Cloud Computing and Big Data - Fall 18

Homework Assignment 2

Assignment:

In this assignment you will continue to iterate over your application from the first assignment. You have to build intelligence into your chatbot API. In addition, you will design a truly scalable distributed application leveraging message queues and notifications.

Specifically, you have to build a Dining Concierge chatbot, that will email you restaurant suggestions given a set of preferences that you give the chatbot through conversation.

Outline:

This assignment has four components:

1. **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
2. **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
            - Phone number

3. **Build a suggestions module, that is decoupled from the Lex chatbot.**
    a. During the fulfillment step (https://docs.aws.amazon.com/lex/latest/dg/API_FulfillmentActivity.html) of the DiningSuggestionsIntent, push the information collected from the user (location, cuisine, etc.) to an SQS queue. More on SQS queues here: https://aws.amazon.com/sqs/
    b. Create a new Lambda function (LF2) that acts as a queue worker. Whenever it is invoked it 1. pulls a message from the SQS queue, 2. gets restaurant suggestions based on its parameters using one or more APIs such as Yelp or Google Places, 3. stores the suggestions in a DynamoDB

    table alongside the message (for recordkeeping), 4. formats the suggestions, and 5. sends them over text message to the phone number included in the SQS message, using SNS (https://docs.aws.amazon.com/sns/latest/dg/SMSMessages.html).
        i. More on DynamoDB table creation: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.CreateTables.html
    c. Set up a CloudWatch event trigger that runs every minute and invokes the Lambda function as a result: https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/RunLambdaSchedule.html. This automates the queue worker Lambda (LF2) to poll and process suggestion requests on its own.

4. **Integrate the Lex chatbot into your chat API from Assignment 1.**
    a. Use the AWS SDK to call your Lex chatbot from the Lambda function that you created in Assignment 1.
    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.
    c. If you did everything correctly, you should be able to leverage the frontend from Assignment 1, with no additional modifications.

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: Great. Lastly, I need your phone number so I can send you my findings.
User: 123-456-7890
Bot: You're all set. Expect my recommendations shortly! Have a good day.
User: Thank you!
Bot: You're welcome.

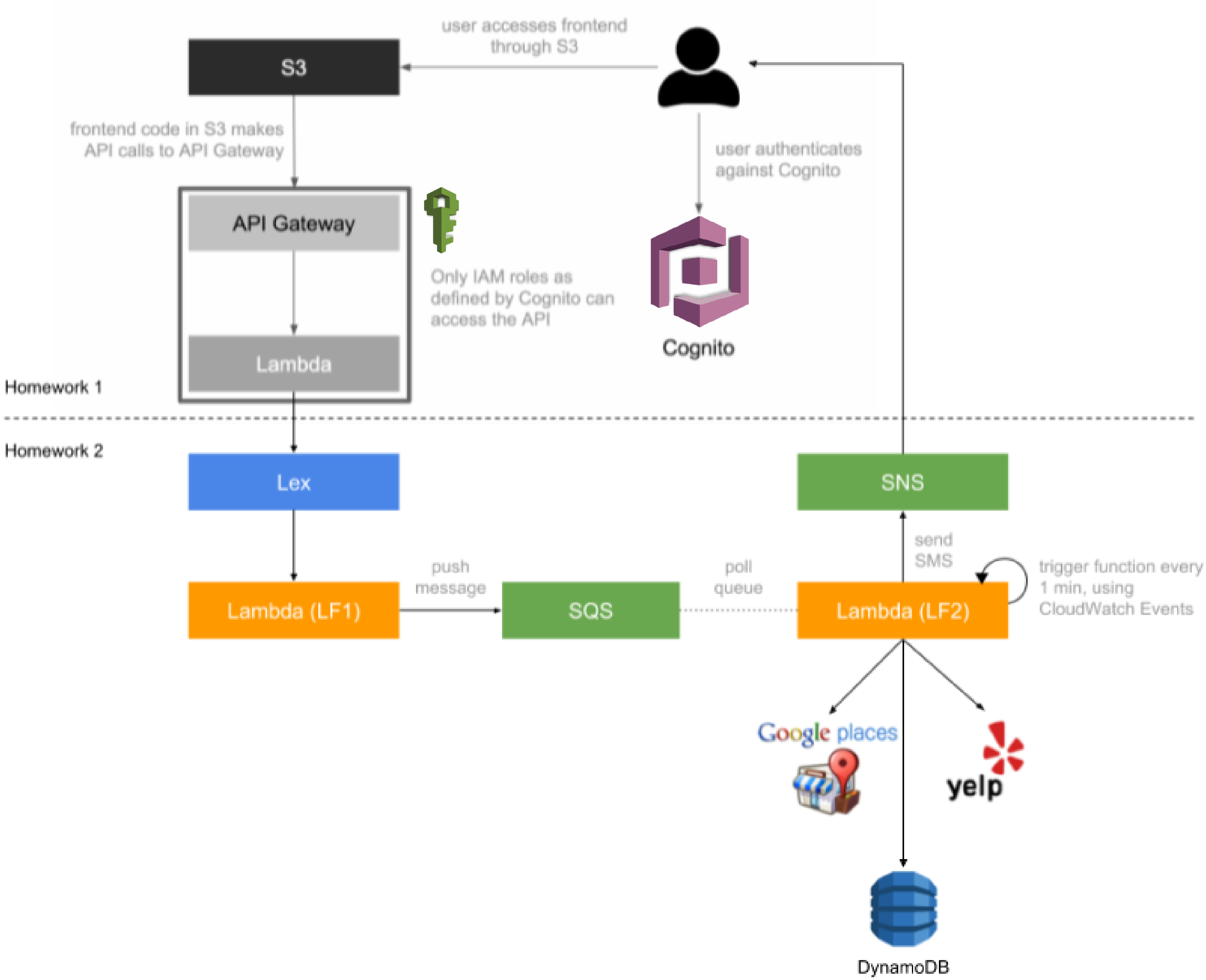
(a few moments later)

User gets the following text message:

"Hello! 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[1]



user accesses frontend through S3
S3
frontend code in S3 makes API calls to API Gateway
API Gateway
Lambda
Only IAM roles as defined by Cognito can access the API
user authenticates against Cognito
Cognito
Homework 1
Homework 2
Lex
SNS
send SMS
trigger function every 1 min, using CloudWatch Events
Lambda (LF1)
push message
SQS
poll queue
Lambda (LF2)
Google places
yelp
DynamoDB

---

[1] Homework 2 components are in shown in color.