CSE - 546 - Project 2 - Portfolio Report

Priyadarshini Ramakrishnan(ASU ID - 1225407339)

My primary responsibility in this project was to create a table in AWS DynamoDB and upload the given student information to it. Additionally, in the handler.py module, which is the primary module of this application, I developed the logic to retrieve details of the recognized names after matching with the encoding file from the created table, convert the data obtained into a CSV file, and upload the file into the output S3 bucket and also to test and evaluate the application as a whole.

To accomplish the function of uploading the given JSON data to a DynamoDB table, I utilized the boto3 python library to connect to the DynamoDB resource. The store_items() function reads a JSON file named student_data.json that contains data on multiple students. It then establishes a connection with the DynamoDB table and for each student record in the JSON file, the function employs the put_item() method of the DynamoDB table to insert a new item with corresponding attribute-value pairs. I also developed a retrieve_item() function for obtaining a specific student record from the student-data DynamoDB table to verify if the data was uploaded successfully. In this case, it retrieves the record of a student named "mr_bean." The function prints the retrieved item and some attributes, such as name, major, and year, for display purposes.

To retrieve and store information from DynamoDB as a CSV, in the dynamo_db function, I utilized the table.get_item() method to retrieve an item from the table based on the person_name key. A Key parameter is a dictionary that specifies the key-value pair for the item to retrieve. If the item exists in the table, it is returned in the response variable. If the item exists, the csv_content string is constructed using f-strings. The values for name, major, and year are accessed using item['name'], item['major'], and item['year'], respectively.

Finally, the csv_content is uploaded to the output S3 bucket using the s3.put_object() method. The s3_object_key.replace(".mp4", ".csv") method replaces the .mp4 extension of the original S3 object key with .csv. The csv_content string is then returned to the face_recognition_handler function, which is returned as the result of the Lambda function.

Finally, I contributed to the overall testing and evaluation of the application by collaborating with the team on each stage of the application from constructing the handler file and testing it locally using the provided workload generator, building the docker image, storing it in the ECR registry, and deploying it as a Lambda function, setting up the S3 trigger and running the workload generator. On each step, we verified if the application worked without breaking and made sure the process was completed within a reasonable time, which in our case was less than 3 minutes.