# CSE 546 — Project2 Report

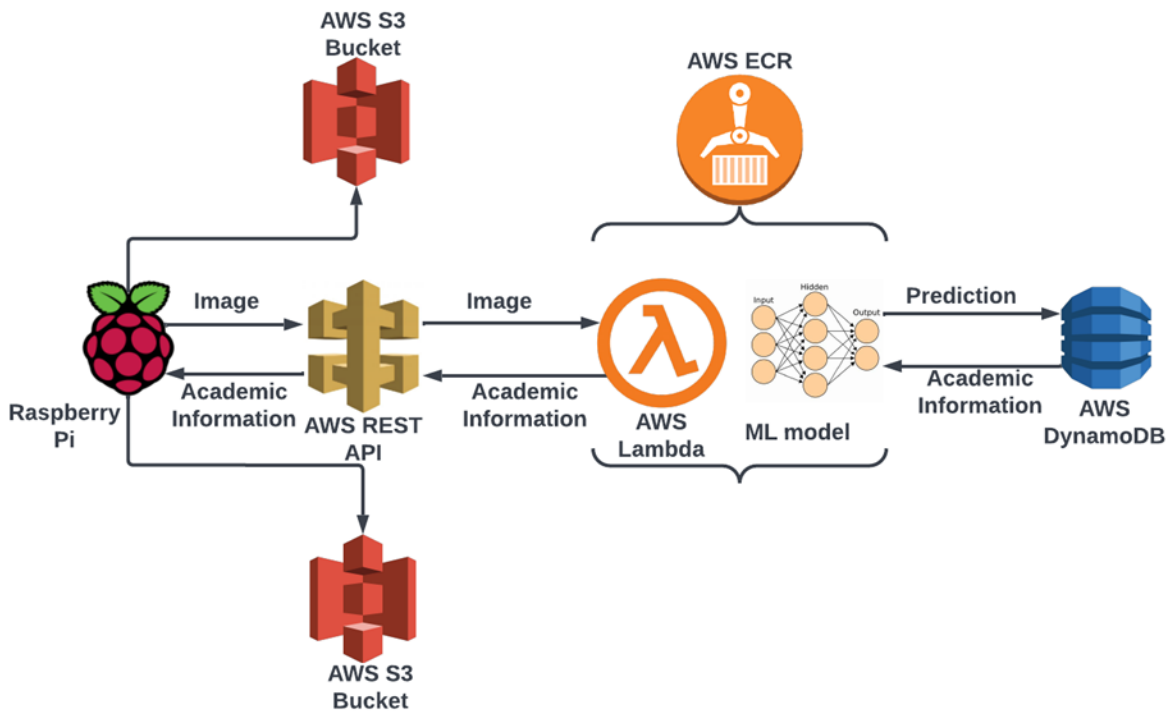
*Gompa Divya (1221003043)*
*Nithin Jayakar Padala (1220061427)*
*Sai Varun Vaka (1223499368)*

## 1. Problem statement

A distributed cloud application which performs real-time face recognition on videos recorded by IoT devices which utilizes Platform as a Services (PaaS). We build the distributed application with PaaS service Amazon Lambda functions which are the most popular and widely used Function as a Service and IoT device Raspberry Pi which is popularly used for IoT development platform. This distributed cloud application is providing experience with developing-real world, IoT data driven cloud applications.

## 2. Design and implementation

### 2.1 Architecture



Architecture Diagram for the face-recognition distributed cloud application

**Raspberry PI:**

The Raspberry Pi camera records the video for every 0.5 second for 5 mins long and uploads each of the 0.5 second video to S3. While recording the video, frames are also being extracted at the same time with the help of 0.5 second video. Then it sends the frame to S3 bucket and makes a call to the AWS Rest API using the image name and gets the Academic information of the student which is displayed on the Raspberry PI terminal.

**Aws Lambda:**

Aws Lambda function is created with a container image of the docker consisting of the training data which has been produced while training the Machine Learning algorithm. This also consists of a lambda function handler which triggers with the API gateway, which takes the input image and sends to predict the image and sends the response to DynamoDB and gets the academic information of the students.

**Aws S3 Buckets:**

‘videosfile’: This s3 bucket consists of all the 0.5 seconds video which were sent from the Raspberry pi Camera. This bucket consists of all the videos stored in the .h264 video format.

