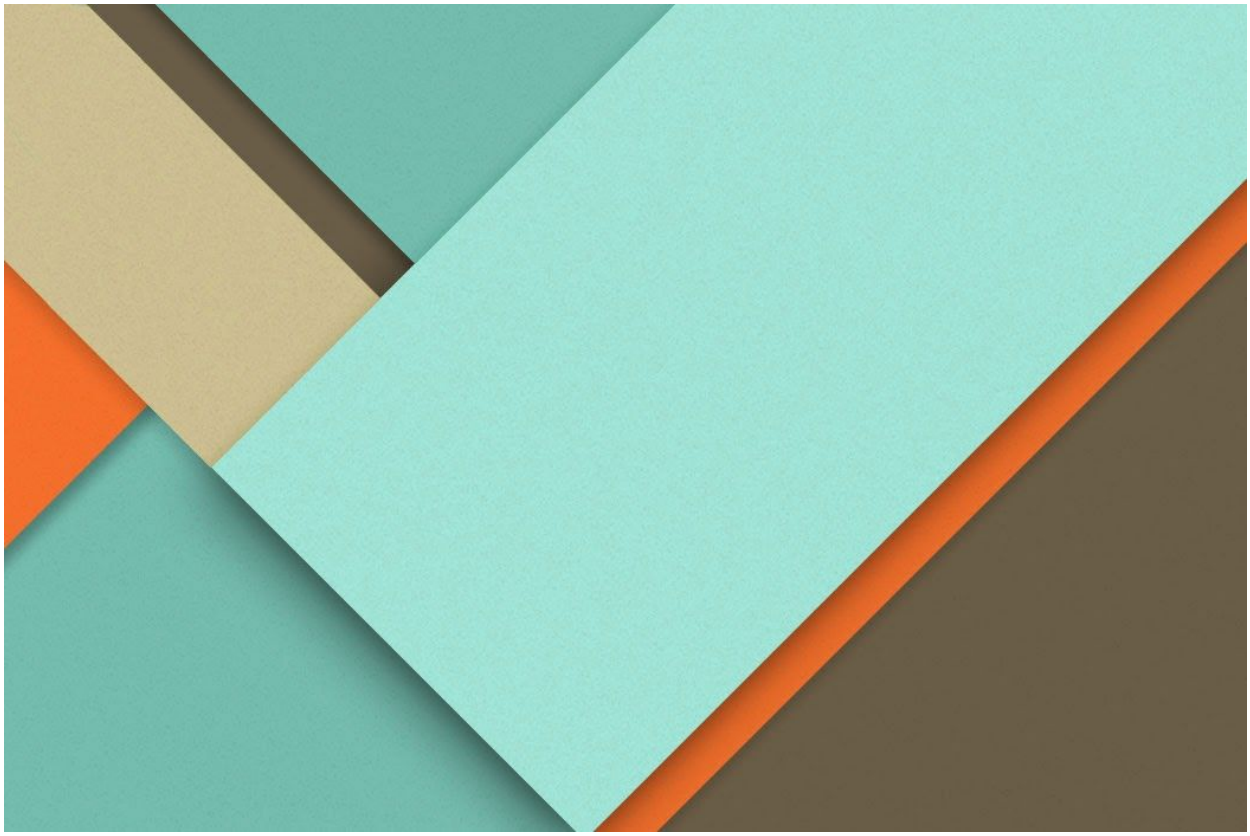


Meet-Up Assistant

Project Report (Cloud Computing and Big Data - Fall 19)

Team:

Mohit Patel(msp552)

Saqib Patel(sip257)

Shivam Raval(sgr341)

Tirupal Rao Ravilla (trr321)

Dept. of CSE, NYU Tandon School of Engineering

Introduction

Meet-up Assistant is a virtual assistant that helps a host organize a meet-up (or any gathering of any scale) and help them plan ahead by giving insights of the people who have gathered (either at the current gathering or over all the past gatherings). It recognizes faces from a real time video streaming, gathers the required details of the guests with minimal interaction, with the help of data APIs, checks the guests in for the meet-up, and runs an analysis on the background on the checked-in guests list, for the host to consider and make any decisions for future gatherings.

