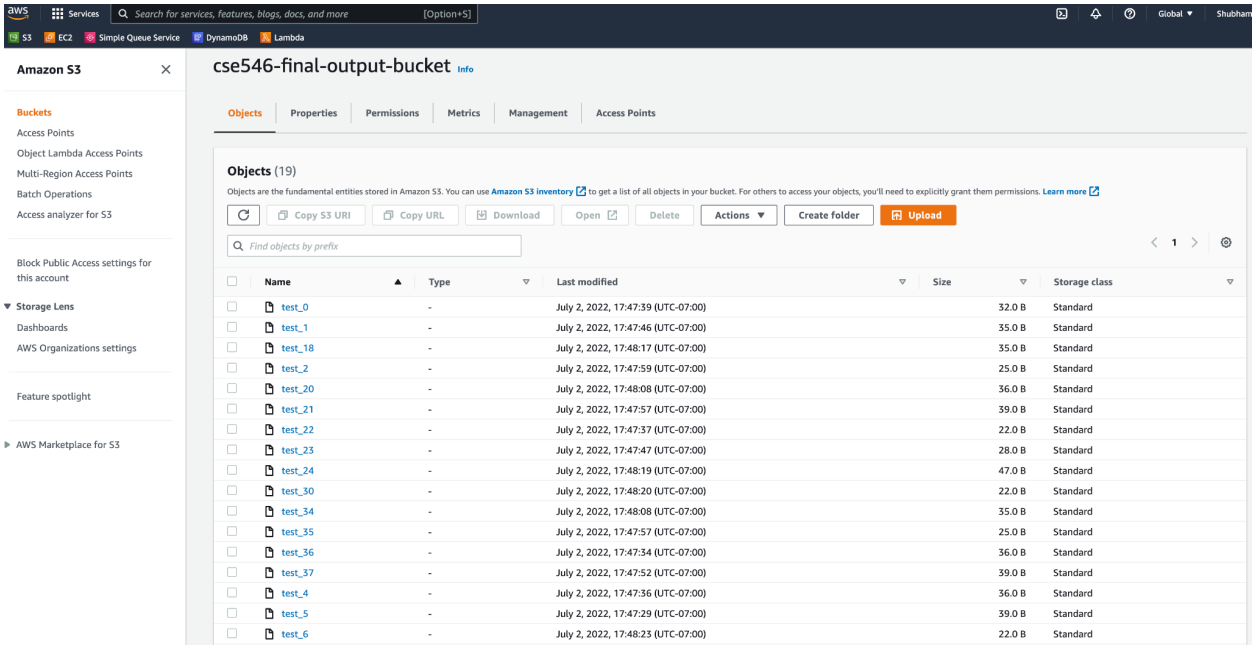
Project Report Contributions:-

1. Helped to implement the design of the architecture and the report.
2. Wrote the problem statement as it was a must thing in order to describe the project precisely and concisely.
3. Also I completed the ReadMe file with all the required and important information details.
4. Furthermore, added all the details related to the testing and evaluation, along with the screenshot.