‘framesfile’: This s3 bucket consists of all the frames extracted from each of the 0.5 second videos present in the bucket ‘videosfile’. This bucket consists of the frames stored in the .png image format.

**Aws DynamoDB:**

This database consists of a table ‘StudentAcademicInfo’ which consists of items having Students academic information. This information is stored in a table consisting of columns ‘Student Name’, ‘Major’, ‘Year’.

**Aws ECR:**

Elastic Container Registry is being used for storing the Docker container which consists of all the files which need to be run in the lambda function.

**Aws Api Gateway:**

We are using the Rest API of the AWS Apis. We are sending the image information to API gateway which in turn triggers the AWS lambda function which consists of the code related to prediction and response from DynamoDB and responds to the Raspberry Pi with the Student Academic Information.

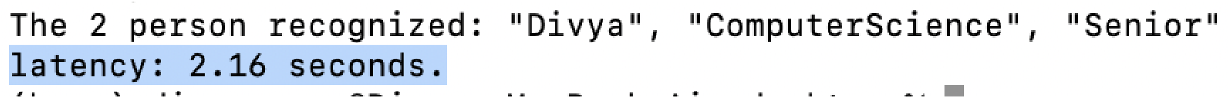
## 2.2 Concurrency and Latency

In this application we are recording a 0.5 second video in the Raspberry PI, once the first video is completed a separate thread is called to upload the recorded video, second thread to process the function where extraction of the frames happens. Then the third thread processes the request to API Gateway which triggers the AWS lambda function which performs the remaining steps. These steps include downloading the frame from the S3 bucket by comparing the key value with the input request from the Rest API. Then sending the image for evaluating the face recognition for the person, this sends the output as a string which is the name of the person being present in the recorded video. Then the predicted output is sent to the DynamoDB to fetch the academic information of that student. Students Academic Information is sent as the API response to the Raspberry PI, which displays the information and Latency of that video.

Here we are achieving concurrency by performing the multithreading which works in such a way that when first iteration is performed only the 0.5 second video is being recorded, when second iteration is performed then 0.5 second video is recorded with the previous video being sent to upload it to s3 and extract a frame with other thread, And third iteration is performed with third 0.5 second video being recorded, uploading and frame extraction of the second video and request being sent to AWS lambda function through the API Gateway. Similarly, three threads process concurrently by performing their individual task. Since multiple processes execute parallely these affect the latency of the task being performed. By making use of the edge computing device where small computations can be performed we are doing frame extraction in Raspberry PI by making the communication time to decrease. This process helped to reach the best latency possible making the effective use of resources.

