Web Application Design: FocusMusic

Jiachen Du
jd3488@columbia.edu

Yuqin Xu
yx2478@columbia.edu

Tian Xia
tx2178@columbia.edu

Abstract

Our project is to build a web application to help users find music they want. We mainly use IAM, Cognito, SES, Lambda, API Gateway, SQS, S3, DynamoDB, Elasticsearch, Rekognition, Amazon Lex to implement.

1 EXTERNAL LINKS

youtube video: https://www.youtube.com/watch?v=JJXNjR8_Cdsfeature=youtu.be

slide: https://docs.google.com/presentation/d/1hlp2ZZ0aY9yLou85kbwQy4kMX_g-FExVAGmG6

TKz9Ic/edit?usp=sharing

github: https://github.com/lmxs1237/6998-Project_MusicFocus

2 INTRODUCTION

Our project focuses on building a single music-focused web application based on Amazon Web Service. After registering and logging in to the web application through email or social media account, the users are able to search by album name, key word, album cover as well as upload album covers. With one album page open, users can play the music in or comment on it conveniently. Their comments should be saved and be visible for later users. It is also implemented with an intelligent chat bot which can recommend music list based on users' mood, condition, music preference and so on. In the end, it is easy to contact us in real time in case there are any suggestions given to us. Our project mainly use IAM, Cognito, SES, Lambda, API Gateway, SQS, S3, DynamoDB, Elasticsearch, Rekognition, Amazon Lex to implement.

3 AWS ARCHITECTURE

The following pictures are the two parts of our application architecture.

The picture above shows the architecture of the searching part, in this part, we used API Gateway to connect the frontend in S3 and backend in lambda. Then, we used the Lambda function to call the AWS Rekognition to detect the picture feature labels and used the results to search the matching pictures in elastic searching area. Besides, we also built a table containing the album information we scraped and used the cover ID fetched from elasticsearch domain to get the infomation from DynamoDB.

The architecture above can be divided into several parts, the frontend part can directly used S3 and DynamoDB to achieve the comment and songs playing function. And the Lex Bot can be triggered by frontend, get the keywords, Lambda fuction LF4 hooks them and sends to SQS query, then the query is catched by LF5, searched corresponding playlist in elasticsearch domain, finally send the information back to user via SES.

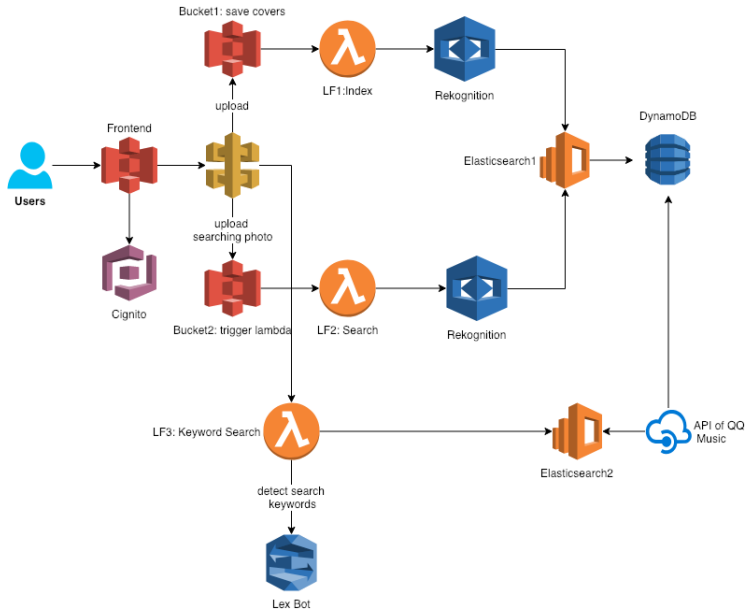


Fig. 1. Part1:The search function architecture

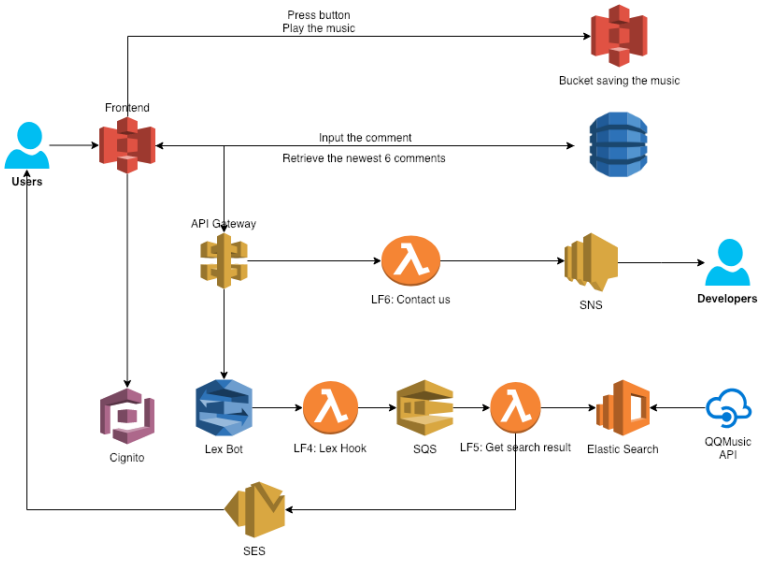


Fig. 2. Part2:The songlist recommendation and contact function architecture

4 FEATURES IMPLEMENTATION AND RESULTS

4.1 Frontend

Goals want to achieve:

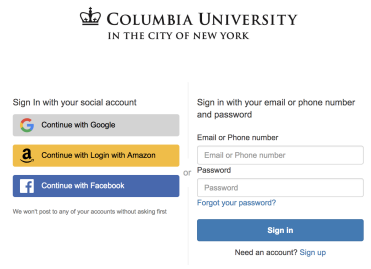
We use html, css and js build a concise web application. Users need to sign in first to use this app. After signing in, users will have the permission to listen to every song in every album and leave their comments.

Detailed Steps:

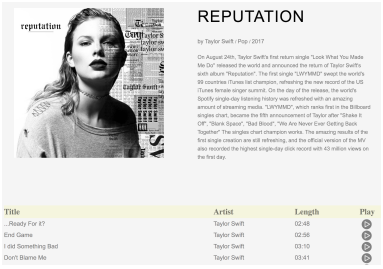
a. Using AWS lambda Cognito User Pool to manage users. Users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, or Google. Then create an Identity Pool to provide IAM credentials to logged in users.

b. Create a S3 bucket to store the songs. When users click on the play or pause button in the song list in Album Info, it will link to the object URL in S3 and play the song.

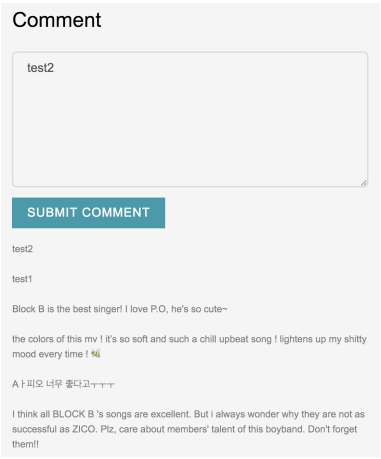
c. We use Dynamodb to realize comment. When an user writes a comment under an album and uploads it, we will first get the item count in this album's Dynamodb table, and store this comment with id = 'itemcount' in table. Then retrieve the newest 6 comments and show under this album.



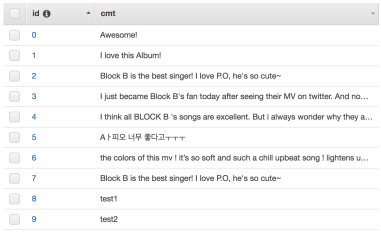
(i) Log in



(ii) Album Info



(iii) Comment



(iv) Comment stored in Dynamodb

Fig. 3. Features realized in frontend

4.2 Index

Goals want to achieve:

This part is the initial step we achieved in building the application, when we upload the cover pictures into the bucket, the lambda function of indexing pictures will be triggered, used AWS Rekognition to detect the picture labels and then stored them into the elasticsearch area. This function is designed for web designers so that it is not an open function to users.

Detailed Steps:

a. Create a Lambda function to index cover pictures.Create a S3 bucket to store the photos.

b. Create a S3 bucket to store the covers, and set a PUT event trigger to lambda function.

c. Inside the lambda function, we call AWS Rekognition to detect picture labels, then form an index message as the following stucture:

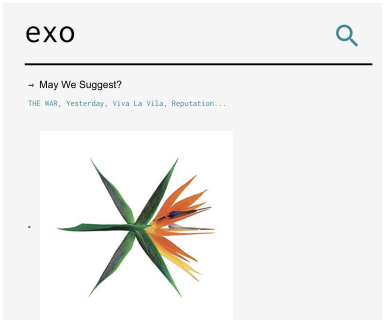
d. Create a elasticsearch area, where to put the formed index messages.

4.3 Search

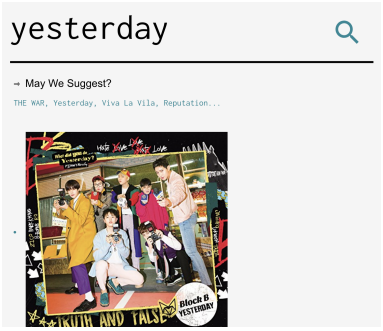
Goals want to achieve:

In this part, we will achieve the searching functions of our application, the users can either upload a cover picture or just input the album's name or singer's name to search the corresponding albums.

