# CSE 546: Cloud Computing (2023 Spring) Project 2 MCS Portfolio Report - PaaS

Anurag Banerjee (ASU ID: 1225497546)

*School of Computing and Augmented Intelligence,*
*Ira A. Fulton Schools of Engineering*
*Arizona State University*
*Tempe, United States*
abaner40@asu.edu

***Abstract*— This document details the contribution of Anurag Banerjee to Project 2 of the course Cloud Computing offered in Spring 2023. The project was to be done in a group of 3. The main idea of the assignment was to create an application that was elastic and could scale out and in based on the demand in a cost-efficient manner with minimal human intervention with the help of PaaS cloud. This application was to be built using AWS (Amazon Web Services) Lambda and other tools and services offered by AWS since AWS Lambda was the most widely used function-based serverless computing at the time of the project.**

***Keywords*— Paas, Amazon Web Services (AWS), Simple Storage Service (S3), AWS Lambda, Python, Cloud Computing, Face Recognition**

## I. Contribution

The project aimed to implement a smart classroom assistant for educators. The application's purpose was to collect videos from the educator's classroom, perform face recognition on the collected videos, look up the recognized students in the database, and return the relevant academic information to the educator. The application workflow consisted of uploading a video to an upload bucket in S3, which would trigger the lambda function. The lambda function would then download the video from S3, separate it into its frames, recognize the faces, fetch the students' academic information from DynamoDB, and write it to the output bucket in S3.

The project was divided into three major parts: setting up services and other data in AWS, developing the python script for the handler, and dockerizing the script and validating its functionality. Within the group, I was responsible for the python script for the handler, which was invoked as soon as a video was uploaded into the upload bucket in S3. The handler received an event argument containing the necessary details of the uploaded image.

To begin, the script created a temporary directory with the prefix as the video filename, using the tempfile library in Python. The video from the triggering event was then downloaded into this temporary folder, and the ffmpeg tool was utilized to extract frames from the video. These images were saved in the same temporary directory created above, and the list of image names present in the temporary directory was stored in a list.

Furthermore, the encoding file provided to us was utilized to extract the known face names and known face encodings using the numpy library. A flag was then created with a value of false, which helped to stop the code's execution when a face was recognized. For each image in the temporary directory which was stored in the list, the face_recognition library was utilized to load the image file, extract the face locations, and extract the face encodings from these locations. The encodings were then matched with the known encodings, and once a match was found, the name was stored in a variable, and the flag was set to true, so the loop would not execute further.

Finally, the details of the recognized face were fetched from DynamoDB, and on successful extraction, the name, major, and year were stored as a comma-separated values (CSV) file in the output bucket with the name of the file as the name of the video that was uploaded. The python script for the handler played a crucial role in the application's workflow by recognizing the faces in the uploaded videos and extracting their corresponding academic information from the database.