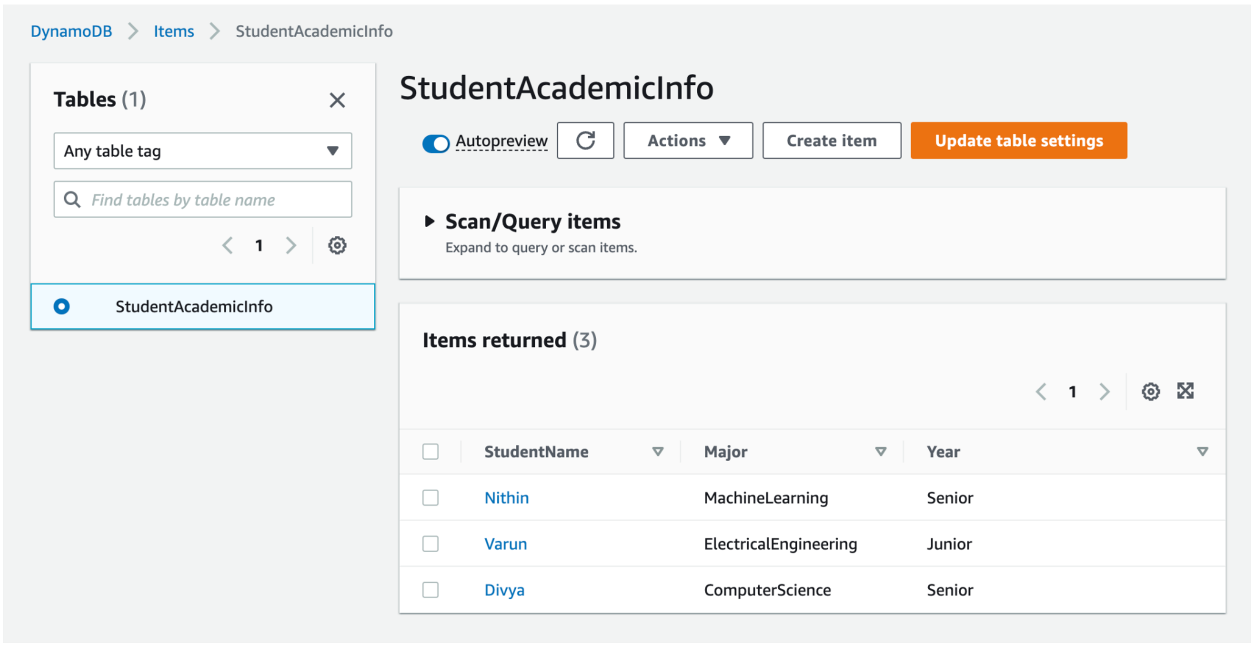
## 3. Testing and evaluation

- *End-to-end latency for each request:*
  We are getting on average 2 seconds latency from recording 0.5 second video to getting predicted person's details from dynamoDB and displaying on Raspberry Pi.



```
The 2 person recognized: "Divya", "ComputerScience", "Senior"
latency: 2.16 seconds.
```

- *The correctness of the returned information:*
  When a face-recognition algorithm runs and predicts a person, we are fetching the Academic information of the person by passing the predicted person name as key to DynamoDB. The academic information displayed on the Raspberry pi and we are cross checking the info with the Academic info in DynamoDB below.

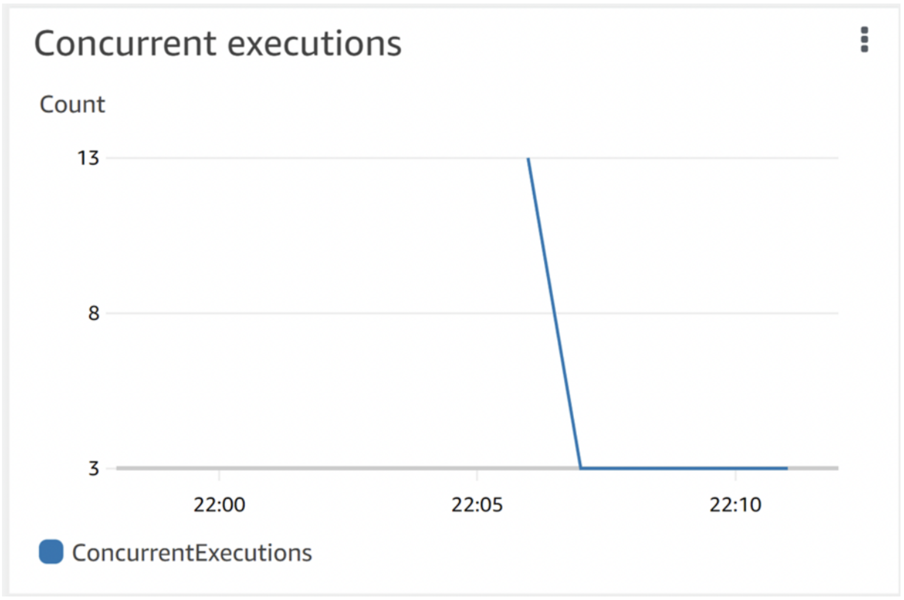


| StudentName | Major | Year |
|---|---|---|
| Nithin | MachineLearning | Senior |
| Varun | ElectricalEngineering | Junior |
| Divya | ComputerScience | Senior |