Meet-ups are social gatherings of people who have common interests. They generally attract a variety of people from different professional and educational backgrounds. Typically, the hosts setup a date and time for which interested members of the meetup group can choose to attend and typically even bring along others who may be interested, so they can see how the group is like and join the group and attend later meetings.

In each meetup gathering, the host may select a topic for discussion, or an activity for the time the meetup happens, or even a seminar by anyone interested. So, deciding on what topic to choose for each meetup could be a difficult task. It could also get difficult for the host to keep track of members, their interests and to keep track of frequent visitors in order to give priority to the topics of interests that the frequent attendees would prefer. By doing that, the host can attract more guests to attend frequently.

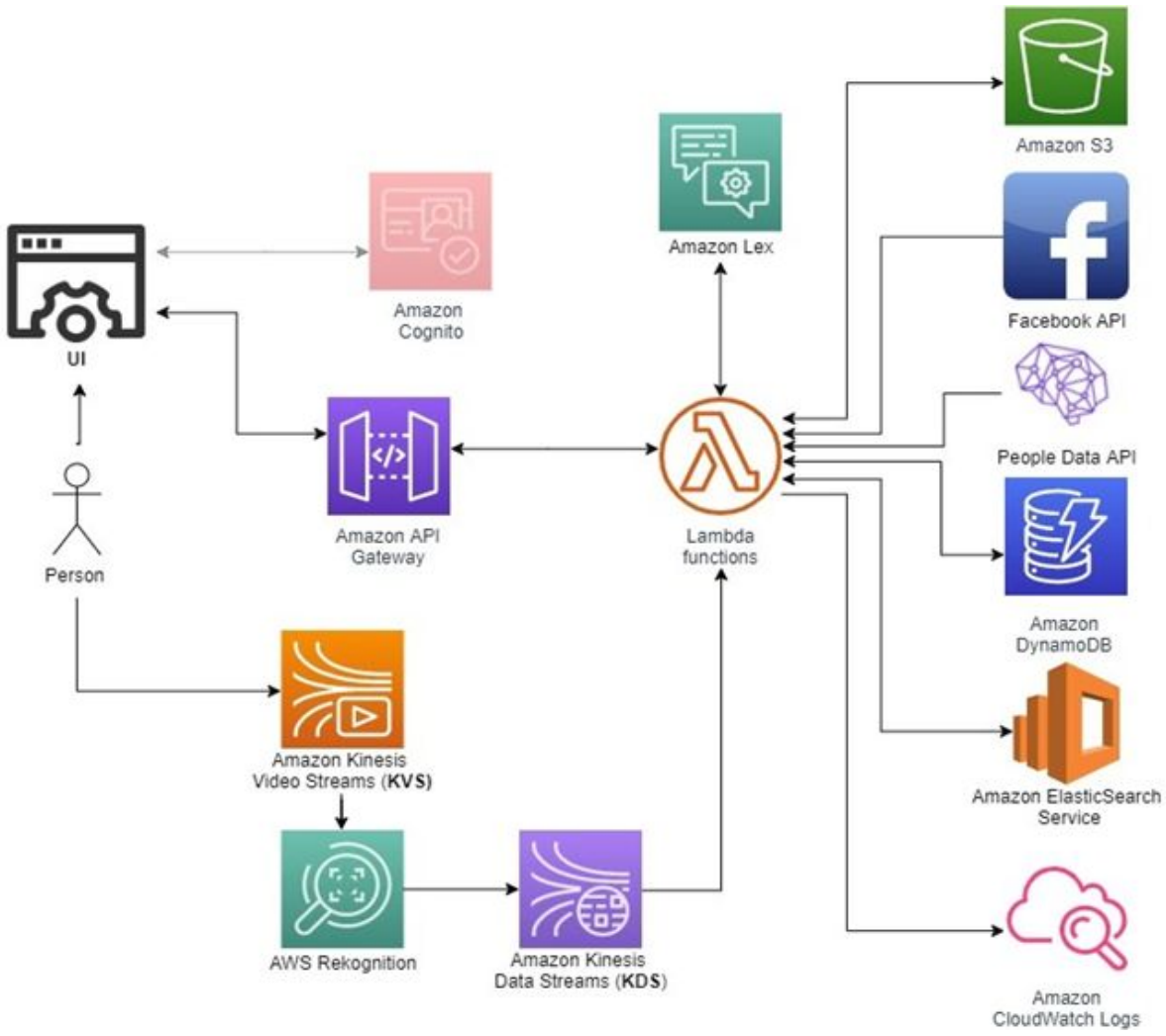
In this light, we introduce the Meet-up Assistant to help the host make easy decisions, keep track of attendees at each meetup gathering, gather their data from open source professional and educational APIs in order to not only make decisions, but also promote the meetup group to attract more people.

