

Cloud Computing and Big Data - Spring 19

Homework Assignment 1

Assignment:

Customer Service is a core service for a lot of businesses around the world and it is getting disrupted at the moment by Natural Language Processing-powered applications.

In this first assignment you will implement a serverless, microservice-driven web application. Specifically, you will build a Dining Concierge chatbot, that gives you restaurant suggestions given a set of preferences that you provide the chatbot with through conversation.

Outline:

This assignment has five components:

1. Build and deploy the frontend of the application

- a. **Implement a chat user interface**, where the user can write messages and get responses back. You can use open source libraries and frameworks that give you this UI and UX out of the box.
- b. **Host your frontend in an AWS S3 bucket**
 - i. Set the bucket up for website hosting
 - ii. <https://docs.aws.amazon.com/AmazonS3/latest/dev/HostingWebsiteOnS3Setup.html>

2. Build the API for the application

- a. Use API Gateway to setup your API
 - i. use the following API/Swagger specification for your API
 - <https://github.com/001000001/aics-columbia-s2018/blob/master/aics-swagger.yaml>
 - Use <http://editor.swagger.io/> to visualize this file
 - You can **import the Swagger file into API Gateway**
 - <https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-import-api.html>
 - **Create a Lambda function (LF0)** that performs the chat operation

- Use the request/response model (interfaces) specified in the API specification above
- ii. For now, just implement a boilerplate response to all messages:
 - ex. User says anything, Bot responds: "I'm still under development. Please come back later."
- b. Notes
 - i. You will need to **enable CORS on your API methods**
 - <https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-cors.html>
 - ii. API Gateway can **generate an SDK for your API**, which you can use in your frontend. It will take care of calling your API, as well as session signing the API calls -- an important security feature
 - <https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-generate-sdk-javascript.html>

3. Setup Authentication using Cognito

- a. Setup AWS Cognito to manage your users, using User Pools
 - i. <https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html>
- b. Create a Cognito-generated login page to authenticate the users
 - i. <https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-ux.html>
- c. Create an Identity Pool and configure Cognito Identity Provider Manager to provision temporary IAM credentials to your logged in users
 - i. <https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-identity.html>
- d. Enable IAM Authentication on each API method of your API Gateway API
 - i. Once you enable this, only authenticated users in your application should be able to access your API
- e. Add execute permissions to the Authenticated Cognito IAM role to call your API
 - i. <https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-iam-policy-examples-for-api-execution.html>

4. Build a Dining Concierge chatbot using Amazon Lex.

- a. Create a new bot using the Amazon Lex service. Read up the documentation on all things Lex, for more information:
 <https://docs.aws.amazon.com/lex/latest/dg/getting-started.html>
- b. Create a Lambda function (LF1) 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 -- this gives you the chance to manipulate and

validate parameters as well as format the bot's responses. More documentation on Lambda code hooks at the following link:

<https://docs.aws.amazon.com/lex/latest/dg/using-lambda.html>

c. Bot Requirements:

- i. Implement at least the following three intents:
 - GreetingIntent
 - ThankYouIntent
 - DiningSuggestionsIntent
- ii. The implementation of an intent entails its setup in Amazon Lex as well as handling its response in the Lambda function code hook.
 - Example: for the GreetingIntent you need to 1. create the intent in Lex, 2. train and test the intent in the Lex console, 3. implement the handler for the GreetingIntent in the Lambda code hook, such that when you receive a request for the GreetingIntent you compose a response such as "Hi there, how can I help?"
- iii. For the DiningSuggestionsIntent, you need to collect at least the following pieces of information from the user, through conversation:
 - Location
 - Cuisine
 - Dining Time
 - Number of people
- iv. Based on the parameters collected from the user, get restaurant suggestions using the Yelp API. Format the top few results and return them to the user.

5. Integrate the Lex chatbot into your chat API

- a. Use the AWS SDK to call your Lex chatbot from the API Lambda (LF0).
- b. When the API receives a request, you should 1. extract the text message from the API request, 2. send it to your Lex chatbot, 3. wait for the response, 4. send back the response from Lex as the API response.

Example Interaction

User: Hello

Bot: Hi there, how can I help?

User: I need some restaurant suggestions.

Bot: Great. I can help you with that. What city or city area are you looking to dine in?

User: Manhattan

Bot: Got it, Manhattan. What cuisine would you like to try?

User: Japanese

Bot: Ok, how many people are in your party?

User: Two people

Bot: A few more to go. What date?

User: Today

Bot: What time?

User: 7 pm, please

Bot: Here are my Japanese restaurant suggestions for 2 people, for today at 7 pm: 1. Sushi Nakazawa, located at 23 Commerce St, 2. Jin Ramen, located at 3183 Broadway, 3. Nikko, located at 1280 Amsterdam Ave. Enjoy your meal!"

ANNEX

Architecture Diagram