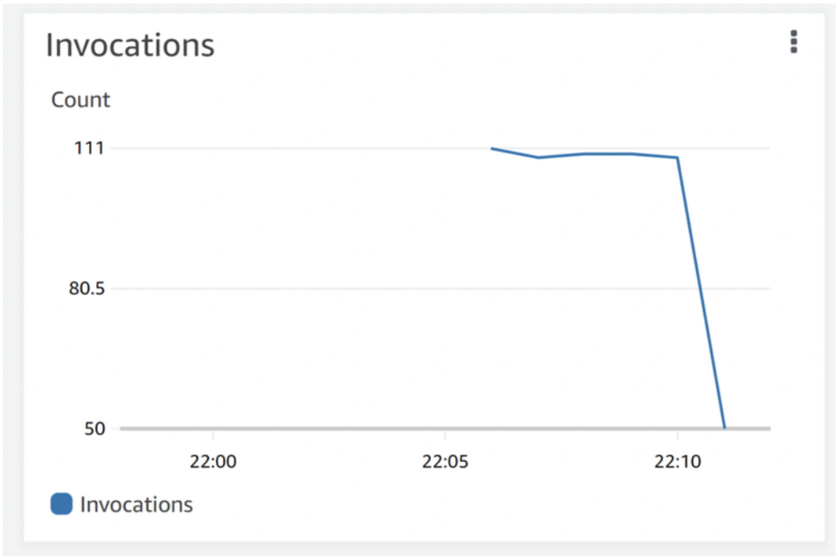
```
The 2 person recognized: "Divya", "ComputerScience", "Senior"
latency: 2.16 seconds.
```

With the training data in the machine learning model, we achieved 71% of the accuracy for face-recognition prediction.

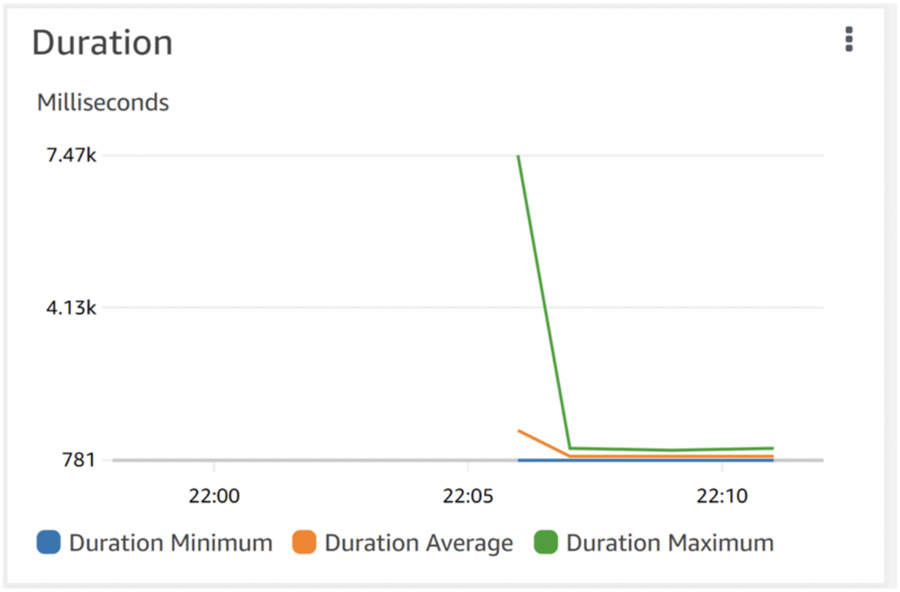
- *The autoscaling of Lambda functions in the Lambda Web UI:*
  Concurrency: 12