With the real time facial recognition, minimal interaction with the guests, real time email notification about the new attendees, automated information scraping using data APIs, and insights into the data of the attendees, Meet-Up assistant can reduce the burden of organization on the host to a very high degree. In addition, usage of AWS services provides a guarantee of scalability.

Features

- ❏ Real Time Face Recognition
- ❏ Minimal Interaction check-in of the guests
- ❏ Real time analysis of guest data
- ❏ Real Time Email Notifications
- ❏ Easily Scalable
- ❏ User friendly
- ❏ Can easily add new features

Architecture



Key Services

KINESIS VIDEO STREAMING

We have used Kinesis for Live video streaming. We have used web-cam on our laptop as hardware to stream the video. We have followed the following steps to configure our hardware:

- Create a new video stream on Amazon Kinesis Video Stream (KVS)
- Download C++ Producer SDK for KVS and configure it. Configure GStreamer libraries on the laptop
- Run GStreamer to ingest video data from the webcam into Kinesis Video Stream.

AMAZON REKOGNITION VIDEO

A Rekognition Data processor is created with KVS stream as the input and Kinesis Data stream as output for continuously analyzing the streaming video. The rekognition data processor hooked to read the KVS stream identifies if the person in the video belongs to the configured face collection or not.

There are two kinds of guests at this point, a known guest whose face is present in the collection, who might have attended at least one meet-up gathering previously, and the unknown guest who is coming in for the first time to the meetup.

KINESIS DATA STREAM

The output from the AWS Rekognition is then ingested into Kinesis Data Stream. This triggers the face detection lambda, which takes the process forward.

LAMBDA FUNCTIONS

We have used lambda functions for the following main purposes,

- **The face detection lambda**: triggered when KDS starts receiving data from Rekognition. This lambda first processes the KDS stream data, and accesses the KVS video records to extract the image of the guest's face from the frame Rekognition detected a face. Then, the face-id and the meetup id is used to create an appearance id which is used to search in the elastic search service, which is on the MeetUp data table, to check if the person is already checked in. If the person is not checked in, meaning meetup data elastic search returns nothing, then a second search is made on the Elastic search service on the 'people data table' to see if the face is of someone known. If the face is known, then is only a matter of checking them in, so a message is pushed onto SQS queue with the face-id to asking the UI to check-in the guest. If the face is unknown, then we need to gather the details of the new person, so first their face id is added into the people data table, then an appearance id is created to add an entry into the meetup data table, a message with face-id is pushed to the SQS queue to register the unknown person, and finally the link to the detected face image that is stored in S3 bucket is sent to the email of the host.
- **The Lex lambdas**: one to process the API calls and extract the input message to Lex and send it to Lex after formatting. And the other is to handle the dialog management on Lex bot, in which People data labs API is called first on the email id collected from the user, and if the education and work data is provided, then

updates the people data and meet up data table entries for that guest, otherwise, continues the conversation with the guest to gather their details that are required, such as the name of the educational institution where they pursued their latest education, and the place where they are currently working or last worked, and even their name (in case people data labs api does not return even that). Clearbit API and CollegeScroreCard API are used to gather further information about the university and place of work respectively, and they are stored against the the entries made for the person in the meet up data table and people data table.