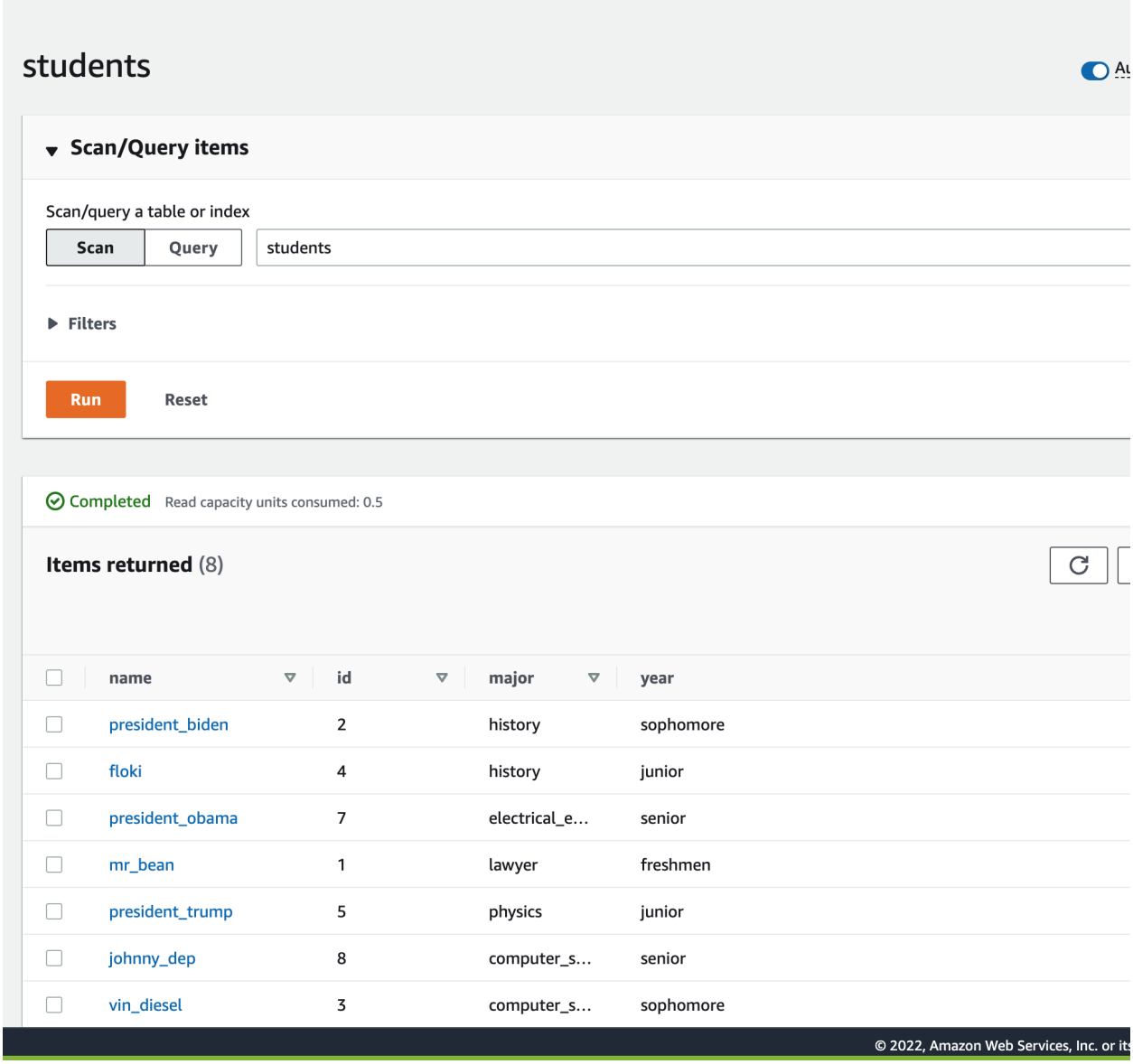
3. Testing and evaluation

Input and output buckets of the application. Input buckets were used to upload video, while the output buckets were used to store the students information received from DynamoDB.