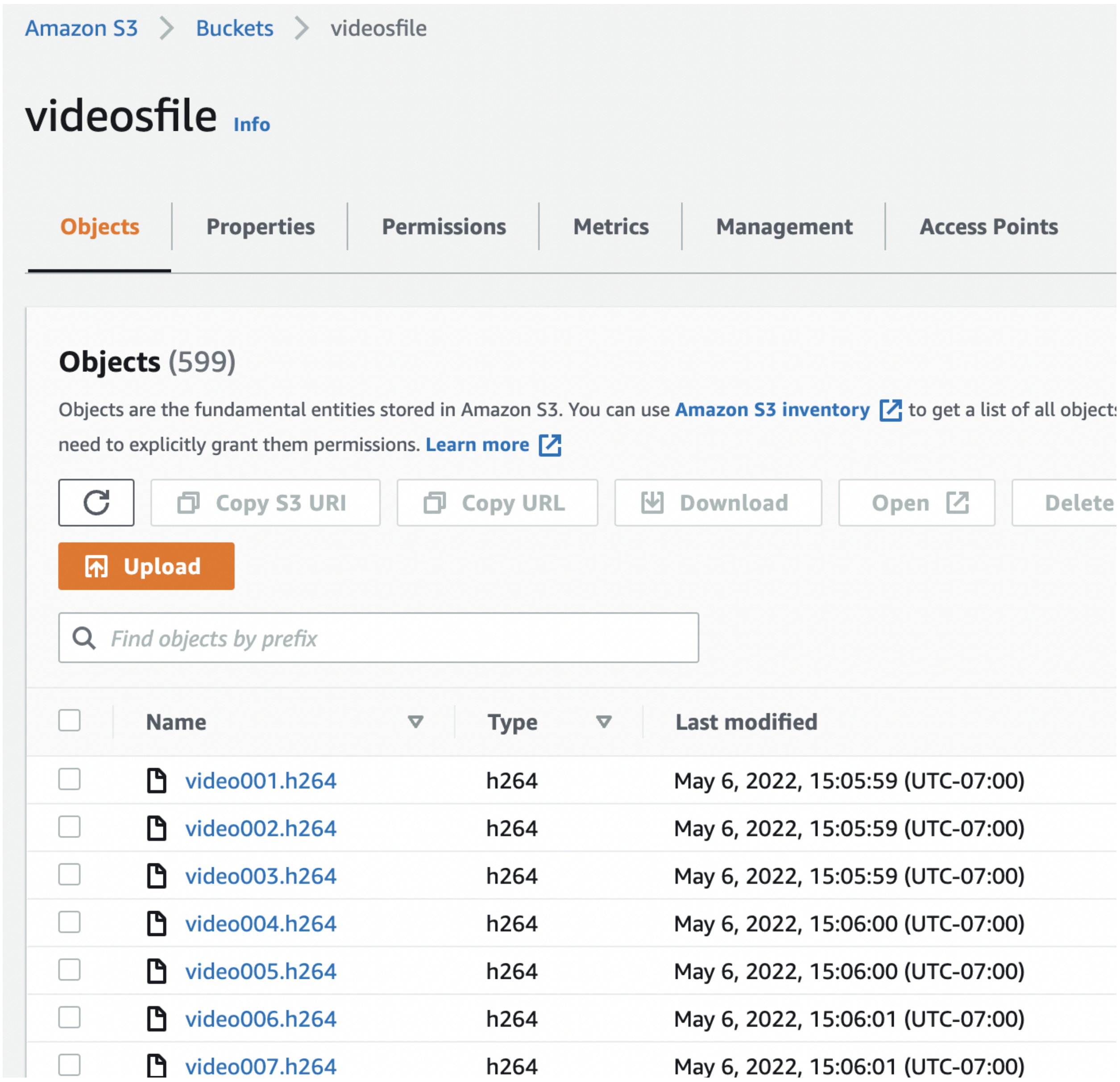
The number of invocations: 600



Duration: 1.4 seconds



- *Received videos in S3:*
  The 0.5 sec videos recorded from raspberry pi are stored in s3 bucket and we are verifying that each video taken is storing it to s3. Also, checking the prediction result by comparing the video in s3 bucket by downloading the videos. Verified the total 5 minute videos in s3 bucket.