- **Check-in lambda**: It is used to add an entry to the check in data based on people database.
- **DynamoToS3 Lambda:** To send the json data from DynamoDB to S3 so that Quicksight can access it from there to visualize the data and provide insights on the professional and educational data gathered about the guests who are present at the meetup.

S3 BUCKET

We have used three separate buckets: one to store images of guests, one to store QuickSight analyses, and one to host the UI

Notes:

- For images of the guests, only S3 links are stored in the database
- For QuickSight analyses, periodically a Lambda is triggered to read the data file and update the manifest file on the S3 bucket which is used by Quicksight.

DynamoDB

People Data Table: This table is used as a global people data table which stores information regarding the people who have attended previous events and may or may not be yet checked in for the current event. The primary key used here is the *faceId* for the people, which we get from Rekognition.

Meet Up Table: This table is used to store data for the people who have checked in for a particular event and provides information regarding each individual event. The primary key used here is *app-id* which is a combination of meetupId and faceId(meetupId-faceId).

AMAZON QUICKSIGHT

We have used Quicksight to create and analyze visualizations for the data collected from the people visiting the event. Quicksight provides an interactive dashboard that include customized visualizations for the data and provides valuable ML insights which the host can then consider to make decisions for any future gatherings. The main highlights provided are the work and education backgrounds of the people attending the event and which sector they belong to.

LEX

Amazon Lex is used as the conversational bot behind the MeetUp Assistant. Only one intent is created - meetup

Slots are as follows:

- Email
- Work place
- University
- Name

For unknown guests, upon Fulfilment, their data is processed and stored in DB along with additional data.

API-GATEWAY
Two endpoints were created:

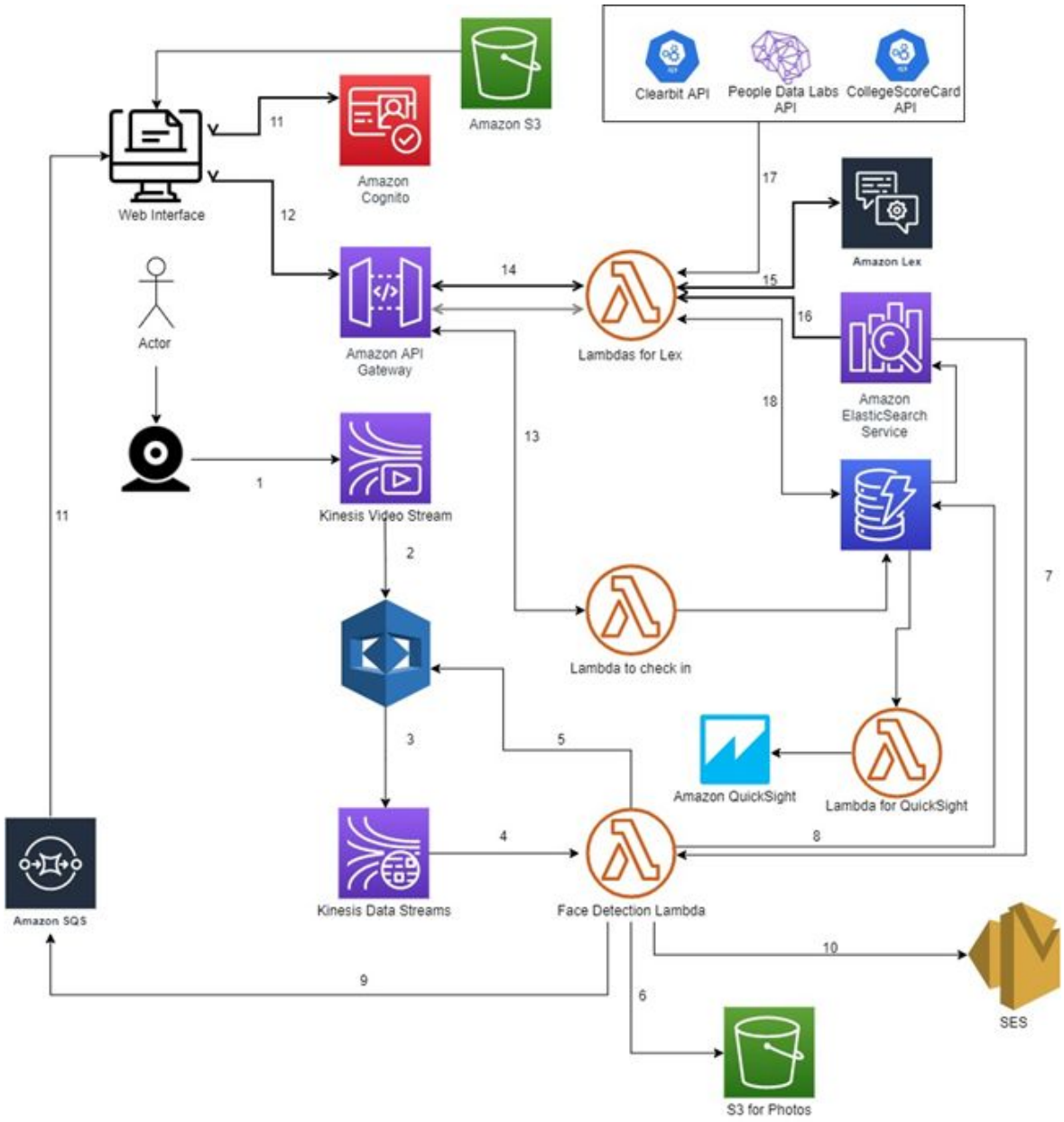
/unknown

- POST method
- Handles the case of unknown guests by calling the lex bot.
- {“message”:””, “faceid”:””} is the request data json format

/checkin

- POST Method
- To handle the case where the user is known and has to be checked in
- {“faceid”:””} is the request data json format.