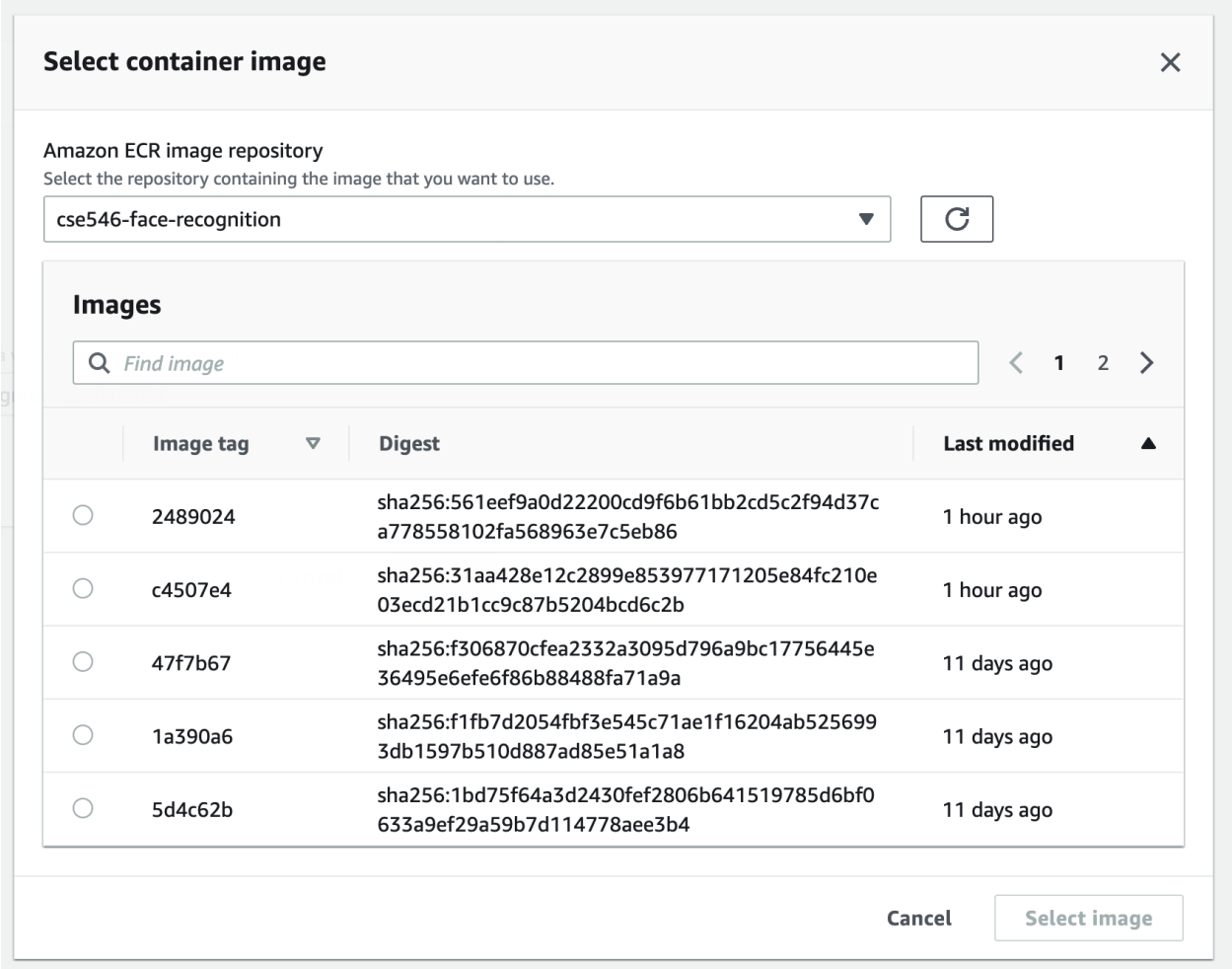




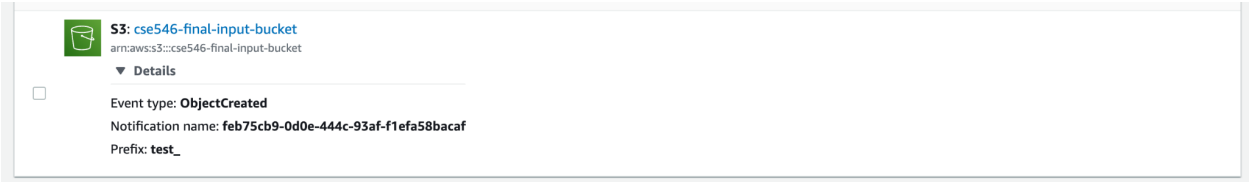
DynamoDB for the student information. There is a table named students which returns all the information like name, id, major, and year of the students. Made sure by testing that the data from the given students_data.json file was correctly being written in the table.



