

Cloud Computing and Big Data - Spring 18

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. This assignment is the first in a series of three assignments, that will have you build an AI Customer Service experience.

Outline:

This assignment has four main components:

- Frontend
 - **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.
 - **Host your frontend in an AWS S3 bucket**
 - Set the bucket up for website hosting
 - <https://docs.aws.amazon.com/AmazonS3/latest/dev/HostingWebsiteOnS3Setup.html>
- Backend
 - **Use API Gateway to setup your API**
 - 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** that performs the chat operation
 - Use the request/response model (interfaces) specified in the API specification above

- **Upon receiving a message from the user, respond with an appropriate response**
 - ex. User says: "Hello", Bot responds: "Hi there, how can I help?"
 - You should have a few simple examples for this interaction. In a later assignment we will be enhancing the bot's capabilities using NLP technology
- Notes
 - You will need to **enable CORS on your API methods**
 - <https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-cors.html>
 - 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>
- Authentication
 - **Setup AWS Cognito to manage your users**, using User Pools
 - <https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html>
 - **Create a Cognito-generated login page** to authenticate the users
 - <https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-ux.html>
 - **Create an Identity Pool** and configure **Cognito Identity Provider Manager** to provision temporary IAM credentials to your logged in users
 - <https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-identity.html>
- Security
 - **Enable IAM Authentication on each API method** of your API Gateway API
 - Once you enable this, only authenticated users in your application should be able to access your API
 - **Add execute permissions to the Authenticated Cognito IAM role** to call your API
 - <https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-iam-policy-examples-for-api-execution.html>

ANNEX

Architecture Diagram