Flow

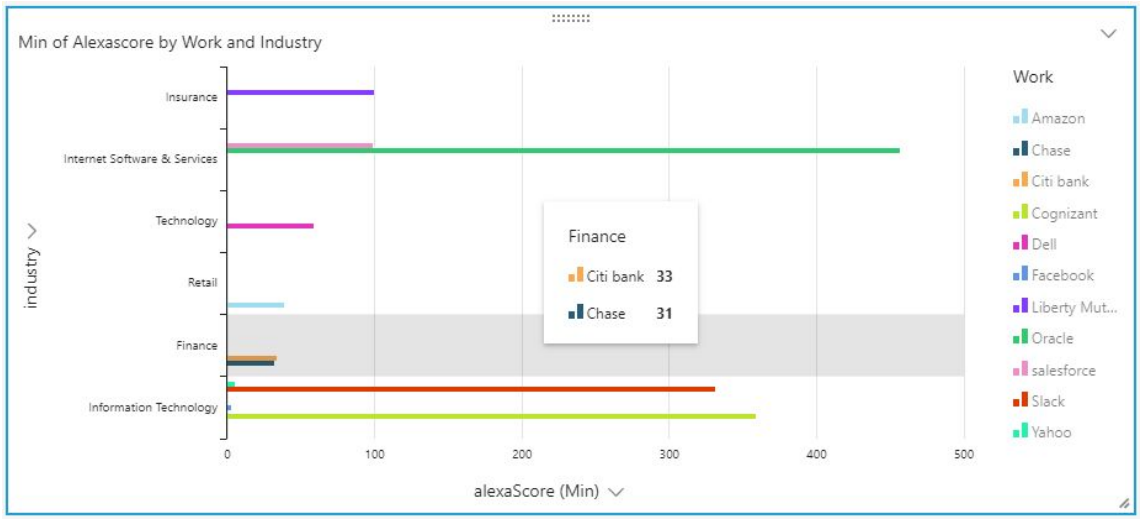


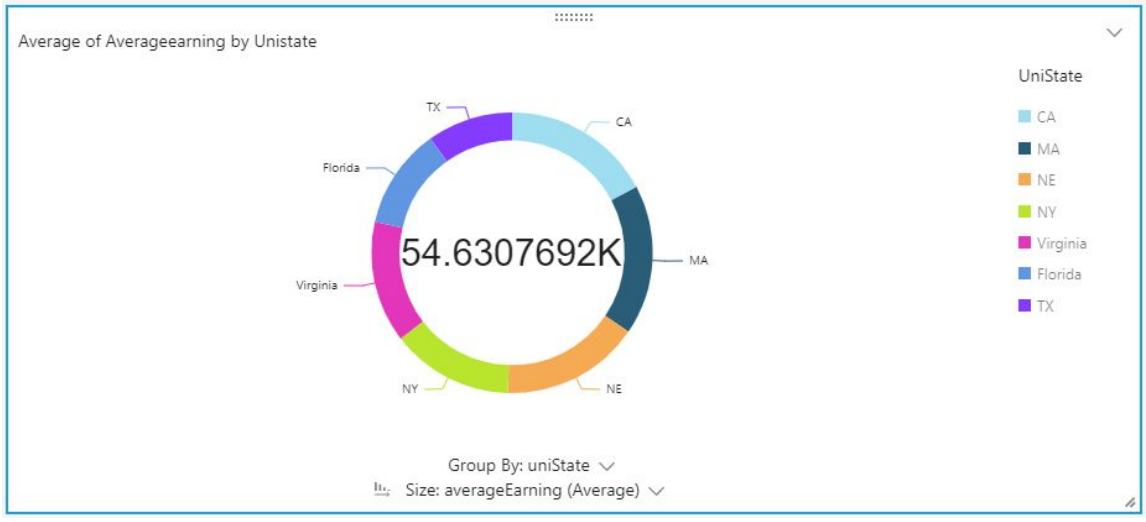
The figure above shows how the flow of the service is in real time, and the numbers indicate the order in which each of those actions take place. The following is the corresponding explanation for each of those.

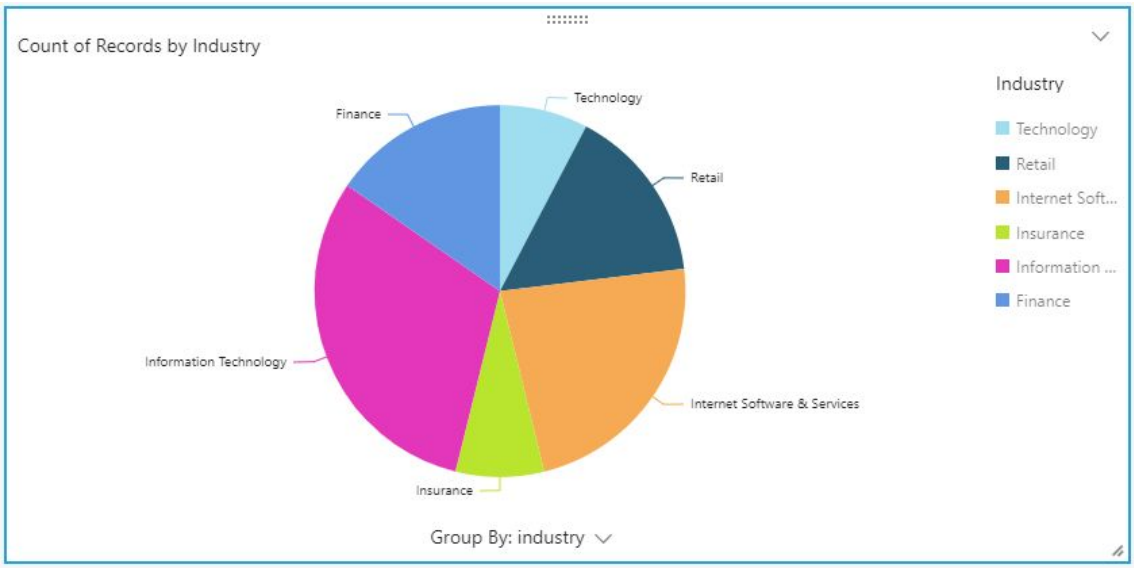
1. Video data is streamed from the webcam to the Kinesis Video stream.
2. Video Stream data is ingested into the Rekognition data processor
3. The frames of the video in which faces are detected are streamlined and ingested to the Kinesis Data Stream by Rekognition data processor.