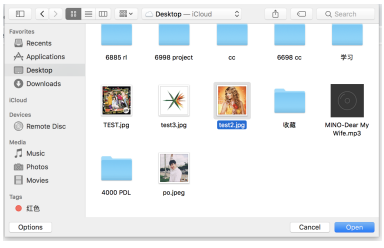
- Search by key word
  This function is achieved by Lex, lambda together with elastic search.
  Detailed Steps:
  a. Create a Lambda function to use the input keywords to search corresponding results.
  b. Create an Amazon Lex bot to handle search queries.
  c. Implement the indexing Lambda function, if the Lex disambiguation request yields any keywords search the ElasticSearch index for results, and return them accordingly. Otherwise, return an empty array of results.
- Search by album cover
  This function is achieved by lambda function, Rekognition and elasticsearch index.
  Detailed Steps:
  a. Create a S3 bucket to store the upload covers.
  b. Create a Lambda function triggered S3's PUT event.
  c. Use the Rekognition function to get the cover's labels, then search the corresponding result in elasticsearch area.
  d. Use the cover ID fetched from elasticsearch area, trace the record from DynamoDB, get the album's website url and send back to user.



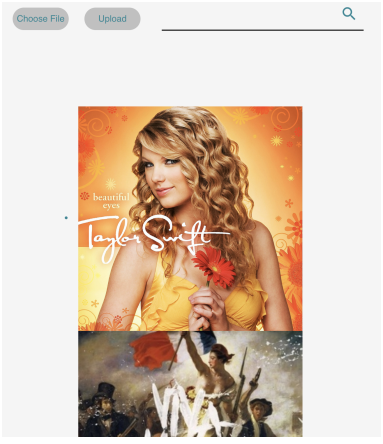
(i) Search by singer's name



(ii) Search by album's name



(iii) Search by upload an album cover: upload



(iv) Search by upload an album cover: result

Fig. 4. Search function

4.4 Chatbot

Goals want to achieve:

When users want to find a music list which is suitable for the mood at this moment and situation, they can use this web application's chatbot function. We will recommend it for you.

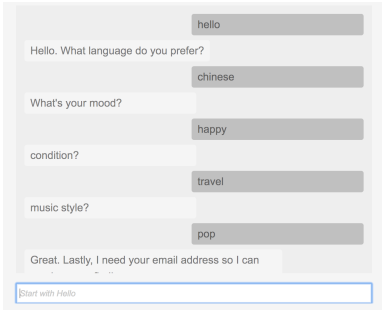
Detailed Steps:

a. Build a chatbot using AWS Lex. It will collect information about the music list users want: music language, mood, situation and music style.

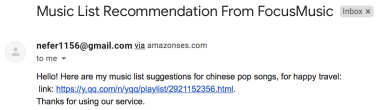
b. Create a Lambda function (chat) and use it as a code hook for Lex, which essentially entails the invocation of your Lambda before Lex responds to any of your requests. Then we collect the information and push it to an SQS queue.

c. Use QQ music API build an ElasticSearch which contains label about language, mood, situation and music style.

d. Create another Lambda function (songlist). This Lambda will be triggered every 1 min, Using Cloudwatch Event. When being triggered, it will pull information from the SQS queue and send it to ElasticSearch. Get the most match music list link and send an email to user using SES.



(i) Recommend music list



(ii) Receive an email containing list link

Fig. 5. Chatbot function

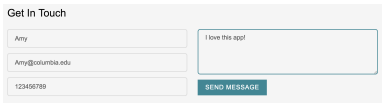
4.5 Contact

Goals want to achieve:

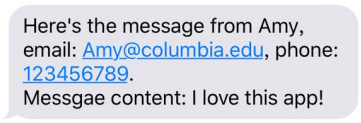
After using this web application, users can leave their suggestions and opinions to us.

Detailed Steps:

a. When users leave their basic information and message, the frontend will invoke Lambda function and send message to us with SNS



(i) Leave message to developers



(ii) Receive a message containing message

Fig. 6. Contact function

5 CONCLUSIONS

In this project, we successfully designed a music searching and recommendation web application with the functions of AWS. While doing the project, we used the lambda function, API Gateway, Rekognition, elasticsearch and other method we learnt from the class. Finally, we achieved the designed functions of our application, besides, compared the existing music search websites, our project has an innovation of searching albums via album cover. Although we've achieved the aims we made, there are still many other functions we can improve, such as searching songs via music and so on, which can be completed in the future.

6 Future Work

For future work, we can do several development: Firstly, the newest version of this web application can only support songs of Chinese, English, Korean and Japanese. We need to use QQ music API to get more songs with different language. Secondly, the function of search by photo can not exactly return the album we want, but return several similar albums. We will consider use Machine Learning to improve the recognition capability for album covers.

References

[1] AWS Document. https://docs.aws.amazon.com/index.htmllang/en_us