Amazon S3 > Buckets > videosfile

videosfile Info

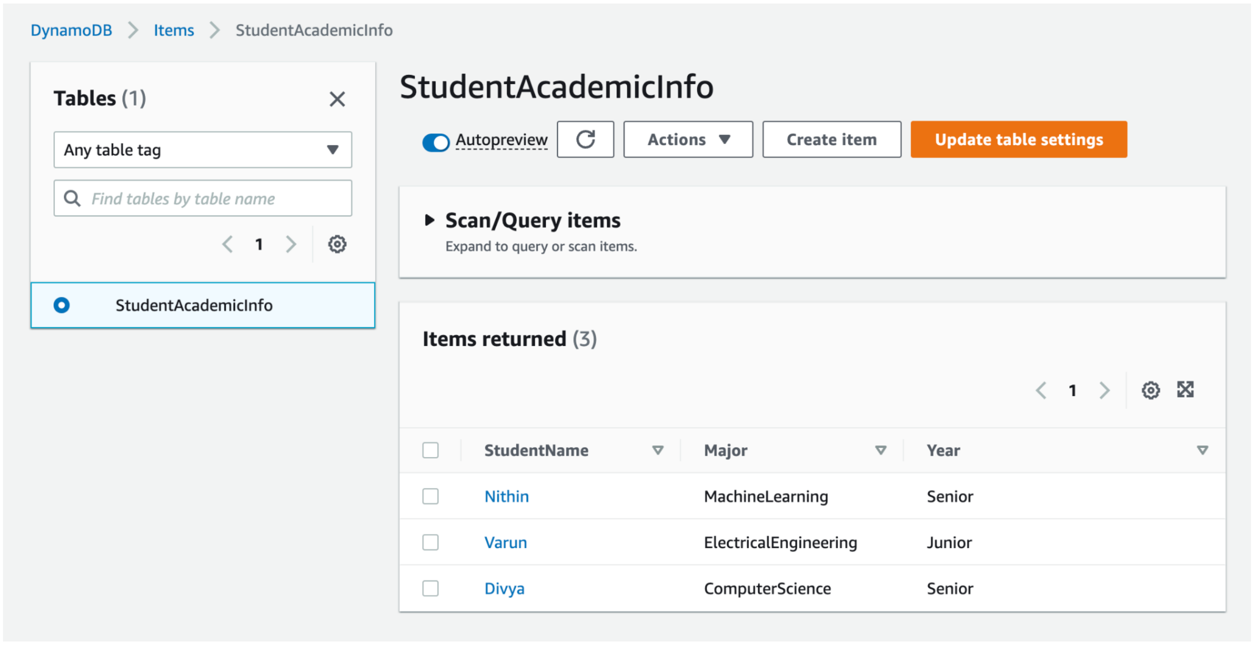
Objects | Properties | Permissions | Metrics | Management | Access Points

**Objects** (599)

Objects are the fundamental entities stored in Amazon S3. You can use **Amazon S3 inventory** to get a list of all object need to explicitly grant them permissions. **Learn more**

| Name | Type | Last modified |
|---|---|---|
| video001.h264 | h264 | May 6, 2022, 15:05:59 (UTC-07:00) |
| video002.h264 | h264 | May 6, 2022, 15:05:59 (UTC-07:00) |
| video003.h264 | h264 | May 6, 2022, 15:05:59 (UTC-07:00) |
| video004.h264 | h264 | May 6, 2022, 15:06:00 (UTC-07:00) |
| video005.h264 | h264 | May 6, 2022, 15:06:00 (UTC-07:00) |
| video006.h264 | h264 | May 6, 2022, 15:06:01 (UTC-07:00) |
| video007.h264 | h264 | May 6, 2022, 15:06:01 (UTC-07:00) |

- *The academic information stored in the database:*
  The academic information of the students are stored in DynamoDb and this information is retrieved when a student is recognized by face-recognition algorithm.



StudentAcademicInfo

| StudentName | Major | Year |
|---|---|---|
| Nithin | MachineLearning | Senior |
| Varun | ElectricalEngineering | Junior |
| Divya | ComputerScience | Senior |

## 4. Code

### Training model:

The machine learning model is trained on 30 images of each person plus validated on 6 images of each person. After a great exploration, we decided to go with Adam optimizer to attain an accuracy of 71% on the test data. We used 0.001 as the learning rate and 12 as the batch size.

### Function_handler.py

This function takes the frame name as the input, with the input key it tries to download a frame from S3 converts the image from rgba to rgb. Then it sends the downloaded image to eval_face_recognition.py where the face recognition evaluation is performed on the image and output is predicted based on the previously trained model and will be one of the team member's names. This output is sent to DynamoDB which consists of a 'StudentAcademicInfo' table which has the academic details of each of the students, these academic details are added to the response object which acts as the REST API response.

### Dockerfile

Docker file consists of the following steps:

1) Installs a fresh copy of the image and GCC.

2) Installs aws-lambda-cpp build dependencies

3) Installs Lambda Runtime Interfaces Client for Python

4) Consists of a shell script entry point which acts as an entry point for the lambda function to trigger.

5) Function_handler.py file is triggered at the end of the docker file completion.

**RaspberryPiVideoProcessing.py**

In this file the Picamera starts recording the sequence of 0.5 videos in the 5 mins duration. While recording each of the 0.5 videos we are creating two new threads to perform multithreading. This threading consists of the first thread being used to upload the 0.5 second video to the 'videosfile' s3 bucket. Second thread extracts the frame from the recorded video, uploads it to 'framesfile' s3 bucket and sends the frame name as the input to the REST API gateway and gets the academic information of the student present in the recorded video.