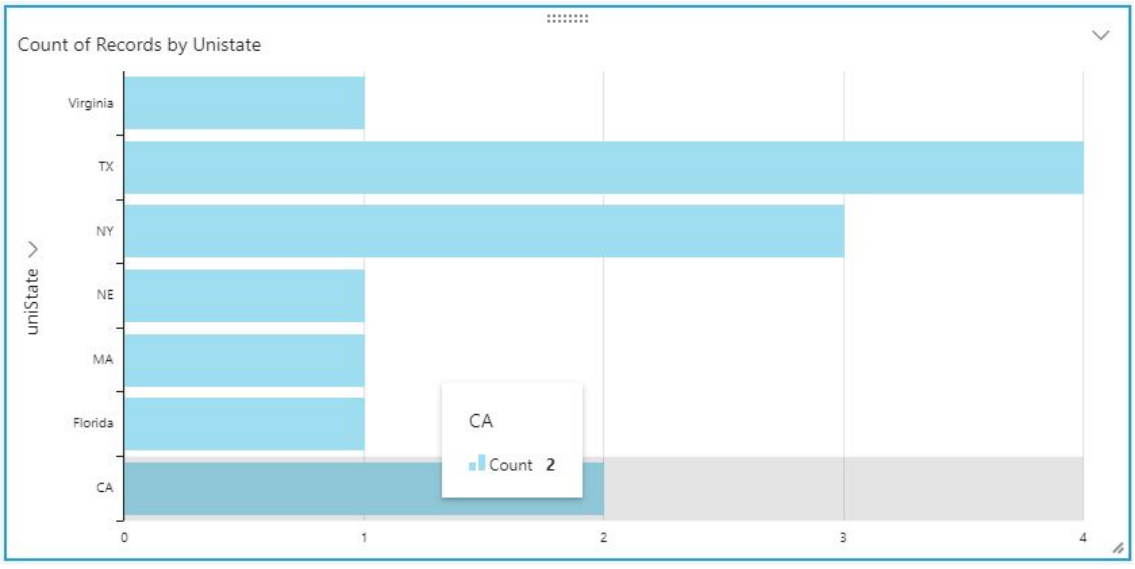
4. Face Detection Lambda gets triggered by the reception of detected faced on KDS.
5. The details of the frame for detected face is read from KDS and used to extract the actual frame from the Video Stream, and then the image is indexed on Rekognition face collection
6. When the person is unknown, the face image is stored in the S3 bucket.
7. Two elastic searches are made: (see face detection lambda) one on the MeetUp data table Elastic search, to see if the person is already checked in, in which case there is no need to process or trigger anything, otherwise, second is made on the people data table elastic search, to check if the person is known or unknown. If there is an entry, the person is known, they only need to be checked in.
8. If there is no entry in people data table, then the person is unknown, which means there must be an entry added onto people data table using face id, along with the image link.
9. When the person is known, a message indicating to 'check in" the guest along with their face id is sent to the SQS queue. And if the person is unknown, then a message indicating to 'register the unknown' guest along with their face id is sent to the SQS queue.
10. To let the host know that a new guest attended their meetup, an email is sent with their photo link
11. Cognito Identity pool is given access to the SQS, which is used by the UI to poll the messages on the queue
12. When the SQS message is to check in the user, the check in API is called sending the face-id received from the SQS queue.
13. The check-in lambda is triggered by the check-in api and an entry is added in the meetup data table by reading the details of the known person from the people data table, using the face-id received.
14. When the SQS message indicated that the guest is unknown, then the lex api is triggered which starts to request for guest's email address, and that is sent to the lex api
15. After Lex Lambda receives the email address, it hits the People Data Labs API with the email address, and processes any data received from it. If the API returns name, work place, and latest educational institution attended, then those slots are filled and a fulfillment hook is invoked. Otherwise, all the data that is not returned by the API is asked from the guest, to fulfill the slots.
16. After the details are gathered from the user, upon fulfilment, these details are indexed on the elastic search services
17. Based on the information gathered from the user, clearbit API and CollegeScoreCard API are hit to gather more information about the institutions.
18. The details gathered from the APIs are stored in the data tables as well, both to mark the user as known, and to mark the user as checked in.

