Food First: Personal Restaurant Recommendation application

Zihui Yuan(zy1028), Guangwei Zhang(gz667), Weijun Zhai(wz1131)
Instructor: Sambit Sahu
New York University

Abstract

A personal restaurant recommendation web application was built in this project. A group of AWS services were used to build the serverless architecture system. Various recommendation schemas including machine learning classification, keywords search and Kinesis user log analysis were adopted.

1 Introduction

Food First is a personal restaurant recommendation web application based on several AWS components such as S3, Cognito, Api Gateway, Lambda, ElasticSearch Service, AWS Machine Learning, Kinesis,etc.

When using this application, users can set their own profile and food preference. Users can search restaurants by key in keywords or choosing filters, by the food preferences they set or see restaurants most user like. Locations of recommended restaurants will show in a google map window. Users can jump to the yelp pages of our recommendation restaurants or sent the detail information of these restaurants to their personal phone as texts. The results are recommended based on AWS Machine Learning and log information from Kinesis.

2 Related Technology

Amazon Simple Storage Service is storage for the Internet. It is designed to make web-scale computing easier for developers.
AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes code only when needed and scales automatically, from a few requests per day to thousands per second.
Amazon API Gateway is an AWS service that enables developers to create, publish, maintain, monitor, and secure APIs at any scale.
Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps.
Elasticsearch is an open-source, RESTful, distributed search and analytics engine built on Apache Lucene.
Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
Amazon Kinesis Data Analytics enables to quickly author and run powerful SQL code against streaming sources to perform time series analytics, feed real-time dashboards, and create real-time metrics.
Amazon Simple Queue Service is a fully managed message queuing service that makes it easy to decouple and scale

microservices, distributed systems, and serverless applications.

Amazon Simple Notification Service is a web service that coordinates and manages the delivery or sending of messages to subscribe endpoints or clients.

Amazon Machine Learning is a robust, cloud-based service that makes it easy for developers of all skill levels to use machine learning technology.

This system was built on top of various AWS services with a serverless web application architecture (Fig 1). The static front-end web page was hosted on S3. The authentication process was deployed by AWS Cognito. Application calls were made by HTTP request through AWS API Gateway. The backend logic was deployed by using Lambda function.

3 Architecture

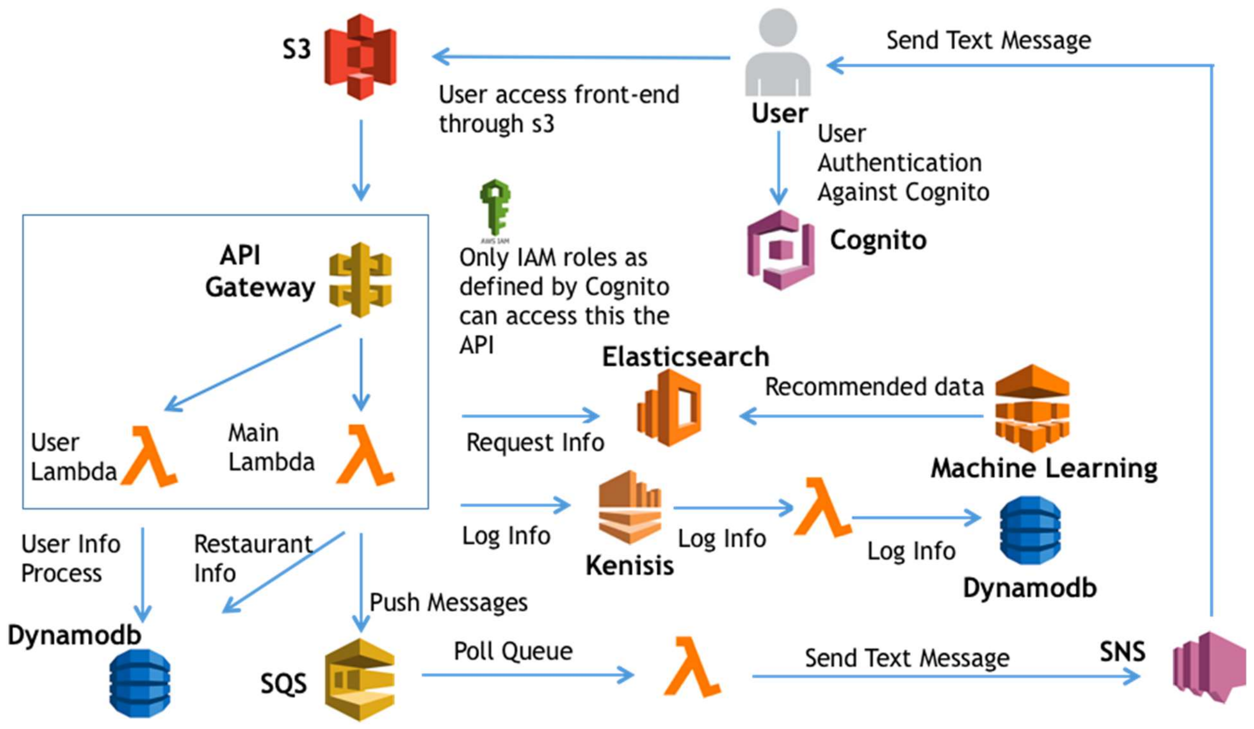

Fig 1. System Architecture

4 Implementation

4.1 Front End

Our front end is implemented by html and JavaScript, AWS JavaScript SDK, AWS Cognito SDK for JavaScript and google map api. It contains three pages: login page, main search page and user profile setting page. On login page, front end connected AWS Cognito user pools and identity pools by AWS Cognito SDK for JavaScript to get authentication to access Api Gateway on other two pages. On main search page, users can get their recommendation lists and see their locations in a google map window. On user profile setting page, users can set their basic information and their own food preference. These two pages are connected with the back-end lambda functions by Amazon Api Gateway.