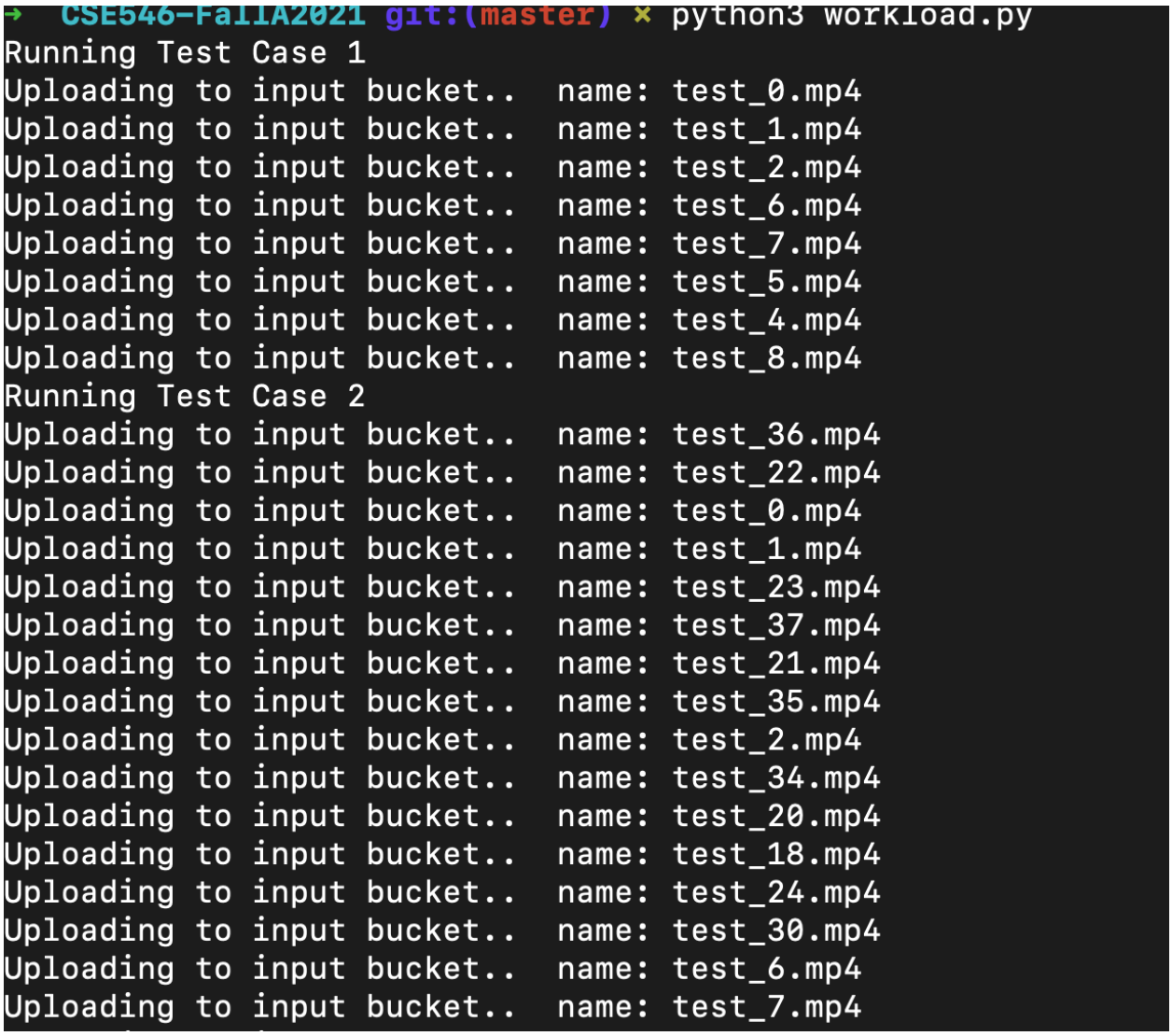
Deployed the custom docker image for Lambda function for handling face recognition, frame extraction, reading and writing data to S3 buckets.



Configure the S3 trigger for the lambda function, which is triggered when the put method occurs on the input S3 bucket.



Command line representation that the test-cases are running and all the images are being uploaded to the bucket and system working correctly as a whole.



4. Code

The project code architecture follows a simple python format. The main files which are important for the project and their functionalities are described below.

main.py:

It is responsible for creating the input and output buckets and creating the dynamoDB table. It has the functions to write data into the dynamoDB table. The table named ‘students’ will be created for holding the student data and will have four columns namely id, name, major, and year. The table will be indexed on the ‘name’ column.

Handler.py

The handler.py file contains the main code for the lambda function. The function ‘face_recognition_handler’ handles each event triggered by the lambda function. It executes a sequence of steps which can be referred to in the Architecture Description section.

1. Get the video from the input bucket,
2. Stores it in a temporary folder (/tmp)
3. Extracts the first frame from the video using ffmpeg library
4. Runs facial recognition algorithm on the extracted frame
5. Recognizes the face and encodes it
6. Loads the ‘encoding’ file and uses it to get the name of the corresponding recognized face in step 5
7. Searches for the recognized student name in the dynamoDB table
8. Puts the information about the student in the csv file, and uploads it to output S3 bucket

Individual Contribution Report

Shubham Bansal (1224102750)

Design Contributions:-

1. Designed the architecture of the project including AWS lambda function integration with DynamoDB and S3 buckets.
2. Discussed the language to use for building lambda function within the time constraint that we had.
3. Managed the project deliverables, code quality, and team members.

Code Contributions:-

1. AWS Lambda function - handler.py
    a. Wrote down reading video from the input bucket.
    b. Wrote down writing the output to the output bucket.
    c. Wrote down the DynamoDB query to extract students information.
2. Setup - main.py
    a. Wrote down *create_bucket* function to create S3 buckets which is used to create both input and output buckets.
    b. Wrote down *check_bucket* function to check the existence of S3 buckets before creating them.

Testing Contributions:-

1. The first round of testing on AWS with setting up the input S3 bucket ‘cse546-final-input-bucket’ and output bucket ‘cse546-final-output-bucket’.
2. Also, setting up the lambda function and the trigger for the lambda function. Providing those setup instructions to the rest of the team for final rounds of testing.
3. Finally verified the service output in the output bucket ‘cse546-final-output-bucket’.

Project Report Contributions:-

1. Helped with writing design and architecture documentation in the project report.
2. Also, I wrote down the testing setup instructions and execution setup instructions for the project report.
3. Provided the screenshot of the live testing of the service to be included in the project report.