19. There is another lambda that reads the data from DynamoDB occasionally and updates the manifest fine on S3
20. The manifest file on S3 is read by the Quicksight service and insights are provided as follows:






Count of Records by Industry
Technology
Finance
Retail
Information Technology
Internet Software & Services
Insurance
Industry
Technology
Retail
Internet Soft...
Insurance
Information ...
Finance
Group By: industry


Count of Records by Unistate
Virginia
TX
NY
NE
MA
Florida
CA
uniState
CA
Count 2
0
1
2
3
4

Key Differences between Assignment 2 and MeetUp Assistant:

Use case Point of View:

- Assignment 2 was primarily a simulation of a smart door, while the MeetUp assistant is focused on helping anyone organizing an event to check-in guests automatically and also visualize the category of people attending the event.
- MeetUp Assistant not only determines whether the guest is known or unknown, but also automatically triggers a conversation with the guest detected, and only gathers information that is absolutely necessary, nothing more, keeping the interaction as minimal as possible. This was possible from the help of all the APIs that are being used. For example, at first only the email id is asked of the guest, and if the People Data API is returning all the necessary data for the API, then the assistant simply checks in the guest and gathers their data in the background.
- MeetUp Assistant aims to help the host in all possible ways in organizing the events, one of which is to provide quick analyses on the guests who have attended the event. There is also a possibility of analyzing the details of all entries of the guests from all the past meetups, so that a forecast can be made, or a frequent attendee list can be prepared.

Technical Point of View:

- Using an SQS between the face detection lambda and the UI, automated the process further giving an opportunity to make the interaction with the guests as minimal as possible.
- Unlike in the assignment 2, where the owner enters the details of the visitor, MeetUp Assistant sparks a very short conversation automatically upon recognition and keeps the chat short.
- Maintaining two data tables, one for check-in and one for all the known guests, gave the MeetUp Assistant a way to overcome the issue of Rekognition continually streaming the detected faces repeatedly.
- MeetUp Assistant periodically collects the meetup guests data and pushes it to QuickSight for detailed analyses.

Challenges:

- Facebook Graph API is out of service. Our initial idea was more elaborate, including an analysis of the personal details of the guests as well, upon their consent. This had to be reduced to only professional and education information using the other APIs available.

- People Data API is inconsistent with many user accounts. This had made us resort to extending the chat with the user to an additional extent in cases where the API does not provide any data.
- Kinesis Video Streaming with Rekognition is sometimes unstable, which made us repeat a lot of processes in order to even work with the service.
- Rekognition processing abruptly stops at times. This called for restarting the Rekognition data processor many a time.
- Rekognition identifies faces continuously. This pipeline of KVS service seems to have a better use for processes like camera surveillance. While Rekognition is not customizable to process that kind of stream of video data differently. This is very useful in many cases, but in use cases like MeetUp assistant, it gets challenging.
- Rekognition is poor at times.
- Amazon QuickSight Standard edition does not support many crucial features (sending analyses to UI)
- Challenges in preprocessing the api data to make a uniform and consistent model for all people

Future Scope:

- Meet-Up is only one application for this idea.
- Any ad-hoc meeting can use this application
- Parties, unofficial gatherings can use a quick check-in using facial recognition
- An extension of this idea is to verify students who misplaced their id cards
- Discussion Topic Ranking (choosing the best topic from among the favorite topics gathered from used upon check-in)
- An extra layer of verifying the guests attending for any red flags, like criminal records, can be added. This could help avoid a potential threat.