4.2 Search Module

This module acts like a main control to provide main functions. Currently, there are four functions related to it:

1. Restaurant search
Search restaurants by Elasticsearch Service. The user can key in keywords, set preference, select tags, etc. to modify the search conditions. Elastic Search will return several, currently we set it to 20, restaurant IDs most relevant to the given conditions. Then complete information of corresponding restaurants is fetched from DynamoDB to display.
Then fetch complete information of corresponding restaurants from DynamoDB to the user.

2. Text sender
Send a text reminder at request of the user. The request message with restaurant id and user's phone number is sent to SQS. A message worker polls SQS every minute and sends text messages to users according to requests on SQS.

3. Click log
If interested in a restaurant, the user can click the button to send a text reminder. This action is logged and sent to a Kinesis stream for real time data analytics. Top 20 most frequently occurring restaurants over a 60-second tumbling window are delivered to destination lambda function. We keep this information as a DynamoDB item to simplify the query process.

4. Log query
Query the analytics result from DynamoDB to show restaurants most frequently interested. It's like the search function.

4.3 User Module

Cognito was adopted to provide user authentication service. Some of the user information was stored in Cognito. When user login into the system, the system will access the user name from the Cognito. After that the application will request user preference setting from DynamoDB based on the username. Then the system will provide recommendation information based on the user information provide by DynamoDB.

5 Code Details

5.1 Front End

All-important front-end codes are in three html files and all main function are implemented by JavaScript.

index.html:
Function 'login()' and 'authentication()' get the user login inputs, connect Amazon Cognito User pool to check the user information and then use the id_token to get the AccessKeyId, SecretKey and SessionToken. Also there are some tool functions,more details please see 'index.html'.

search-list.html:
Function 'GetQuery()' get the search input based on the search type the user choose. Function 'Mysearch()' access the backend

lambda function through Api Gateway which authentified by the AccessKeyId, SecretKey and SessionToken we get above. Function 'showList()' will generate a functional list to show use the recommendation results. Also, there are some tool functions like google map functions, more details please see 'search-list.html'.

edit-profile.html:
Function 'loaduser()' access the backend lamda function through Api Gateway which authentified by the AccessKeyId, SecretKey and SessionToken we get above, it load the current login user's personal profile and show them on the front page. Function 'updateUser()' access the backend lamda function through Api Gateway which authentified by the AccessKeyId, SecretKey and SessionToken we get above, it update the user's profile according to user's changes in front end. Also there are several tool functions such as 'addPrefer()' which help users to add or delete their preferences. For more details see 'edit-profile.html'

5.2 Lambda Function 1 – userHandler

This lambda function handles the functions of putting, updating user information in the DynamoDB and getting information from it. When the user login into the system and put or update his/her restaurant preference in the system, the front-end web application will send a HTTP post request to this lambda function, then the lambda function will update this information in the DynamoDB accordingly.

5.3 Lambda Function 2 – Message Handler

The lambda function, messageHandler, act as the main control of the search module. It firstly parses the messages coming in and identify the "type" of the message. The message type is kept in the attribute called "searchtype". There are 4 types in total:

1. normal
A normal restaurant search with searching conditions. The handler will call search_list(), to search for a list of restaurants qualifies the search. In search_list(), keywords are parsed into query conditions to refer to the Elasticsearch and gets an id list via query_es(). Then the id list is mapped to DynamoDB items and get corresponding information with query_dynamodb(). The result with detailed restaurants information is packed into a response message as a list and send back to client.

2. sendtext
A request to send text reminder. The handler will call send_message(), to pares and get restaurant id and user's phone number. Then query_dynamodb() with the restaurant id to get detailed information such as name and location. The information together with user's phone will be sent to SQS.

3. clicklog
A signal to indicate that the user has clicked a restaurant. The handler will call log_click() which connects to Kinesis stream and put in the restaurant id as a record.

4. logquery
A query to retrieve the Kinesis analytics result we put in DynamoDB. The handler will call log_query() to get the id list and

then call map() and query_dynamodb() for detailed information.

5.4Lambda Function 3 – ccproject_click_count

The lambda function, ccproject_click_count, is set to be the destination of Kinesis data analytics.
Click log information sent to Kinesis as in-application stream. The Kinesis Analytics application continuously performs real time analytics with SQL on the source data, to find the most frequently occurring values in a stream using the Space Saving algorithm. The result is delivered to the lambda function via destination SQL stream. In the lambda function, stream data is decoded and then store as a special item into DynamoDB.

5.5 Lambda Function 4 – text_sender

The lambda function, text_sender, is triggered every minute to poll messages from SQS. Then the message is sent to user’s given phone number.

6 Discussion and Future Work

With the highly scalable serverless architecture properly built, more function can be integrated into the system easily. For instance, a Lex NLP function can be integrated into the keywords input component to improve the system’s service. A restaurant reservation component can be added to the application. The service also can be scaled to provide hotel recommendation and reservation.