Explain in detail how to install your programs and how to run them.

1) Download Docker from the link below and install.

   https://docs.docker.com/get-docker/

2) Go to the project directory in the terminal where it is extracted and run the below command:

   "docker build -t docker_container ."

3) Inorder to test the code in local machine, run the docker using the below command:

   "docker run -p 9000:8080 docker_container"

4) To test the docker container's invocation and output we need to run the below command terminal:

   "curl -XPOST "http://localhost:9000/2015-03-31/functions/function/invocations" -d {"queryStringParameters":{"key":"frame.png"}}"

5) This should give the output of the Student Academic Information of the student present in the image.

6) In the Raspberry PI run the below commands in the terminal to install the dependencies related to python file:

   "Pip install boto3"

   "Pip install ffmpeg-python"

7) Then copy the file "RaspberryPiVideoProcessing.py" to RaspberryPI. Go to the directory where the file is present and run the below command in the terminal

   "Python3 RaspberryPiVideoProcessing.py"

8) Keep the face in front of the camera to predict the face with a raspberry pi camera by machine learning module.
9) The prediction person academic details from DynamoDB along with the latency gets displayed on the Raspberry Pi.

## 5. Individual contributions

**DIVYA GOMPA (1221003043)**

## *Design:*

- Proposed several possible architectural designs for the face-recognition distributed cloud application and discussed with final design possibilities and outcomes.
- Added a change to the architecture by extracting frame from 0.5 second video on the raspberry pi and uploading it to s3 bucket to offload the work on cloud.
- When deciding to use API Gateway to communicate with lambda functions, researched with possible options available and choosed Rest API to serve the purpose.
- Added another Amazon s3 bucket to the design to store extracted frames from 0.5 sec video to give a dedicated s3 bucket for videos and frames.

## *Implementation:*

- Built a docker container image and installed the required software to it to function the lambda function and face-recognition machine learning model.
- Pushed the created docker image to the Amazon ECR repository.
- Created Rest API and linked to lambda function to trigger the lambda function whenever request from API comes.
- Developed a lambda function to run the face-recognition machine learning model on frame and retrieving the predicted person's info from dynamoDb.
- Built a response object in lambda function to respond to the API gateway request whenever it comes.
- Implemented code to extract frames while raspberry pi is recording.
- Created a table and added academic info of the students in DynamoDB.
- Solved some application bugs raised while testing.

## *Testing:*

- Validated connections between rest API and Lambda functions.
- Tested the end-to-end application whenever the machine learning model changed for accuracy.
- Verified the format and accuracy of the academic information displayed on the raspberry pi.
- Tested the machine learning model with different sets of train data.
- Tested the number of videos uploading to the S3 bucket are matching with overall 5 minute length for multiple runs.

**Sai Varun Vaka (*1223499368*)**

## *Design:*

- I implemented the final architecture of the project which we are following currently in this application after brainstorming through several designs.
- Designed few ways to achieve the best concurrency by making the effective use of multithreading.
- Suggested the effective way to extract the best images by considering the external constraints which might affect machine learning algorithms.
- Solved the challenge of recording a 0.5 second video in very less time frame.
- Created scenarios where the output responses from API gateway or uploading to s3 bucket can fail and handling them.
- Done some research how images data can be divided between train and val to provide the best accuracy.

## *Implementation:*

- Implemented multithreading by using different threads processing concurrently which perform different operations.
- Implemented a code to record 0.5 seconds video effectively which makes usage of lambda in a less time consuming manner.
- Improved the accuracy of the face recognition model to 72% by changing the learning rate, batch size and step size.
- Reduced the end-to-end latency for each video by making invocation to the aws lambda by handling the exceptions.
- Added the code for uploading the data related to S3 and creating S3 buckets from the RaspberryPI.
- Implemented the changes for making the call AWS Rest API with the required parameters.

## *Testing:*

- Tested the end-to-end application with multiple changes in multi threading to test the latency.
- Tested the docker container locally by giving some sample inputs and comparing the result with expected output.
- Tested and cross checked the academic info displayed on raspberry pi with info in the DynamoDB.
- Verified there are no override requests or non uploading files to S3 due to previous runs.
- Evaluated the stored videos in s3 buckets to compare with predicted results from face-recognition machine learning model.